

# Improving Users' Acceptance in Recommender System

Chen Wei

B.Eng. in Software Engineering

South China University of Technology

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2013

---

# ACKNOWLEDGEMENTS

First and foremost I would like to thank my supervisors, Professor Wynne Hsu and Professor Mong Li Lee for their valuable guidance, continuous support, encouragement and freedom to pursue independent work throughout my Ph.D study. Above all, they are like my friend, which I appreciate them from my heart.

I would also like to thank my thesis committee, Professor Anthony K. H. Tung and Professor Chew Lim Tan, who provided encouraging and constructive feedback. To the many anonymous reviewers at the various conferences, thank you for helping to shape and guide the direction of my work with your careful and detailed comments.

I would also like to thank my classmates in the Database Research Lab for their supports and friendship especially during the many sleepless night rushing to complete experiments before conference deadline. Specially, I would like to thank my parents for supporting me spiritually throughout my life.

Last but not the least, I would like to thank my wife Zhou Ye for her personal support and great patience. Without her encouragement and understanding, it would have been impossible for me to finish my Ph.D study.



---

# TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Improving users' acceptance using Rating and Tagging Data . . . . .	2
1.2	Improving users' acceptance using Cross Domain Data . . . . .	4
1.3	Improving users' acceptance using Social Trust Data . . . . .	6
1.4	Contributions . . . . .	7
1.5	Organization . . . . .	9
<b>2</b>	<b>Literature Review</b>	<b>11</b>
2.1	Recommender System . . . . .	11
2.2	Techniques of Recommender System . . . . .	12
2.2.1	Content Filtering . . . . .	12
2.2.2	Collaborative Filtering . . . . .	14
2.2.3	Measurement of Users' Acceptance . . . . .	19
2.3	Recommender System using Rating and Tagging Data . . . . .	21
2.4	Recommender System using Cross Domain Data . . . . .	25
2.4.1	Latent feature shares . . . . .	26
2.4.2	Binary Knowledge Transfer using Cross Domain Data . . . . .	27
2.4.3	Ternary Knowledge Transfer using Cross Domain Data . . . . .	28
2.5	Recommender System using Social Trust Data . . . . .	28
2.5.1	Neighborhood-Based Model using Social Trust Data . . . . .	29
2.5.2	Model-Based using Social Trust Data . . . . .	31
<b>3</b>	<b>Improving users' acceptance using Rating and Tagging Data</b>	<b>33</b>
3.1	Motivation . . . . .	34
3.2	Tensor algebra and multilinear analysis . . . . .	36
3.3	Recommender System Overview . . . . .	41

3.3.1	Recommender Engine - Quaternary Semantic Analysis . . . . .	44
3.3.2	Top-N Recommendation and Prediction . . . . .	49
3.3.3	Tag-based Explanation and Feedback . . . . .	51
3.4	Experimental Studies . . . . .	60
3.4.1	Experiments on Users' Acceptance . . . . .	61
3.4.2	Sensitivity Experiments . . . . .	71
3.5	Summary . . . . .	72
<b>4</b>	<b>Improving users' acceptance using Cross Domain Data</b>	<b>75</b>
4.1	Motivation . . . . .	76
4.2	Problem Formulation . . . . .	77
4.3	Cross Domain Framework . . . . .	80
4.3.1	Cluster-Level Tensor . . . . .	80
4.3.2	Fusing Social Network Information . . . . .	84
4.4	Experiments . . . . .	88
4.4.1	Experiments on Users' Acceptance . . . . .	89
4.4.2	Sensitivity Experiments . . . . .	95
4.4.3	Case Study . . . . .	97
4.4.4	Scalability . . . . .	98
4.5	Summary . . . . .	99
<b>5</b>	<b>Improving users' acceptance using Social Trust Data</b>	<b>101</b>
5.1	Motivation . . . . .	102
5.2	Problem Formulation . . . . .	104
5.3	Proposed Method . . . . .	105
5.3.1	Receptiveness over Time Model . . . . .	105
5.3.2	Applications of RTM . . . . .	115
5.4	Experimental results . . . . .	117
5.4.1	Experiments on Users' Acceptance . . . . .	119
5.4.2	User Interest Change Case Study . . . . .	121
5.4.3	User Receptiveness Case Study . . . . .	122
5.4.4	Sensitivity Experiments . . . . .	123
5.5	Summary . . . . .	124
<b>6</b>	<b>Conclusion</b>	<b>125</b>
6.1	Future Work . . . . .	126

---

# SUMMARY

Personalized recommender systems aim to push only the relevant items and information directly to the users without requiring them to browse through millions of web resources. The challenge of these systems is to achieve a high user acceptance rate on their recommendations. Collaborative filtering is a method of increasing user' acceptance towards recommendation (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). In this thesis, we focus on improving user's acceptance by collaborative filtering on three popular user-generated data types: social tagging and rating data, cross domain data and social trust data. We outline our approaches as follows.

First, we study the problem of increasing the user's acceptance using social tagging and rating data. We show that ternary relationships such as users-items-ratings, or users-items-tags, are insufficient to increase user' acceptance towards recommendations. Instead, we model the quaternary relationship among users, items, tags and ratings as a 4-order tensor and cast the recommendation problem as a multi-way latent semantic analysis problem. A unified framework for user recommendation, item recommendation, tag recommendation and item rating prediction is proposed. Besides that, we also provide the explanation for the recommendation by using tags. Tags are used as intermediary entities that not only relate target users to the recommended items but also

understand users intents. Our system also allows tag-based online relevance feedback. Experiment results on a real world Movielens dataset show that the proposed approach is able to increase the user acceptance compared to the state-of-the-art recommendation techniques.

Next, we study the problem of increasing the user’s acceptance using cross domain data, which enables more accurate recommendation by leveraging the knowledge in the other domain. We first show that high dimension relationships transfer without decomposition may decrease user’ acceptance towards recommendations. Instead, we model the high dimension relationship transfer without decomposition. We propose a generalized cross domain collaborative filtering framework that integrates social network information seamlessly with cross domain data. This is achieved by utilizing tensor factorization with topic based social regularization. This framework is able to transfer high dimensional data without the need for decomposition by finding shared implicit cluster-level tensor from multiple domains. Extensive experiments conducted on real world datasets indicate that the proposed framework outperforms state-of-art algorithms for item recommendation, user recommendation and tag recommendation.

Finally, we study the problem of increasing the user’s acceptance using social trust data. We show that the complex interaction between user interests and the social relationship over time is important to increase the user’s acceptance toward recommendation, which is ignored by existing recommender systems model. We propose a probabilistic generative model, called *Receptiveness over Time Model* (RTM), to capture this interaction. We design a Gibbs sampling algorithm to learn the receptiveness and interest distributions among users over time. The results of experiments on a real world dataset demonstrate that RTM-based recommendation outperforms the state-of-the-art recommendation methods. Case studies also show that RTM is able to discover the user interest shift and receptiveness change over time.

---

# LIST OF TABLES

1.1	Ternary relations among user, rating and item in Book Domain . . . . .	2
1.2	Ternary relations among user, tags, and item in Book Domain . . . . .	3
1.3	Quaternary relations among users, tags, ratings and items in Book Domain	4
1.4	Ternary relations among users, tags, and items in Movies Domain . . .	5
1.5	Social Trust in Books Domain . . . . .	5
1.6	Example of Table 1.2 over Time . . . . .	6
3.1	Meanings of symbols used . . . . .	37
3.2	Example dataset of a 3-order tensor . . . . .	37
3.3	Quaternary relations among users, tags, ratings and items in Book Domain	46
3.4	Data of the tensor $\mathcal{A}$ . . . . .	46
3.5	Output of the approximate tensor $\hat{\mathcal{A}}$ . . . . .	48
3.6	Latent features of users, tags and items extracted. . . . .	55
3.7	Output of the updated approximate tensor $\hat{\mathcal{A}}$ . . . . .	59
3.8	Updated Latent features of users, tags, items and ratings extracted. . . .	60
3.9	Statistics of rating data . . . . .	61
3.10	Comparison of intra- and inter- similarity between QSA and TSA . . .	64
3.11	MAE and Coverage . . . . .	67
3.12	Example explanations for recommended movie. . . . .	68
3.13	Difference between explanation ratings and actual ratings . . . . .	69
3.14	User ratings of preferred explanation style . . . . .	70
3.15	Results of User Feedback . . . . .	71
4.1	Book domain dataset . . . . .	78
4.2	Ternary relations among users, tags, and items in Movies Domain . . .	78
4.3	Clusters for the Movie domain in Table 4.2 . . . . .	81
4.4	Cluster-level tensor in Movie domain. . . . .	82



4.5	Mapping between Book and Movie domains. . . . .	83
4.6	Output tensor $\mathcal{A}_{tgt}^*$ . . . . .	87
4.7	Characteristics of datasets. . . . .	89
4.8	Intra- and inter- similarity between FUSE and TSA . . . . .	95
4.9	Example of Top 10 representative tags for 5 groups in movies and books domain . . . . .	97
5.1	Example datasets . . . . .	103
5.2	Meanings of symbols used . . . . .	106
5.3	Summary of methods. . . . .	119
5.4	Statistics of rating dataset. . . . .	119
5.5	Effect of $K$ and $L$ on RMSE . . . . .	123

---

## LIST OF FIGURES

2-1	User-based CF . . . . .	15
2-2	Latent factor model illustration . . . . .	18
2-3	Tags in Flickr . . . . .	22
2-4	Extend user item matrix by including user tags as items and item tags as users (Tso-Sutter et al. 2008) . . . . .	23
2-5	Tensor representation left (Symeonidis et al. 2008), right (Rendle et al. 2009) . . . . .	24
2-6	Tensor Factorization . . . . .	25
2-7	The correspondence of transfer from Movie Domain to Book Domain . . . . .	26
2-8	User Feedback, Social Relation and its Matrix representation . . . . .	29
2-9	Recommendation based on Social Trust Data . . . . .	29
3-1	Recommendation System Overview . . . . .	42
3-2	Screenshots of recommendation system . . . . .	43
3-3	Distribution of users, tags, and items in $r = 2$ dimensional space. . . . .	53
3-4	Hit ratio for Top N item recommendation . . . . .	63
3-5	Precision and recall for tag recommendation . . . . .	65
3-6	Run time at each time stamp for the incremental and non-incremental algorithms . . . . .	67
3-7	Effect of core tensor dimensions on hit ratio . . . . .	72
4-1	Results for Item Recommendation. . . . .	91
4-2	Results for Item Recommendation. . . . .	91
4-3	Results for Item Recommendation. . . . .	92
4-4	Tag recommendation . . . . .	93
4-5	User recommendation . . . . .	94
4-6	Sensitivity analysis . . . . .	96

4-7	Sensitivity analysis on $\lambda$	96
4-8	Scalability analysis	98
5-1	Graphical model representation of Bi-LDA <sup>social</sup>	108
5-2	Graphical model representation of RTM Model	113
5-3	Accuracy of Rating Prediction	120
5-4	User interest change over time	121
5-5	User interest profiles and their trust relationships	122
5-6	Receptiveness change over time	123
5-7	Sensitivity analysis on $\lambda$	124

---

---

# CHAPTER 1

---

## INTRODUCTION

As we enter the age of social networks, social media has been expanding rapidly, leading to a massive amount of user-generated data. Applications of recommender system typically involve different kinds of data such as rating data from Netflix<sup>1</sup>, social tagging data from Digg<sup>2</sup>, web click log from Google<sup>3</sup>, purchase and review data from Amazon<sup>4</sup>, and location data from Foursquare<sup>5</sup>, etc. At the same time, the growth of crowdsourcing, where knowledge can be harvested from the masses, gives rise to new ways to build intelligent recommender system to increase user' acceptance towards recommendation. While there have been some research works that focus on the mining the knowledge from different kinds of user generated data, more works need to be done. In this thesis, we focus on three types of user generated data. They are social tagging and rating data, cross domain data and social trust data respectively.

---

<sup>1</sup>[www.netflix.com](http://www.netflix.com)

<sup>2</sup>[digg.com](http://digg.com)

<sup>3</sup>[www.google.com](http://www.google.com)

<sup>4</sup>[www.amazon.com](http://www.amazon.com)

<sup>5</sup><https://foursquare.com>

## 1.1 Improving users' acceptance using Rating and Tagging Data

Social network systems such as FaceBook and YouTube have played a significant role in capturing both explicit and implicit user preferences for different items in the form of ratings and tags. This forms a quaternary relationship among users, items, tags and ratings. Existing systems have utilized only ternary relationships such as users-items-ratings [30, 4, 42, 66], or users-items-tags [74, 68, 59] to derive their recommendations. However, recommendations based on ternary relationships which would have missed out important associations and may decrease users' acceptance as it is not accurate.

Table 1.1: Ternary relations among user, rating and item in Book Domain

User	Rating	Item
$U_1$	like	Forrest Gump
$U_1$	like	Beautiful Mind
$U_2$	like	Forrest Gump
$U_2$	like	Groundhog Day
$U_2$	like	Groundhog Day
$U_3$	like	Forrest Gump
$U_4$	dislike	Forrest Gump
$U_4$	dislike	Toy Story
$U_5$	like	New moon
$U_6$	like	New moon
$U_7$	like	Good omens
$U_8$	like	James Bonds Girls
$U_9$	like	Ghost rider
$U_9$	like	James Bonds Girls
$U_9$	like	Scorpia

Let us consider the ternary relationship users-rating-items in Table 1.1. From this table, we conclude that users  $U_1$ , and  $U_2$  have common interests with  $U_3$  since they all like the movie “*Forrest Gump*”. Hence, the movies “*Beautiful Mind*” and “*Groundhog Day*” will be recommended to  $U_3$  because  $U_1$  and  $U_2$  also like “*Beautiful Mind*” and “*Groundhog Day*”.

On the other hand, if we consider the ternary relationship users-tags-items in Table 1.2. The users  $U_2$  and  $U_4$  are said to have common interests with  $U_3$  because they both

Table 1.2: Ternary relations among user, tags, and item in Book Domain

User	Tag	Item
$U_1$	psychology	Forrest Gump
$U_1$	psychology	Beautiful Mind
$U_2$	comedy	Forrest Gump
$U_2$	excellent	Groundhog Day
$U_2$	comedy	Groundhog Day
$U_3$	comedy	Forrest Gump
$U_4$	comedy	Forrest Gump
$U_4$	comedy	Toy Story
$U_4$	overrated	Toy Story
$U_5$	fantasy	New moon
$U_6$	romance	New moon
$U_7$	drama	Good omens
$U_8$	action	James Bonds Girls
$U_9$	action	Ghost rider
$U_9$	action	James Bonds Girls
$U_9$	adventure	Scorpia

tag the movie “*Forrest Gump*” as “*comedy*”. As a result, “*Groundhog Day*” and “*Toy story*” will be recommended to  $U_3$  since  $U_2$  and  $U_4$  also tag “*Groundhog Day*” and “*Toy story*” as “*comedy*”.

Now, instead of the two ternary relationships, we consider the quaternary relationships among users, tags, ratings, and items as shown in Table 1.3. We note that only users  $U_2$  would be highlighted to  $U_3$  and the only movie recommended to  $U_3$  is “*Groundhog Day*”. This is because although  $U_1$  likes “*Forrest Gump*”, he likes it for its psychology aspects as shown by the tag he used psychology, whereas  $U_3$  likes the movie “*Forrest Gump*” as a comedy. Hence,  $U_1$  does not share a common interest with  $U_3$ . As a result,  $U_1$ ’s item “*Beautiful Mind*” will not be recommended to  $U_3$ .

Similarly, although  $U_4$  tags “*Toy Story*” with “*comedy*”, the rating given by  $U_4$  for the movie is “dislike”. In other words  $U_3$  and  $U_4$  have different opinions on “*Forrest Gump*” even though they both use the tag “*comedy*”,  $U_4$  should not be considered as having common interests with  $U_3$ .

Clearly, there is a need to capture the quaternary relationship among users, items, tags and ratings so as to develop more accurate recommender system.

Table 1.3: Quaternary relations among users, tags, ratings and items in Book Domain

User	Tag	Rating	Item
$U_1$	psychology	like	Forrest Gump
$U_1$	psychology	like	Beautiful Mind
$U_2$	comedy	like	Forrest Gump
$U_2$	excellent	like	Groundhog Day
$U_2$	comedy	like	Groundhog Day
$U_3$	comedy	like	Forrest Gump
$U_4$	comedy	dislike	Forrest Gump
$U_4$	comedy	dislike	Toy Story
$U_4$	overrated	dislike	Toy Story
$U_5$	fantasy	like	New moon
$U_6$	romance	like	New moon
$U_7$	drama	like	Good omens
$U_8$	action	like	James Bonds Girls
$U_9$	action	like	Ghost rider
$U_9$	action	like	James Bonds Girls
$U_9$	adventure	like	Scorpia

## 1.2 Improving users' acceptance using Cross Domain Data

With the increasing popularity of social media communities, we now have data repositories from various domains such as user-item-tag data from social tagging in book and movie domains [39] [40], and friendship data between users in social networks [44, 28, 69, 86]. The joint analysis of information from various domains and social networks has the potential to improve our understanding of the underlying relationships among users, items and tags and increase users' acceptance in recommender systems.

For example, users who like to read *romance* books generally have similar preferences as users who like to watch *romance* movies. By learning the characteristics of *romance* lovers from the Movie domain and transferring the learned characteristics to the Book domain, recommender systems can predict users' preferences more accurately and provide more customized recommendations. Besides the cross domain knowledge, another major source of information that has yet to be fully utilized is that of social network data. For example, users interests may be affected by their friends.

Let us consider Table 1.4 and Table 1.5 which show sample data from the auxiliary

Table 1.4: Ternary relations among users, tags, and items in Movies Domain

User	Tag	Item
$U'_1$	fantasy	Twilight
$U'_1$	romance	Twilight
$U'_1$	drama	Big Daddy
$U'_2$	fantasy	Spider man
$U'_2$	adventure	Spider man
$U'_2$	action	Iron Man
$U'_3$	drama	Big Daddy
$U'_3$	comedy	Little man
$U'_4$	action	Iron Man
$U'_4$	action	Star war
$U'_5$	adventure	Die hard
$U'_5$	adventure	Braveheart

Table 1.5: Social Trust in Books Domain

User	User
$U_3$	$U_2$
$U_5$	$U_7$
$U_8$	$U_9$

Movie domains and social network respectively. Suppose we want to recommend some book to user  $U_5$  in Table 1.2. Unfortunately, we cannot find similar users in the Book domain to base the recommendation on since  $U_5$  is the only user who uses the tag fantasy. However, we can utilize the denser Movie domain dataset to learn the characteristics of users and make suitable recommendations to  $U_5$ .

For example, Table 1.4 show the ternary relationship in the Movie domain. Based on the relationship, we see that  $U_5$  is similar to  $U'_1$  and  $U'_2$  because they all like fantasy items. Further, we observe that the book ‘New moon’, read by  $U_5$ , has been tagged as fantasy and romance. Between users  $U'_1$  and  $U'_2$ , we observe that  $U'_1$  watches fantasy, romance and comedy type of movies, while  $U'_2$  watches fantasy, adventure and action type of movies. Thus, we conclude that  $U_5$  is more similar to  $U'_1$  than  $U'_2$ . In addition, from the Movie domain, we realize that users who like fantasy and romance type of movies also like comedy movies. Thus, we should recommend comedy books “Good omens” to  $U_5$ . This is further strengthened by the friend relationship in Table 1.5, As we



know from some social network website that  $U_5$  is a friend of  $U_7$ , we may infer that  $U_5$  is influenced by  $U_7$ . As such, we will recommend the same book “Good omens” to  $U_5$  which have been tagged by  $U_7$  before.

### 1.3 Improving users’ acceptance using Social Trust Data

With the advent of online social networks, social trust based CF approaches to recommendation have emerged [28, 69, 47]. The assumption is that friends tend to influence their friends to exhibit similar likes and dislikes. Hence, we can also increase user acceptance in recommender systems by taking into account the social relationships.

Table 1.6: Example of Table 1.2 over Time

(a) Ternary relations among user, rating and item over Time in Book Domain

User	Rating	Item	Time
$U_1$	like	Forrest Gump	$T_1$
$U_1$	like	Beautiful Mind	$T_1$
$U_2$	like	Forrest Gump	$T_1$
$U_2$	like	Groundhog Day	$T_1$
$U_2$	like	Groundhog Day	$T_1$
$U_3$	like	Forrest Gump	$T_1$
$U_3$	like	Toy Story	$T_2$
$U_4$	dislike	Forrest Gump	$T_1$
$U_4$	dislike	Toy Story	$T_1$
$U_5$	like	New moon	$T_1$
$U_6$	like	New moon	$T_1$
$U_7$	like	Good omens	$T_1$
$U_8$	like	James Bonds Girls	$T_1$
$U_9$	like	Ghost rider	$T_1$
$U_9$	like	James Bonds Girls	$T_1$
$U_9$	like	Scorpia	$T_1$
$U_{10}$	like	Toy Story	$T_2$
$U_{10}$	like	Shrek	$T_2$

(b) Social Trust Over Time

User	User	Time
$U_3$	$U_2$	$T_1$
$U_5$	$U_7$	$T_1$
$U_8$	$U_9$	$T_1$
$U_3$	$U_{10}$	$T_2$
$U_{10}$	$U_3$	$T_2$

Let us consider the snapshots of users’ item ratings of Table 1.1 at time points  $T_1$  and  $T_2$  in Table 1.6(a). Besides that, we also have additional social relationship at time points  $T_1$  and  $T_2$  in Table 1.6(b). Suppose our target user is  $U_3$ . At time point  $T_1$ , both users

$U_1$  and  $U_2$  have watched and rated the Book “*Forrest Gump*”. Traditional CF methods [63, 66, 57] will group  $U_1$ ,  $U_2$  and  $U_3$  as similar users and recommend “*Beautiful Mind*” and “*Groundhog Day*” to  $U_3$  since  $U_1/U_2$  has watched these books previously. Yet,  $U_3$ ’s interest does not remain static. We observe that at time point  $T_2$ , his interest has shifted from comedy book to animation book as he rates a new item “*Toy Story*”. Recognizing this, CF with temporal dynamics will recommend another animation book “*Shrek*” to  $U_2$  instead. On the other hand, looking at the social relationships among users, we realize that  $U_1$  and  $U_3$  are friends. Hence, social network based CF will conclude that  $U_3$  probably like “*Groundhog Day*” since his friend  $U_2$  has read and rated this book. Each of the different methods arrive at different items to recommend. How do we reconcile the different recommendations? To complicate matter, social relationships are not static but evolve over time as a user can make new friends and old friends do grow apart. We observe that at time point  $T_1$ ,  $U_3$  has only one friend  $U_2$ , whereas at time point  $T_2$ , his friends are  $\{U_2, U_{10}\}$ . Now if we want to give a recommendation to  $U_3$  at time point  $T_2$ , what item should we recommend so that it is most likely to be accepted by  $U_3$ ?

To answer this question, we must be able to quantify the degree of influence on a user’s decision making process from his/her long term and short term interests, as well as his/her social trust relationships over time. Note that these two factors are not independent. We advocate that when two users’ long term and short term interests are aligned, they are likely to become friends, and they will tend to be more receptive towards each other’s preferences. Conversely, if the users’ interests are not aligned, they will grow apart after some time and become less receptive towards the preferences of the other user. Clearly, there is a need to quantify the dynamic interaction between user interest and social trust so as to develop a more accurate recommender system.

## 1.4 Contributions

The contributions of this thesis are stated as follows:

This thesis examines three ways to improve users' acceptance towards recommendation. First, *Quaternary Semantic Analysis* (QSA) algorithm that utilizes social rating and tagging data is proposed. Second, FUSE algorithm is proposed to allow knowledge transfer from other domain to the target domain. Third, *Receptiveness over Time Model* (RTM) algorithm is proposed by modeling the interaction between users' interest and social relation. The major contributions are summarized as follows.

- We show that ternary relationships are insufficient to provide accurate recommendations which may decrease users' acceptance. Instead, we model the quaternary relationship among users, items, tags and ratings as a 4-order tensor and cast the recommendation problem as a multi-way latent semantic analysis problem [81, 84]. A unified framework *Quaternary semantic analysis*(QSA) for user recommendation, item recommendation, tag recommendation and item rating prediction is proposed. The results of extensive experiments performed on a real world dataset demonstrate that our unified framework outperforms the state-of-the-art techniques in all the four recommendation tasks.
- We show that cross domain data can be transferred without decomposition may decrease user' acceptance towards recommendations and propose a generalized cross domain collaborative filtering framework FUSE that integrates social network information seamlessly with cross domain data [82]. We find shared implicit cluster-level tensor from multiple domains and perform tensor factorization with topic based social regularization. Extensive experiments conducted on real world datasets indicate that the proposed framework outperforms state-of-art algorithms for item recommendation, user recommendation and tag recommendation.
- We show that the complex interaction between user interests and the social relationship over time is important to increase the user's acceptance toward recommendation [83]. We propose a probabilistic generative model, called *Receptiveness over Time Model* (RTM), to capture this interaction. We design a Gibbs sam-

pling algorithm to learn the receptiveness and interest distributions among users over time. Experimental results on a real world dataset demonstrate that RTM-based recommendation outperforms the state-of-the-art recommendation methods. Case studies also show that RTM is able to discover the user interest shift and receptiveness change over time.

## **1.5 Organization**

The rest of this thesis is organized as follows. Chapter 2 includes a literature review covering some existing recommendation algorithms on different types of data. Their strengths and weaknesses are discussed. Based on the literature review, we present the unified framework for social tagging and rating data in detail in Chapter 3. In Chapter 4, we develop a cross domain framework that is applicable in transferring knowledge from different domain. Further in Chapter 5, we describe methods for improving users' acceptance by modeling the social trust over the time. Finally, Chapter 6 concludes the thesis and provides future work.



---

---

## CHAPTER 2

---

### LITERATURE REVIEW

#### 2.1 Recommender System

Recommender system help user to choose items by predicting user's interest on an item based on various sorts of information including item, user information and interactions between users and items. Resnick and Varian [62] describe a recommender system as a system which can acquire users' opinions about different items and also use these opinions to direct users to those items that might be interesting to them. Herlocker [22] says that a recommender system is one that predicts what items a user might find interesting or suitable to his/her needs. Burke [13] put forward his definition that a recommender system is any system that can produce individualized recommendations and have the ability to guide users in a personalized manner to find interesting information items in a large space of possible options.

## 2.2 Techniques of Recommender System

Broadly speaking, recommender systems can be classified into two types: (1) Content based [5, 51, 50, 55, 49, 56, 6, 38] (2) Collaborative Filtering [66, 61, 12, 77, 29, 73, 79, 88, 89, 35, 26, 25, 10, 72, 54, 33, 34, 37, 64, 87].

### 2.2.1 Content Filtering

The content filtering approach [5, 51] creates a profile for each user or item by building a vector space whereby both the items and users are represented as points in this space. Given a target user, we obtain the set of the most relevant items for the target user by comparing the distances between the items and the user profiles and retrieving the items' points in the space that are nearest to user profile.

More formally, assuming there is a set of attributes (keywords)  $\{a_1 \cdots a_k\}$  characterizing item  $i$ . The attributes are usually computed by extracting a set of features from item  $i$  (its content) which is useful for recommendation purposes. Let  $Content(i)$  denotes the profile of item  $i$ , we have

$$Content(i) = \{w_1, w_2 \cdots w_k\} \quad (2.1)$$

where  $w_k$  is the weight of  $k$  th attribute of item  $i$ , this weight can be attained based on the calculation of TF-IDF [65].

Similarly, Let  $User(u)$  be the vector of weights built for user  $u$  as follows:

$$User(u) = \frac{1}{|Item_{like}|} \sum_{i \in Item_{like}} Content(i) \quad (2.2)$$

where  $Item_{like}$  define the sets of items that users  $u$  has previously purchased. Given the user  $u$  profile vector and an item  $l$ , we calculate the user  $u$ 's preference towards to item  $l$  (similarity between user  $u$  and item  $l$ ) as follows:

$$p(u, i) = \text{sim}(\text{User}(u), \text{Content}(i)) \quad (2.3)$$

where  $\text{sim}(\cdot, \cdot)$  denotes the cosine similarity between two vectors. Consequently, a recommender system will determine the appropriateness of recommendation by the similarity.

Instead of TF-IDF to obtain the weight of attributes for items/users, Pazzani et al, [55] try to learn “importance” of attribute from the underlying data using statistical learning which is Bayesian classifier. Similarly, Mooney et al, [49] have applied text categorization to extracted users/items attributes and their weights. This is done by simple Bayesian text-categorization algorithm extended to efficiently handle set-valued features. Balabanovic et al, [5] represents users/items with the 100 most important attributes instead of all attributes. Pazzani et al, [56] design machine learning approach for learning a linear classifier which try to represent each user as a vector of weighted words derived from positive training examples using the Winnow algorithm. Besides bayesian and machine learning approach, Basu et al, [6] use rule induction to represent the relation between user and items. They design *Ripper* to learn a function (sets of rules) that takes a user and item as input and predicts whether the movie will be liked or disliked. In order to incorporate other information such as rating information, Lee [38] treats the recommending task as the learning of a user’s preference function that exploits item content as well as the ratings of similar users. They perform a study of several mixture models for this task.

In terms of scalability, Berry et al, [7] pointed out the need to introduce some dimensionality reduction technique such as latent semantic analysis [17] and probabilistic latent semantic indexing [25] for the vector space model. Recently, Wang et al, [80] try to further extent the latent semantic indexing in the large-scale data.

One of the disadvantage of content-based techniques is that it is limited by the features that are explicitly associated with the items that these systems recommend. Therefore, in order to have a sufficient set of features, the content must either be in a form that



can be parsed automatically by a computer (e.g., text), or the features should be assigned to items manually. While information retrieval techniques work well in extracting features from text documents, some other domains have an inherent problem with automatic feature extraction. For example, automatic feature extraction methods are much harder to apply to the multimedia data, e.g., graphical images, audio and video streams. Moreover, it is often not practical to assign attributes by hand due to limitations of resources [2].

Another disadvantage is that, if two different items are represented by the same set of features, they are indistinguishable. Therefore, since text-based documents are usually represented by their most important keywords, content-based systems cannot distinguish between a well-written article and a badly written one, if they happen to use the same terms [2].

Besides content filtering, collaborative filtering (CF) is another important class of recommender system techniques. The major difference between collaborative filtering and content-based recommender systems is that collaborative filtering only uses the user-item ratings data to make predictions and recommendations, while content-based recommender systems rely on the features of users and items for predictions. Both content-based recommender systems and CF systems have limitations. While CF systems do not explicitly incorporate feature information, content-based systems do not necessarily incorporate the information in preference similarity across individuals.

## **2.2.2 Collaborative Filtering**

Collaborative filtering (CF) in recommender systems can be roughly divided into two major categories. Memory-based methods aim at finding like-minded users to predict the active user's preference [66, 61, 12, 77, 29, 73, 79, 88, 89]. Model-based methods [35, 26, 25, 10, 72, 54, 33, 34, 37, 64, 87] model the user-item-rating or user-item-tagging interaction based on the observed rating or tagging.

## Memory-based

**User based** GroupLens published the first paper [61] in collaborative filtering, which is also called user-based collaborative filtering. However, user-based CF fails when the databases are large and sparse. In 2000, Amazon proposed the item-based collaborative filtering [66], which is more scalable compared to the user-based CF. User-based CF believes that target user will have similar preference to users with similar interests. Cosine similarity and Pearson correlation [66] are two typical measures to evaluate the similarity of interests. Two users are represented as two vectors in the  $m$  dimensional item-space, where  $m$  is the total number of items in the data. The cosine similarity between user  $i$  and  $j$  is defined:

$$sim(i, j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| * \|\vec{j}\|} \quad (2.4)$$

where  $\cdot$  denotes the dot-product of the two vectors.

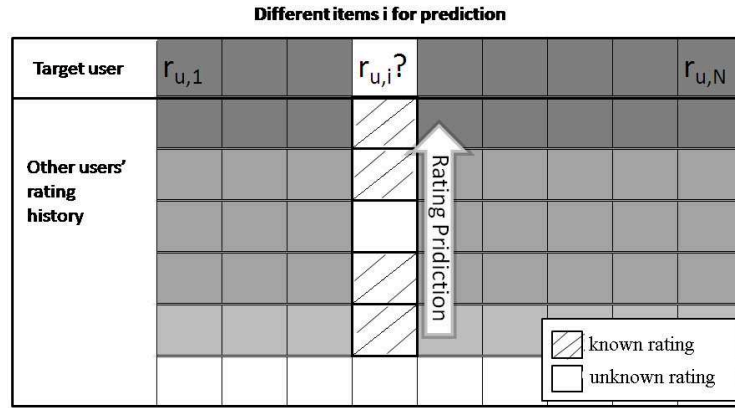


Figure 2-1: User-based CF

As illustrated in Figure 2-1, target user  $u$ 's rating on item  $i$  depends on other similar user rating on item  $i$ . Ratings by users who are more similar are weighted more and contribute more towards the prediction of the item rating. The set of similar users can be identified by employing a threshold or selecting the top-N. The most similar users  $N_u(u_k)$  is defined as follows:

$$N_u = \{u_k | \text{rank sim}(u, u_k) \leq N, R_{u_k,i} \neq \emptyset\} \quad (2.5)$$

where  $R_{u_k,i}$  is the rating of user  $u_k$  on item  $i$ .

Consequently, the predicted rating  $\hat{R}_{u,i}$  of test item  $i$  by test user  $u$  is computed as [66, 61]:

$$\hat{R}_{u,i} = \bar{u} + \frac{\sum_{u_k \in N_u} \text{sim}(u, u_k)(R_{u_k,i} - \bar{u}_k)}{\sum_{u_k \in N_u} \text{sim}(u, u_k)} \quad (2.6)$$

where  $\bar{u}$  and  $\bar{u}_k$  denote the average rating made by user  $u$  and  $u_k$ , respectively. Existing methods differ in their treatment of unknown ratings from similar users ( $R_{u,i} = \emptyset$ ).

**Item based** CF algorithms [61, 66] use similarity between items instead of users to predicted the rating of the items. The assumption is that people who agreed in the past tend to agree again in the future. Users who usually give similar ratings to the same items are considered to be similar. Two items are represented as two vectors in the  $m$  dimensional user-space, where  $m$  is the total user in the data. In cosine similarity, the similarity between item  $i$  and  $j$  is defined:

$$\text{sim}(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| * \|\vec{j}\|} \quad (2.7)$$

where  $\cdot$  denotes the dot-product of the two vectors.

The prediction (preference) of user  $u$  given to item  $i$  can be obtained by computing the sum of the ratings given by the user on the items similar to  $i$ . Each ratings is weighted by the corresponding similarity  $\text{sim}(i, j)$  between items  $i$  and  $j$ .

$$P_{u,i} = \frac{\sum_{\text{all\_similar\_items},j} (\text{sim}(i, j) * R_{u,j})}{\sum_{\text{all\_similar\_items},j} |\text{sim}(i, j)|} \quad (2.8)$$

The weighted sum is one of the representation of calculated the prediction, there can be other approaches such as regression [66]

A number of extensions on memory-based collaborative filtering have been pro-

posed, Breese et al, [12] design several similarity measurements, including techniques based on correlation coefficients, vector-based similarity calculations, and statistical Bayesian methods. In order to address data sparsity problem, Ungar et al, [77] group users into clusters based on the items they have purchased and making recommendations at the cluster level rather than individual level. Taking into account the impact of rating discrepancies among different users, Jin et al, [29] propose an optimization algorithm to automatically compute the weights for different items based on the clustered distribution of user vectors in item space. For example, an item that is highly favored by most users should have a smaller impact on the user-similarity than an item for which different types of users tend to give different ratings. Su et al, [73] extend Bayesian belief nets (BNs) to handle multi-class data and apply it on memory based collaborative filtering tasks. Wang et al, [79] unify the user based CF and item based CF in a generative probabilistic framework. Recently, there are many other researchers looked into the incorporation of the tagging data to improve memory based collaborative filtering [88, 89].

## **Model-based**

Latent factor models are an alternative approach that tries to explain the ratings by characterizing both items and users on, say, 20 to 100 factors inferred from the ratings patterns. Figure 2-2 illustrates this idea for a simplified example in two dimensions [35]. Consider two hypothetical dimensions characterized as female- versus male-oriented and serious versus escapist. For this model, a user's predicted rating for a movie, relative to the movie's average rating, is equal to the dot product of the movie's and user's locations on the graph. For example, we expect Gus to like "Dumb and Dumber", hate "The Color Purple", and do not mind "Lethal Weapon".

Model-based CF method utilizes singular value decomposition and its variants. Recently, a number of research have investigated the use of Latent Semantic Analysis (LSA) [26], probabilistic LSA [25], latent Dirichlet allocation (LDA) [10]. Latent Se-

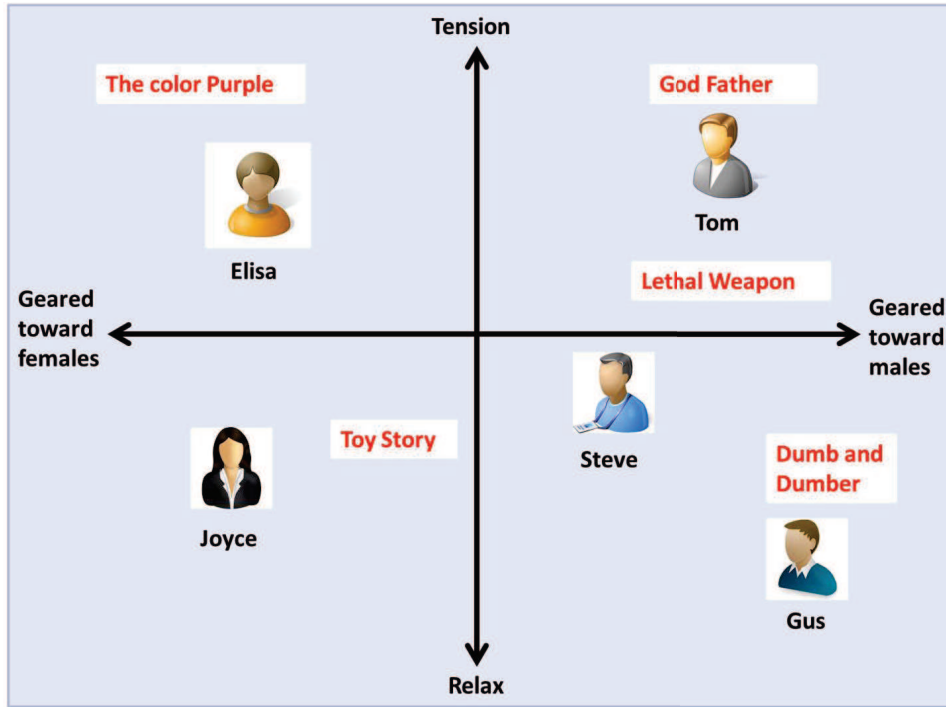


Figure 2-2: Latent factor model illustration

mantic Analysis (LSA) [26] is first proposed to use in the language and information retrieval communities and later applied in recommender system. Based on the LSA, probabilistic LSA [25] was proposed to provide the probabilistic modeling, and further latent Dirichlet allocation (LDA) [10] provides a Bayesian treatment of the generative process.

Along another direction, several attempts have been made to improve the recommendation accuracy based on the matrix factorization model. Specifically, matrix factorization methods usually seek to associate both users and items with latent profiles represented by vectors in a low dimension space that can capture their characteristics. Low-rank matrix factorization algorithms for collaborative filtering can be roughly grouped into non-probabilistic and probabilistic (non-negative) approaches.

For non-probabilistic approach, [72] approach uses margin based loss functions such as the hinge loss used in SVM classification, and its ordinal extensions for handling multiple ordered rating categories. For ratings that span over  $K$  values, this reduces to finding  $K - 1$  thresholds that divide the real line into consecutive intervals specifying

rating bins to which the output is mapped, with a penalty for insufficient margin of separation. Rennie and Srebro [72] suggest a non-linear Conjugate Gradient algorithm to minimize a smoothed version of this objective function. Fueled by the Netflix competition, several improvements have been proposed including the use of regularized SVD [54], and the idea of matrix factorization combined with neighborhood based methods [33]. Koren [34] extend his work in [33] to incorporate time information and name it as timeSVD++. The timeSVD++ method assumes that the latent features consist of some components that are evolving over time and some others that are dedicated bias for each user at each specific time point. This model can effectively capture local changes of user preference which the authors claim to be vital for improving the performance.

Another class of techniques is the non-negative matrix factorization popularized by the work of Lee and Seung [37] where non-negativity constraints are imposed on user/item latent profile. NMF is in fact essentially equivalent to Probabilistic Latent Semantic Analysis (pLSA) [25] which has also previously been used for Collaborative Filtering tasks. Different from [72] which is non-probabilistic framework, Ruslan et al, [63] present probabilistic algorithms that scale linearly with the number of observations and perform well on very sparse and imbalanced datasets. Bayesian PMF (BPMF) [64] provides a Bayesian treatment for PMF to achieve automatic model complexity control. It demonstrates the effectiveness and efficiency of Bayesian methods and MCMC in real-world large-scale data mining tasks. Yu et al, [87] develop nonparametric matrix factorization methods by allowing the latent factors of two low-rank matrix factorization methods, the singular value decomposition (SVD) and probabilistic principal component analysis (pPCA) [75], to be data-driven, with the dimensionality increasing with data size.

### **2.2.3 Measurement of Users' Acceptance**

The measurement of users' acceptance determines on the quality of a recommendation system. According to Herlocker [24], metrics evaluating recommendation systems can

be broadly classified into the following broad categories: predictive accuracy metrics, such as Mean Absolute Error (MAE) and its variants; classification accuracy metrics, such as precision, recall, F1-measure, and ROC sensitivity and other metrics such as transparency [9], [22], trustworthiness [18], scalability [3], [21], [66], [67], or privacy [58], [67]. In this thesis, our focus is on predictive and classification accuracy.

Predictive accuracy metrics mainly compare the estimated ratings against the actual ratings e.g. Mean Absolute Error (MAE), root mean squared error (RMSE).

**Mean Absolute Error** (often referred to as MAE) measures the average absolute deviation between a predicted rating and the user's true rating. Mean absolute error (Eq. 2.9) has been used to evaluate recommender systems in several cases [66, 63, 4, 44]. The MAE is given by:

$$MAE = \frac{\sum_{r_{u,i} \in D} |r_{u,i} - \hat{r}_{u,i}|}{|D|} \quad (2.9)$$

where  $r_{u,i}$  is the rating given by user  $u$  for item  $i$ ,  $\hat{r}_{u,i}$  is the predicted rating and  $D$  is the size of the testing dataset.

**Root Mean Squared Error** (often referred to as RMSE) are variant of MAE by squaring the error before summing it. The RMSE is given by:

$$RMSE = \sqrt{\frac{\sum_{r_{u,i} \in D} (r_{u,i} - \hat{r}_{u,i})^2}{|D|}}$$

where  $r_{u,i}$  is the rating given by user  $u$  for item  $i$ ,  $\hat{r}_{u,i}$  is the predicted rating and  $D$  is the size of the testing dataset.

In many applications, the task is to recommend to users items that they may adopt. In this case we are interested in the classification accuracy of the recommendation.

Common classification accuracy metrics include precision [23], recall [23], F-measure [23], and Receiver Operating Characteristic (ROC) [23]. A recommendation is true positive (TP) if an item recommended has been adopted by the user. It is true negative (TN), if an item that has not recommended is not adopted by the user. It is false negative (FN),

if an item that has not recommended the user is adopted by the user. It is false positive (FP), if an item that has recommended is not adopted by the user.

Based on this, we have:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F = \frac{2 * Precision * Recall}{Precision + Recall}$$

Receiver Operating Characteristic (ROC) is a graphical technique that uses two metrics, true positive rate (TPR) and false positive rate (FPR) where:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

The curve is obtained by plotting TPR against FPR as we vary the number of item recommended to the user.

## 2.3 Recommender System using Rating and Tagging Data

Collaborative tagging systems, also known as folksonomies are web-based systems that allow users to upload their resources, and to label them with arbitrary words, so-called tags. These systems are becoming more common among web users. For example popular web services such as Flickr<sup>1</sup>, del.icio.us<sup>2</sup>, Last.fm<sup>3</sup> etc, allow users to tag or label an

---

<sup>1</sup>[www.flickr.com](http://www.flickr.com)

<sup>2</sup>[delicious.com](http://delicious.com)

<sup>3</sup>[www.last.fm](http://www.last.fm)



item of interest as shown in Figure 2-3.



Figure 2-3: Tags in Flickr

Bogers [11] has attempted to extend existing CF algorithms to tag-based collaborative filtering where the user and item similarities are computed based on their overlaps in tagging behavior. For instance, users who have many of the same tags and thus have more tag overlap between them, can be seen as rather similar. Items that are often assigned the same tags are also more likely to be similar than items that share no tag overlap at all.

For Tag-based CF using user similarity, they calculate tag overlap on the User-Tag matrix or on the binarized User-Tag matrix, depending on the metric. The user similarity in equation 2.4 is changed to Jaccard overlap  $sim_{jaccard}(i, j)$  between user  $i$  and user  $j$ . Let two users be represented as two vectors in the  $t$  dimensional tag-space, where  $t$  is the total number of items in the data, the similarity between user  $i$  and user  $j$  is defined as

$$sim_{jaccard}(i, j) = \frac{|\vec{i} \cap \vec{j}|}{|\vec{i} \cup \vec{j}|} \quad (2.10)$$

where  $\vec{i}$  and  $\vec{j}$  are user and item vector respectively.

Likewise, for Tag-based CF using item similarity, they calculate tag overlap on the item-tag matrix or on the binarized item-Tag matrix, and Jacard overlap between items is used for item similarity. However if we only applied the standard memory-based CF algorithms to the data sets, we would be neglecting the extra layer of information formed by the tags. In other words, we will lose the tagging information which not only tells what a user likes, but also why he or she likes it.

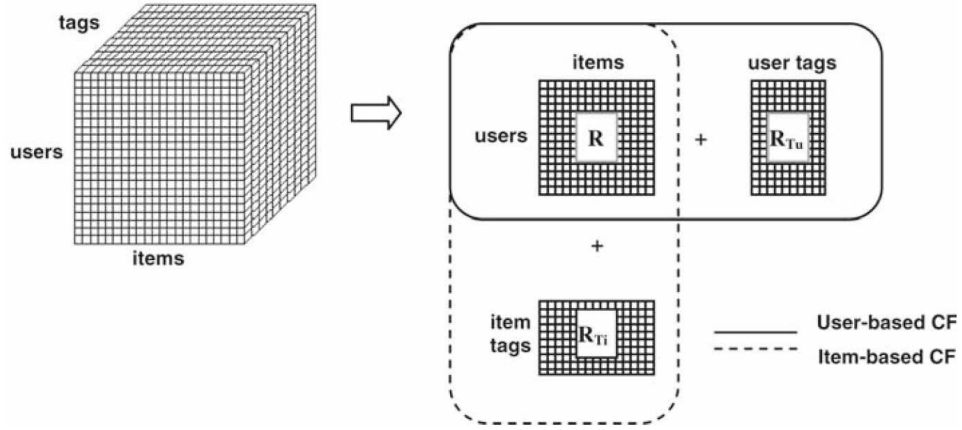


Figure 2-4: Extend user item matrix by including user tags as items and item tags as users (Tso-Sutter et al. 2008)

To address the problem, Tso-Sutter et al. [76] propose a generic method that allows tags to be incorporated into standard CF algorithms, by decomposing the three-dimensional  $\langle \text{user-item-tag} \rangle$  correlations into three two-dimensional correlations, which is  $\langle \text{user, tag} \rangle$  and  $\langle \text{item, tag} \rangle$  and  $\langle \text{user, item} \rangle$  as shown in Figure 2-4.

However, decomposing the three dimensions all together without reducing them into lower dimensions result in information loss. Symeonidis et al. (2008) [74] and Rendle et al. (2009) [59] proposed tensor factorization based approach for folksonomy data structure. By representing user-item-tag as a 3-order tensor  $\mathcal{A}$ , one is able to exploit the underlying latent semantic structure and obtain the multi-way correlations between users, tags and items (See Figure 2-5).

The factorization of  $\mathcal{A}$  is expressed in Equation 2.11.  $U^{(i)}$  are orthonormal matrices corresponding to the dominant singular vectors at  $i$ -mode.  $\mathcal{S}$  is the core tensor that contains the singular values, thus it has the same size as  $\mathcal{A}$  and the property of all or-

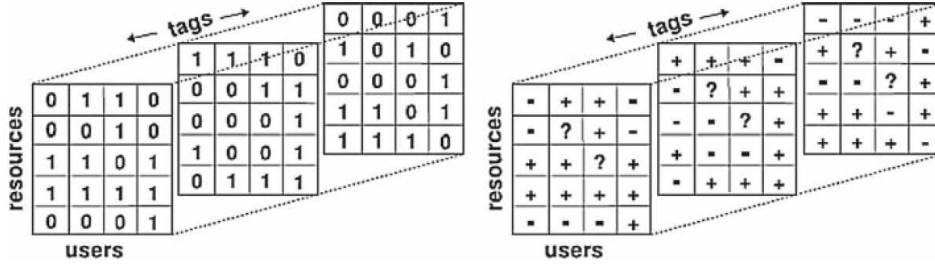


Figure 2-5: Tensor representation left (Symeonidis et al. 2008), right (Rendle et al. 2009)

thogonality. The symbol  $\times_i$  denotes the  $i$ -mode multiplication between a tensor and a matrix.

$$\mathcal{A} = \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \quad (2.11)$$

After decomposing  $\mathcal{A}$ , the matrices  $\mathbf{U}^{(1)}$ ,  $\mathbf{U}^{(2)}$ ,  $\mathbf{U}^{(3)}$  and the core tensor  $\mathcal{S}$  are truncated by maintaining only the highest  $D$  singular values and the corresponding singular vectors per mode (henceforth,  $D$  denotes the fraction, e.g., 0.7, of the maintained values divided by the original number of values). This produces the truncated matrices  $\hat{\mathbf{U}}^{(1)} \in \mathbb{R}^{|User| \times D_1}$ ,  $\hat{\mathbf{U}}^{(2)} \in \mathbb{R}^{|Item| \times D_2}$ ,  $\hat{\mathbf{U}}^{(3)} \in \mathbb{R}^{|Tag| \times D_3}$ . and the truncated core tensor  $\hat{\mathcal{S}} \in \mathbb{R}^{D_1 \times D_2 \times D_3}$ . Using truncation we can approximate with the reconstructed tensor  $\hat{\mathcal{A}}$  as expressed in Eq. 2.12 and illustrated in Figure 2-6.

$$\hat{\mathcal{A}} = \mathcal{S} \times_1 \hat{\mathbf{U}}^{(1)} \times_2 \hat{\mathbf{U}}^{(2)} \times_3 \hat{\mathbf{U}}^{(3)} \quad (2.12)$$

Once is computed, the list with the  $N$  highest scoring tags for a given user  $u$  and a given item  $i$  can be calculated by:

$$Top(u, i, N) = \underset{t \in T}{\operatorname{argmax}}^N \hat{\mathcal{A}}_{u,i,t} \quad (2.13)$$

Recommending  $N$  resources to a given user  $u$  for a particular tag  $t$  can be done in a similar manner. Moreover, other users can be recommended to a particular user  $u$  given a specific tag  $t$ , according to the total score that results by aggregating all resources that

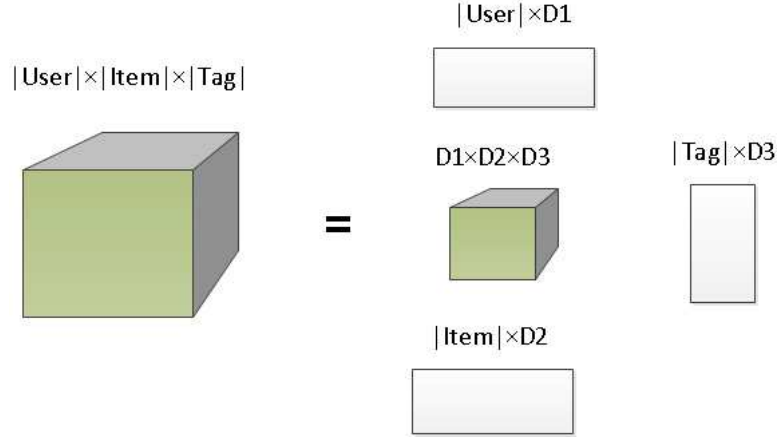


Figure 2-6: Tensor Factorization

are tagged with  $t$  by  $u$ .

Different from Symeonidis et al., Rendle et al. (2009) distinguish between positive and negative examples and missing values in order to learn personalized ranking of tags. The idea is that positive and negative examples are only generated from observed tag assignments. Observed tag assignments are interpreted as positive feedback, whereas the non-observed tag assignments of an already tagged resource are negative evidences. All other entries, i.e., all tags for a resource that a user has not tagged yet, are assumed to be missing values (Figure 2-5).

## 2.4 Recommender System using Cross Domain Data

In real-world recommender systems, users can rate only a limited number of items, so the rating matrix is always extremely sparse. The available rating data that can be used for k-NN search, probabilistic modeling, or matrix factorization are clearly insufficient. The sparsity problem has become a major bottleneck for most collaborative filtering methods. Cross-domain collaborative filtering is an emerging research topic in recommender systems. It aims to alleviate the sparsity problem in individual CF domains by transferring knowledge among related domains. For example, users who like to read *romance* books generally have similar preferences as users who like to watch *romance* movies as

shown in Figure 2-7. By learning the characteristics of *romance* lovers from the Movie domain and transferring the learned characteristics to the Book domain, recommender systems can predict users' preferences more accurately and provide more customized recommendations.

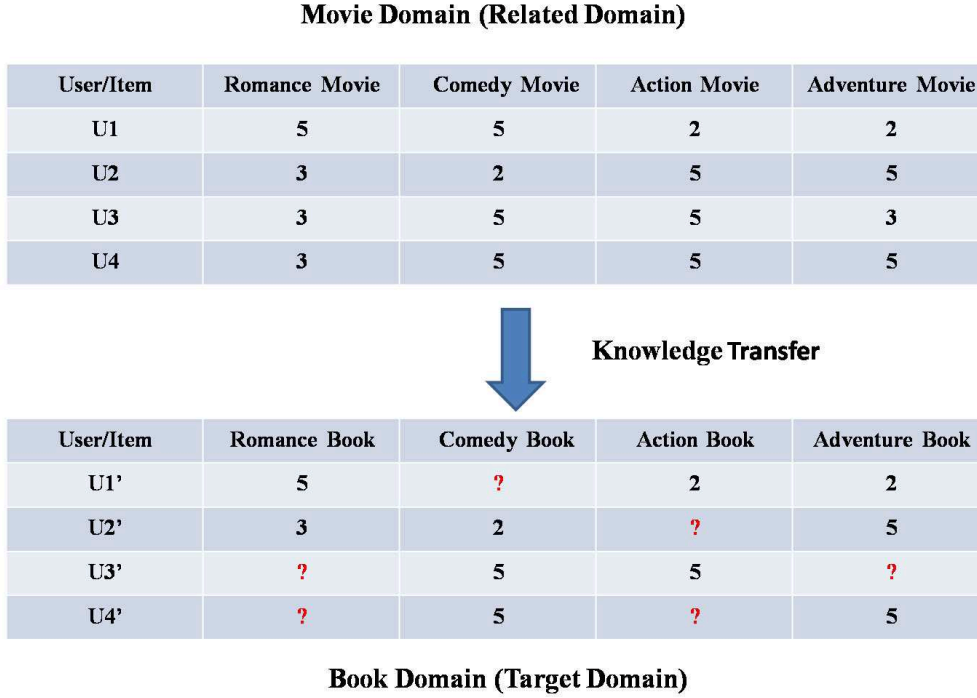


Figure 2-7: The correspondence of transfer from Movie Domain to Book Domain

Cross domain collaborative filtering methods can be categorized into (a) latent-feature sharing [71] [14][45][78] (b) binary relationships knowledge transfer [52][40]; and (c) ternary relationship knowledge transfer with decomposition [70].

### 2.4.1 Latent feature shares

A common cross-domain CF scenario is that the data in one domain (e.g., a new book website) are very sparse while the data in some related domain are abundant (e.g., a popular movie website). In such cases, knowledge can be transferred over related system domains to the domain where data is sparse and help to improve the recommendation accuracy. A system domain is further decomposed into two sub-domains: user domain and item domain. For the item domain knowledge transfer, [71] aimed at making use of

relation in the item domain such as movie and genres, and actors and movie etc. These multiple relations in item domain are represented as multiple matrices, they try to improve predictive accuracy by exploiting information from one relation while predicting another. To this end, they propose a collective matrix factorization model.

For the user domain knowledge transfer, [14] jointly considering multiple heterogeneous link prediction tasks such as predicting links between users and different types of items including books, movies and songs. A nonparametric Bayesian framework is proposed for solving the collective link prediction problem, which allows knowledge to be adaptively transferred across heterogeneous tasks while taking into account the similarities between tasks. Ma et al, [45] considering the connections among users which is trust relation. They propose framework to incorporate the social trust as restrictions on the recommender system. Recently, Vasuki et al, [78] consider recommendation problem given the the current state of the friendship and affiliation networks. these two networks are used as user domain knowledge transfer. In particular, they design two models of user-community affinity for the purpose of making recommendations: one based on graph proximity, and another using latent factors to model users and communities.

## **2.4.2 Binary Knowledge Transfer using Cross Domain Data**

For binary relationships knowledge transfer, Li et al [39] design Rating-pattern sharing which is also called CodeBook Transfer (CBT) for solving adaptive transfer learning (domain adaptation) problems in CF. Then the idea was incorporated into a probabilistic model, Rating-Matrix Generative Model (RMGM)[40], for solving collective transfer learning (multi-task learning) problems in CF.[52] introduces a coordinate system transfer over multiple domains and transfer framework consisting of multiple data domains. These approaches share user/item latent feature spaces across CF domains and knowledge can be transferred through the shared latent features.

### **2.4.3 Ternary Knowledge Transfer using Cross Domain Data**

With the rapid development of Web 2.0, Tagging has become a ubiquitous function in most of today's recommender systems. Social tags have also been used to link domains since they can be used as an agreed vocabulary to describe items from any domain in a simple, generic way. Y.Shi [70] exploited tags to improve recommendation by proposing a matrix factorization based method use tags as bridge for cross domain transfer, by reducing the ternary relation to two 2D correlations and use these for regularization.

In particular, they utilize tags to build user-user and item-item similarity matrices. The similarity between two users/items from different domains is proportional to the number of tags shared by their annotation profiles. Computed similarities are incorporated as constraints into a probabilistic model based on matrix factorization and collaborative filtering.

## **2.5 Recommender System using Social Trust Data**

In the past few years, the dramatic expanding of Web 2.0 Web sites and applications pose new challenges for traditional recommender systems. Traditional recommender systems always ignore social relationships among users by utilizing users' feedback data such as rating data as shown in Figure 2-8(a). The Facebook and Twitter, Research have tried to make recommendation based on social relation as shown in Figure 2-8(b) and 2-8(c). They believe that users' interest and item selection are often influenced by their friends. In order to improve recommender systems and to provide more personalized recommendation results, it is necessary to incorporate social network information among users in recommender system.

Figure 2-9 shows how Amaazon make recommendation by using the social trust in Facebook. The list of friends who also like the recommendation is listed at the bottom of each recommendation. Generally, trust-based CF can be categorized into neighborhood-based [20, 48, 27] and model based method [44, 28, 69, 86] .

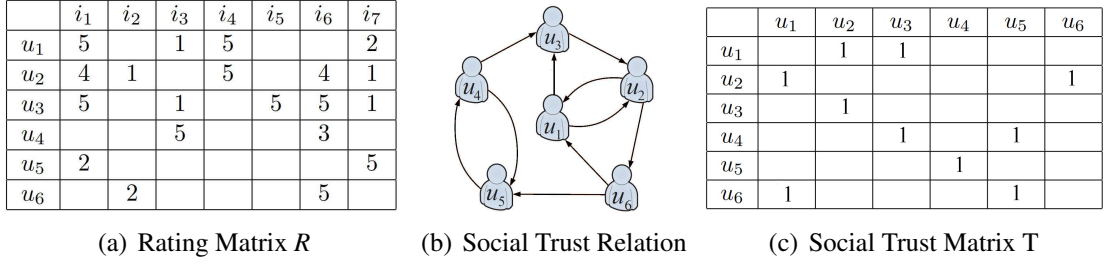


Figure 2-8: User Feedback, Social Relation and its Matrix representation



Figure 2-9: Recommendation based on Social Trust Data

### 2.5.1 Neighborhood-Based Model using Social Trust Data

Given user  $u$ , let  $F(u)$  denote the friend of user  $u$ , and  $N(u)$  denote the set of items user  $u$  likes. The preference of user  $u$  on item  $i$  can be defined as number of user  $u$ ' friends who like item  $i$  :

$$p_{ui} = \sum_{f \in F(u), i \in N(f)} 1$$

This simple model only considers direct trust. It does not consider the indirect trust (Friend-of-a-Friend). Some memory based approaches have been proposed for recommendation in social rating networks [20, 48, 27].

Golbeck [20] analyzed some of the properties of trust in social networks to design a trust propagation algorithm that took the indirect trust into account and propose TidalTrust. TidalTrust performs a modified breadth first search in the trust network to



compute a prediction of item's rating. Basically, it finds all raters with the shortest path distance from the source user and aggregates their ratings weighted by the trust between the source user and these raters. To compute the trust value  $T_{u,v}$  between user  $u$  and  $v$  who are not directly connected, TidalTrust aggregates the trust value between  $u$ 's direct neighbors and  $v$  weighted by the direct trust values of  $u$  and its direct neighbors, that is

$$T_{u,v} = \frac{\sum_{w \in F(u)} T_{u,w} T_{w,v}}{\sum_{w \in F(u)} T_{u,w}}$$

Once the raters  $T$  have been selected (e.g.  $T_{u,v}$  must be larger than some threshold), the rating prediction of user  $u$  on item  $i$  is calculated as the weighted average of all raters  $T$ ' ratings:

$$\hat{R}_{u,i} = \frac{\sum_{w \in T} T_{u,w} R_{w,i}}{\sum_{w \in T} T_{u,w}}$$

P. Massa [48] introduces MoleTrust. The ideas used in MoleTrust and TidalTrust are similar except MoleTrust considers all raters up to a maximum-depth given as an input. maximum-depth is independent of any specic user and item. Also, to compute the trust value between  $u$  and  $v$  in MoleTrust, they perform a backward exploration. It means that the trust value from  $u$  to  $v$  is the aggregation of trust values between  $u$  and users directly trusting  $v$  weighted by the direct trust values.

Similarly, M. Jamali [27] proposes a random walk method (TrustWalker) which combines trust-based and item-based recommendation. Specifically, TrustWalker consists of two major components: random walk in the trust network and probabilistic item rating selection on each visited node. During the random walk, a user's direct and indirect friends are visited in the trust network. Whenever a friend is visited, if she has rated the target item, her rating is logged; if she has not rated the target item, but has rated an item similar to the target item, her rating is logged with certain probability. The probability of using a rating of a similar item in place of a rating for the target item increases as the length of random walk increases. This probabilistic item rating selection aims to avoid

traverse deeply in the network when no user in a close neighborhood has rated the target item.

M. Jamali et.al employ the Pearson Correlation Coefficient of ratings expressed for two items to calculate the similarity value between item  $i$  and item  $j$ ,

$$corr(i, j) = \frac{\sum_{u \in U_c} (R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U_c} (R_{u,i} - \bar{R}_u)^2 \sum_{u \in U_c} (R_{u,j} - \bar{R}_u)^2}} \quad (2.14)$$

where  $U_c$  is the set of users who have rated both  $i$  and  $j$ ,  $R_{u,i}$  and  $R_{u,j}$  are ratings of  $u$  assigned to item  $i$  and  $j$  respectively.  $\bar{R}_u$  is the average rating issued by user  $u$ . Values of the Pearson correlation are in the range of  $[0, 1]$ . Only items with positive correlation with the target item are considered. The similarity value is then calculated as:

$$sim(i, j) = \frac{1}{1 + e^{-\frac{|U_c|}{2}}} \times corr(i, j) \quad (2.15)$$

where  $|U_c|$  is the number of users who rated both  $i$  and  $j$ .

## 2.5.2 Model-Based using Social Trust Data

In contrast to the neighborhood-based approaches, the model-based approaches use the observed user-item ratings and social trust to train a compact model that explains the given data, so that ratings could be predicted via the model instead of directly manipulating the original rating database as the neighborhood-based approaches do.

H.Ma [44] proposed a matrix factorization approach for social network based recommendation, called STE. Their method is a linear combination of basic matrix factorization approach and a social network based approach. The predicted ratings  $\hat{R}_{u,i}$  of user  $u$  on item  $i$  is obtained as follows:

$$\hat{R}_{u,i} = \alpha U_u V_i^U + (1 - \alpha) \sum_{v \in F(u)} T_{u,v} U_u V_i^T$$

where  $U \in \mathbb{R}^{number\_of\_users \times R}$  and  $V \in \mathbb{R}^{number\_of\_items \times R}$  is the latent feature profiles

for users and items respectively.  $F(u)$  denotes user  $u$ 's direct friends and  $T_{u,v}$  denotes the trust level between user  $u$  and  $v$ . The trade-off between the feedback data (ratings) and the influence from social network is determined by  $\alpha \in [0, 1]$ . Obviously, the social influence is ignored for  $\alpha = 1$ , while  $\alpha = 0$  assigns the highest possible weight to the social influence.

The STE model does not consider the transitivity of trust in social networks. M. Jamali [28] propose SocialMF model that addresses the transitivity of trust in social networks. In SocialMF model, the dependence of a user's feature vector on the direct neighbors' feature vectors can propagate through the network, making a user's feature vector dependent on possibly all users in the network (with decaying weights for more distant users). The training objective function to optimize RMSE is as following:

$$\min_{U,V} \sum_{\text{all observed rating } R_{u,i}} (R_{u,i} - \hat{R}_{u,i})^2 + \beta \sum_{\text{all user } u} \|U_u - \sum_{v \in F(u)} T_{u,v} U_v\|_F^2 + \lambda(\|U\|_F^2 + \|V\|_F^2)$$

where  $U \in \mathbb{R}^{[\text{number\_of\_users} \times R]}$  and  $V \in \mathbb{R}^{[\text{number\_of\_items} \times R]}$  is the latent feature profiles for users and items respectively.  $F(u)$  denotes user  $u$ 's direct friends and  $T_{u,v}$  denotes the trust level between user  $u$  and  $v$ . The trade-off between the feedback data (ratings) and the influence from social network is determined by  $\beta > 0$ . Obviously, the social influence is ignored for  $\beta = 0$ , while increasing  $\beta$  will put more weight to the social influence.

Besides these model, a number of extensions for social recommendation have been proposed and the detail can be obtained from the bibliography given in [69, 86].

---

---

## CHAPTER 3

---

# IMPROVING USERS' ACCEPTANCE USING RATING AND TAGGING DATA

Users' rating and tagging data help to improve users' acceptance towards recommendation system. Existing systems have utilized only ternary relationships such as in rating network (users-items-ratings) [30, 4, 66, 42], or social tagging network (users-items-tags) [74, 68, 59] to increase the accuracy in recommender system respectively. However, to the best of our knowledge, there is no existing work which considers the quaternary relationship among users, items tags and ratings. This relationship is important to understand the user's interest. Besides that, we also help in improving the users' acceptance towards recommendation system by explaining their recommendations in our framework.

In this chapter, we propose to improve users' acceptance which take the quaternary relation which is user-item-tag-rating into account. We model the quaternary relationship among users, items, tags and ratings as a 4-order tensor and cast the recommendation problem as a multi-way latent semantic analysis problem. A unified framework for user recommendation, item recommendation, tag recommendation and item rating prediction

is proposed. In addition, we also provide explanations on why the items are recommended and provide adaptive feedback scheme to further increase users' acceptance.

### 3.1 Motivation

The amount of information on the Web is increasing at a lightning pace. In order to adequately cope with this information overload, recommendation systems are needed to bring the relevant resources to the attention of the users automatically. Recommendation systems are typically classified according to the type of tasks they are intended for, which include:

1. User recommendation - Here the task is to identify users with common interests so as to extend the connection among users with similar interests (e.g., Amazon <sup>1</sup> and Facebook <sup>2</sup>). Existing user recommendation systems (e.g., Amazon) determine users with common interests either through the fact that these users often give the same rating to similar resources, or they use similar tags to describe the resources.
2. Item recommendation - Instead of stopping at identifying users with common interests, this task goes one step further. Based on the identified set of users with common interests, the items that this set of users are interested in become the candidates for recommending to the target user (e.g., Amazon and YouTube <sup>3</sup>).
3. Tag recommendation - This task has emerged recently due to the popularity of social tagging activity. Users typically use a ubiquitous vocabulary as tags to reflect the semantics of the items from his/her point of view. In order to improve the selection of vocabulary to be used as tags, tag recommendation is now a hot research area which aims to provide users with a good set of tags to describe items (e.g., Amazon, Facebook and Flickr <sup>4</sup>). Most tag recommendation systems rely on

---

<sup>1</sup><http://www.amazon.com>

<sup>2</sup><http://www.facebook.com>

<sup>3</sup><http://www.youtube.com>

<sup>4</sup><http://www.flickr.com>

identifying similar users and recommending the tags used by these similar users on similar resources.

4. Item rating prediction - Here, the task goes beyond just determining whether an item should be recommended to a user. Instead, the recommendation systems need to predict the degree of preference a user is likely to exhibit for an item (e.g., Netflix <sup>5</sup>).

Till now, most, if not all, recommendation systems utilize only ternary relationships in generating their recommendations. The collaborative filtering-based recommendation systems [30, 4, 42, 66] typically make use of the users-rating-items relationship to group users based on their ratings on items, whereas the tag-based recommendation systems utilize the users-tags-items relationship to perform the various tasks [74, 68, 59]. We argue that recommendations based on ternary relationships are not accurate as they would have missed out important associations. The quaternary relationships can reveal semantics that cannot be obtained otherwise. This is reinforced by the following observations:

1. Users may use the same tag for an item but have different ratings for it. For example, users  $U_2$  and  $U_4$  both use the same tag “*comedy*” on item “*ForrestGump*”. However, user  $U_2$  likes the book but  $U_4$  does not. Hence using tag information alone is insufficient.
2. Items may have multiple tags indicating their different facets. This could give rise to varied ratings, depending on the facet considered by the user. For example, “*ForrestGump*” is tagged as a “*psychology*” book by user  $U_1$  and a “*comedy*” by  $U_4$ . However, the book may be not a good comedy book as  $U_4$  dislike it. Yet this book could be an interesting psychology book since user  $U_1$  like it. In other words, rating information alone is insufficient.
3. Some tags may carry implicit semantics that can reveal the users’ preferences. For example, user  $U_2$  tags book “*GroundhogDay*” with the tag “*excellent*” implying

---

<sup>5</sup><http://www.netflix.com>

that  $U_2$  likes the book. Similarly, the tag “*overrated*” which is tagged by the user  $U_4$  will imply that  $U_4$  dislikes the book “Toy Story”. This observation tells us that the combination of tag and rating information gives extra insights into the users’ preferences.

In order to capture the quaternary relationship among users, items, tags and ratings, we propose a model based on the 4-order tensor. We apply the Higher-Order Singular Value Decomposition (HOSVD) [36] in the 4-order tensor to reveal the latent semantic associations among users, items, tags and rating. With this model, we design a unified framework to perform item rating prediction as well as user, item and tag recommendations.

In addition, we also provide explanations on why the items are recommended to users proved to be impossible. We achieve this by utilizing the PARAFAC model to extract latent features of users and items and map them to a common basis in terms of tags. We then generate profiles of users and items in the form of tagclouds. These tags capture the semantic features of items from users’ point of views and allow us to generate explanations that are intuitive to users in the form of tagclouds. Our recommender system also allows users to provide feedback on the recommended items. Based on the feedback, we design an incremental algorithm to update the approximate core tensor to generate a new list of recommendations. We carry out experiments on a real world dataset to demonstrate the effectiveness of our proposed approach and explanation. To the best of our knowledge, this is a first work to explore the use of the quaternary relationship among user, items, tags and ratings for recommendation tasks.

## 3.2 Tensor algebra and multilinear analysis

Tensor algebra and multilinear analysis have been applied successfully in many domains [32, 36]. In this section, we review the concepts and terminologies used in the thesis. For the sake of simplicity, Table 3.1 summarizes the symbols used in describing the

following section.

Table 3.1: Meanings of symbols used

Symbol	Meaning
$u$	A user
$U$	The set of all users
$v$	A item
$V$	The set of all items
$t$	A tag
$T$	The set of all tags
$\mathcal{A}$	the tensor containing the ratings and taggings will be a 4-dimensional tensor
$\hat{\mathcal{A}}$	the approximate tensor
$A_{(n)}$	n-mode matrix unfolding of tensor $\mathcal{A}$
$\times_n$	n-mode product
$U^{(i)}$	Latent feature matrix of tensor $\mathcal{A}$ at mode $i$
$(U^{(i)})^T$	Transpose of latent feature matrix of tensor $\mathcal{A}$ at mode $i$

A tensor is a multidimensional array. An N-order tensor  $\mathcal{A}$  is denoted as  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N}$  with elements  $a_{i_1 \dots i_n}$  and dimensions  $I_1, I_2, \dots, I_N$ .

Table 3.2: Example dataset of a 3-order tensor

$I_1$	$I_2$	$I_3$	element value
1	1	1	1
1	2	2	1
2	3	3	1
3	1	1	1

For example, the corresponding 3-order tensor  $\mathcal{A}$  for the example dataset in Table 3.2 is:

$$\mathcal{A}(:, :, 1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \mathcal{A}(:, :, 2) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \mathcal{A}(:, :, 3) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

**Definition 1** The *matrix unfolding* of an N-order tensor  $\mathcal{A} = \mathbb{R}^{I_1 \times \dots \times I_N}$  along the dimension  $d$  are vectors obtained by keeping the index  $d$  fixed while varying the other indices.



The unfolding of our example 3-order tensor  $\mathcal{A}$  along each dimension is:

$$\mathbf{A}_{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{A}_{(2)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{A}_{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Note that the definition of the matrix unfolding involves the tensor dimensions  $I_1, I_2, I_3$  in a cyclic way. Hence, for the unfolding of dimension  $I_c \times I_a I_b$ , the index  $I_b$  varies more slowly than  $I_a$ .

**Definition 2** The *n-mode product* of a tensor  $\mathcal{A} = \mathbb{R}^{I_1 \times \dots \times I_N}$  by a matrix  $U = \mathbb{R}^{J_n \times I_n}$ , denoted by  $\mathcal{A} \times_n U$ , is a  $(I_1 \times I_2 \cdots I_{n-1} \times J_n \times I_{n+1} \cdots I_N)$ -tensor where the entries are given by

$$\begin{aligned} & (\mathcal{A} \times_n U)_{i_1 i_2 i_3 \dots i_{n-1} j_n i_{n+1} \dots i_N} \\ &= \sum_{i_n} a_{i_1 i_2 i_3 \dots i_{n-1} i_n i_{n+1} \dots i_N} \cdot u_{j_n i_n} \end{aligned}$$

For example, the 1-mode product of a tensor  $\mathcal{A} = \mathbb{R}^{2 \times 3 \times 4}$  by a matrix  $U = \mathbb{R}^{5 \times 2}$ ,

denoted as  $\mathcal{A} \times_1 \mathbf{U}$  is an  $5 \times 3 \times 4$  tensor in which the entries are given by

$$(\mathcal{A} \times_1 \mathbf{U})_{ji_2i_3} = \sum_{k=1}^2 a_{ki_2i_3} \cdot u_{jk} \quad 1 \leq j \leq 5$$

**PROPERTY 1.** Given the tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N}$  and the matrices  $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$  and  $\mathbf{V} \in \mathbb{R}^{J_m \times I_m}$  ( $n \neq m$ ), we have

$$(\mathcal{A} \times_n \mathbf{U}) \times_m \mathbf{V} = (\mathcal{A} \times_m \mathbf{V}) \times_n \mathbf{U} = \mathcal{A} \times_n \mathbf{U} \times_m \mathbf{V}$$

**PROPERTY 2.** Given the tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N}$  and the matrices  $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$  and  $\mathbf{V} \in \mathbb{R}^{K_n \times J_n}$ , then

$$(\mathcal{A} \times_n \mathbf{U}) \times_n \mathbf{V} = \mathcal{A} \times_n (\mathbf{V} \cdot \mathbf{U})$$

**Definition 3** Let  $\vec{V}^{(1)} \in \mathbb{R}^{1 \times I_1}$ ,  $\vec{V}^{(2)} \in \mathbb{R}^{1 \times I_2} \dots \vec{V}^{(N)} \in \mathbb{R}^{1 \times I_N}$  be vectors. Then the **outer product** of two or more vectors, denoted as  $\mathcal{A} = \vec{V}^{(1)} \otimes \vec{V}^{(2)} \dots \otimes \vec{V}^{(N)}$ , is a  $(I_1 \times I_2 \dots \times I_N)$ -tensor where the entries are given by

$$\mathcal{A}(i_1, \dots, i_N) = \vec{V}_{i_1}^{(1)} \times \vec{V}_{i_2}^{(2)} \dots \times \vec{V}_{i_N}^{(N)}$$

for all  $1 \leq i_n \leq I_n$ ,  $1 \leq n \leq N$ .

The Higher-Order Singular Value Decomposition (HOSVD) is a generalization of the Singular Value Decomposition (SVD) to higher-order tensors [36] and can be written as n-mode product:

$$\mathcal{A} = \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \dots \times_N \mathbf{U}^{(N)}$$

where  $\mathbf{U}^{(n)}$  contain the orthonormal vectors (or n-mode singular vectors) spanning the column space of the  $\mathcal{A}_{(n)}$  (n-mode matrix unfolding of  $\mathcal{A}$ ).  $\mathcal{S}$  is the core tensor and has the property of all orthogonality.

Consider our example tensor  $\mathcal{A}$  and its matrix unfolding  $\mathcal{A}_{(1)}$ . We perform SVD on  $\mathcal{A}_{(1)}$  and obtain the resultant left singular matrix:

$$\mathbf{U}^{(1)} = \begin{pmatrix} -0.85 & 0 & 0.53 \\ 0 & -1 & 0 \\ -0.53 & 0 & -0.85 \end{pmatrix} \quad \mathbf{U}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{U}^{(3)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

With this, the core tensor  $\mathcal{S} \in \mathbb{R}^{3 \times 3 \times 3}$  can be constructed as described in [36]. We have  $\mathcal{S} = \mathcal{A} \times_1 (\mathbf{U}^{(1)})^T \times_2 (\mathbf{U}^{(2)})^T \times_3 (\mathbf{U}^{(3)})^T$  where

$$\mathcal{S}(:, :, 1) = \begin{pmatrix} -1.38 & 0 & 0 \\ 0 & 1 & 0 \\ -0.32 & 0 & 0 \end{pmatrix} \quad \mathcal{S}(:, :, 2) = \begin{pmatrix} 0 & -0.85 & 0 \\ 0 & 0 & 0 \\ 0 & -0.53 & 0 \end{pmatrix} \quad \mathcal{S}(:, :, 3) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

**Definition 4** The  $n$ -rank of tensor denoted by  $R_N = \text{rank}_n(\mathcal{A})$ , is the dimension of the vector space spanned by the  $n$ -mode matrix. We denote the rank of tensor  $\mathcal{A}$  as  $\text{rank}(\mathcal{A}) = (\text{rank}_1(\mathcal{A}) \cdots, \text{rank}_n(\mathcal{A}))$

**Definition 5** Given a tensor  $\mathcal{A} = R^{I_1 \times \cdots \times I_N}$ , the RANK- $(R_1 \cdots, R_N)$  approximation  $\hat{\mathcal{A}}$  is defined as  $\min_{\mathcal{B} \in \mathcal{S}} \|\mathcal{A} - \mathcal{B}\|_F^2$ ,  $\mathcal{S} = \{\mathcal{B} | \text{rank}(\mathcal{B}) \leq (R_1 \cdots, R_N)\}$  where  $\|\mathcal{A} - \mathcal{B}\|_F^2$  is the least-square cost.<sup>6</sup>

Suppose we want to get the RANK-(2,3,3) approximation, we first retain the first  $c_i$  column of matrix  $\mathbf{U}^{(i)}$  at mode  $i$  ( $1 \leq i \leq 3$ ) as follows:

$$\hat{\mathbf{U}}^{(1)} = \begin{pmatrix} -0.85 & 0 \\ 0 & -1 \\ -0.53 & 0 \end{pmatrix} \quad \hat{\mathbf{U}}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \hat{\mathbf{U}}^{(3)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

---

<sup>6</sup>the square frobenius norm is defined as  $\|\mathcal{A}\|_F^2 = \sum_{i_1=1}^{R_1} \cdots \sum_{i_N=1}^{R_N} \mathcal{A}(i_1 \cdots, i_N)^2$

We can now construct the approximate core tensor  $\hat{\mathcal{S}} \in \mathbb{R}^{2 \times 3 \times 3}$  using  $\mathcal{S} = \mathcal{A} \times_1 (\hat{\mathcal{U}}^{(1)})^T \times_2 (\hat{\mathcal{U}}^{(2)})^T \times_3 (\hat{\mathcal{U}}^{(3)})^T$  :

$$\hat{\mathcal{S}}(:, :, 1) = \begin{pmatrix} -1.38 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \hat{\mathcal{S}}(:, :, 2) = \begin{pmatrix} 0 & -0.85 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \hat{\mathcal{S}}(:, :, 3) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Finally, we obtain the RANK-(2,3,3) approximation  $\hat{\mathcal{A}} = \hat{\mathcal{S}} \times_1 \hat{\mathcal{U}}^{(1)} \times_2 \hat{\mathcal{U}}^{(2)} \times_3 \hat{\mathcal{U}}^{(3)}$ :

$$\hat{\mathcal{A}}(:, :, 1) = \begin{pmatrix} 1.2 & 0 & 0 \\ 0 & 1 & 0 \\ 0.72 & 0 & 0 \end{pmatrix} \quad \hat{\mathcal{A}}(:, :, 2) = \begin{pmatrix} 0 & 0.72 & 0 \\ 0 & 0 & 0 \\ 0 & 0.45 & 0 \end{pmatrix} \quad \hat{\mathcal{A}}(:, :, 3) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

where  $\|\mathcal{A} - \hat{\mathcal{A}}\|_F^2 = 0.618$  which is minimized.

### 3.3 Recommender System Overview

Figure 3-1 shows the framework of the proposed recommender system. The repository contains users' rating and social tagging activities and user profiles. The Watcher monitors log user activities such as the tags they use, the ratings they give to items, etc. We provide an interface for users to choose tags that are used by other users (e.g, Pixar, Disney, animaton, TomHanks, cgi.etc) or add their own tags to the books (see Figure 3-2(a)). Besides that, user can rate the item based on his/her opinion.

Based on the user profiles, the recommender utilizes the 4-order tensor model to generate personalized recommendation using Quaternary Semantic Analysis *QSA* and provide explanation for the recommended items. The Advisor shows the top-N items to the user, and accepts feedback from the user if s/he is not satisfied with the rec-

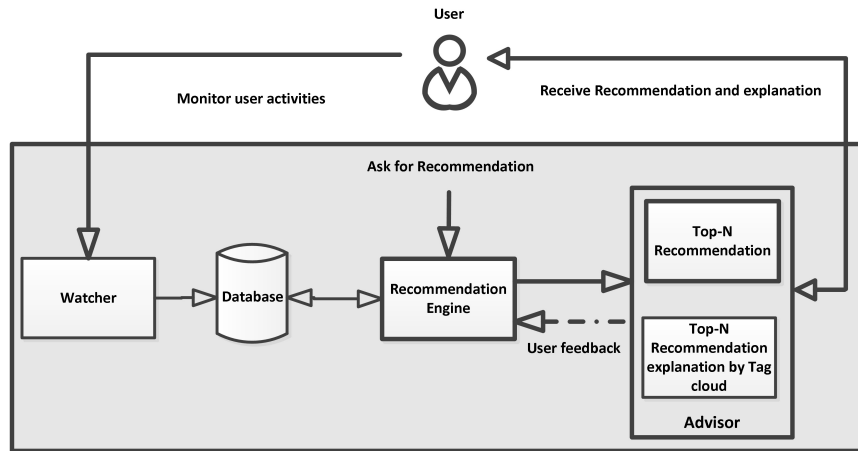


Figure 3-1: Recommendation System Overview

ommendations. Based on the feedback, the recommender will compute a new list of recommendations for the user.

After mapping the users, tags, items and ratings to a common basis, we compute the  $k$  nearest tags for each user. These tags are displayed as a tagcloud<sup>7</sup> that summarizes the user's interest. Similarly, we also compute the  $k$  nearest tags for each item and display them as a tagcloud to summarize the items' topic. A red (blue) colored tag indicates that it is often associated with positively (negatively) rated items. A black colored tag means that it is neutral.

Figure 3-2(a) shows a screenshot of our recommender system. From the user and item tagclouds, a user will realize that "*Toy story*" is recommended to him/her because these clouds have tags in common "*classic*", "*disney*" "*imdb top 250*" and "*animation*". The context to aid user understanding, e.g., "*disney*" "*animation*", "*classic*", "*imdb top 250*" and "*Oscar*" are key factors that characterize the user. Note that a user can choose different levels of summarization by controlling the number of tags displayed.




Figure 3-2(b) shows the new list of items recommended after the user clicks on the thumb-up icon for "*Tom Hanks*" and "*Adventure*".

In summary, the system consists of several components. The main three sub-components are described: (1) Recommender Engine- *QSA* (2) Advisor-Top-N recommendation (3)

<sup>7</sup>Tagclouds are generated using WordCram (<http://wordcram.org/>)

MovieLens Movie

Home | Recommendation to you | Rate and Tag | My Rating History | My Tagging History




Recommendation	User ID:	Top N:
<p>User's Interests tag cloud</p> <p>Max Words: 10</p> <p>Toy Story 2</p>  <p>Your tags: <input type="text" value="Toy Story 2"/> Add</p> <p>Your Rating: <input type="text" value="5"/> Add</p> <p>imdb top 250 animation classic tim allen sequel tom hanks Pixar adventure</p>		
<p>Mulan</p>  <p>Your tags: <input type="text" value="Mulan"/> Add</p> <p>Your Rating: <input type="text" value="5"/> Add</p> <p>oscar disney animation imdb top 250 drama family classic comedy action war</p> <p>children good 70mm animation fun book musical disney classic family</p>		
<p>Lion King</p>  <p>Your tags: <input type="text" value="Lion King"/> Add</p> <p>Your Rating: <input type="text" value="5"/> Add</p> <p>disney animation imdb top 250 best fun children 70mm Oscar musical</p>		

comedy, oscar, disney, animation, drama, classic, action, imdb top 250, family, war, Pixar, tim allen, sequel, adventure, cgi, tom hanks, children, musical, fun, good, 70mm, book, best, kids

(a) Before feedback

MovieLens Movie

Home | Recommendation to you | Rate and Tag | My Rating History | My Tagging History

User Recommendation	User ID:	Top N:
<p>User's Interests tag cloud</p> <p>Max Words: 10</p> <p>Forrest Gump</p>  <p>Your tags: <input type="text" value="Forrest Gump"/> Add</p> <p>Your Rating: <input type="text" value="5"/> Add</p> <p>adventure classic war comedy vietnam tom hanks drama story oscar</p>		
<p>Saving Private Ryan</p>  <p>Your tags: <input type="text" value="Saving Private Ryan"/> Add</p> <p>Your Rating: <input type="text" value="5"/> Add</p> <p>oscar drama tom hanks imdb top 250 action war classic comedy adventure</p> <p>aids drama divorce tom hanks oscar classic law AFI political</p>		
<p>Philadelphia</p>  <p>Your tags: <input type="text" value="Philadelphia"/> Add</p> <p>Your Rating: <input type="text" value="5"/> Add</p> <p>steven spielberg action war ww2 imdb top 250 drama tom hanks history oscar</p>		

comedy, oscar, tom hanks, animation, drama, classic, action, imdb top 250, adventure, war, story, vietnam, steven spielberg, ww2, AFI, history, gay, aids, political, divorce, law, AFI

(b) After feedback

Figure 3-2: Screenshots of recommendation system

Advisor-Top N Recommendation Explanation and FeedBack In the following sections, we describe each sub-components in details in the following sections.

### 3.3.1 Recommender Engine - Quaternary Semantic Analysis

In this section, we will show our Recommender Engine - Quaternary Semantic Analysis. The main idea behind the quaternary semantic analysis is to capture the underlying relationships among users-tags-items-ratings. Suppose we have a list of quadruples  $\langle u, t, r, v \rangle$  denoting that a user  $u$  will provide tag  $t$  to a book  $v$  and give the rating  $r$  if he has watched  $v$  before. We first model this list of quadruples as a 4-order tensor  $\mathcal{A} \in U \times T \times R \times V$ , where  $U$  is the set of all users,  $V$  the set of all items/resources,  $T$  the set of all tags and  $R$  the set of ratings. An entry  $\mathcal{A}(u, t, r, v)$  has a value 1 if the quadruple  $\langle u, t, r, v \rangle$  exists, otherwise it has a value of 0.

We reduce the rank of the original tensor to minimize the effect of noise on the underlying population and reduce sparseness. This is achieved by approximating the tensor  $\mathcal{A}$  to a lower rank tensor. Given the dimensions of users, tags, ratings and items, namely,  $c_1, c_2, c_3, c_4$ , we want to obtain the RANK- $(c_1, c_2, c_3, c_4)$  approximation of  $\hat{\mathcal{A}}$  such that the square frobenius norm defined as:

$$\|\hat{\mathcal{A}}\|_F^2 = \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} \hat{\mathcal{A}}(i_1, \dots, i_N)^2$$

is minimized.

In our experiments, we set  $c_1, c_2, c_3, c_4$  to preserve 70%, 90%, 80%, 90% of the original tensor information in each dimension respectively.

#### Tensor Approximation Algorithm

Algorithm 1 shows the details for approximating a tensor. We first apply SVD on the four matrix unfoldings  $\mathbf{A}_{(1)}, \mathbf{A}_{(2)}, \mathbf{A}_{(3)}$ , and  $\mathbf{A}_{(4)}$ . Note that:

$$\mathbf{A}_{(i)} = \hat{\mathbf{U}}^{(i)} \cdot \mathbf{S}^{(i)} \cdot (\hat{\mathbf{V}}^{(i)})^T, \quad 1 \leq i \leq 4 \quad (3.1)$$

---

**Algorithm 1:** Quaternary Semantic Analysis

---

**Input:**

List of quadruples  $\langle \text{users}, \text{tags}, \text{rating}, \text{items} \rangle$ ;  
Dimensions of users, tags, ratings and items  $c_1, c_2, c_3, c_4$ ;

**Output:**

Approximate Tensor  $\hat{\mathcal{A}}$ ;

- 1: Initialization: From the quadruple (users, items tag and rating), we construct tensor  $\mathcal{A} \in \mathbb{R}^{|U| \times |T| \times |R| \times |V|}$ , where  $|U|, |V|, |T|$  and  $|R|$  are the number of users, items and tags and rating respectively
- 2: Calculate the matrix unfolding  $\mathbf{A}_{(1)}, \mathbf{A}_{(2)}, \mathbf{A}_{(3)}$ , and  $\mathbf{A}_{(4)}$  from tensor  $\mathcal{A}$ .
- 3: Construct the variance matrix  $\mathbf{C}_i = \mathbf{A}_{(i)} \mathbf{A}_{(i)}^T$  for each mode  $1 \leq i \leq 4$
- 4: Compute  $\mathbf{U}^{(i)}$  by diagonalizing  $\mathbf{C}_i, 1 \leq i \leq 4$
- 5: Remove the least significant rows  $|U| - c_1, |V| - c_2, |T| - c_3$  and  $|R| - c_4$  from  $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}$ , and  $\mathbf{U}^{(4)}$ , respectively. Denote the result as  $\hat{\mathbf{U}}^{(1)}, \hat{\mathbf{U}}^{(2)}, \hat{\mathbf{U}}^{(3)}$ , and  $\hat{\mathbf{U}}^{(4)}$ .
- 6: Calculate the approximate core tensor  $\hat{\mathcal{S}}$  as follows:

$$\hat{\mathcal{S}} = \mathcal{A} \times_1 (\hat{\mathbf{U}}^{(1)})^T \times_2 (\hat{\mathbf{U}}^{(2)})^T \times_3 (\hat{\mathbf{U}}^{(3)})^T \times_4 (\hat{\mathbf{U}}^{(4)})^T$$

- 7: Approximate the original tensor by:

$$\hat{\mathcal{A}} = \hat{\mathcal{S}} \times_1 \hat{\mathbf{U}}^{(1)} \times_2 \hat{\mathbf{U}}^{(2)} \times_3 \hat{\mathbf{U}}^{(3)} \times_4 \hat{\mathbf{U}}^{(4)}$$

---

In order to obtain the left matrix of the SVD, we first define a matrix  $\mathbf{C}_i$  as follows:

$$\mathbf{C}_i = \mathbf{A}_i \cdot \mathbf{A}_i^T, \quad 1 \leq i \leq 4, \quad (3.2)$$

Since each  $\hat{\mathbf{U}}^{(i)}$  and  $\hat{\mathbf{V}}^{(i)}$  are orthogonal and each  $\mathbf{S}^{(i)}$  is diagonal, we substitute (3.1) into (3.2):

$$\begin{aligned} \mathbf{C}_i &= \mathbf{A}_i \mathbf{A}_i^T \\ &= (\hat{\mathbf{U}}^{(i)} \cdot \mathbf{S}^{(i)} \cdot (\hat{\mathbf{V}}^{(i)})^T) \cdot (\hat{\mathbf{U}}^{(i)} \cdot \mathbf{S}^{(i)} \cdot (\hat{\mathbf{V}}^{(i)})^T)^T \\ &= \hat{\mathbf{U}}^{(i)} (\mathbf{S}^{(i)})^2 (\hat{\mathbf{U}}^{(i)})^T \end{aligned}$$

Therefore, each required  $\mathbf{U}^{(i)}$  can be computed by diagonalizing each  $\mathbf{C}_i$  and taking its eigenvectors (Lines 3-4).

Consider our example quaternary relations in Table 3.3 which is the subset of Table



Table 3.3: Quaternary relations among users, tags, ratings and items in Book Domain

User	Tag	Rating	Item
$U_1$	psychology	like	Forrest Gump
$U_1$	psychology	like	Beautiful Mind
$U_2$	comedy	like	Forrest Gump
$U_2$	excellent	like	Groundhog Day
$U_2$	comedy	like	Groundhog Day
$U_3$	comedy	like	Forrest Gump
$U_4$	comedy	dislike	Forrest Gump
$U_4$	comedy	dislike	Toy Story
$U_4$	overrated	dislike	Toy Story

1.3. We initialize the the weights of the quadruples to 1, as shown in Table 3.4. A 4-order tensor  $\mathcal{A} \in \mathbb{R}^{4 \times 4 \times 4 \times 2}$  can be constructed from this table. For example, the first quadruple  $\langle U_1, psychology, like, ForrestGump \rangle$  will correspond to the entry  $\mathcal{A}(1, 1, 1, 1)=1$ .

Table 3.4: Data of the tensor  $\mathcal{A}$

User	Tag	Rating	Item	Val
$U_1$	psychology	like	Forest Gump	1
$U_1$	psychology	like	Beautiful Mind	1
$U_2$	comedy	like	Forest Gump	1
$U_2$	excellent	like	Groundhog Day	1
$U_2$	comedy	like	Groundhog Day	1
$U_3$	comedy	like	Forest Gump	1
$U_4$	comedy	dislike	Forest Gump	1
$U_4$	comedy	dislike	Toy Story	1
$U_4$	overrated	dislike	Toy Story	1

For each matrix unfolding  $A_{(i)}$ ,  $1 \leq i \leq 4$ , we compute  $U^{(i)}$  as follows:

$$U^{(1)} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0.92 & 0 & 0 & -0.38 \\ 0.38 & 0 & 0 & 0.92 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad U^{(2)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0.95 & 0 & 0 & -0.30 \\ 0.21 & 0 & 0.71 & 0.67 \\ 0.21 & 0 & -0.71 & 0.67 \end{pmatrix}$$

$$U^{(3)} = \begin{pmatrix} 0.88 & 0 & -0.27 & -0.40 \\ 0.22 & 0 & -0.50 & 0.84 \\ 0.29 & -0.71 & 0.58 & 0.27 \\ 0.29 & 0.71 & 0.58 & 0.27 \end{pmatrix} \quad U^{(4)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

We maintain only a subset of the original dimensions in each of the four modes (Line 5). Here, we choose  $c_1 = 3, c_2 = 4, c_3 = 4, c_4 = 2$ . The resulting  $\hat{U}^{(i)}$  are shown as follows:

$$U^{(1)} = \begin{pmatrix} 0 & 0 & 1 \\ 0.92 & 0 & 0 \\ 0.38 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad U^{(2)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0.95 & 0 & 0 & -0.30 \\ 0.21 & 0 & 0.71 & 0.67 \\ 0.21 & 0 & -0.71 & 0.67 \end{pmatrix}$$

$$U^{(3)} = \begin{pmatrix} 0.88 & 0 & -0.27 & -0.40 \\ 0.22 & 0 & -0.50 & 0.84 \\ 0.29 & -0.71 & 0.58 & 0.27 \\ 0.29 & 0.71 & 0.58 & 0.27 \end{pmatrix} \quad U^{(4)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Lines 6-7 compute the approximate tensor  $\hat{\mathcal{A}}$ . The final weights of the quadruples are shown in Table 3.5. We observe that the algorithm has added the following two quadruples:

$$\langle U_3, \text{comedy}, \text{like}, \text{Groundhog Day} \rangle$$

$$\langle U_3, \text{excellent}, \text{like}, \text{Groundhog Day} \rangle$$

Note that user  $U_3$  has not used the tag “*excellent*” previously, and there is no indication on which item should be recommend to  $U_3$  based on the tags “*comedy*” and “*excellent*” in the original table (recall Table 3.4). However, the newly added quadruples indicate that the book “*Groundhog Day*” is associated with user  $U_3$  and tags “*comedy*” and “*excellent*” with a weight of 0.35. Hence, the book “*Groundhog Day*” will be recommended to  $U_3$ .

Table 3.5: Output of the approximate tensor  $\hat{\mathcal{A}}$

User	Tag	Rating	Item	Val
$U_1$	psychology	like	Forest Gump	1.01
$U_1$	psychology	like	Beautiful Mind	1
$U_2$	comedy	like	Forest Gump	1.2
$U_2$	excellent	like	Groundhog Day	0.85
$U_2$	comedy	like	Groundhog Day	0.85
$U_3$	comedy	like	Forest Gump	0.50
$U_4$	comedy	dislike	Forest Gump	1
$U_4$	comedy	dislike	Toy story	1
$U_4$	overrated	dislike	Toy story	1
$U_3$	comedy	like	Groundhog Day	0.35
$U_3$	excellent	like	Groundhog Day	0.35

We observe that latent associations such as the newly added quadruples in Table 3.5 may not be found if the tensor data is sparse, that is, most of the entries are 0. This problem is particularly acute as we are working with the quaternary relationship. We overcome this problem by applying a smoothing technique to Line 1 in Algorithm 1. The smoothing method is based on the similarity between items. For each user  $\langle u, r, t \rangle$  in the tensor, let  $S_1$  be the set of items that are rated and tagged by user  $u$ , and  $S_2 = V - S_1$  where  $V$  is the set of all items/resources. We assign  $\langle u, r, t, v_j \rangle$  with the overall

similarly between item  $v_j \in S_2$  and the items in  $S_1$ .

The overall similarity between item  $v_j \in S_2$  and the items in  $S_1$  can be calculated as follows:

$$SIM(v_j, S_1) = \frac{\sum_{v_i \in S_1} sim(v_i, v_j)}{|S_1|} \quad (3.3)$$

where  $sim(v_i, v_j)$  is the cosine similarity between items  $v_i$  and  $v_j$ , assuming the items are represented by vectors of word weights.

The most time consuming steps in Algorithm 1 are the diagonalization of the unfolding matrices and the computation of the approximate core tensor. For real world applications involving large tensors, the work in [32] utilizes parallel architectures to optimize memory usage and reduce computation time. Note that the approximate tensor needs to be updated when we have new users, items, or tags. We adopt the methods described in [74] to incrementally update the approximate tensor.

### 3.3.2 Top-N Recommendation and Prediction

We describe how the proposed quaternary semantic analysis can provide a unified framework for the 4 common tasks: item recommendation, item rating prediction, user recommendation and tag recommendation.

- **User Recommendation.**

This is achieved in the proposed framework as follows: We first initialize the set  $Q$  to be empty. For each quadruple  $\langle u, r, t, i \rangle$  involving the target user  $u$ , we find the set of quadruples that have the same  $r$ ,  $t$ , and  $i$  values and add them to  $Q$ . Next, we group the quadruples in  $Q$  according to the user and aggregate the weights for each user. The top N users with the highest weights are recommended to  $u$ .

- **Item Recommendation.**

Here, we assume that a user *likes* a item if he/she has given a 5-star rating to the item [33]. Let  $T_u$  be the set of tags that a user  $u$  has used to tag items which s/he

likes. For each item  $i$  in  $V$ , we compute its total weight

$$w_i = \sum_{t \in T_u} \hat{\mathcal{A}}(u, t, r, i)$$

Then we sort the items according to their  $w_i$  and return the top  $N$  items with the highest weights.

- **Tag Recommendation.**

In our framework, the task of tag recommendation is reduced to examining the weights of the quadruples in the approximate tensor  $\hat{\mathcal{A}}$  which indicate how likely a user  $u$  would use the tag  $t$  for an item  $i$  if he has given the rating  $r$  before. Hence, we sort the quadruples involving  $u$ ,  $r$  and  $i$  according to their weights and return the top  $N$  tags.

- **Item Rating Prediction.**

We use the approximate tensor  $\hat{\mathcal{A}}$  to predict the item rating as follows. Let the rating scale be  $[1, R_{max}]$ . Let  $T_u$  be the set of tags that a user  $u$  has used to tag items. The rating that a user  $u$  will give to an item  $i$  such as  $r_{u,i}$  ( $1 \leq r_{u,i} \leq 5$ ) is given by:

$$r_{u,i} = \frac{\sum_{j=1}^{R_{max}} \sum_{t \in T_u} j \cdot \hat{\mathcal{A}}(u, t, j, i)}{\sum_{j=1}^{R_{max}} \sum_{t \in T_u} \hat{\mathcal{A}}(u, t, j, i)}$$

### 3.3.3 Tag-based Explanation and Feedback

While achieving accurate recommendation is good, this does not automatically lead to high users' acceptance of the recommended list. In the section, we describe how our system further increases users' acceptance of recommendations by providing a more intuitive tag-based explanations of why the items are recommended and adaptive scheme.

In particular, we try to utilize the PARAFAC model to extract latent features of users and items and map them to a common basis in terms of tags. We then generate profiles of users and items in the form of tagclouds. These tags capture the semantic features of items from users' point of views and allow us to generate explanations that are intuitive to users in the form of tagclouds. Our recommender system also allows users to provide feedback on the recommended items. Based on the feedback, we design an incremental algorithm to update the approximate core tensor to generate a new list of recommendations.

#### Tag-Based Explanation

Social tagging has become a common online activity of web users. This has generated a rich set of tags for products <sup>8</sup>, movies <sup>9</sup>, videos clips <sup>10</sup>, news articles <sup>11</sup>, blogs <sup>12</sup>. All

---

<sup>8</sup>[www.amazon.com](http://www.amazon.com)

<sup>9</sup>[www.movielens.org](http://www.movielens.org)

<sup>10</sup>[www.youtube.com](http://www.youtube.com)

<sup>11</sup>[digg.com](http://digg.com)

<sup>12</sup>[technorati.com](http://technorati.com)

these tags capture the semantics of items from users' points of view, using a ubiquitous vocabulary for heterogeneous domains of objects. We propose to use social tags to explain the recommendation made by QSA.

Tags are good rich semantic feature space of both items and users. For better illustration, For example, in Table 3.3,  $U_1$  likes the book "*Forrest Gump*" and tags it with the tag "*Psychology*". We observe that an item is often associated with tags that provide the semantic features for characterizing the item. As shown in Table 3.3, the tag "*comedy*" highlights the light-hearted nature of the book "*Forrest Gump*". We can also infer users' preference for certain aspects of an item based on the tags used and the rating information. For example,  $U_4$  does not seem to like comedies since he tags "*Forrest Gump*" as "*comedy*" and rates them with "*dislike*" (see Table 3.3). In contrast,  $U_2$  tags the same book as "*comedy*" and rates them with "*like*". In addition, tags can serve to highlight the latent associations between an item and the user. For example, a system with QSA engine may recommend "*Groundhog Day*" to  $U_3$  since  $U_3$  likes and tags "*Forrest Gump*" as comedy.

Tags can be used as explanation for the recommendation. In order to provide an intuitive explanation for the recommendation, our idea is to map the underlying relationships among user-tag-item-rating to a common basis in terms of tags so that it is meaningful and understandable to the users.

To achieve this, we need to extract the latent features of users, tags, items and map them into a common space. Figure 3-3 shows the mapped 2D space of users, items, and tags for our running example.

From the distribution, we observe that "*Groundhog Day*" and  $U_3$  are close together, suggesting that the two are rather similar. In addition, the closest tag to  $U_3$ , and "*Groundhog Day*" is *comedy*. In other words, the dominant feature in  $U_3$  and "*Groundhog Day*" is *comedy*. Hence, the recommender system can explain to  $u_3$  that "*Groundhog Day*" is recommended because he/she likes *comedy* and "*Groundhog Day*" is a *comedy*.

The extraction of the latent features of users, tags, and items and mapping them into

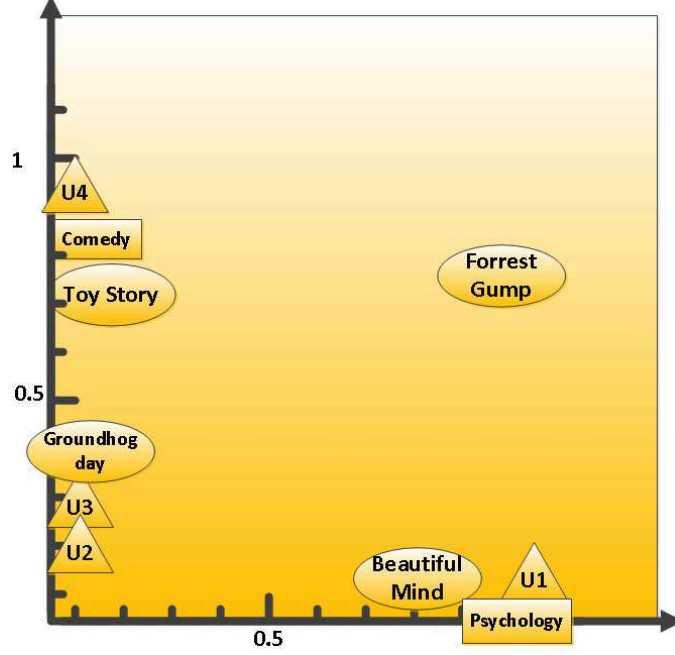


Figure 3-3: Distribution of users, tags, and items in  $r = 2$  dimensional space.

a common space requires a special decomposition model that allows only one to one mapping of dimension across each mode. In this thesis, we adopt the PARAFAC model to carry out the tensor decomposition. Given a tensor  $\mathcal{A}$  of size  $I_1 \times I_2 \cdots \times I_N$  and an input rank  $r$ , the PARAFAC tensor decomposition is defined as

$$\hat{\mathcal{A}} = \sum_{j=1}^r \vec{u}_j^{(1)} \otimes \vec{u}_j^{(2)} \otimes \cdots \otimes \vec{u}_j^{(N)} \quad (3.4)$$

where  $\vec{u}_j^{(n)}$  denotes the  $j^{th}$  column of matrix  $\mathbf{U}^{(n)}$  of size  $I_n \times r$ ,  $\|\mathcal{A} - \hat{\mathcal{A}}\|^2$  is minimized.

Recall in previous step, we obtain an approximate tensor  $\hat{\mathcal{A}}$  by step 7 of Algorithm with

$$\hat{\mathcal{A}} = \hat{\mathcal{S}} \times_1 \hat{\mathbf{U}}^{(1)} \times_2 \hat{\mathbf{U}}^{(2)} \times_3 \hat{\mathbf{U}}^{(3)} \times_4 \hat{\mathbf{U}}^{(4)} \quad (3.5)$$

where  $\hat{\mathbf{U}}^{(i)}$  ( $1 \leq i \leq 4$ ) is  $\mathbf{U}^{(i)}$  after removing the least significant column.

We use the PARAFAC model to carry out a tensor decomposition on the tensor  $\hat{\mathcal{S}}$  to



obtain a set of projection matrices  $\hat{\mathbf{P}}^{(i)}$  ( $1 \leq i \leq 4$ ) as shown below.

$$\begin{aligned}\hat{\mathbf{S}} &= \sum_{j=1}^R \vec{\hat{\mathbf{P}}}_j^{(1)} \otimes \vec{\hat{\mathbf{P}}}_j^{(2)} \otimes \dots \otimes \vec{\hat{\mathbf{P}}}_j^{(4)} \\ &= \hat{\mathbf{S}}' \times_1 \hat{\mathbf{P}}^{(1)} \times_2 \hat{\mathbf{P}}^{(2)} \times_3 \hat{\mathbf{P}}^{(3)} \times_4 \hat{\mathbf{P}}^{(4)}\end{aligned}\tag{3.6}$$

where  $\vec{\hat{\mathbf{P}}}_j^{(i)}$  denotes the  $j$  th column of matrix  $\hat{\mathbf{P}}^{(i)}$  and  $\hat{\mathbf{P}}^{(i)} \in \mathbb{R}^{c_i \times r}$  ( $1 \leq i \leq 4$ ), also we have a core tensor  $\hat{\mathbf{S}}' \in \mathbb{R}^{r \times r \times r \times r}$  and  $\hat{\mathbf{S}}'(i, j, k, l) = 1$  if and only if  $i = j = k = l$ .

In our example, the four projection matrices obtained are as follows:

$$\begin{aligned}\hat{\mathbf{P}}^{(1)} &= \begin{pmatrix} 0 & -0.01 \\ 1.61 & 0 \\ 0 & 1.41 \end{pmatrix} & \hat{\mathbf{P}}^{(2)} &= \begin{pmatrix} 1.51 & -0.01 \\ 0 & 1.41 \\ 0.45 & 0 \\ 0.07 & 0.01 \end{pmatrix} \\ \hat{\mathbf{P}}^{(3)} &= \begin{pmatrix} 0.76 & 0.81 \\ 0.74 & -0.19 \\ 0.18 & -0.84 \\ 0 & 0.27 \end{pmatrix} & \hat{\mathbf{P}}^{(4)} &= \begin{pmatrix} 0 & 1.41 \\ 1.61 & 0 \end{pmatrix}\end{aligned}$$

With the projection matrices, we can now map the latent feature matrices to a common space. Recall Equation (3.7), we replace the core tensor  $\hat{\mathbf{S}}$  with  $\hat{\mathbf{S}}'$  and obtain

$$\begin{aligned}\hat{\mathcal{A}} &= (\hat{\mathbf{S}}' \times_1 \hat{\mathbf{P}}^{(1)} \times_2 \hat{\mathbf{P}}^{(2)} \times_3 \hat{\mathbf{P}}^{(3)} \times_4 \hat{\mathbf{P}}^{(4)}) \times_1 \dots \\ &\quad \hat{\mathcal{U}}^{(1)} \times_2 \hat{\mathcal{U}}^{(2)} \times_3 \hat{\mathcal{U}}^{(3)} \times_4 \hat{\mathcal{U}}^{(4)}\end{aligned}\tag{3.7}$$

By **Property 1** and **Property 2**, we rearrange Equation (3.7) to get

$$\begin{aligned} \hat{\mathcal{A}} = & \hat{\mathcal{S}}' \times_1 (\hat{\mathcal{U}}^{(1)} \cdot \hat{\mathcal{P}}^{(1)}) \times_2 (\hat{\mathcal{U}}^{(2)} \cdot \hat{\mathcal{P}}^{(2)}) \times_3 \\ & (\hat{\mathcal{U}}^{(3)} \cdot \hat{\mathcal{P}}^{(3)}) \times_4 (\hat{\mathcal{U}}^{(4)} \cdot \hat{\mathcal{P}}^{(4)}) \end{aligned} \quad (3.8)$$

Let  $\hat{\mathcal{U}}'^{(i)} = \hat{\mathcal{U}}^{(i)} \cdot \hat{\mathcal{P}}^{(i)}$  ( $1 \leq i \leq 4$ ). By substituting  $\hat{\mathcal{U}}'^{(i)}$  into Equation (3.8), we have

$$\hat{\mathcal{A}} = \hat{\mathcal{S}}' \times_1 \hat{\mathcal{U}}'^{(1)} \times_2 \hat{\mathcal{U}}'^{(2)} \times_3 \hat{\mathcal{U}}'^{(3)} \times_4 \hat{\mathcal{U}}'^{(4)} \quad (3.9)$$

where  $\hat{\mathcal{U}}'^{(1)} \in \mathbb{R}^{U \times r}$ ,  $\hat{\mathcal{U}}'^{(2)} \in \mathbb{R}^{V \times r}$ ,  $\hat{\mathcal{U}}'^{(3)} \in \mathbb{R}^{R \times r}$  and  $\hat{\mathcal{U}}'^{(4)} \in \mathbb{R}^{T \times r}$

Table 3.6 shows the resultant mapped 2D space for users, tags, and items.

Table 3.6: Latent features of users, tags and items extracted.

Entity	Dimension 1	Dimension 2
$U_1$	0.01	1.09
$U_2$	0.22	0
$U_3$	0.39	0
$U_4$	0.96	0
Psychology	0	1.09
Comedy	0.88	0.01
Forrest Gump	0.96	0.77
Groundhog Day	0	0.45
Beautiful Mind	0.85	0
Toy Story	0	0.85

In this mapped space, we use the cosine similarity to compute the distance between a user  $u$  and a tag  $t \in T$ .

$$sim(u, t) = \frac{V_u \cdot V_t}{||V_u|| ||V_t||}$$

where  $V_u$  and  $V_t$  are the vectors for  $u$  and  $t$  in the 2-D space respectively.

Similarly, we can obtain the distance between an item  $i$  and a tag  $t$  as follows:

$$sim(i, t) = \frac{V_i \cdot V_t}{||V_i|| ||V_t||}$$

where  $V_i$  and  $V_t$  are the vectors for  $i$  and  $t$  in the 2-D space respectively.

Let us consider user  $U_3$  and item “*Groundhog Day*”. Since we have

$$\text{sim}(U_3, \text{“comedy”})=1$$

$$\text{sim}(U_3, \text{“psychology”})=0$$

$$\text{sim}(\text{“Groundhog Day”}, \text{“comedy”})=1$$

$$\text{sim}(\text{“Groundhog Day”}, \text{“psychology”})=0$$

we will characterize both  $u_3$  and “*Toy Story*” using the tag “*comedy*”. Once this is done, the recommender system can automatically generate the explanation for the ”why” question as follows: “This recommendation is made because your profile indicates a high preference for “*comedy*” books and “*Groundhog Day*” is a “*comedy*” book”.

To provide further insights on the recommendation, we categorize tags into three groups depending on how often they are associated with positively rated items, negatively rated items, or mixed rating by the user. The categorization of a tag  $t$  for a user  $u$  is obtained by computing the total tensor values for each rating over all the items.

$$\text{pref}(u, t) = \sum_{i \in V} \hat{\mathcal{A}}(u, i, t, \text{“like”}) - \sum_{i \in V} \hat{\mathcal{A}}(u, i, t, \text{“dislike”})$$

We say that  $t$  is a positive tag for  $u$  if  $\text{pref}(u, t) > 0$ , negative if  $\text{pref}(u, t) < 0$ , and neutral if  $\text{pref}(u, t) = 0$ .

### Adaptive Feedback Recommendation

After receiving the recommendations and the corresponding explanations, sometimes the users may find the recommendations unsuitable due to inaccurate profile descriptions. A novel feature of our recommendation framework is its ability to allow users to provide feedback and dynamically adjust the recommendation list based on the feedback. Back to our running example, suppose  $U_3$  is not happy with the recommendation of “*Groundhog Day*”. He/She is able to rate the recommendation “*Groundhog Day*” with the rating “*dislike*”. This is equivalent to changing the weight of the tensor element

$\mathcal{A}(U_3, \text{"Comedy"}, \text{"dislike"}, \text{"Groundhog Day"})$  to 1.

Alternatively, the user may choose to adjust his/her profile description to more accurately reflect his/her current interests. Figure 3-2(a) shows an interface that allows a user to adjust the weight of individual tag description in the profile description. Suppose a user  $u$  likes the artist “*Tim Allen*”, he can click on the thumb-up icon. If he does not like “*Tim Allen*”, he will click on the thumb-down icon. For each thumb-up on the tag  $t$ , we search for book  $m$  that has been tagged using  $t$  by users other than  $u$  and replace weight of the tensor element  $\mathcal{A}(u, t, \text{"like"}, m)$  by a small constant  $c = 1/q$  ( $q$  is the count of users other than  $u$  who tagged book  $m$  with tag  $t$ ). Similarly, for each thumb-down action, the weight of tensor element  $\mathcal{A}(u, t, \text{"dislike"}, m)$  will be replaced by a small constant  $c = 1/q$ .

With the updates of tensor elements, we need to re-compute the latent feature matrices. However, the high computational complexity of HOSVD [81] renders this approach infeasible for online application, especially the digitalization and variance updating step, i.e. Step 3 and Step 4 of Algorithm 1 in [81]. For most time-critical applications, when the change of the variance matrix is small, it is not worthy digitalizing that matrix.

Here, we address the problem of incrementally updating the latent feature matrices as new tensor elements are inserted over time in large volumes. Our idea is to continuously track the changes to the latent feature matrices  $\hat{\mathcal{U}}^{(i)}$  ( $1 \leq i \leq 4$ ) using the online PCA technique [53]. The details of our algorithm is presented in Algorithm 2. For each mode  $i$ , we read in a updated column of a matrix unfolding  $\mathbf{A}_{(i)}$ , denoted as  $\vec{X}$ . We adjust the latent feature matrix  $\hat{\mathcal{U}}^{(i)}$  by performing the following three steps for each dimension incrementally:

- Project  $\vec{X}$  onto  $\vec{\mathcal{U}}_j^{(i)}$  to obtain  $\mathbf{y}$  (Line 7)
- Estimate the reconstruction error ( $\vec{E}$ ) and the energy based on the  $\mathbf{y}$  values and  $\vec{\mathcal{U}}_j^{(i)}$  (Lines 8-9)
- Update  $\vec{\mathcal{U}}_j^{(i)}$  based on the estimate (Lines 10)

Intuitively, the goal is to adaptively update  $\hat{\mathbf{U}}^{(i)}$  ( $1 \leq i \leq 4$ ) quickly based on the new update. The larger the error  $\vec{E}$ , the more  $\vec{\mathbf{U}}_j^{(i)}$  is updated. However, the magnitude of this update should also take into account the past data currently "captured" by  $\hat{\mathbf{U}}^{(i)}$ . For this reason, the update is inversely proportional to the current energy  $\mathbf{S}^{(i)}$

The details of the algorithm is presented in Algorithm 2.

---

**Algorithm 2:** Online Tensor Analysis

---

**Input:**

Latent Feature Matrix  $\hat{\mathbf{U}}^{(i)}$  ( $1 \leq i \leq 4$ )

Energy Matrices  $\mathbf{S}^{(i)} \in \mathbb{R}^{c_i \times c_i}$

Tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$  ;

**Output:**

Updated Latent Feature Matrix  $\hat{\mathbf{U}}^{(i)}$  ( $1 \leq i \leq 4$ )

Updated Energy Matrices  $\mathbf{S}^{(i)} \in \mathbb{R}^{c_i \times c_i}$

Approximate Core tensor  $\hat{\mathbf{S}}$ ;

- 1: **for**  $i = 1$  **to** 4 **do**
- 2:   Calculate the matrix unfolding  $\mathbf{A}_{(i)}$  from tensor  $\mathcal{A}$ .
- 3:   **for** each updated column vector  $\vec{X}$  in  $\mathbf{A}_{(i)}$  **do**
- 4:      $\vec{X}' = \vec{X}$
- 5:     **for**  $j = 1$  **to**  $c_i$  **do**
- 6:       Let  $\vec{\mathbf{U}}_j^{(i)}$  denotes the  $j^{th}$  column of Matrix  $\hat{\mathbf{U}}^{(i)}$  and  $\mathbf{S}^{(i)}(j)$  be the  $j^{th}$  eigenvalue of  $\mathbf{S}^{(i)}$
- 7:        $\mathbf{y} = (\vec{\mathbf{U}}_j^{(i)})^T \vec{X}'$  (project  $X$  on to  $\vec{\mathbf{U}}_j^{(i)}$ )
- 8:        $\mathbf{S}^{(i)}(j) = \mathbf{S}^{(i)}(j) + \mathbf{y}^2$  (update the energy at  $j^{th}$  eigenvalue)
- 9:        $\vec{E} = \vec{X}' - \mathbf{y} \cdot \vec{\mathbf{U}}_j^{(i)}$  (calculate the approximation error)
- 10:        $\vec{\mathbf{U}}_j^{(i)} = \vec{\mathbf{U}}_j^{(i)} + \frac{\mathbf{y} \cdot \vec{E}}{\mathbf{S}^{(i)}(j)}$  (update the latent feature matrix)
- 11:        $\vec{X}' = \vec{X} - \mathbf{y} \vec{\mathbf{U}}_j^{(i)}$  (repeat with residual)
- 12:     **end for**
- 13:   **end for**
- 14: **end for**
- 15: Calculate the approximate core tensor  $\hat{\mathbf{S}}$  as follows:

$$\hat{\mathbf{S}} = \mathcal{A} \times_1 (\hat{\mathbf{U}}^{(1)})^T \times_2 (\hat{\mathbf{U}}^{(2)})^T \times_3 (\hat{\mathbf{U}}^{(3)})^T \times_4 (\hat{\mathbf{U}}^{(4)})^T$$


---

For our running example, suppose  $U_3$  changes

$\mathcal{A}(u_3, \text{"Comedy"}, \text{"dislike"}, \text{"Groundhog Day"})$  to 1 , the updated latent feature matrices are:

$$\hat{\mathcal{U}}^{(1)} = \begin{pmatrix} -0.74 & 0.59 \\ -0.33 & -0.74 \\ -0.59 & -0.33 \\ -0.32 & -0.31 \end{pmatrix} \quad \hat{\mathcal{U}}^{(2)} = \begin{pmatrix} 0 & 1.05 \\ 0.88 & 0 \\ 0 & 0.42 \\ 0.33 & 0 \end{pmatrix}$$

$$\hat{\mathcal{U}}^{(3)} = \begin{pmatrix} -0.74 & 0.52 & -0.43 \\ -0.63 & -0.76 & 0.17 \\ -0.23 & 0.40 & 0.89 \\ -0.28 & 0.53 & -0.73 \end{pmatrix} \quad \hat{\mathcal{U}}^{(4)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Based on the updated latent feature matrices, we compute the new approximate core tensor and the result is shown in Table 3.7.

Table 3.7: Output of the updated approximate tensor  $\hat{\mathcal{A}}$

User	Tag	Rating	Item	Val
$U_1$	psychology	like	Forrest Gump	1.13
$U_1$	psychology	like	Beautiful Mind	0.93
$U_1$	comedy	like	Forrest Gump	0.89
$U_2$	comedy	like	Forrest Gump	1.18
$U_2$	excellent	like	Groundhog Day	0.75
$U_2$	comedy	like	Groundhog Day	0.75
$U_3$	comedy	like	Forrest Gump	0.89
$U_3$	comedy	dislike	Toy Story	0.70
$U_4$	comedy	dislike	Forest Gump	0.82
$U_4$	comedy	dislike	Toy story	0.72
$U_4$	overrated	dislike	Toy story	0.78
$U_3$	psychology	like	Beautiful Mind	0.24
$U_3$	comedy	dislike	Toy story	0.51
$U_3$	overrated	dislike	Toy story	0.51

Once again, we map this tensor to a common basis as described in Section 3 and obtain Table 3.8.

Table 3.8: Updated Latent features of users, tags, items and ratings extracted.

Entity	Dimension 1	Dimension 2
$U_1$	0.05	1.12
$U_2$	0.77	0.12
$U_3$	0.52	0.27
$U_4$	0.22	0.17
Psychology	0	1.08
Comedy	1.13	0.01
Forrest Gump	0.74	0.56
Toy Story	0.52	0
Beautiful Mind	0	0.76
Groundhog Day	0.42	0

Hence, after the feedback from  $U_3$ , we have

$$\text{sim}(U_3, \text{"comedy"})=0.83$$

$$\text{sim}(U_3, \text{"psychology"})=0.37$$

$$\text{sim}(\text{"Beautiful Mind"}, \text{"comedy"})=0$$

$$\text{sim}(\text{"Beautiful Mind"}, \text{"psychology"})=1$$

The profile descriptions of both  $u_3$  and “*Beautiful Mind*” are now updated to tags “*comedy*” and “*psychology*”. Consequently, the Book “*Beautiful Mind*” will now be recommended to  $U_3$ .

### 3.4 Experimental Studies

We conduct experiments to evaluate the effectiveness of our proposed framework for item recommendation, item rating prediction and tag recommendation. We implemented our framework in MATLAB and run the experiments on a 2.33Ghz Intel Core 2 CPU with 4GB RAM, running Windows 7-64 bit.

**Experimental Dataset** We use the publicly available MovieLens dataset available at [http : //www.grouplens.org/node/73](http://www.grouplens.org/node/73). This dataset comprises of two files. The first file contains users’ tags on different movies. The second file contains users’ ratings on differ-

ent movies on a scale of 1 to 5, with 1 being bad and 5 being excellent. By joining these two files over user and movie, we obtain the quadruples  $\langle user, movie, tag, rating \rangle$ . We have a total of 24563 quadruples with 2,026 users, 5,088 movies, and 9,078 tags. We pre-process these quadruples to generate a subset such that each user, movie and tag occur at least 10 times in the dataset. The resulting dataset has 11122 tuples with 201 users, 501 movies, and 404 tags. Table 4.7 shows the statistics of the users' ratings after pre-processing.

We carried out three sets of experiments to evaluate our proposed approach. The first set of experiments evaluates the effectiveness of users' acceptance in terms of accuracy on item, user and tag recommendation task. The second set of experiments is a user study to demonstrate the effectiveness of users' acceptance in terms of explanation style. Finally, the third set of experiments show that updating the recommendation through user feedback is able to increase user acceptance in terms of accuracy.

Table 3.9: Statistics of rating data

Statistics	Users	Movies
Min. # of ratings	5	1
Max. # of ratings	203	58
Mean. # of ratings	$32.58 \pm 35.61$	$13.06 \pm 8.67$

### 3.4.1 Experiments on Users' Acceptance

We demonstrate five sets of experiments to show that our proposed approach increases the effectiveness of users' acceptance on item, user, tag recommendation task, explanation style and adaptive feedback scheme.

#### Experiments on Item Recommendation

We first evaluate the task of users' acceptance on item recommendation. We compare our method with the following existing methods:



1. **UPCC** [60]. This method uses the Pearson's Correlation Coefficient to cluster similar users and recommend items based on these similar users.
2. **IPCC** [66]. This method uses the Pearson's Correlation Coefficient to cluster similar items for recommendation.
3. **Probabilistic Matrix Factorization (PMF)** [63]. This is a state-of-the art collaborative filtering algorithm that utilizes the ternary relationship among user, item and ratings.
4. **Ternary Semantic Analysis (TSA)** [74]. This method recommends items based on the ternary semantic analysis on users-items-tags.

We use the Hit Ratio [30] as the metric to evaluate the users' acceptance on various item recommendation methods. For each user  $u \in U$ , we randomly choose one item  $i$  that has a rating of 5 and withhold the quadruples involving  $u$  and  $i$ . Then we run the 5 methods to generate the top N items recommended for this user. If the item  $i$  is among the top N recommended items, then we say that a hit has occurred. The hit ratio of a method is given by:

$$HitRatio = \frac{Number\ of\ hits}{|U|}$$

Figure 3-4 shows the hit ratio of the 5 methods as we vary N. We observe that the proposed QSA method has a higher hit ratio compared to the other methods. In particular, QSA outperforms TSA, PMF, IPCC and UPCC by more than 23%, 50%, 60% and 80% respectively. This is because QSA can find more accurate latent associations using quaternary relationships compared to ternary relationships of either users-items-ratings or users-items-tags. UPCC and IPCC find similar users or items (neighbors) by calculating Pearson correlation coefficient. If a user has few ratings for items, then it will be difficult for UPCC and IPCC to find neighbors. The PMF approach suffers from the data sparsity problem and is unable to extract sufficient feature information. On the other

hand, TSA captures user's interest (topic) by using tag, but does not judge how much he likes these topics (rating). By utilizing quaternary relations, the proposed QSA overcomes the data sparsity problem and captures both users' opinions and interests with the rating and tagging information.

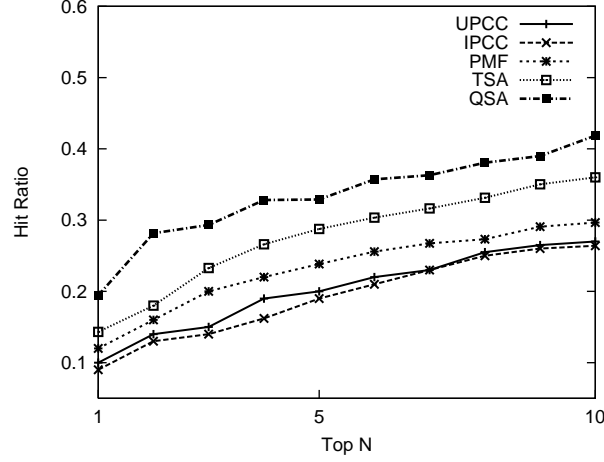


Figure 3-4: Hit ratio for Top N item recommendation

## Experiments on User Recommendation

For the task of users' acceptance on interesting user recommendation, we determine the similarity of items among the recommended top  $N$  users [74] since users with shared interests are more likely to tag and rate similar items. We compute the item similarity as the average of the cosine similarity of their  $TF \times IDF$  tag term vector [74] and cosine similarity of the rating vector [66].

Let  $NB_u$  be the set of top  $N$  users recommended to  $u$ . The intra-neighborhood similarity is given by the average cosine similarity of all items for the users in  $NB_u$ :

$$IntraSim(NB_u) = \frac{\sum_{w \in NB_u} \sum_{i \in I_u, j \in I_w} sim(i, j)}{\sum_{w \in NB_u} |I_u| |I_w|}$$

where  $I_u$  and  $I_w$  are the sets of items tags by users  $u$  and  $w$ .

Let  $Random_u$  be the set of  $N$  users randomly chosen from the set of users  $U - \{u\}$ .

We can determine the inter-neighborhood similarity as follows:

$$InterSim(Random_u) = \frac{\sum_{w \in Random_u} \sum_{i \in I_u, j \in I_w} sim(i, j)}{\sum_{w \in Random_u} |I_u| |I_w|}$$

where  $I_u$  and  $I_w$  are the sets of items tags by users  $u$  and  $w$  respectively.

Table 3.10: Comparison of intra- and inter- similarity between QSA and TSA

Method	<i>Intra – similarity</i>	<i>Inter – similarity</i>
TSA	0.10	0.08
QSA	<b>0.145</b>	<b>0.065</b>

Table 3.10 shows the intra-similarity and inter-similarity of QSA and TSA. We observe that the average intra-similarity is consistently higher than the average inter-similarity for both QSA and TSA. In particular, QSA outperforms TSA in intra-similarity indicating that more relevant users are found by QSA. Table 3.10 shows that the average of intra-similarity for QSA is about 0.15, while the average inter-similarity is only 0.065.

## Experiments on Tag Recommendation

For the task of users' acceptance on tag recommendation, we evaluate our algorithm QSA against the two state-of-the-art methods: TSA [74] and RTF [59]. For each user  $u \in U$ , we randomly choose one item  $i$  and remove all quadruples involving  $u$  and  $i$  from the dataset. Then we run the 3 methods to generate the top N tags recommended for this user.

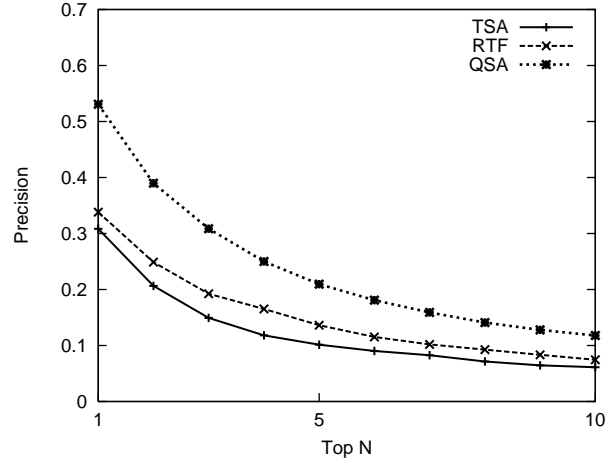
We use the standard recall and precision measures to evaluate the results:

$$Precision = \frac{Number\ of\ Hits}{N}$$

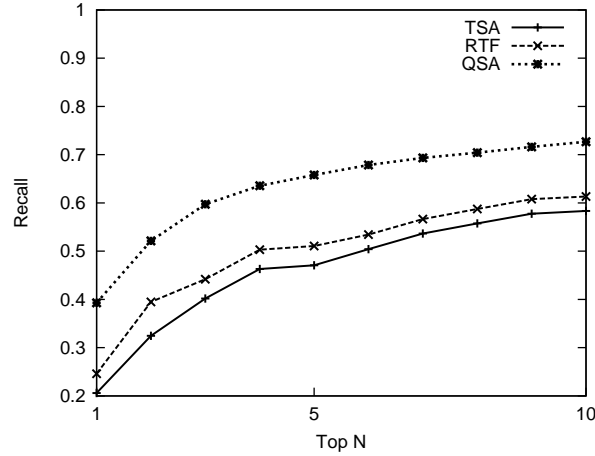
$$Recall = \frac{Number\ of\ Hits}{|T_{u,i}|}$$

where  $T_{u,i}$  is the set of tags used by user  $u$  on item  $i$ .

Figures 4-4(a) and 4-4(b) show the precision and recall of the 3 methods for varying values of  $N$ . It is clear that QSA is able to achieve a higher recall and precision compared to the other two methods.



(a)



(b)

Figure 3-5: Precision and recall for tag recommendation

### Experiments on Item Rating Prediction

For the task of users' acceptance on item recommendation, we also evaluate the predictive performance of QSA for item ratings. We compare QSA with UPCC, IPCC and

PMF only because TSA is based on user-item-tag relationship and does not use rating information.

We use the Mean Absolute Error (MAE) and Coverage as the evaluation metrics [4]. Coverage refers to the fraction of items that an algorithm is able to give a predicted rating. The MAE is given by:

$$MAE = \frac{\sum_{i \in T} |r_{u,i} - \hat{r}_{u,i}|}{|D|}$$

where  $r_{u,i}$  is the rating given by user  $u$  for item  $i$ ,  $\hat{r}_{u,i}$  is the predicted rating and  $D$  is the size of the testing dataset.

We use 80% of the dataset as training set and 20% as the testing set, and compute the MAE and coverage for different methods. The five-fold cross validation results are shown in Table 3.11. We observe that the coverage is not 100% for UPCC and IPCC, which confirms that these two methods are unable to deal with the problem of data sparsity effectively. On the other hand, QSA alleviates the data sparsity problem with the help of tagging information, thus achieving 100% coverage with a lower MAE.

To analyze the statistical significance of the results, we conduct a paired t-test. Let  $a_i, b_i, i = 1, 2, \dots, n$ , be the MAE values obtained using methods A and B respectively. Let  $d_i = a_i - b_i$  and  $\bar{d}$  be the average value of  $d_i, i = 1, 2, \dots, n$ . We set the null hypothesis as  $\bar{d} = 0$ . The *p-value* is computed using the t-statistics:

$$T = \frac{\bar{d}}{s/\sqrt{n}}$$

where  $s$  is the standard deviation of  $d$ . A *p-value* that is less than 0.01 indicates the existence of statistically significant evidence against the null hypothesis. We compare the results of QSA against UPCC, IPCC and PMF and obtain the p-values of 3.52E-06, 4.02E-06 and 1.70E-03 respectively. These results indicate that the improvement in the MAE values for QSA is statistically significant compared to UPCC, IPCC and PMF.

Table 3.11: MAE and Coverage

Method	MAE	Coverage
UPCC	0.7424	97.29%
IPCC	0.7458	98.05%
PMF	0.692	100%
QSA	<b>0.673</b>	<b>100%</b>

### Efficiency of Online Recommendation

In this section, we conduct experiments to compare the efficiency of incremental versus non-incremental algorithms for latent feature computation. We divide the data from Dec 2005 to Jan 2009 into 5 time points and measure the computational time spent on building the latent feature model at each time point. Figure 3-6 shows the runtime of both incremental and non-incremental algorithms. We observe that the runtime of the non-incremental algorithm increases over time as more data arrives. On the other hand, the runtime of the incremental algorithm does not increase over time. On average, the non-incremental algorithm takes thrice as long to build the latent feature model as compared to the incremental algorithm. This shows that the incremental algorithm is efficient and able to handle highly dynamic and continuously growing data.

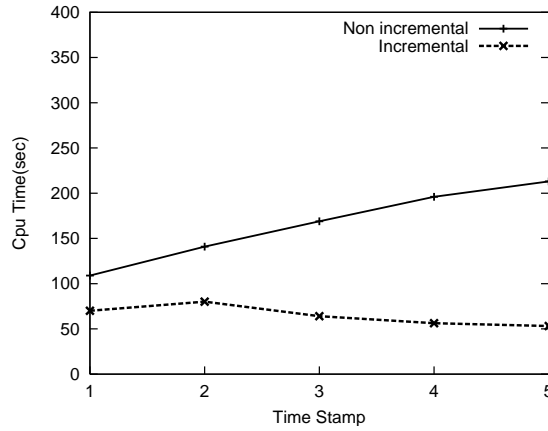


Figure 3-6: Run time at each time stamp for the incremental and non-incremental algorithms


## Experiment on Explanation Styles

In order to evaluate the user acceptance of the explanation style of items recommended, we conduct an extensive user study to compare 3 styles of explanations.

1. **Item-based Explanation.** This approach computes the *top-k* most similar items. It has the format "Item *X* is recommended, because you have tagged and rated items *Y*." The similarity between *X* and *Y* is computed based on the cosine similarity of the items' ratings, and is provided along with the recommendation.
2. **Feature-based Explanation.** This approach shows the features<sup>13</sup> of items recommended to the user. It has the format "Item *X* is recommended, because it contains features *a, b, ...*".
3. **TagCloud Explanation.** This approach uses a tagcloud to summarize the tags used for characterizing the user profile and the recommended item.

Table 3.12 shows the different explanations for the recommended item "Jurassic Park".

Table 3.12: Example explanations for recommended movie.

Method	Recommendation	Explanation
Item-based	Jurassic Park	<b>Because you tagged and rated</b> "I Heart Huckabees" (0.63), "1984" (0.62), "Saving Private Ryan" (0.56)
Feature-based	Jurassic Park	<b>Because it contains</b> Steven Spielberg, Sam Neill, Laura Dern, Dinosaur, Island, adventure, family and Sci-fi
TagCloud	Jurassic Park	

<sup>13</sup>The extraction of the features are done by joining with the Internet movie database (<http://www.imdb.com/interfaces>)

We have a total of 30 participants in our online user study. This study has two phases: Data Collection Phase and Evaluation Phase.

In the Data Collection Phase, the goal is to construct users' profile based on their rating and tagging activities. A user is asked to rate (on a scale of 1 to 5) and tag two sets of movies. The first set of movies contains the top-40 most popular movies in MovieLens. The second set of movies is selected from the top-200 movies in MovieLens. These movies have the highest variance in their user ratings. In order to ensure that there is a reasonable overlap in the tags used, users have to choose from a pre-determined list of tags.

In the Evaluation Phase, the system will show each user the 3 different styles of explanation that corresponds to a list of movies. We hide the movie title and ask the user to look at the information provided in the explanation to decide if this is the type of movie that he/she will like to watch. The ratings obtained based entirely on the information provided in the explanation is called the **explanation ratings** and the ratings obtained during the Data Collection Phase the **actual ratings**.

Table 3.13: Difference between explanation ratings and actual ratings

Explanation style	$\mu_d$	$\sigma_d$
Item-based	0.56	0.82
Feature-based	0.47	1.22
Tagcloud-based	0.03	0.73

The best explanation style is one that minimizes the difference between the explanation ratings and actual ratings [8]. Table 3.13 shows the results of the user study. We use  $\mu_d$  and  $\sigma_d$  to denote the mean and standard deviation of the differences between explanation ratings and actual ratings. It is clear that tagcloud-based explanation has the lowest mean and standard deviation compared to item-based and feature-based explanation, indicating that it is the most intuitive and easily understood by the users.

We also ask the participants to explicitly express their preferences for each explanation style for **user' explanation preference measurement**. The rating is performed on



a scale of 1 to 5 with 1 being the least preferred and 5 being the most preferred. Table 3.14 shows the survey results with  $\mu_q$  and  $\sigma_q$  denote the mean and standard deviation of the survey ratings respectively. We observe that the participants strongly preferred tagcloud-based explanation style.

Table 3.14: User ratings of preferred explanation style

Explanation style	$\mu_q$	$\sigma_q$
Item-based	2.56	0.91
Feature-based	2.17	0.86
Tagcloud-based	4.06	0.74

### Experiment on Adaptive Feedback

In this set of experiments, we evaluate the effectiveness of users' feedback in improving user acceptance of our recommended items. Users are given a list of top 10 recommendations by our system, and asked to provide ratings for these items. They are allowed to give feedback on the recommendations that they are not satisfied with. After each feedback, the system will generate a new list of top 10 recommendation list to the user. We repeat these rounds of feedback up to 5 times.

We use the following evaluation metrics.

1. Mean Absolute Error (MAE) which is given by

$$MAE = \frac{\sum_{i \in Items} 5 - \hat{r}_{u,i}}{N}$$

where  $\hat{r}_{u,i}$  is the rating given by user  $u$  for item  $i$  and  $N$  is the number of items recommended.

2. Precision measures the proportion of the correctly recommended items for top- $N$  items and is defined as

$$Precision(N) = \frac{\text{Number of items with rating 5}}{N}$$

3. Average precision (AP) is the average of the precision for top- $k$  items,  $1 \leq k \leq N$ .

$$AP = \frac{\sum_{k=1}^N Precision(k)}{N}$$

Table 3.15 shows the results for each round of feedback. We observe a **9%** improvement in MAE, **13.8%** improvement in Precision(10) and **41 %** improvement in AP between round 1 and round 5.

Note that 50 % of the users stop giving feedback after round 2, implying that they are satisfied with the recommended items. The results obtained in round 3 is comparable to round 5, indicating that we are able to achieve near optimal values in just a few rounds of feedback.

Table 3.15: Results of User Feedback

	round 1	round 2	round 3	round 4	round 5
MAE	1.04	1.06	0.91	1	0.94
P(10)	0.36	0.37	0.41	0.41	0.41
AP	0.25	0.31	0.34	0.35	0.36

In addition, we observe that MAE and AP increase slightly in round 4 while Precision(10) remains unchanged. With Precision(10) remains unchanged, we know that the total number of hits is the same in round 3, 4 and 5. The increase in AP suggests that more relevant movies are recommended at the top in each round. Closer examination of the data reveals that in round 4, a less well-known movie has been recommended to the users. Due to the unfamiliarity of this movie, many users give a 3 star rating to this movie. As a result, MAE has increased slightly.

### 3.4.2 Sensitivity Experiments

We also conduct experiments to study the effect of core tensor dimensions  $c_1, c_2, c_3$ , and  $c_4$  on the performance of our algorithm QSA. We first vary each dimension to find the settings that give the best performance. This occurs when  $c_1 = 45, c_2 = 125, c_3 = 165, c_4 = 4$ .

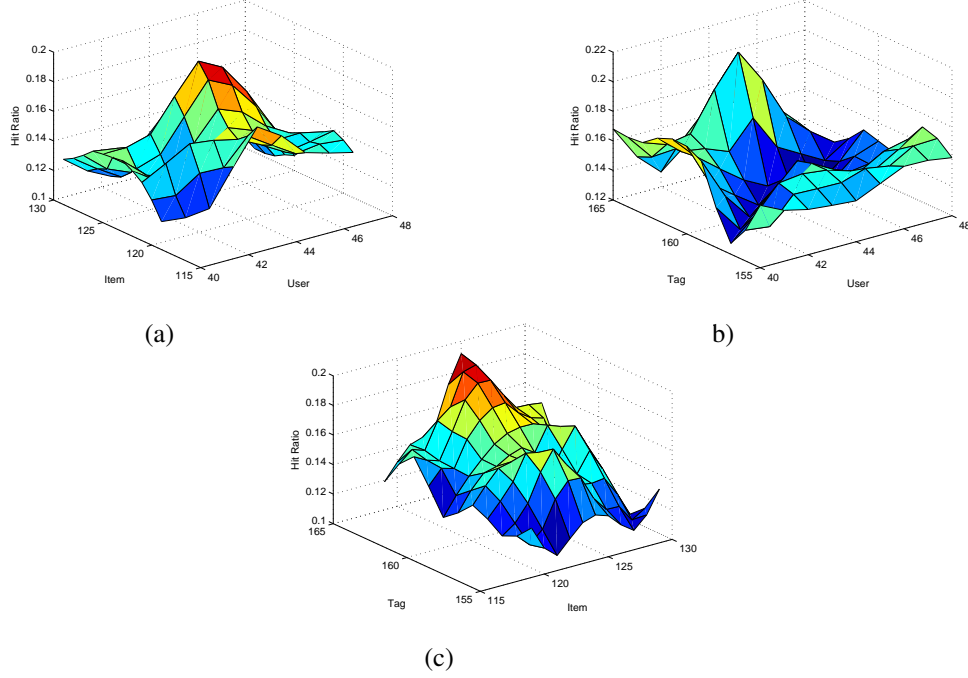


Figure 3-7: Effect of core tensor dimensions on hit ratio

For ease of visualization, we vary two of the four dimensions and keep the other two dimensions fixed at their optimal values. The results are shown in Figure 3-7. Figure 3-7(a) shows the effect on hit ratio as we vary  $c_1$  and  $c_2$  while keeping  $c_3$  fixed at 165 and  $c_4$  fixed at 4. Figure 3-7(b) shows the effect on hit ratio as we vary  $c_1$  and  $c_3$  while keeping  $c_2$  fixed at 125 and  $c_4$  fixed at 4. Figure 3-7(c) shows that results of varying  $c_2$  and  $c_3$  while  $c_1$  and  $c_4$  are fixed at 45 and 4 respectively. From the figures, we observe that a good approximation of the original diagonal can be achieved by preserving 70%, 90%, 80%, 90% of the original tensor information in each dimension respectively, that is,  $c_1 = 45, c_2 = 125, c_3 = 165, c_4 = 4$ .

### 3.5 Summary

In this chapter, we have shown that quaternary semantic analysis can lead to more accurate recommendation. We have proposed using a 4-order tensor to model the four heterogeneous entities: users, items, tags and ratings. We further employed the higher order

singular value decomposition to reduce the dimensionality of the 4-order tensor, thereby casting the recommendation problem as a multiway latent semantic analysis problem. Extensive experiments have been conducted on a real world dataset for item recommendation, user recommendation, tag recommendation, and item rating prediction. The results demonstrated that quaternary semantic analysis outperforms state-of-the-art algorithms in all the four tasks.



---

---

## CHAPTER 4

---

# IMPROVING USERS' ACCEPTANCE USING CROSS DOMAIN DATA

Collaborative Filtering techniques purely rely on the observed rating/tagging data, the sparsity problem has become a major bottleneck for CF methods. In real-world scenarios, we can easily find related CF domains that recommend similar items with the target one. For example, movies, books, and music are related in entertainment; mobile phones, notebook PCs, and digital cameras are related in electronic products. Can we establish a bridge between related CF domains and transfer useful knowledge from one another to improve the performance? Since movies and books are somewhat related (they have some correspondence in genre and the users of the both rating websites may reflect similar social aspects [16]), we believe that the users' behaviors in different domains can share similar patterns. Thus, transfer of knowledge can be beneficial. Furthermore, this knowledge-transfer idea can be generalized to any related real-world domains. Besides the rating and social tagging data, it is necessary to incorporate cross domain data to better understand users' interest and make better recommendation. In this chapter, we extend the work of previous chapter to improving the users' acceptance

by the help of related cross domain information.

## 4.1 Motivation

With the increasing popularity of social media communities, we now have data repositories from various domains such as user-item-tag data from social tagging in book and movie domains, and friendship data between users in social networks. The joint analysis of information from various domains and social networks has the potential to improve our understanding of the underlying relationships among users, items and tags and increase user acceptance in recommender systems.

For example, users who like to read *romance* books generally have similar preferences as users who like to watch *romance* movies. By learning the characteristics of *romance* lovers from the Movie domain and transferring the learned characteristics to the Book domain, recommender systems can predict users' preferences more accurately and provide more customized recommendations.

Recent works [52, 40] apply transfer learning methods to utilize data in some auxiliary domain such as Movie domain, and transfer knowledge that are consistent in this domain to a target domain such as Book domain. However, they are limited to transferring only binary relationships, e.g. user-item, in the form of matrices. Shi et al. [70] use tags as a bridge for cross domain transfer by decomposing the ternary user-tag-item relation into two binary relations user-tag and item-tag. Unfortunately, these decomposition is lossy and may lead to inaccurate recommendations. Thus, we advocate that recommendation using cross domain data should be carried out without decomposition.

Another major source of information that has yet to be fully utilized is that of social network data. Researchers have proposed to use data from the social network domain to increase user acceptance in recommender systems [28, 47]. The assumption is that the social network structure is useful for predicting users' preferences because users' interests may be affected by their friends. However, this assumption is not realistic as it

implies that if two users, say  $u_i$  and  $u_j$ , are friends, then  $u_i$  will be influenced by  $u_j$  on all topics/aspects.

In this chapter, we propose a tensor factorization based framework to fuse knowledge from different domains. We design a topic-based social trust regularization to integrate social network information with cross domain data. Our contributions are as follows:

- For cross domain data, we construct a shared three dimensional cluster level tensor as a bridge to uncover the hidden knowledge between the target domain and auxiliary domain. In particular, we extend tensor factorization to the setting of transfer learning.
- For social network information, we construct a shared users' latent feature space and design a topic based social trust regularization model, which has not been well studied in cross domain recommender systems.
- Experiments on real world datasets demonstrate the effectiveness of using multiple domains and social network for recommendation.

To the best of our knowledge, this is the first study that combines cross domain recommendation and social network in a unified framework. The rest of this chapter is organized as follows. Section 4.2 gives the problem formulation. Section 4.3 describes the cross domain framework. Section 4.4 presents the experimental results. We summarize in Section 4.5.

## 4.2 Problem Formulation

A tensor is a multidimensional array. An  $N$ -order tensor  $\mathcal{A}$  is denoted as  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N}$  with elements  $a_{i_1, \dots, i_n}$  and dimensions  $I_1, I_2, \dots, I_N$ . Let the target domain dataset be a list of tuples  $\langle u, t, v \rangle$  denoting that a user  $u$  tags an item  $v$  with tag  $t$ . We model this target domain dataset as a 3-order tensor  $\mathcal{A}_{tgt} \in U_{tgt} \times T_{tgt} \times V_{tgt}$ , where  $U_{tgt}$  is the set of users,



$T_{tgt}$  is the set of tags, and  $V_{tgt}$  is the set of items/resources.  $\mathcal{A}_{tgt}(u, t, v)$  has a value of 1 if the tuple  $\langle u, t, v \rangle$  exists, otherwise it has a value of 0.

For example, let's consider the Table 4.1 for simplicity. we can model it the tagging activities of users in Table 4.1 as a 3-order tensor  $\mathcal{A}$  with dimensions  $5 \times 5 \times 5$ . The entry  $\mathcal{A}(1,1,1)$  has a value of 1 since it corresponds to the tuple  $\langle U_5, 'fantasy', 'New moon' \rangle$  which is found in Table 4.1. On the other hand, the entry  $\mathcal{A}(1,3,1)$  has a value of 0 since its corresponding tuple  $\langle U_5, 'drama', 'New moon' \rangle$  does not exist in Table 4.1.

Table 4.1: Book domain dataset

User	Tag	Item
$U_5$	fantasy	New moon
$U_6$	romance	New moon
$U_7$	drama	Good omens
$U_8$	action	James Bonds Girls
$U_9$	action	Ghost rider
$U_9$	action	James Bonds Girls
$U_9$	adventure	Scorpia

Similarly, we model the dataset in the auxiliary domain such as movie domain in Table 4.2 as  $\mathcal{A}_{aux} \in U_{aux} \times T_{aux} \times V_{aux}$ . Note that our proposed approach can handle the case when  $U_{tgt} \cap U_{aux} = \emptyset$  and/or  $V_{tgt} \cap V_{aux} = \emptyset$ .

Table 4.2: Ternary relations among users, tags, and items in Movies Domain

User	Tag	Item
$U'_1$	fantasy	Twilight
$U'_1$	romance	Twilight
$U'_1$	drama	Big Daddy
$U'_2$	fantasy	Spider man
$U'_2$	adventure	Spider man
$U'_2$	action	Iron Man
$U'_3$	drama	Big Daddy
$U'_3$	comedy	Little man
$U'_4$	action	Iron Man
$U'_4$	action	Star war
$U'_5$	adventure	Die hard
$U'_5$	adventure	Braveheart

At the same time, suppose the users in the target domain are connected to each other

via some social network. We model the user connections as a  $U_{tgt} \times U_{tgt}$  trust matrix,  $\mathbf{F} = [f_{u,w}]$  where  $u, w \in U_{tgt}$  and  $f_{u,w} \in [0, 1]$  denotes the degree of social trust that  $u$  has on  $w$ . A value of 0 implies  $u$  does not trust  $w$  while a value of 1 suggests that  $u$  trusts  $w$  completely.

We formulate the recommendation problem as a tensor missing value prediction problem. The goal is to generate a ranked list of users/items/tags based on the predicted value in the tensor. Here, we show how to extract the informative, yet compact cluster-level tensor (knowledge we want to transfer) from the auxiliary domain along with the mappings of users, items and tags between target and auxiliary domains, and the social trust knowledge in the target domain to enable better prediction results in the target domain. In other words, we want to predict the missing values in  $\mathcal{A}_{tgt}$  with knowledge from  $\mathcal{A}_{aux}$  and the trust matrix  $\mathbf{F} = [f_{u,w}]$ .

Let  $\mathcal{A}_{tgt}^*$  be the tensor obtained. Based on  $\mathcal{A}_{tgt}^*$ , we can use it to perform the following recommendation tasks.

- Tag recommendation. This is to find the top-N tags that user  $u$  is most likely to use for an item  $v$  and can be derived from

$$\operatorname{argmax}_{t \in T_{tgt}}^N [\mathcal{A}_{tgt}^*]_{u,t,v}$$

- Item recommendation. This task recommends the top-N items for user  $u$  based on the set of tags  $T_u$  s/he has used previously. The top-N items is determined from

$$\operatorname{argmax}_{v \in V_{tgt}}^N \sum_{t \in T_u} [\mathcal{A}_{tgt}^*]_{u,t,v}$$

- User recommendation. This task recommends the top-N most likely friends for user  $u$  as follows:

$$\operatorname{argmax}_{u' \in \{U_{tgt} - \{u\}\}}^N \sum_{(u,t,v) \in \mathcal{A}_{tgt}} [\mathcal{A}_{tgt}^*]_{u',t,v}$$

## 4.3 Cross Domain Framework

In this section, we first describe our approach to establish a bridge from the auxiliary domain to the target domain. Then we present our framework to fuse the social network information and the cross domain data to generate recommendations.

### 4.3.1 Cluster-Level Tensor

The key to a successful knowledge transfer from the auxiliary domain to the target domain lies in extracting the appropriate information from the auxiliary domain and establishing a mapping from the extracted knowledge back to the target domain. Here, the knowledge we want to extract are groupings of users, items, and tags that have similar characteristics. Our proposed method will construct a cluster tensor in the auxiliary domain. Then we will map the users, tags and items in the target domain to the clusters in the auxiliary domain.

We first perform a PARAFAC tensor decomposition on the auxiliary tensor  $\mathcal{A}_{aux}$ . This decomposition maps users, items and tags into a shared latent feature space. In this shared space, we perform clustering to obtain groups of similar users, items, and tags.

The PARAFAC tensor decomposition for a tensor  $\mathcal{A}$  of size  $I_1 \times I_2 \cdots \times I_N$  with an input rank  $R$  is defined as [36]:

$$\hat{\mathcal{A}} \approx \sum_{j=1}^R [\hat{\mathbf{U}}^{(1)}]_{*j} \circ [\hat{\mathbf{U}}^{(2)}]_{*j} \circ \cdots \circ [\hat{\mathbf{U}}^{(N)}]_{*j}$$

where  $\hat{\mathbf{U}}^{(n)}$  of size  $I_n \times R$  for  $n = 1, \dots, N$  and  $[\hat{\mathbf{U}}^{(i)}]_{*j}$  denotes the  $j^{th}$  column of matrix  $\hat{\mathbf{U}}^{(n)}$  and  $\|\mathcal{A} - \hat{\mathcal{A}}\|_F^2$  is minimized.  $\|\cdot\|_F^2$  is the square frobenius norm. It is defined as  $\|\mathcal{A}\|_F^2 = \sum_{i_1=1}^{R_1} \cdots \sum_{i_N=1}^{R_N} \mathcal{A}(i_1 \cdots, i_N)^2$ .  $\circ$  is the outer product between vectors. The entry  $\mathcal{A}(i_1, \cdots, i_N)$  is equal to  $\sum_{j=1}^R [\hat{\mathbf{U}}^{(1)}]_{i_1 j} \times [\hat{\mathbf{U}}^{(2)}]_{i_2 j} \cdots \times [\hat{\mathbf{U}}^{(N)}]_{i_N j}$

For the Movie dataset in Table 4.2, the PARAFAC tensor decomposition factorizes the auxiliary tensor  $\mathcal{A}_{aux}$  in the form of the latent feature representation  $\hat{\mathbf{U}}^{(i)}$  ( $1 \leq i \leq 3$ )

as follows:

$$\mathcal{A}_{aux} \approx \sum_{j=1}^{R=5} [\hat{U}^{(1)}]_{*j} \circ [\hat{U}^{(2)}]_{*j} \circ [\hat{U}^{(3)}]_{*j}$$

where  $[\hat{U}^{(i)}]_{*j}$  denotes the  $j^{th}$  column of matrix  $\hat{U}^{(i)}$ , and  $\hat{U}^{(1)} \in \mathbb{R}^{|U_{aux}| \times 5}$ ,  $\hat{U}^{(2)} \in \mathbb{R}^{|T_{aux}| \times 5}$ , and  $\hat{U}^{(3)} \in \mathbb{R}^{|V_{aux}| \times 5}$ .

The projection matrices  $\hat{U}^{(i)}$  ( $1 \leq i \leq 3$ ) obtained for the Movie dataset are as follows:

$$\hat{U}^{(1)} = \begin{pmatrix} 0 & 0.53 & 0 & 1 & 0 \\ 0.53 & 0 & 1 & 0 & 0 \\ 0 & 0.85 & 0 & 0 & 0 \\ 0.85 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \hat{U}^{(2)} = \begin{pmatrix} 0 & 0 & 0.71 & 0.71 & 0 \\ 0 & 0 & 0 & 0.71 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.71 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \hat{U}^{(3)} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0.85 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0.85 & 0 & 0 & 0 & 0 \\ 0 & 0.52 & 0 & 0 & 0 \\ 0.52 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.71 \\ 0 & 0 & 0 & 0 & 0.71 \end{pmatrix}$$

Based on the projection matrices, we apply some existing clustering algorithm to cluster the users, items, and tags. Table 4.3 shows the clusters obtained.

Table 4.3: Clusters for the Movie domain in Table 4.2

(a) Users		(b) Tags	
Cluster ID	Cluster	Cluster ID	Cluster
User-G1	{ $U'_1$ }	Tag-G1	{ fantasy }
User-G2	{ $U'_2$ }	Tag-G2	{ romance }
User-G3	{ $U'_3$ }	Tag-G3	{ drama }
User-G4	{ $U'_4$ }	Tag-G4	{ adventure }
User-G5	{ $U'_5$ }	Tag-G5	{ action }

(c) Items	
Cluster ID	Cluster
Item-G1	{ Twilight }
Item-G2	{ Big Daddy, Little man }
Item-G3	{ Spider man }
Item-G4	{ Iron man, Star war }
Item-G5	{ Die hard, Braveheart }

With this, we replace the ids of user, item and tags in the auxiliary dataset with their

Table 4.4: Cluster-level tensor in Movie domain.

User	Tag	Item	Val
User-G1	Tag-G1	Item-G1	0.5
User-G1	Tag-G2	Item-G1	0.5
User-G1	Tag-G3	Item-G2	0.5
User-G2	Tag-G1	Item-G3	0.5
User-G2	Tag-G4	Item-G3	0.5
User-G2	Tag-G5	Item-G4	0.5
User-G3	Tag-G3	Item-G2	1
User-G4	Tag-G5	Item-G4	1
User-G5	Tag-G4	Item-G5	1

respective cluster id to obtain a cluster-level tensor, denoted as  $\mathcal{A}_{aux}^{cluster} \in \mathbb{R}^{R \times R \times R}$ . Table 4.4 shows the cluster-level tensor obtained for the Movie dataset. The *Val* column is the normalized count of the duplicate tuples obtained after replacing the ids. We use this tensor to transfer the knowledge from the auxiliary domain (Movie) to the target domain (Book).

Transferring knowledge from  $\mathcal{A}_{aux}$  to  $\mathcal{A}_{tgt}$  is achieved through a reverse process of summarization in the auxiliary domain. By assuming that there exists implicit correspondence between the user/ tag/item group of the auxiliary domain and those of the target domain. Based on the cluster-level tensor and the correspondence of user/tag/item group, we reconstruct the tensor  $\mathcal{A}_{tgt}^*$  as follows:

$$\mathcal{A}_{tgt}^* = \mathcal{A}_{aux}^{cluster} \times_1 \hat{U}_{tgt}^{(1)} \times_2 \hat{U}_{tgt}^{(2)} \times_3 \hat{U}_{tgt}^{(3)} \quad (4.1)$$

where  $\hat{U}_{tgt}^{(1)} \in \mathbb{R}^{|U_{tgt}| \times R}$ ,  $\hat{U}_{tgt}^{(2)} \in \mathbb{R}^{|T_{tgt}| \times R}$ , and  $\hat{U}_{tgt}^{(3)} \in \mathbb{R}^{|V_{tgt}| \times R}$  are user latent feature matrix, tag latent feature matrix and item latent feature matrix which we want to learn respectively.  $\times_n$  is n-mode product. The **n-mode product** of a tensor  $\mathcal{A} = \mathbb{R}^{I_1 \times \dots \times I_N}$  by a matrix  $U = \mathbb{R}^{J_n \times I_n}$ , denoted by  $\mathcal{A} \times_n U$ , is a  $(I_1 \times I_2 \dots I_{n-1} \times J_n \times I_{n+1} \dots I_N)$ -tensor where the

entries are given by [36]:

$$\begin{aligned}
& (\mathcal{A} \times_n \mathbf{U})_{i_1 i_2 i_3 \dots i_{n-1} j_n i_{n+1} \dots i_N} \\
&= \sum_{i_n} a_{i_1 i_2 i_3 \dots i_{n-1} i_n i_{n+1} \dots i_N} \cdot u_{j_n i_n}
\end{aligned}$$

To compute the optimal  $\mathcal{A}_{tgt}^*$  for recommendation, we need to find the  $\hat{\mathbf{U}}_{tgt}^{(i)}$  ( $1 \leq i \leq 3$ ) such that the difference between the observed tensor  $\mathcal{A}_{tgt}$  and the reconstructed tensor  $\mathcal{A}_{tgt}^*$  is minimized, that is,

$$\min_{\hat{\mathbf{U}}_{tgt}^{(1)} \dots \hat{\mathbf{U}}_{tgt}^{(3)}} \|\mathcal{A}_{tgt} - \mathcal{A}_{tgt}^*\|_F^2 \quad (4.2)$$

Table 4.5: Mapping between Book and Movie domains.

(a) Users			(b) Tags		
User	Cluster ID	Weight	Tag	Cluster ID	Weight
$U_5$	User-G1	0.2	fantasy	Tag-G1	0.3
$U'_6$	User-G1	0.61	romance	Tag-G2	2.68
$U'_7$	User-G3	1.8	drama	Tag-G3	0.50
$U'_8$	User-G4	0.46	adventure	Tag-G4	1.47
$U'_9$	User-G2	1.5	action	Tag-G5	1.24

(c) Items		
Item	Cluster ID	Weight
New moon	Item-G1	1.22
Good omens	Item-G2	1
Scorpia	Item-G3	0.89
James Bonds Girls	Item-G4	1.24
Ghost rider	Item-G4	0.76

Table 4.5 shows the correspondence  $\hat{\mathbf{U}}_{tgt}^{(i)}$  ( $1 \leq i \leq 3$ ) between the users, items and tags in the Book domain and the user, item, tag clusters in the Movie domain. The *Weight* column indicates how similar a user/item/tag is to the cluster.

Suppose we want to recommend some books to user  $U_5$  in Table 4.1. User  $u_1$  in the Movie domain forms a cluster User-G1. From Table 4.5, we observe that the mapping between user  $U_5$  and cluster User-G1 has a weight of 0.2, indicating that  $U_5$  has similar interests as the users in cluster User-G1. Since users in User-G1 like drama movies 'Big

Daddy’, we may infer that user  $U_5$  may also like drama books and thus recommend the book ‘Good Omens’ to  $U_5$ .

### 4.3.2 Fusing Social Network Information

Besides cross domain data, another valuable source of information is the social network information. Existing works on social recommendations [28][47] are all based on the assumption that friends in the social network will have similar interests in all topics and areas. They incorporate such a network-based similarity property among users to regulate the latent factor modeling as follows.

$$\begin{aligned}
h &= \sum_{i=1}^N \sum_{j=1}^N F_{ij} \| [\hat{\mathbf{U}}_{tgt}^{(1)}]_{i*} - [\hat{\mathbf{U}}_{tgt}^{(1)}]_{j*} \|^2 \\
&= \sum_{i=1}^N \sum_{j=1}^N F_{ij} \left[ \sum_{r=1}^R ([\hat{\mathbf{U}}_{tgt}^{(1)}]_{ir} - [\hat{\mathbf{U}}_{tgt}^{(1)}]_{jr})^2 \right] \\
&= \sum_{r=1}^R [\hat{\mathbf{U}}_{tgt}^{(1)}]_{*r}^T (\mathbf{D} - \mathbf{F}) [\hat{\mathbf{U}}_{tgt}^{(1)}]_{*r} \\
&= \text{tr}([\hat{\mathbf{U}}_{tgt}^{(1)}]^T (\mathbf{D} - \mathbf{F}) \hat{\mathbf{U}}_{tgt}^{(1)})
\end{aligned} \tag{4.3}$$

where  $F_{ij}$  is the similarity between users  $u_i$  and  $u_j$  (defined in terms of either Vector Space Similarity (VSS) or Person Correction Coefficient (PCC) [47]),  $N$  is the number of users in target domain, and  $\mathbf{D}$  is a diagonal matrix whose diagonal elements  $D_{ii} = \sum_j F_{ij}$  and  $\text{tr}(\cdot)$  denotes the trace of a matrix. The terms targets to minimize the difference between the latent vectors of  $[\hat{\mathbf{U}}_{tgt}^{(1)}]_{i*} \in \mathbb{R}^{1 \times R}$  and her or his friend  $[\hat{\mathbf{U}}_{tgt}^{(1)}]_{j*} \in \mathbb{R}^{1 \times R}$  for all  $r$  topics ( $1 \leq r \leq R$ ) with same weight.

Here, we want to differentiate user interest based on topics. We define a similarity matrix  $\mathbf{F}^{(r)}$  for each topic  $r$  ( $1 \leq r \leq R$ ), where  $R$  is the dimension for the users’ latent feature  $\hat{\mathbf{U}}_{tgt}^{(1)}$ . If users  $i$  and  $j$  are friends, then we define their similarity on a topic  $r$ ,

denoted by  $\mathbf{F}_{ij}^{(r)}$ , as follows:

$$\mathbf{F}_{ij}^{(r)} = \frac{[\hat{\mathbf{U}}_{tgt}^{(1)}]_{ir}[\hat{\mathbf{U}}_{tgt}^{(1)}]_{jr}}{\sqrt{\sum_{k=1}^R [\hat{\mathbf{U}}_{tgt}^{(1)}]_{ik}} \sqrt{\sum_{k=1}^R [\hat{\mathbf{U}}_{tgt}^{(1)}]_{jk}}}$$

Otherwise their similarity  $\mathbf{F}_{ij}^{(r)} = 0$ .

We introduce the topic-based similarity function into the latent factor model and modify Eq (4.3) to the following:

$$\begin{aligned} h &= \sum_{i=1}^N \sum_{j=1}^N \sum_{r=1}^R \mathbf{F}_{ij}^{(r)} \| [\hat{\mathbf{U}}_{tgt}^{(1)}]_{ir} - [\hat{\mathbf{U}}_{tgt}^{(1)}]_{jr} \|^2 \\ &= \sum_{i=1}^N \sum_{j=1}^N \mathbf{F}_{ij}^{(r)} \left[ \sum_{r=1}^R ([\hat{\mathbf{U}}_{tgt}^{(1)}]_{ir} - [\hat{\mathbf{U}}_{tgt}^{(1)}]_{jr})^2 \right] \\ &= \sum_{r=1}^R [\hat{\mathbf{U}}_{tgt}^{(1)}]_{*r}^T (\mathbf{D}^{(r)} - \mathbf{F}^{(r)}) [\hat{\mathbf{U}}_{tgt}^{(1)}]_{*r} \\ &= \text{tr}([\hat{\mathbf{U}}_{tgt}^{(1)}]^T (\mathbf{D}^{(r)} - \mathbf{F}^{(r)}) \hat{\mathbf{U}}_{tgt}^{(1)}) \end{aligned} \quad (4.4)$$

where  $N$  is the number of users in the target domain,  $\mathbf{D}^{(r)}$  is a topic-based diagonal matrix whose diagonal elements  $\mathbf{D}^{(r)}_{ii} = \sum_j \mathbf{F}_{ij}^{(r)}$  and  $\text{tr}(\cdot)$  denotes the trace of a matrix.

By combining Equations (4.2) and (4.4), we obtain the objective function for minimization:

$$\begin{aligned} f &= \min_{\hat{\mathbf{U}}_{tgt}^{(1)} \dots \hat{\mathbf{U}}_{tgt}^{(3)}} \|\mathcal{A} - \mathcal{A}_{tgt}^*\|_F^2 \\ &\quad + \lambda \cdot \sum_{r=1}^R \text{tr}([\hat{\mathbf{U}}_{tgt}^{(1)}]^T (\mathbf{D}^{(r)} - \mathbf{F}^{(r)}) \hat{\mathbf{U}}_{tgt}^{(1)}) \end{aligned} \quad (4.5)$$

Equation (4.5) can be reduced to a non-negative tensor factorization problem with regularization [1]. We derive the multiplicative updating rules for  $\hat{\mathbf{U}}_{tgt}^{(i)}$  ( $1 \leq i \leq 3$ ) as follows:

$$[\hat{\mathbf{U}}_{tgt}^{(1)}]_{*r} \leftarrow [\hat{\mathbf{U}}_{tgt}^{(1)}]_{*r} \otimes \frac{[\mathbf{A}_{(1)}[\hat{\mathbf{S}}_{(1)}^A]^T]_{*r} + \lambda \mathbf{F}^{(r)}[\hat{\mathbf{U}}_{tgt}^{(1)}]_{*r}}{[\hat{\mathbf{U}}_{tgt}^{(1)}\hat{\mathbf{S}}_{(1)}^A([\hat{\mathbf{S}}_{(1)}^A]^T]_{*r} + \lambda \mathbf{D}^{(r)}[\hat{\mathbf{U}}_{tgt}^{(1)}]_{*r}} \quad (4.6)$$



---

**Algorithm 3:** FUSE
 

---

**Input:**

 List of tuples <users, tags, items>;  $\lambda$ ;

 Cluster-level tensor  $\mathcal{A}_{aux}^{cluster} \in \mathbb{R}^{R \times R \times R}$ 
**Output:**

 Tensor  $\mathcal{A}_{tgt}^*$ ;

- 1: Initialization: From the tuple (users, items tag), we construct tensor  $\mathcal{A}_{tgt} \in \mathbb{R}^{|U| \times |T| \times |V|}$ , where  $|U|$ ,  $|V|$  and  $|T|$  are the number of users, items and tags respectively
  - 2: Random initialize  $\left[\hat{U}_{tgt}^{(1)}\right]^0$ ,  $\left[\hat{U}_{tgt}^{(2)}\right]^0$  and  $\left[\hat{U}_{tgt}^{(3)}\right]^0$  to random nonnegative value.
  - 3: **for**  $t = 1$  to *Max\_iteration* **do**
  - 4:   **for**  $r = 1$  to  $R$  **do**
  - 5:     Update  $[\hat{U}_{tgt}^{(1)}]_{*r}^t$  using Equation (4.6).
  - 6:   **end for**
  - 7:   Update  $[\hat{U}_{tgt}^{(2)}]^t$  using Equation (4.7).
  - 8:   Update  $[\hat{U}_{tgt}^{(3)}]^t$  using Equation (4.8).
  - 9: **end for**
  - 10:  $\mathcal{A}_{tgt}^* \approx \mathcal{A}_{aux}^{cluster} \times_1 \hat{U}_{tgt}^{(1)} \times_2 \hat{U}_{tgt}^{(2)} \times_3 \hat{U}_{tgt}^{(3)}$
- 

$$\hat{U}_{tgt}^{(2)} \leftarrow \hat{U}_{tgt}^{(2)} \otimes \frac{[A_{(2)}[\hat{S}_{(2)}^A]^T]}{[\hat{U}_{tgt}^{(2)}\hat{S}_{(2)}^A[\hat{S}_{(2)}^A]^T]} \quad (4.7)$$

$$\hat{U}_{tgt}^{(3)} \leftarrow \hat{U}_{tgt}^{(3)} \otimes \frac{[A_{(3)}[\hat{S}_{(3)}^A]^T]}{[\hat{U}_{tgt}^{(3)}\hat{S}_{(3)}^A[\hat{S}_{(3)}^A]^T]} \quad (4.8)$$

where  $A_{(n)}$  ( $1 \leq n \leq 3$ ) is matrix unfolding of tensor  $\mathcal{A}$  at mode  $n$ ,  $\hat{S}_{(n)}^A = [\mathcal{A}_{aux}^{cluster} \times_{m \neq n} \hat{U}_{tgt}^{(m)}]_{(n)}$ , and  $\otimes$  is the Hadamard product. The matrix unfolding of an N-order tensor  $\mathcal{A} = \mathbb{R}^{I_1 \times \dots \times I_N}$  along the dimension  $d$  are vectors obtained by keeping the index  $d$  fixed while varying the other indices and is denoted as  $A_{(d)}$ . The Hadamard product of a matrix  $U = \mathbb{R}^{I \times J}$  by a matrix  $V = \mathbb{R}^{I \times J}$ , denoted as  $U \otimes V = \mathbb{R}^{I \times J}$  where the entries are given by

$$[U \otimes V]_{ij} = [U]_{ij} \cdot [V]_{ij}$$

where  $1 \leq i \leq I$  and  $1 \leq j \leq J$ .

These multiplicative update rules have stationary points at local minimum, and will not break the non-negativity constraint for the matrix  $\hat{U}_{tgt}^{(i)}$  ( $1 \leq i \leq 3$ ) [37]. The con-

vergence of the above multiplicative update rules can be proven by using the auxiliary function method similar to the work in [37].

Table 4.6: Output tensor  $\mathcal{A}_{tgt}^*$

User	Tag	Item	Val
$U_5$	fantasy	New moon	0.04
$U_6$	romance	New moon	1
$U_7$	drama	Good Omens	0.97
$U_8$	action	James Bonds Girls	0.72
$U_9$	action	Ghost rider	1.17
$U_9$	action	James Bonds Girls	0.72
$U_9$	adventure	Scorpia	1
$U_5$	romance	New moon	0.33
$U_5$	drama	Good Omens	0.05
$U_6$	fantasy	New moon	0.11
$U_6$	drama	Good Omens	0.15
$U'_8$	action	Ghost rider	0.45
$U_9$	fantasy	Scorpia	0.20

Based on the above multiplicative update rules, we design an iterative algorithm to obtain  $\hat{\mathbf{U}}_{tgt}^{(i)}$  ( $1 \leq i \leq 3$ ) which minimize the objective function. The complete algorithm is shown in Algorithm 3. The most time consuming steps in Algorithm 1 are Steps 5, 7 and 8 with a complexity of  $O(|U| \times |V| \times |T| \times R)$ . However, since the matrices  $\mathbf{A}_{(i)}$  and  $\hat{\mathbf{S}}_{(i)}^A$  are sparse, we will utilize the sparse matrix property to reduce the complexity. Let  $N_1$  denote the non-zeros of  $\mathbf{A}_{(i)}$  and  $N_2$  the non-zeros of matrix  $\hat{\mathbf{S}}_{(i)}^A$ . If both matrices are sparse, the complexity is  $O(R \times N_1 + |U| \times N_2)$ ,  $O(R \times N_1 + |T| \times N_2)$  and  $O(R \times N_1 + |V| \times N_2)$  for  $i=1, 2$  and  $3$  respectively. As such, the complexity for Algorithm 1 is bounded by  $O(Max\_iteration \times (R \times N_1 + (|U|+|V|+|T|) \times N_2))$ . Our experiments show that *Max\_iteration* is typically less than 15.

Table 4.6 shows the  $\mathcal{A}_{tgt}^*$  obtained using cross domain information with social network. Note that the last 6 tuples are newly added. Previously, we are unable to recommend any books to  $U_5$  since s/he is the only one who has used the tag 'fantasy'. However, the new tuple  $\langle U_5, drama, Good Omens, 0.05 \rangle$  associates the book 'Good Omens' and the tag 'drama' with user  $U_5$  with a weight of 0.05. Thus, we can now recommend the

drama book 'Good Omens' to  $U_5$ . In addition, although  $U_8$  and  $U_9$  are friends, 'Scorpia' is not recommended to  $U_8$  since 'action' is their only common topic of interest. With the new tuples, we can recommend 'Ghost rider' to  $U_8$ .

## 4.4 Experiments

In this section, we evaluate the effectiveness of the proposed framework for recommendation. We implemented 3 versions of FUSE for the various recommendation tasks:

- **FUSE<sup>+</sup>**: the algorithm utilizes topic-based social regularization
- **FUSE**: the algorithm does not utilize topic-based social regularization
- **FUSE<sup>-</sup>**: the algorithm does not utilize social network information. This is achieved by setting  $\lambda = 0$  for FUSE

We implement our framework in MATLAB and perform the experiments on a 2.33Ghz Intel Core 2 CPU with 4GB RAM, running Windows 7-64 bit. By default,  $R = 50$  and  $\lambda = 10$  in our experiments. We use the following data sets in our experiments:

- **MovieLens dataset<sup>1</sup> (Auxiliary domain)**: This is a publicly available dataset which comprises of two files. The first file contains users' tags on different movies. The second file contains users' ratings on different movies on a scale of 1 to 5, with 1 being bad and 5 being excellent. By joining these two files over user and movie, we obtain the quadruples  $\langle user, movie, tag, rating \rangle$ . We have a total of 24563 quadruples with 2,026 users, 5,088 movies, and 9,078 tags. We pre-process these quadruples to generate a subset such that each user, movie and tag occur at least 10 times in the dataset. The resulting dataset has 24,185 tuples with 339 users, 982 movies, and 582 tags.

---

<sup>1</sup><http://www.grouplens.org/node/73>

- **LibraryThing dataset<sup>2</sup>** (Target domain): Librarything is an online book review website. This dataset also comprises of two files. The first file contains users’ tags and ratings on a scale of 1 to 5, with 1 being bad and 5 being excellent on different books. The second file contains users’ trust statements on different users (binary value is recorded here to indicate the friendship). We have a total of 2,056,487 tuples with 7,279 users, 37,232 books, and 10,559 tags. We pre-process these tuples to generate a subset such that each user, book and tag occur at least 5 times. The resulting dataset has 402,246 tuples with 2,834 users, 2,768 books, 1,012 tags and 7,279 trust statements.

Table 4.7 summarizes the characteristics of these two datasets.

Table 4.7: Characteristics of datasets.

Statistics	Movie	Books
Users	339	2,834
Items	982	2,768
Tags	582	1,012
Social Relations	N.A	7,279
# of tuples	24,185	402,246

We carried out three sets of experiments to evaluate our proposed approach. The first set of experiments evaluates the effectiveness of users’ acceptance in terms of accuracy on item recommendation task. The second set of experiments evaluates the effectiveness of users’ acceptance in terms of accuracy on tag recommendation task. Finally, the third set of experiments show the effectiveness of users’ acceptance in terms of accuracy on user recommendation task.

#### 4.4.1 Experiments on Users’ Acceptance

We demonstrates five sets of experiments to show that our proposed approach increase the effectiveness of users’ acceptance on item, user, tag recommendation task.

---

<sup>2</sup><http://www.librarything.com/>

## Experiments on Item Recommendation

We first evaluate the users' acceptance on item recommendation. We compare our methods with the following existing methods:

1. **UPCC** [60]. This method uses the Pearson's Correlation Coefficient to cluster similar users and recommend items based on these similar users.
2. **IPCC** [19]. This method uses the Pearson's Correlation Coefficient to cluster similar items for recommendation.
3. **TSA** [74]. This method recommends items based on the target domain data only, which is a ternary semantic analysis on users-items-tags.
4. **RMGM** [40]. This is a state-of-the art cross domain collaborative filtering algorithm that utilizes the user-item networks. Latent factor is set to 50.
5. **TagCDCF** [70]. This is a state-of-the art cross domain collaborative filtering algorithm that utilizes the tagging networks by reducing the three-dimensional correlations to two 2D correlations. Latent factor is also set to 50.

We use the Hit Ratio [19] as the metric to evaluate the effectiveness of the various item recommendation methods. Noted that, MAE is not used here as the rating information is not available. Besides, We compare FUSE with UPCC, IPCC, ISA , RMGM and TagCDCF. We do not compare FUSE with QSA as QSA requires both tagging and rating information to be available, yet we are unable to obtain cross domain datasets that have both rating and tagging information. For each user  $u \in U$ , we randomly choose one item  $v$  that has tagged by user previously and withhold the tuples involving  $u$  and  $v$  [19]. Then we run the various methods to generate the top N items recommended for this user. If the item  $v$  is among the top N recommended items, then we say that a hit has occurred. The hit ratio of a method is given by:

$$HitRatio = \frac{Number\ of\ hits}{|U|}$$

Figure 4-1 shows the performance of FUSE<sup>+</sup> with existing recommendation algorithms such as UPCC, IPCC, and TSA. We observe that FUSE<sup>+</sup> is a clear winner, indicating that the joint analysis of cross domain information and social network are useful in understanding the users' interests better and providing better item recommendation compared to TSA which makes use of the social tagging network, and UPCC/IPCC which make use of the rating network only.

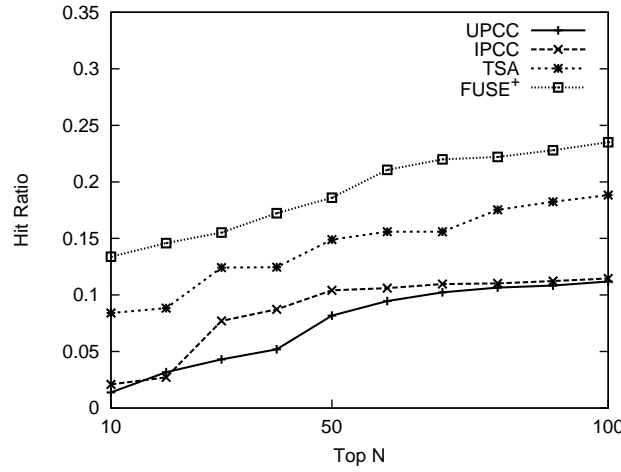


Figure 4-1: Results for Item Recommendation.

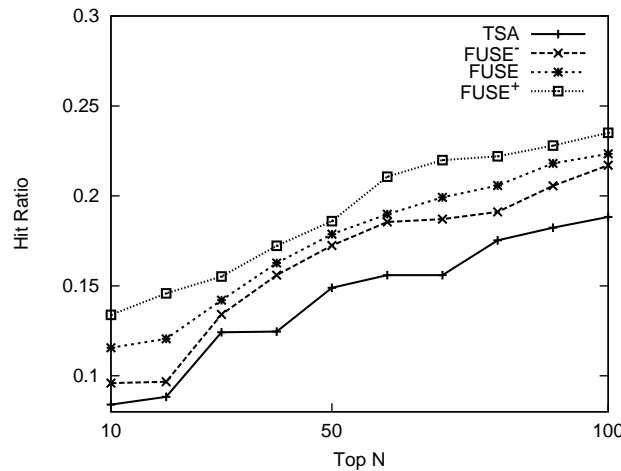


Figure 4-2: Results for Item Recommendation.

We investigate the effectiveness of utilizing topic-specific social regularization for recommendation. Figure 4-2 shows the results when we vary N from 10 to 100. We

observe that FUSE consistently outperforms FUSE<sup>-</sup> indicating the benefits of incorporating social trust information in recommendation. Further, FUSE<sup>+</sup> outperforms FUSE by an average of at least 8 % demonstrating that accurate modeling of topic-specific trust relationships leads to more accurate item recommendation.

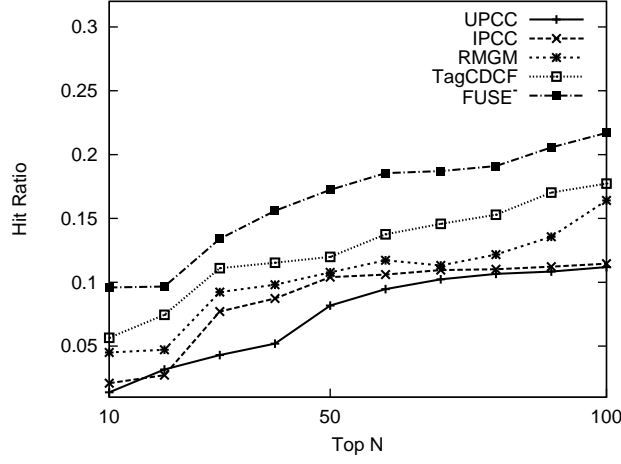


Figure 4-3: Results for Item Recommendation.

Figure 4-3 shows the effectiveness of utilizing cluster-level tensor in cross domain recommendation. We observe that FUSE<sup>-</sup> consistently outperforms TagCDCF, RMGM, UPCC and IPCC as we vary N from 10 to 100. In particular, RMGM outperforms UPCC and IPCC, indicating that cross domain transfer of binary relationships (user-rating) can improve recommendation accuracy. Further, TagCDCF outperforms RMGM demonstrating that tag information is useful in cross domain recommendation. However, since TagCDCF requires the decomposition of ternary relationship into two binary relationships (user-item and item-tag), there is information loss resulting in reduced accuracy compared to FUSE<sup>-</sup>.

### Experiments on Tag Recommendation

For the task of users' acceptance on tag recommendation, we evaluate our algorithm against two state-of-the-art methods: TSA [74] and RTF [59]. For each user  $u \in U$ , we randomly choose one item  $v$  and remove all tuples involving  $u$  and  $v$  from the dataset

[59]. Then we run the methods to generate the top N tags recommended for this user.

We use the standard recall and precision measures to evaluate the results:

$$Precision = \frac{Number\ of\ Hits}{N}$$

$$Recall = \frac{Number\ of\ Hits}{|T_{u,v}|}$$

where  $T_{u,v}$  is the set of tags used by user  $u$  on item  $v$ .

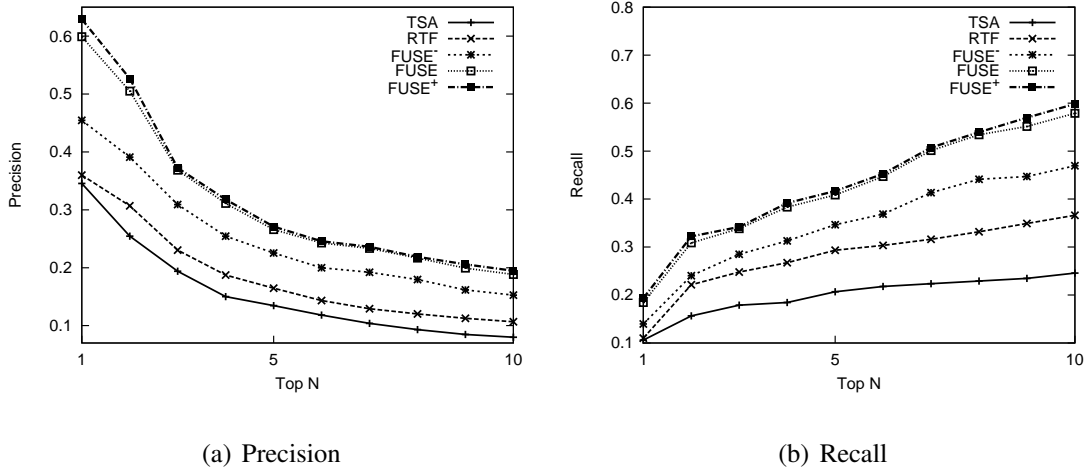


Figure 4-4: Tag recommendation

Figures 4-4(a) and 4-4(b) show the precision and recall of the methods for varying values of  $N$ . We see that FUSE<sup>+</sup> is able to achieve a higher recall and precision compared to the other three methods. FUSE<sup>+</sup> outperforms FUSE by 2.5% on average in both recall and precision, indicating that topic-specific trust regularization can improve tag recommendation compared to traditional trust regularization. Both FUSE<sup>+</sup> and FUSE outperform FUSE<sup>-</sup>, indicating the effectiveness of incorporating social trust in tag recommendation. All our methods outperform state-of-the-art TSA demonstrating the effectiveness of using cluster-level tensor in transferring knowledge from the Movie domain to Book domain.



## Experiments on User Recommendation

For the task of users' acceptance on interesting user recommendation, we compare our algorithm with TSA [74]. For each user  $u \in U$ , we randomly choose one of his/her friend  $u_f$  and remove  $u_f$  from  $u$ 's friendship list. Then we run the algorithms to generate the top  $N$  users recommended for this user. We use the standard recall measures to evaluate the results:

$$Recall = \frac{Number\ of\ Hits}{|U|}$$

Figure 4-5 shows the results for varying values of  $N$ . We observe that  $FUSE^+$  achieves the best performance and outperforms  $FUSE$  by 10% on average, while the performance of  $FUSE^-$  is very close to TSA. This confirms that both topic-specific trust and social network information are useful in user recommendation task.

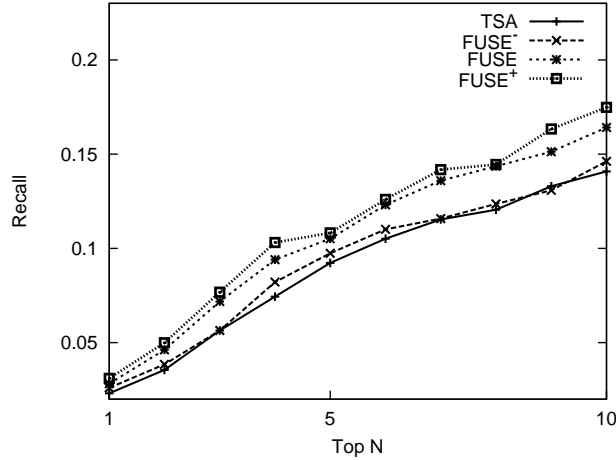


Figure 4-5: User recommendation

In order to evaluate the effectiveness of our methods in recommending interesting users, we first determine the similarity of items among the recommended top  $N$  users [81] since users with shared interests are more likely to tag and rate similar items and with similar friends. We compute the item similarity as the cosine similarity of their  $TF \times IDF$  tag term vector [81].

Let  $NB_u$  be the set of top  $N$  users recommended to  $u$ . The intra-neighborhood simi-

larity is given by the average cosine similarity of all items for the users in  $NB_u$ :

$$IntraSim(NB_u) = \frac{\sum_{w \in NB_u} \sum_{i \in I_u, j \in I_w} sim(i, j)}{\sum_{w \in NB_u} |I_u| |I_w|}$$

where  $I_u$  and  $I_w$  are the sets of items tagged by users  $u$  and  $w$ .

Let  $Random_u$  be the set of  $N$  users randomly chosen from the set of users  $U - \{u\}$ .

We can determine the inter-neighborhood similarity as follows:

$$InterSim(Random_u) = \frac{\sum_{w \in Random_u} \sum_{i \in I_u, j \in I_w} sim(i, j)}{\sum_{w \in Random_u} |I_u| |I_w|}$$

where  $I_u$  and  $I_w$  are the sets of items tagged by users  $u$  and  $w$  respectively.

Table 4.8 shows the intra-similarity and inter-similarity of FUSE<sup>+</sup> and TSA. We observe that the average intra-similarity is generally higher than the average inter-similarity for all the three methods. Furthermore, FUSE<sup>+</sup> have much higher intra-similarity and inter-similarity as compared to TSA. This indicates that more relevant users are found by FUSE<sup>+</sup> and hence lead to more accurate user recommendation.

Table 4.8: Intra- and inter- similarity between FUSE and TSA

Method	Intra-similarity	Inter-similarity
TSA	0.15	0.09
FUSE <sup>+</sup>	<b>0.225</b>	<b>0.037</b>

#### 4.4.2 Sensitivity Experiments

We also examine the effect of various parameters on the performance of Algorithm FUSE and FUSE<sup>+</sup> for item recommendation. Figure 4-6(a) shows the results as we vary the tensor dimension  $R$ . We observe that the proposed method FUSE<sup>+</sup> consistently outperforms the FUSE. This provides a evidence that the topic-based social recommendation is useful and can be used to improve the recommendation accuracy. We also find that the hit ratio of both FUSE and FUSE<sup>+</sup> increase as  $R$  increases, but decrease after  $R = 50$  which

may be caused by model over-fitting when the latent dimensions are large. Thus we set  $R = 50$ .

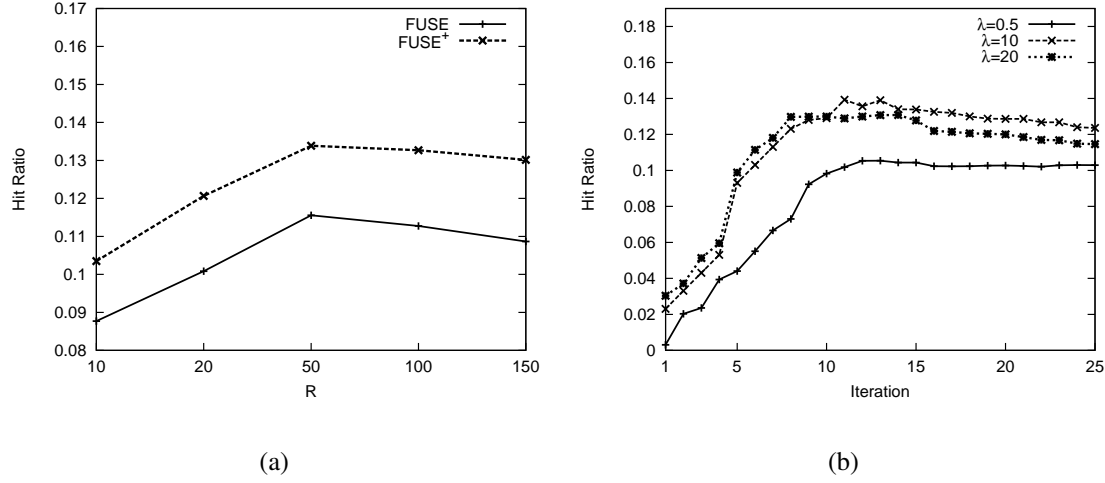


Figure 4-6: Sensitivity analysis

Figure 4-6(b) shows the hit ratio for various values of  $\lambda$  as we vary the number of iterations from 1 to 25. We observe that when we increase the iteration to be around 10, there seem to be little improvement for any large iteration. This suggests that a small number of iteration (such as 10) is enough for models. In other words, our algorithms typically converge after 10 iterations.

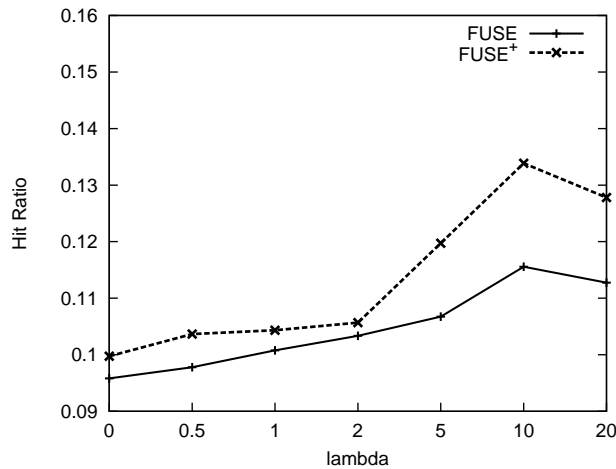


Figure 4-7: Sensitivity analysis on  $\lambda$

Figure 4-7 shows the impact of  $\lambda$  on the recall rate of our algorithms. Recall that the parameter  $\lambda$  control how much the information from social network will dominate

the learning process. In the extreme case where  $\lambda = 0$ , the social network information is not used. As we can see from Figure 4-7, adopting a larger  $\lambda$  value can help to avoid the sparsity problem suffered by most MF-based CF methods. When we set  $\lambda > 0$ , we can achieve better results. This clearly demonstrates the impact of social network information, that is, adding more social network information can improve the generalization ability of the model. Moreover, Figure 4-7 also shows that the performance might degrade when  $\lambda$  is too large. In practice, we should choose a moderate value of  $\lambda$ . We observe that the best recall is obtained when  $\lambda = 10$  indicating that social network information helps to improve item recommendation.

Table 4.9: Example of Top 10 representative tags for 5 groups in movies and books domain

Cluster	Tags from LibraryThing	Tags from MovieLens
Cluster 1	humor, england, jane austen, classics, non fiction animals, books, children, british, historical	humorous, jane austen, humor, history, england anime, disney, library, 70mm, classic
Cluster 2	adventure, war, world, history, political philosophy, exploration, action, fun, dark	steven spiberg, adventure, war, action, super hero murder, johny deep, tom hanks, nasa, zombie
Cluster 3	crime, spouse, romance, french, James Bond sherlock holmes, action, series, australian, blake	crime, romance, 007, James Bond , dramma bruce wills, brad pitt, japan, french, pg
Cluster 4	science, aliens, mars, mystery, future space, intelligence, technology, star, robots	sci-fi, aliens, space, future, magic mystery, travel, robots, trip, boring
Cluster 5	relationships, man-woman, friendship, female, family marriage, divorce, royalty, murder, keeper	divorce, romance, love, money, sex family, sweet, france, friendship, sentimental

### 4.4.3 Case Study

Finally, we show a sample of the mappings we obtain between the books and movies domains in our experiments. Table 4.9 shows the top 10 representative tags for 5 clusters. These clusters are randomly chosen from our 50 clusters. For each cluster, we take the top 10 most frequent tags as its representative tags.

By examining the tags for each cluster, we see that our algorithm is able to identify the topic for movies and books and find the correspondence between the different domains. For example, both the book “Jane Austin” and the movie “Jane Austin” are mapped to Cluster 1, while the character “James Bond ” from movies and books are mapped to the same Cluster 3.

#### 4.4.4 Scalability

Finally, we show the scalability of Algorithm 3 after mapping it to the MapReduce framework. The expensive operations in the algorithm are the matrix multiplication in the update formulae in Eq. (4.6), (4.7). Following the idea of [43], we implemented the MapReduce version of Algorithm 3 on our in-house cluster, Awan<sup>3</sup>. The cluster consists of 72 computing nodes, each of which has one Intel X3430 2.4GHz processor, 8GB of memory, two 500GB SATA hard disks and gigabit ethernet. On each node, we install CentOS 5.5 operating system, Java 1.6.0 with a 64-bit server VM, and Hadoop 0.23.6. All the nodes are connected via three high-speed switches.

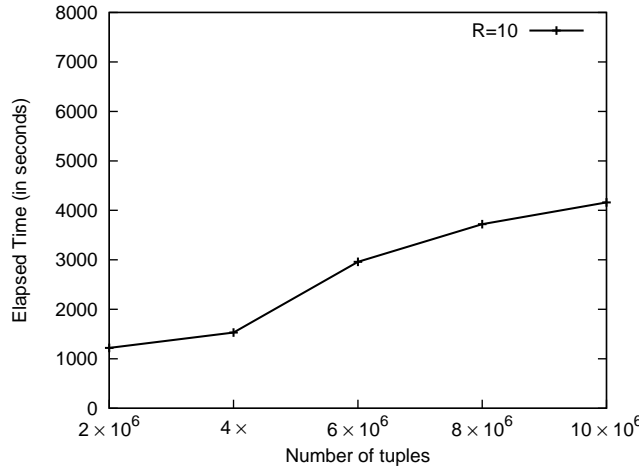


Figure 4-8: Scalability analysis

We vary the dataset size from 2 million to 10 million by duplicating the users, items and tags in the original datasets and run the experiment by setting the model dimension  $R$  to 10 and 20 respectively. Figure 4-8 shows the results. We observe that the runtime increases linearly with respect to the dataset size for both  $R = 10$  and  $R = 20$ . This shows that our algorithm is scalable with respect to the dataset size.

<sup>3</sup><http://awan.ddns.comp.nus.edu.sg/ganglia/>

## 4.5 Summary

In this chapter, we have presented a novel collaborative filtering method for integrating social network and cross domain network in a unified framework via latent feature sharing and cluster-level tensor sharing. This framework utilizes data from multiple domains and allows the transfer of useful knowledge from auxiliary domain to the target domain. The results of extensive experiments performed on a real world dataset show that our unified framework outperforms the state-of-the-art techniques in all the three recommendation tasks. We have also implemented the algorithm on a map-reduce infrastructure and have demonstrated its scalability.



---

---

## CHAPTER 5

---

# IMPROVING USERS' ACCEPTANCE USING SOCIAL TRUST DATA

Online social networks present new opportunities for further improving the users' acceptance of RS. In real life, people often resort to friends in their social networks for advices before purchasing a product or consuming a service. Findings in sociology and psychology fields indicate that human beings tend to associate and bond with similar others, so called homophily. Due to the stable and long-lasting social bindings, people are more willing to share their personal opinions with their friends, and typically trust recommendations from their friends more than those from strangers and vendors. The phenomenally popular online social networks, such as Facebook, Twitter, and Youtube, provide novel ways for people to communicate and build virtual communities. Online social networks not only make it easier for users to share their opinions with each other, but also serve as a platform for developing new RS algorithms to automate the otherwise manual and anecdotal social recommendations in real life social networks. We have seen the existing recommender algorithms which aim to improve users' acceptance in social rating/tagging data, cross domain. The previous two chapters present recommender al-



gorithms for these data. However, they don't study the social trust data in recommender system. In order to improve recommender systems and to provide more personalized recommendation results, we need to incorporate social trust data among users. In this Charter, we focus on recommender systems in Social Trust Data.

## 5.1 Motivation

Recommender systems are fast becoming the tools of choice for a user to sieve through tons of online materials in order to find information that is relevant to him/her. Many of these recommender systems employ collaborative filtering (CF) techniques to identify similar users based on their purchased history or past ratings to generate personalized recommendation. This works well when the users have long term interests that do not change from time to time. However, for users with short term interests, modeling the shift in users' interests has been shown to improve recommendation accuracy. This is achieved by introducing a personalized time factor for each user to capture the shift in users' interests over time [85, 34, 31, 41]. With the advent of online social networks, social network based CF approaches to recommendation have emerged [28, 69, 47]. The assumption is that friends tend to influence their friends to exhibit similar likes and dislikes. Hence, we can further improve recommendation accuracy by taking into account the social relationships.

Let us consider the snapshots of users' item ratings of Table 5.1(a) at time points  $T_1$  and  $T_2$ . Besides that, we also have additional social relationship at time points  $T_1$  and  $T_2$  in Table 5.1(b). Suppose our target user is  $U_3$ . At time point  $T_1$ , both users  $U_1$  and  $U_2$  have watched and rated the Book "*Forrest Gump*". Traditional CF methods [63, 66, 57] will group  $U_1$ ,  $U_2$  and  $U_3$  as similar users and recommend "*Beautiful Mind*" and "*Groundhog Day*" to  $U_3$  since  $U_1/U_2$  has watched these books previously. Yet,  $U_3$ 's interest does not remain static. We observe that at time point  $T_2$ , his interest has shifted from comedy book to animation book as he rates a new item "*Toy Story*". Recognizing

Table 5.1: Example datasets

(a) Ternary relations among user, rating and item over Time in Book Domain

User	Rating	Item	Time
$U_1$	like	Forrest Gump	$T_1$
$U_1$	like	Beautiful Mind	$T_1$
$U_2$	like	Forrest Gump	$T_1$
$U_2$	like	Groundhog Day	$T_1$
$U_2$	like	Groundhog Day	$T_1$
$U_3$	like	Forrest Gump	$T_1$
$U_3$	like	Toy Story	$T_2$
$U_4$	dislike	Forrest Gump	$T_1$
$U_4$	dislike	Toy Story	$T_1$
$U_5$	like	New moon	$T_1$
$U_6$	like	New moon	$T_1$
$U_7$	like	Good omens	$T_1$
$U_8$	like	James Bonds Girls	$T_1$
$U_9$	like	Ghost rider	$T_1$
$U_9$	like	James Bonds Girls	$T_1$
$U_9$	like	Scorpia	$T_1$
$U_{10}$	like	Toy Story	$T_2$
$U_{10}$	like	Shrek	$T_2$

(b) Social Trust Over Time

User	User	Time
$U_3$	$U_2$	$T_1$
$U_8$	$U_9$	$T_1$
$U_3$	$U_{10}$	$T_2$
$U_{10}$	$U_3$	$T_2$

this, CF with temporal dynamics will recommend another animation book "*Shrek*" to  $U_2$  instead. On the other hand, looking at the social relationships among users, we realize that  $U_1$  and  $U_3$  are friends. Hence, social network based CF will conclude that  $U_3$  is likely to like "*Groundhog Day*" since his friend  $U_2$  have read and rated this book. Each of the different methods arrive at different items to recommend. How do we reconcile the different recommendations? To complicate matter, social relationships are not static but evolve over time as a user can make new friends and old friends do grow apart. We observe that at time point  $T_1$ ,  $U_3$  has only one friend  $U_2$ , whereas at time point  $T_2$ , his friends are  $\{U_2, U_{10}\}$ . Now if we want to give a recommendation to  $U_3$  at time point  $T_2$ , what item should we recommend so that it is most likely to be accepted by  $U_3$ ?

To answer this question, we must be able to quantify the degree of influence on a user's decision making process from his/her long term and short term interests, as well as his/her social trust relationships over time. Note that these two factors are not in-

dependent. We advocate that when two users' long term and short term interests are aligned, they are likely to become friends, and they will tend to be more receptive towards each other's preferences. Conversely, if the users' interests are not aligned, they will grow apart after some time and become less receptive towards the preferences of the other user.

In this chapter, we propose a model called the *Receptiveness over Time Model* (RTM), to quantify the dynamic interaction between user interest and social trust. This model utilizes a probabilistic generative approach to leverage on the information embedded in a users' social trust network, and the users' rating history. The RTM captures (1) the degree of receptiveness for each user over time (modeling receptiveness change), (2) the distribution of personal preference over the latent topics for each user over time (modeling users' interests change), (3) the distribution of items for each topic over time. The estimation of the RTM model parameters is performed using Gibbs sampling MCMC procedure. To overcome the data sparsity problem, we design a special Bayesian treatment to the latent variable to ensure that the evolution of latent parameter is smooth and share the topic-level rating knowledge across different time points. We carry out experiments on a real world Epinions dataset to demonstrate the effectiveness of our proposed approach. We also demonstrate how RTM can be used to explicitly track and visualize the change in users' interests and their receptiveness to other users.

The rest of the chapter is organized as follows. Section 5.2 describes our problem formulation. Section 5.3 shows our proposed Method. Section 5.4 presents the experimental results, and we summarize in Section 5.5.

## 5.2 Problem Formulation

In recommender systems, we have a set of users  $U = \{ u_1, \dots, u_a \}$  and a set of items  $M = \{ m_1, \dots, m_b \}$ . At time  $t$ , a user  $u$  expresses his/her preference for an item  $m$  by giving a rating in the range of 1 to 5 with 1 being the least preferred and 5 being the most

preferred. These ratings are represented using the matrix  $R^{(t)}$  with dimensions  $|U| \times |M|$ . Each entry in this matrix,  $r^{(t)}(u, m)$ , corresponds to the rating provided by user  $u$  on item  $m$  at time point  $t$ .

Besides the rating information, we also represent the social relationships among the users at time  $t$  in the form of a matrix  $S^{(t)}$  with dimensions  $|U| \times |U|$  such that its entry  $s^{(t)}(u, v) = 1$  if  $u$  issues trust statement towards user  $v$  at time point  $t$ . Otherwise  $s^{(t)}(u, v) = 0$ . Note that  $S^{(t)}$  is asymmetric in general.

We formulate our task as follows: Given a user  $u \in U$  at time  $t$  and an item  $m \in M$ , we want to predict the rating that  $u$  will give to  $m$  at time  $t$  based on the past rating history and social relationships, i.e.  $R^{(t')}$  and  $S^{(t')}$  for all  $t' \in [1, t)$ .

## 5.3 Proposed Method

In this section, we first give our problem formulation. Then we describe the RTM model and show how the model can be used for various tasks such as rating prediction, tracking receptiveness between friends and user interest change.

### 5.3.1 Receptiveness over Time Model

In order to capture the dynamic interactions between long term and short term interests as well as friendships for recommendation, our model has two parts. The first part models user receptiveness at a single time point. The second part incorporates temporal information to allow for modeling over time.

#### Single Time Point Receptiveness Modeling

Receptiveness captures the dynamic interaction between user interest and social trust. Existing social CF filtering approaches [47, 46, 28, 69] incorporate social trusts as auxiliary data to regulate user preferences for recommendation. In other words, if two users are friends, social CF approaches will assign greater weights to their corresponding pref-

erences. However, we realize that a user's decision making process is not so simplistic. A person may be his/her friend, however, if that friend's interests are not aligned with him/her, the receptiveness to that friend's interest will not be high and vice versa.

Among the approaches that model user's interests for recommendation, Bi-LDA has proven to work well in practice [57]. It is a generative model with several advantages that are suitable for our work:

1. It models the distribution of users' interest within a probabilistic framework, thus allowing a more interpretable explanation compared to the matrix factorization approach.
2. It allows the inclusion of prior knowledge into the generative process and a principled framework to select model structure. This proves to be useful in linking consecutive time points to avoid the data sparsity problem in the second part of RTM.

Table 5.2: Meanings of symbols used

Symbol	Meaning
$u$	A user
$U$	The set of all users
$f$	Friend of user
$F(u)$	The set of user $u$ 's friends
$K$	Number of user topic
$L$	Number of item topic
$z_{u,m}^{friend}$	The receptive friend picked for rating given by user $u$ on item $m$
$z_{u,m}^{user}$	The user topic picked for rating given by user $u$ on item $m$
$z_{u,m}^{item}$	The item topic picked for rating given by user $u$ on item $m$
$\pi_u^{friend}$	Distribution of user $u$ over users' friend
$\pi_u^{user}$	Distribution of user $u$ over users' topics
$\pi_m^{item}$	Distribution of item $m$ over items' topics
$\Phi_{z_{u,m}^{user}, z_{u,m}^{item}}$	Rating-scale mixing proportion of user-item topic joint distribution over values $\{1...R\}$ for topic $z_{u,m}^{user}, z_{u,m}^{item}$

However, Bi-LDA does not incorporate social relationships. For this reason, we extend Bi-LDA to incorporate social relationships and call it Bi-LDA<sup>social</sup>. Table 5.2 summarizes the symbols used in describing the Bi-LDA<sup>social</sup> model. In Bi-LDA<sup>social</sup>, each user  $u$  follows a preference distribution  $\pi_u^{friend}$  that depicts how likely  $u$ 's friend will contribute to  $u$ 's item rating decision. The probability for a friend of  $u$  to influence the item rating decision is proportional to the receptiveness of  $u$  to this friend. Note that, we assume  $u$  is a special friend of himself/herself (i.e.,  $u \in F(u)$ ).

In addition, each user  $u$  and item  $m$  follow the topic distribution parameters  $\pi_u^{user}$  and  $\pi_m^{item}$  respectively. To rate an item  $m$ , a user  $u$  first draws a user topic  $z_{u,m}^{user}$  and the item  $m$  draws a item topic  $z_{u,m}^{item}$  from the corresponding distributions.  $\Phi_{z_{u,m}^{user}, z_{u,m}^{item}}$  is the rating-scale mixing proportion of user-item topic joint distribution over values  $\{1...R\}$  for topic  $z_{u,m}^{user}$ ,  $z_{u,m}^{item}$ .

During the generative process, ratings are generated as follows:

1. Choose a  $K \times L$  distribution over ratings  $\Phi \sim Dir(\beta)$
2. Choose a distribution over friends for each user  $\pi_u^{friend} \sim Dir(\alpha^{friend})$
3. Choose a distribution over  $K$  users' topic for each user  $\pi_u^{user} \sim Dir(\alpha^{user})$
4. Choose a distribution over  $L$  items' topic for each item  $\pi_m^{item} \sim Dir(\alpha^{item})$
5. For each rating  $r_{u,m}$ :

- Choose receptive friend

$$z_{u,m}^{friend} \sim Multinomial(\pi_u^{friend})$$

- Choose user topic

$$z_{u,m}^{user} \sim Multinomial(\pi_u^{user})$$

- Choose item topic

$$z_{u,m}^{item} \sim Multinomial(\pi_m^{item})$$

- Choose a rating  $r_{u,m} \sim \Phi_{z_{u,m}^{user}, z_{u,m}^{item}}$

The graphical model is shown in Figure 5-1. We note that the decision of rating given by user  $u$  on item  $m$  is based on the receptiveness of a friend (including  $u$  himself/herself). The receptive friend  $z_{u,m}^{friend}$  can be drawn from  $Multinomial(\pi_u^{friend})$ . Once the receptive friend  $z_{u,m}^{friend}$  is picked, we randomly draw a topic  $z_{u,m}^{user}$  from user  $z_{u,m}^{friend}$ 's preference based on  $Multinomial(\pi_u^{user})$ . Similarly, a topic  $z_{u,m}^{item}$  from item  $m$  is also drawn based on  $Multinomial(\pi_m^{item})$ . The user and item topics  $z_{u,m}^{user}$  and  $z_{u,m}^{item}$  together with user-item topic joint distribution  $\Phi$  jointly specify the rating, that is,  $r_{u,m} \sim \Phi_{z_{u,m}^{user}, z_{u,m}^{item}}$ .

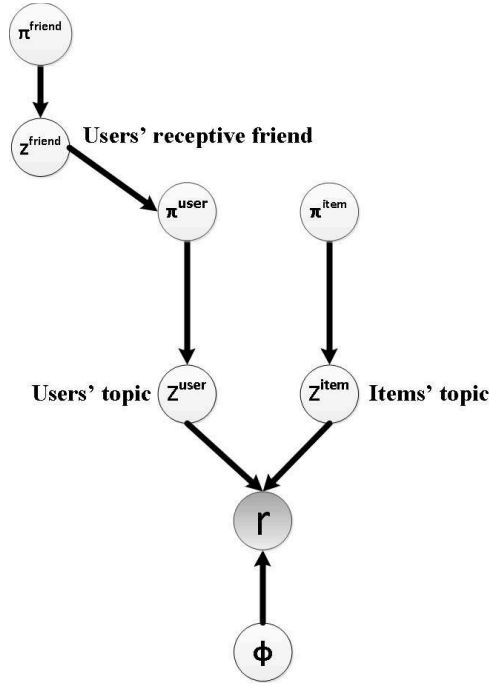


Figure 5-1: Graphical model representation of Bi-LDA<sup>social</sup>

In order to compute the rating  $r_{u,m}$ , we need to obtain a number of model parameters,  $\Theta = \{ \pi_u^{friend}, z_{u,m}^{friend}, \pi_u^{user}, \pi_m^{item}, z_{u,m}^{user}, z_{u,m}^{item}, \Phi_{z_{u,m}^{user}, z_{u,m}^{item}} \}$ . Among them,  $\pi_u^{friend} \in \mathbb{R}^{1 \times |U|}$  with Dirichlet priors  $\alpha^{friend}$  captures distribution of the receptiveness of user  $u$  to his/her friends,  $\pi_u^{user} \in \mathbb{R}^{1 \times K}$  with Dirichlet priors  $\alpha^{user}$  depicts the distribution of user  $u$ 's preferences on the  $K$  users' topics,  $\pi_m^{item} \in \mathbb{R}^{1 \times L}$  with Dirichlet priors  $\alpha^{item}$  captures the distribution of an item  $m$  on the  $L$  items' topics,  $\Phi \in \mathbb{R}^{K \times L}$  with Dirichlet priors  $\beta$  is rating-scale mixing proportion of user-item joint topic,  $z_{u,m}^{friend}$  represents the receptive friend whom user  $u$  has picked for the rating given by user  $u$  on item  $m$ ,  $z_{u,m}^{user}$  represents the topic user

$u$  has picked and  $z_{u,m}^{item}$  represents the topic item  $m$  has picked. The hyper-parameters of the Dirichlet priors  $\alpha^{friend}$ ,  $\alpha^{user}$ ,  $\alpha^{item}$  and  $\beta$  can be simply set to 1 [57].

Let  $X$  be the observed rating for user  $u$  on item  $m$ . Putting everything together, we obtain the joint probability distribution for the Bi-LDA<sup>social</sup> as follows:

$$\begin{aligned}
& Pr(X, z^{friend}, z^{user}, z^{item}, \Phi, \pi^{user}, \pi^{item}) \\
&= Pr(X|z^{user}, z^{item}, \Phi) Pr(\Phi|\beta) Pr(z^{user}|\pi^{user}) \times \\
& Pr(\pi^{user}|\alpha^{user}, z^{friend}) Pr(z^{friend}|\pi^{friend}) \times \\
& Pr(\pi^{friend}|\alpha^r) Pr(z^{item}|\pi^{item}) Pr(\pi^{item}|\alpha^{item})
\end{aligned} \tag{5.1}$$

Solving this equation is intractable, instead we adapt the collapsed Gibbs sampler [15] to learn the model parameters. In particular, we analytically marginalize out all the conjugate distributions  $\Phi$ ,  $\pi^{user}$ ,  $\pi^{item}$  and  $\pi^{friend}$  and obtain an expression for the joint probability  $P(X, z^{user}, z^{item}, z^{friend})$ . With this, we can compute the conditional distributions necessary for Gibbs sampling. We give the explicit forms for the following conditional distributions where  $x = r_{u,m}$  is the observed rating,  $k = z_{u,m}^{user}$  and  $l = z_{u,m}^{item}$ :

$$\begin{aligned}
& P(f = z_{u,m}^{friend} | \Theta \setminus z_{u,m}^{friend}, x) \propto \left( (n_{u,f}^{friend})^{-(u,m)} + \alpha^{friend} \right) \\
& \times \left( (n_{f,k}^{user})^{-(u,m)} + \alpha^{user} \right) \left( \frac{N_{k,l,x}^{-(u,m)} + \beta}{\sum_r (N_{k,l,r}^{-(u,m)} + \beta)} \right)
\end{aligned} \tag{5.2}$$

where  $n_{u,f}^{friend}$  denotes the number of times that user  $u$  is receptive to  $f$  in all the ratings,  $(n_{u,f}^{friend})^{-(u,m)}$  denotes the number of times that user  $u$  is receptive to  $f$  in all the ratings excluding  $r_{u,m}$ ,  $(n_{f,k}^{user})^{-(u,m)}$  denotes the number of times user  $f$  will be assigned to user topic  $k$  in all the ratings except for  $r_{u,m}$ ,  $N_{k,l,r}$  represents the number of times the user with user topic  $k$  has rated item with item topic  $l$  with the rating  $r$ , and  $N_{k,l,x}^{-(u,m)}$  represents the number of times the observed rating  $x$  has been given by user with user topic  $k$  on item with item topic  $l$  excluding  $r_{u,m}$ .



Similarly, we define the conditional distribution for user topic  $z_{u,m}^{user}$ :

$$\begin{aligned} P(z_{u,m}^{user} = k | \Theta \setminus z_{u,m}^{user}, x) \\ \propto \left( (n_{f,k}^{user})^{-\neg(u,m)} + \alpha^{user} \right) \left( \frac{N_{k,l,x}^{-\neg(u,m)} + \beta}{\sum_r (N_{k,l,r}^{-\neg(u,m)} + \beta)} \right) \end{aligned} \quad (5.3)$$

The conditional distribution is the same for the item topic with the role of user and item reversed.

$$\begin{aligned} P(z_{u,m}^{item} = l | \Theta \setminus z_{u,m}^{item}, x) \\ \propto \left( (n_{m,l}^{item})^{-\neg(u,m)} + \alpha^{item} \right) \left( \frac{N_{k,l,x}^{-\neg(u,m)} + \beta}{\sum_r N_{k,l,r}^{-\neg(u,m)} + \beta} \right) \end{aligned} \quad (5.4)$$

where  $(n_{m,l}^{item})^{-\neg(u,m)}$  denotes the number of times that an item  $m$  is assigned to item topic  $l$  in all the ratings except for  $r_{u,m}$ , and  $N_{k,l,r}$  represents the number of times that a user with user topic  $k$  has rated an item with item topic  $l$  with the rating  $r$ .

The parameter  $\pi^{friend}$ ,  $\pi^{user}$ ,  $\Phi$  and  $\pi^{item}$  can be obtained as follows:

$$\begin{aligned} \pi_{u,f}^{friend} &= \frac{n_{u,f}^{friend} + \alpha^{friend}}{\sum_{f \in F(u)} (n_{u,f}^{friend} + \alpha^{friend})} \\ \Phi_{k,l,x} &= \frac{N_{k,l,x} + \beta}{\sum_r (N_{k,l,r} + \beta)} \\ \pi_{u,k}^{user} &= \frac{n_{u,k}^{user} + \alpha^{user}}{\sum_k (n_{u,k}^{user} + \alpha^{user})} \\ \pi_{m,l}^{item} &= \frac{n_{m,l}^{item} + \alpha^{item}}{\sum_l (n_{m,l}^{item} + \alpha^{item})} \end{aligned} \quad (5.5)$$

The algorithm of the collapsed Gibbs sampler for inferring these latent variables  $\{z^{friend}, z^{user}, z^{item}\}$  is shown in Algorithm 4.

## Receptiveness over Time

The second part of RTM is to model the dynamic interaction of users' interest along with the receptiveness among friends over time. Given the users' rating histories at  $T$  different

---

**Algorithm 4:** Gibbs sampling for Bi-LDA<sup>social</sup>


---

**input** : Users' rating histories  $X$ , users' trust relation  $S$ ,  $K$  and  $L$   
**output:**  $\pi^{user}$ ,  $\Phi$ ,  $\pi^{item}$  and  $\pi^{friend}$

- 1 /\*Initialization of the latent variables and counters\*/
- 2 Random initialize  $z^{friend}$ ,  $z^{user}$  and  $z^{item}$
- 3 Initialize  $n^{friend}$ ,  $n^{user}$ ,  $n^{item}$  and  $N$  as 0
- 4 **foreach**  $x = r_{u,m} \in X$  **do**
- 5      $f = z_{u,m}^{friend}$ ,  $k = z_{u,m}^{user}$  and  $l = z_{u,m}^{item}$
- 6     Increase the counter of  $n_{u,f}^{friend}$ ,  $n_{u,k}^{user}$ ,  $n_{m,l}^{item}$  and  $N_{k,l,x}$
- 7 **end**
- 8 **for**  $index=1$  to  $Iter$  **do**
- 9     /\* for each rating  $x = r_{u,m}$  in  $X$  \*/
- 10    **foreach**  $x = r_{u,m} \in X$  **do**
- 11      /\* Sample Friends \*/
- 12       $f = z_{u,m}^{friend} \sim P(z_{u,m}^{friend} | \Theta \setminus z_{u,m}^{friend}, x)$  according to Equation (5.2)
- 13      /\* Sample Users' Topics \*/
- 14       $k = z_{u,m}^{user} \sim P(z_{u,m}^{user} | \Theta \setminus z_{u,m}^{user}, x)$  according to Equation (5.3)
- 15      /\* Sample Items' Topics \*/
- 16       $l = z_{u,m}^{item} \sim P(z_{u,m}^{item} | \Theta \setminus z_{u,m}^{item}, x)$  according to Equation (5.4)
- 17       $f = z_{u,m}^{friend}$ ,  $k = z_{u,m}^{user}$ ,  $l = z_{u,m}^{item}$ ,  $x = r_{u,m}$
- 18      /\* Update Counter \*/
- 19      Update the counter  $n_{u,f}^{friend}$ ,  $n_{u,k}^{user}$ ,  $n_{m,l}^{item}$  and  $N_{k,l,x}$
- 20    **end**
- 21 **end**
- 22 /\* Get the mean estimate for  $\pi^{friend}$ ,  $\pi^{user}$ ,  $\Phi$  and  $\pi^{item}$  \*/
- 23  $\pi^{friend}$ ,  $\pi^{user}$ ,  $\Phi$  and  $\pi^{item}$  can be calculated according to Eq. (5.5).

---

time points, a naive approach is to fit a Bi-LDA<sup>social</sup> model at each time point and learn the receptiveness and user interest distribution at the various time points, denoted as  $\{\pi^{friend(1)}, \dots, \pi^{friend(T)}\}$  and  $\{\pi^{user(1)}, \dots, \pi^{user(T)}\}$ , respectively. However, using this approach, the latent topics learnt at time point  $t_1$  may be totally different from that at time point  $t_2$ . Furthermore, since we regard each user in the different time point as independent, we are unable to make use of his/her ratings in the past. This worsens the data sparsity problem.

To overcome the shortcomings, we assume that the overall interest distribution of the whole user population should remain stable. This enable us to share the  $\Phi$  across the different time points. In addition, we impose constraints on  $\pi^{user}$  and  $\pi^{friend}$  by assuming dependency between two consecutive snapshots as follows:

$$\pi^{friend(t)} \sim P(\pi^{friend(t)} | \pi^{friend(t-1)})$$

In other words, we introduce a prior from  $\pi^{friend(t-1)}$  to  $\pi^{friend(t)}$  so that at time  $t$ , we are drawing from the Dirichlet prior parameterized by  $\pi^{friend(t-1)}$  where

$$\pi^{friend(t)} \sim Dirichlet(\lambda \pi^{friend(t-1)})$$

The intuition of  $\lambda \pi^{friend(t-1)}$  can be interpreted as the prior observed counts that user is receptive to his/her friends before any friend from the current time points is observed. Similarly, we introduce the parameterized prior for both user and item topic distributions:

$$\pi^{user(t)} \sim Dirichlet(\lambda \pi^{user(t-1)}),$$

$$\pi^{item(t)} \sim Dirichlet(\lambda \pi^{item(t-1)})$$

The graphical representation of the RTM model is shown in Figure 5-2. With this change, the conditional distribution of Equation (5.2) is now

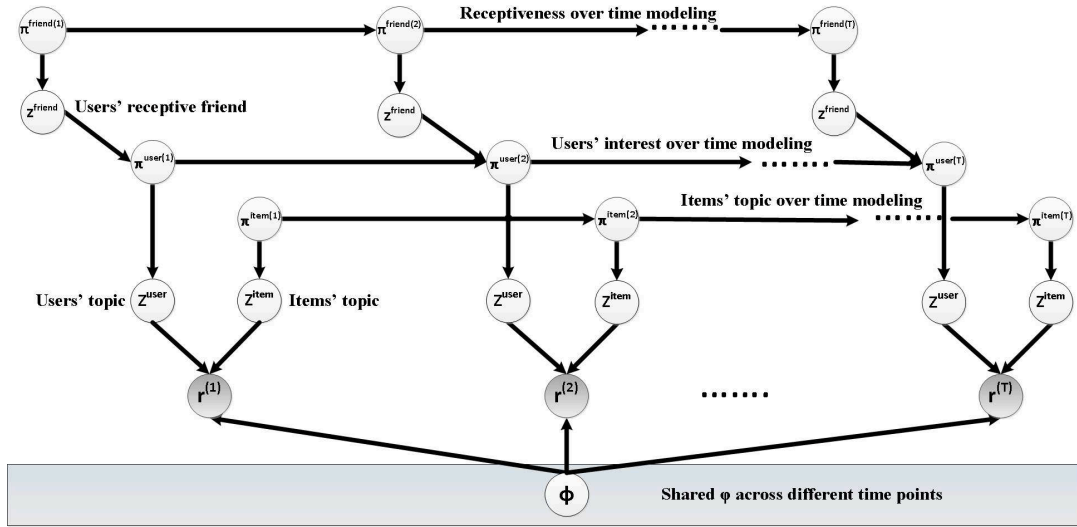


Figure 5-2: Graphical model representation of RTM Model

$$\begin{aligned}
 P(f = z_{u,m}^{friend} | \Theta \setminus z_{u,m}^{friend}, x) &\propto \left( (n_{u,f,t}^{friend})^{-(u,m,t)} + \lambda \pi_{friend(t-1)} \right) \\
 &\times \left( (n_{f,k,t}^{user})^{-(u,m,t)} + \lambda \pi_{user(t-1)} \right) \left( \frac{N_{k,l,x}^{-(u,m,t)} + \beta}{\sum_r (N_{k,l,r}^{-(u,m,t)} + \beta)} \right)
 \end{aligned} \quad (5.6)$$

where  $(n_{u,f,t}^{friend})^{-(u,m,t)}$  denotes the number of times that user  $u$  is receptive to  $f$  in all the ratings at time  $t$  excluding  $r_{u,m}^{(t)}$ .  $(n_{f,k,t}^{user})^{-(u,m,t)}$  denotes the number of times user  $f$  will be assigned to user topic  $k$  in all the ratings at time  $t$  except for  $r_{u,m}^{(t)}$ .  $N_{k,l,r}$  represents the number of times the user with user topic  $k$  has rated item with item topic  $l$  with the rating  $r$ ,  $N_{k,l,x}^{-(u,m,t)}$  represents the number of times the observed rating  $x$  has been given by user with user topic  $k$  on item with item topic  $l$  excluding  $r_{u,m}^{(t)}$ . Similar change can be applied to the Equation (5.3) and (5.4) by introducing an additional temporal dimension for the counter, that is:

$$\begin{aligned}
 P(z_{u,m}^{user} = k | \Theta \setminus z_{u,m}^{user}, x) \\
 \propto \left( (n_{f,k,t}^{user})^{-(u,m,t)} + \lambda \pi_{u,k}^{user(t-1)} \right) \left( \frac{N_{k,l,x}^{-(u,m,t)} + \beta}{\sum_r (N_{k,l,r}^{-(u,m,t)} + \beta)} \right)
 \end{aligned} \quad (5.7)$$

$$\begin{aligned}
P(z_{u,m}^{item} = l | \Theta \setminus z_{u,m}^{item}, x) \\
\propto \left( (n_{m,l,t}^{item})^{-(u,m,t)} + \lambda \pi_{u,k}^{item(t-1)} \right) \left( \frac{N_{k,l,x}^{-(u,m,t)} + \beta}{\sum_r N_{k,l,r}^{-(u,m,t)} + \beta} \right)
\end{aligned} \tag{5.8}$$

With this, based on the Eq. (5.5), the parameter  $\pi^{friend}$ ,  $\pi^{user}$ ,  $\Phi$  and  $\pi^{item}$  can be constructed as:

$$\begin{aligned}
\pi_{u,f}^{friend(t)} &= \frac{n_{u,f,t}^{friend} + \lambda \pi_{u,f}^{friend(t-1)}}{\sum_{f \in F(u)} (n_{u,f,t}^{friend} + \lambda \pi_{u,f}^{friend(t-1)})} \\
\pi_{u,k}^{user(t)} &= \frac{n_{u,k,t}^{user} + \lambda \pi_{u,k}^{user(t-1)}}{\sum_k (n_{u,k,t}^{user} + \lambda \pi_{u,k}^{user(t-1)})} \\
\pi_{m,l,t}^{item(t)} &= \frac{n_{m,l,t}^{item} + \lambda \pi_{m,l}^{item(t-1)}}{\sum_l (n_{m,l,t}^{item} + \lambda \pi_{m,l}^{item(t-1)})} \\
\Phi_{k,l,x} &= \frac{N_{k,l,x} + \beta}{\sum_r (N_{k,l,r} + \beta)}
\end{aligned} \tag{5.9}$$

The process of generating time series ratings is summarized as follows:

1. Choose a  $K \times L$  distribution over ratings  $\Phi \sim Dir(\beta)$
2. For time  $t = 1$ , choose a distribution over friends  $F(u)$  for each user  $\pi_u^{friend(1)} \sim Dir(\alpha^{friend})$ ;  
For time  $t > 1$ , choose a distribution over friends  $F(u)$  for each user  $\pi_u^{friend(t)} \sim Dir(\lambda \pi^{friend(t-1)})$
3. For time  $t = 1$ , choose a distribution over  $K$  users' topic for each user  $\pi_u^{user(1)} \sim Dir(\alpha^{user})$ .  
For time  $t > 1$ , choose a distribution over  $K$  users' topic for each user  $\pi_u^{user(t)} \sim Dir(\lambda \pi^{user(t-1)})$ .

4. For time  $t = 1$ , choose a distribution over  $L$  items' topic for each item  $\pi_m^{item(1)} \sim Dir(\alpha^{item})$ .

For time  $t > 1$ , choose a distribution over  $L$  items' topic for each item  $\pi_m^{item(t)} \sim Dir(\lambda \pi_m^{item(t-1)})$ .

5. For each rating  $x = r_{u,m}^{(t)}$ :

- Choose user friend

$$f = z_{u,m}^{friend} \sim Multinomial(\pi_u^{friend(t)})$$

- Choose user topic

$$k = z_{u,m}^{user} \sim Multinomial(\pi_f^{user(t)})$$

- Choose item topic

$$l = z_{u,m}^{item} \sim Multinomial(\pi_m^{item(t)})$$

- Choose a rating  $r_{u,m}^{(t)} \sim \Phi_{k,l}$

Based on the generative process, we can design Gibbs sampling to infer the latent variables as shown in Algorithm 5.

Note that the cost of running a full Gibbs iteration is  $O(p)$  where  $p$  is the total number of rating observations.

### 5.3.2 Applications of RTM

In this section, we discuss how the RTM model can be used for rating prediction, tracking receptiveness over time, and analyzing users interest change.

- **RTM-based Rating Prediction**

Having obtained the RTM model, we predict the rating made by user  $u$  on item  $m$  at time point  $t$  as follows:

$$\hat{r}_{u,m}^{(t)} = \sum_{f \in F(u)} \pi_{u,f}^{friend(t)} [\pi_u^{user(t)}]^\top \Phi \pi_m^{item(t)}$$

---

**Algorithm 5:** Gibbs sampling for RTM

---

**input** : Users' rating histories over time  $R=\{R^{(1)}, \dots, R^{(T)}\}$ ,  
 users' trust relation over time  
 $S=\{S^{(1)}, \dots, S^{(T)}\}$ ,  
 $K, L$  and  $\lambda$

**output:**  $\pi^{user(t)}, \Phi, \pi^{item(t)}$  and  $\pi^{friend(t)}$

- 1 /\*Initialization of the latent variables and counters\*/
- 2 Random initialize  $z^{friend}, z^{user}$  and  $z^{item}$
- 3 Initialize  $n^{friend}, n^{user}$  and  $n^{item}$  and  $N$  as 0
- 4 **foreach**  $x = r_{u,m}^{(t)} \in X$  **do**
- 5      $f = z_{u,m}^{friend}, k = z_{u,m}^{user}$  and  $l = z_{u,m}^{item}$
- 6     Increase the counter of  $n_{u,f,t}^{friend}, n_{u,k,t}^{user}, n_{m,l,t}^{item}$  and  $N_{k,l,x}$
- 7 **end**
- 8 /\* for each rating  $x = r_{u,m}^{(t)}$  in  $X$  \*/
- 9 **for**  $index=1$  to  $Iter$  **do**
- 10   **foreach**  $x = r_{u,m}^{(t)} \in X$  **do**
- 11     /\* Sample Friends \*/
- 12      $f = z_{u,m}^{friend} \sim p(z_{u,m}^{friend} | \Theta \setminus z_{u,m}^{friend}, x)$  according to Equation (5.6)
- 13     /\* Sample Users' Topics \*/
- 14      $k = z_{u,m}^{user} \sim P(z_{u,m}^{user} | \Theta \setminus z_{u,m}^{user}, x)$  according to Equation (5.7)
- 15     /\* Sample Items' Topics \*/
- 16      $l = z_{u,m}^{item} \sim P(z_{u,m}^{item} | \Theta \setminus z_{u,m}^{item}, x)$  according to Equation (5.8)
- 17     /\* Update Counter \*/
- 18     Update the counter  $n_{u,f,t}^{friend}, n_{u,k,t}^{user}, n_{m,l,t}^{item}$  and  $N_{k,l,x}$
- 19   **end**
- 20 **end**
- 21 /\* Get the mean estimate for  $\pi^{friend(t)}, \pi^{user(t)}, \Phi$  and  $\pi^{item(t)}$  \*/
- 22  $\pi^{friend(t)}, \pi^{user(t)}, \Phi$  and  $\pi^{item(t)}$  can be calculated according to Eq. (5.9).

---

- **Receptiveness Change Analysis**

For a given user  $u$ , the receptiveness of  $u$  to his/her friends (including user  $u$ ) at time point  $t$  is  $\pi_u^{friend(t)}$ . By constructing the receptiveness of other users to user  $u$  in  $T$  time points, we can track the receptiveness of other users on user  $u$  over time:

$$C_u^{friend} = \left[ \pi_u^{friend(1)}, \dots, \pi_u^{friend(t)} \right]$$

where  $C_u^{friend}$  is an  $|U| \times T$  matrix. Each column in  $C_u^{friend}$  can be interpreted as the expected probability where user  $u$  may be receptive to the other  $|U|$  users at time point  $t$ . In other words, we can discover who are the users that  $u$  is most receptive to at the particular time point.

- **User Interest Change Analysis**

For a given user  $u$ , we compute the user  $u$ 's preference over item topic at time point  $t$  as  $[\pi_u^{user(t)}]^\top \Phi$ . By constructing  $u$ 's preference over item topic for all  $T$  time points, we can track  $u$ 's interest change over time:

$$C_u = \left[ [\pi_u^{user(1)}]^\top \Phi, \dots, [\pi_u^{user(t)}]^\top \Phi \right]$$

where  $C_u$  is a  $L \times T$  matrix. Each column in  $C_u$  can be interpreted as the expected ratings provided by user  $u$  on all the  $L$  item topic at time point  $t$ . By sorting the columns, we can discover what kind of items are preferred by a user at a certain time point.

## 5.4 Experimental results

In this section, we evaluate the effectiveness of the proposed RTM model that utilizes both time-stamped rating data and trust over time for users' acceptance on recommendation in terms of rating prediction. We also implement two variants of RTM:



- **RTM-StaticSocial**: this variant assumes that the social trust does not change over time while user interest may shift over time as reflected by the time-stamped rating data. This is achieved by using the same social trust information for all the time points.
- **RTM-StaticInterest**: this variant assumes that the user interest does not change over time and only the social trust changes over time. This is achieved by using the same rating information for all the time points.

We compare the proposed models with the following state-of-the-art recommender methods for rating prediction:

1. **Probabilistic Matrix Factorization (PMF)** [63]. This is a matrix factorization based CF algorithm that utilizes static user ratings on items. No social trust information is used.
2. **Bi-LDA** [57] This is a generative model that also utilizes static ratings for prediction. Again no social trust information is employed.
3. **TimeSVD++**[34] This is temporal CF algorithm that assumes user interests change over time and is the baseline of temporal CF methods. This method does not incorporate social trust information.
4. **SocialMF** [28]. This is a social CF algorithm that utilizes the social trust information and is the baseline of social CF methods. This method does not consider the shift in user interest.

Table 5.3 gives a summary of the various methods. All the experiments are carried out on an Intel Core i7-2600 CPU with 8GB RAM, running Windows 7-64 bit.

We use the Epinions dataset<sup>1</sup> in our experiments. This dataset comprises of two files. The first file contains 717,667 user trust statements with time-stamps, while the second

---

<sup>1</sup><http://www.trustlet.org/wiki/ExtendedEpinionsDataset>

Table 5.3: Summary of methods.

Data used	<i>No social trust</i>	<i>Static social trust</i>	<i>Social trust over time</i>
<i>Static rating</i>	Bi-LDA, PMF	SocialMF	RTM-StaticInterest
<i>Rating over time</i>	TimeSVD++	RTM-StaticSocial	RTM

file contains 13,668,319 users’ ratings provided by 120,492 users on 755,760 articles on a scale of 1 to 6, with 1 being not helpful and 6 being most helpful. Each rating is associated with a time-stamp over the period from February 2001 to July 2002.

We sort the data according to the time-stamps and split the data into 6 equal time slices. Each time slice corresponds to about 3 months. We use the first 5 time slices of data as the training data, and the last time slice for testing. We also filter out users that have made less than 10 unique ratings. After pre-processing, we obtain 5,077,392 users’ ratings with 9,149 users and 116,697 articles and 236,878 social relations. Table 5.4 summarizes the statistics of the rating dataset.

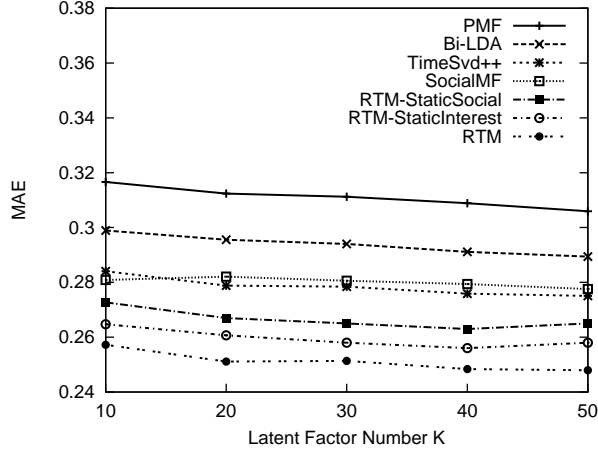
Table 5.4: Statistics of rating dataset.

Statistics	Users	Movies
Min. # of ratings	10	1
Max. # of ratings	39,467	1134
Mean. # of ratings	$554.96 \pm 1681.11$	$43.50 \pm 41.04$

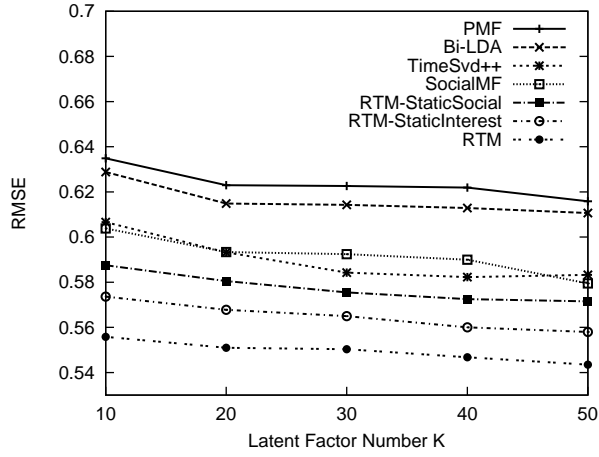
### 5.4.1 Experiments on Users’ Acceptance

We evaluate the task of users’ acceptance on item recommendation utilizing both time-stamped rating data and trust over time. In this set of experiments, we compare the performance of the various methods. We use the standard evaluation metrics Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) as our measurement defined as follows:

$$MAE = \frac{\sum_{r_i \in D} (r_i - \hat{r}_i)}{|D|}$$



(a) MAE



(b) RMSE

Figure 5-3: Accuracy of Rating Prediction

$$RMSE = \sqrt{\frac{\sum_{r_i \in D} (r_i - \hat{r}_i)^2}{|D|}}$$

where  $D$  denotes the test dataset,  $r_i$  is the actual rating and  $\hat{r}_i$  is the predicted rating. A smaller value of MAE or RMSE indicates a better performance.

Figure 5-3 shows the results when we vary the number of user/item dimensions from 10 to 50. We observe that the proposed RTM model has the lowest MAE and RMSE, demonstrating that capturing the dynamic interest between user interest and social trust can improve the rating prediction accuracy. In particular, RTM model lowers the RMSE

(MAE) by as much as 7.71% (8.26%) compared to the SocialMF model, and 8.14% (9.29%), compared to TimeSVD++.

Both SocialMF and RTM-StaticInterest outperform conventional CF models that do not incorporate trust information, namely, Bi-LDA and PMF. This indicates that social trust can help improve the rating prediction accuracy. Both TimeSVD++ and RTM-StaticSocial model user interest over time and thus perform better than Bi-LDA and PMF.

## 5.4.2 User Interest Change Case Study

Here, we visualize the user interest profile obtained from the RTM model over time. Figure 5-4 shows the interest profiles of 2 users from the Epinions dataset. We observe that the user 739's interests remains stable over the time, as indicated by his/her high preference for user latent topic 1 throughout the 6 time points. User 365's main interest is in the latent topic 4 from time points 1 to 3, and changes to latent topics 9 from time point 4 to 6, showing a shift in his/her interest.

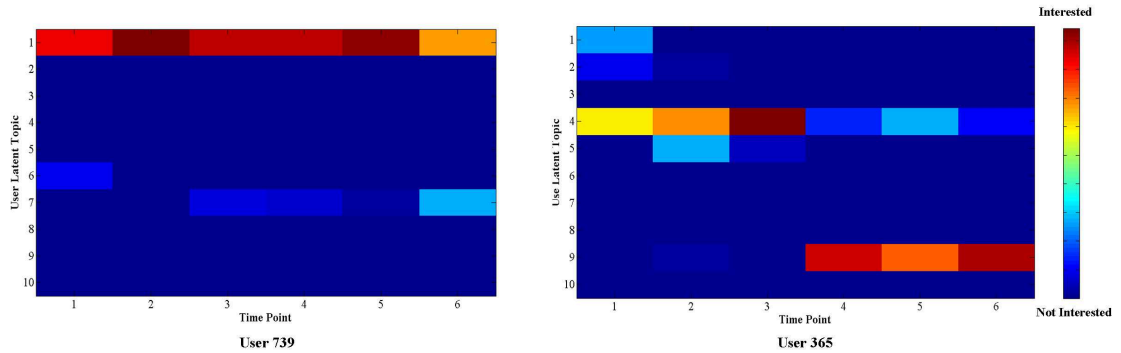


Figure 5-4: User interest change over time

On closer examination, we find that user 739 has rated a lot of reviews in the topic with id 72 for all the time points. On the other hand, user 365 mainly rated reviews on the topic with id 549 from time points 1 to 3, and then change to rate reviews on the topic with id 447 from time points 4 to 6. This confirms that the interest profiles obtained from

the RTM model can capture user interest change.

### 5.4.3 User Receptiveness Case Study

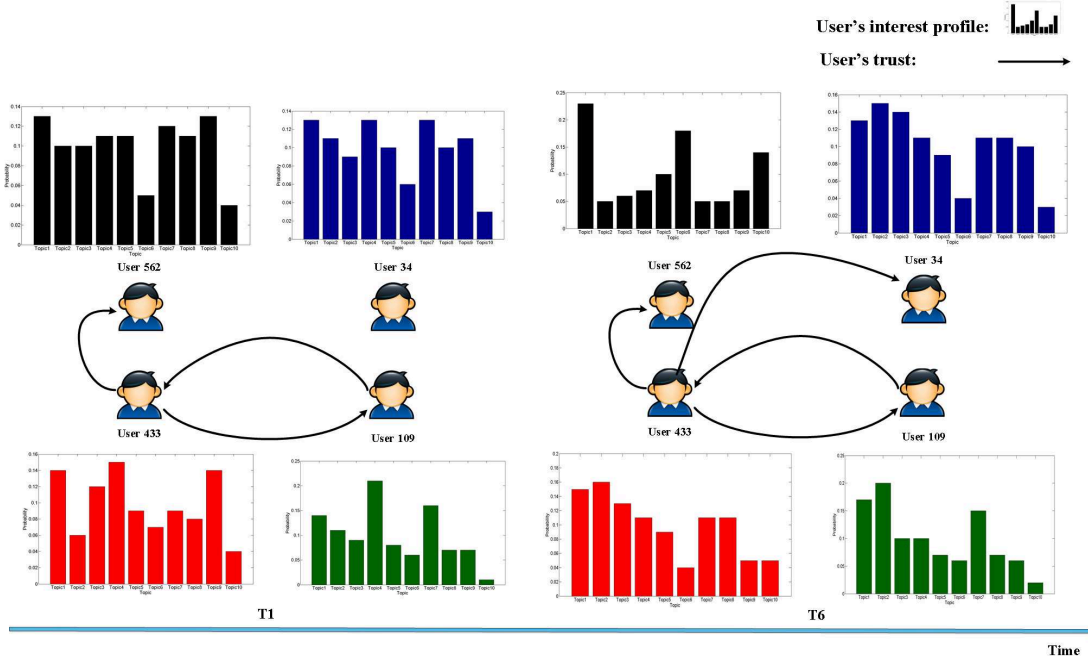


Figure 5-5: User interest profiles and their trust relationships

Next, we analyze the user interest profiles and their social trust relationships over time. Figure 5-5 shows the interest profiles of 4 users and their social trust relationships at time points T1 and T6. Suppose user 433 is our target user. We note that at time point T1, user 433 does not know user 34 and their interest profiles are quite different. However at time point T6, user 34 has become user 344's friend and his/her interest profile has shifted to become similar to that of user 344. Looking at Figure 5-6 which shows the receptiveness of user 433 towards the other 3 users over time, we observe that the receptiveness of user 433 to user 34 increases sharply at T6. This indicates that the RTM model captures the dynamic interaction between user interests and social relationships faithfully.

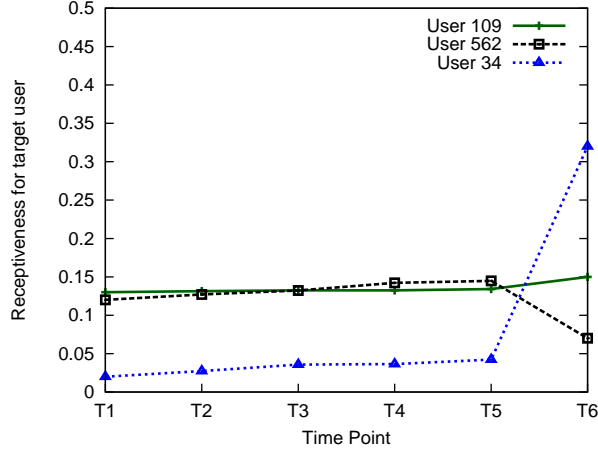


Figure 5-6: Receptiveness change over time

#### 5.4.4 Sensitivity Experiments

In this section, we examine the effect of various parameters on the performance of the RTM model.

##### Effect of varying $K$ and $L$

Table 5.5 shows the RMSE of RTM as we vary the number of user topic  $K$  and the number of item topic  $L$  from 10 to 50. We observe that RMSE does not vary much. The best performance is achieved by setting  $K = 40$  and  $L = 50$ .

Table 5.5: Effect of  $K$  and  $L$  on RMSE

$K \backslash L$	10	20	30	40	50
10	0.5572	0.5512	0.543	0.5419	0.5420
20	0.5532	0.5473	0.5447	0.5428	0.5443
30	0.5718	0.5518	0.5428	0.5434	0.5417
40	0.5534	0.5417	0.5412	0.5431	0.5367
50	0.5521	0.5447	0.5401	0.5414	0.5439

##### Effect of varying $\lambda$

Recall that the parameter  $\lambda$  control how much the prior information is transferred from the previous time slice to the current time slice. When  $\lambda = 0$ , no prior information is

used.

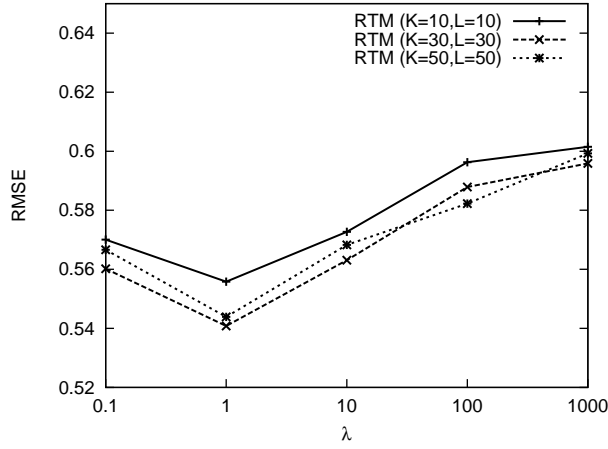


Figure 5-7: Sensitivity analysis on  $\lambda$

Figure 5-7 shows the RMSE obtained for varying  $\lambda$  values. We observe that the best performance is obtained when  $\lambda = 1$ , indicating that prior information helps to improve item rating prediction.

## 5.5 Summary

In this chapter, we have motivated the need to capture the dynamic interaction between trust and user interest for recommendation. We have designed the RTM generative model that incorporates user interest and social trust relationships over time. We have also devised efficient algorithms to learn the latent variables in the RTM model using Gibbs sampling. Experimental results have shown that RTM-based recommendation outperforms state-of-the-art CF methods. In addition, the model provides easy interpretations to allow easy visualization of users' receptiveness and interest change over time.

---

---

## CHAPTER 6

---

### CONCLUSION

In this thesis, we have investigated improving user's acceptance for recommender systems using three popular data. We have reviewed the current work in the area of tagging data, cross domain data and social trust data in recommender system. Although there has been a lot of works in these areas, there remain challenges to be addressed. This thesis has focused on three research problems.

The first research has dealt with increasing the users' acceptance by capturing the explicit and implicit preference with rating and tagging information. We exploit a quaternary relationship among users, items, tags and ratings. We have shown that ternary relationship among user, item and ratings which are insufficient to provide accurate recommendations. Instead, we have modeled the quaternary relationship among users, items, tags and ratings as a 4-order tensor and casted the recommendation problem as a multi-way latent semantic analysis problem. A unified framework for user recommendation, item recommendation, tag recommendation and item rating prediction has been proposed. The results of extensive experiments performed on a real world dataset have demonstrated that our unified framework outperformed the state-of-the-art techniques in all the four recommendation tasks. To the best of our knowledge, this is the first work



to explore the use of the quaternary relationship among users, items, tags and ratings for recommendation tasks.

Second, we have investigated the problem of increasing users' acceptance using cross domain data setting. We have presented a novel collaborative filtering method for integrating social network and cross domain network in a unified framework via latent feature sharing and cluster-level tensor sharing. This framework utilizes data from multiple domains and allows the transfer of useful knowledge from auxiliary domain to the target domain. The results of extensive experiments performed on a real world dataset have demonstrated that our unified framework outperforms the state-of-the-art techniques in all the three recommendation tasks. We have also implemented the algorithm on a map-reduce infrastructure and have shown its scalability.

Finally, we have motivated the need to capture the dynamic interaction between trust and user interest for increasing users' acceptance in recommendation. We have designed the RTM generative model that incorporates user interest and social trust relationships over time. We have also devised efficient algorithms to learn the latent variables in the RTM model using Gibbs sampling. Experimental results have shown that RTM-based recommendation outperforms state-of-the-art CF methods. In addition, the model provides easy interpretations to allow easy visualization of users' receptiveness and interest change over time.

## 6.1 Future Work

First, with the popularity of different social media applications (e.g. foursquare), we have additional user-generated data such as geo-location data. This creates an even more complex relationship that extend beyond quaternary relationships. One possible direction for future work is to extend the QSA framework to create higher-order tensor that can take into consideration geographical influence so as to model users' profiles and capture users' interest more accurately.

Second, *FUSE* assumes that the source and target domains are related to each other in some sense. However, when this assumption is not true, negative transfer may result and the learner can perform worse than if no transfer takes place at all. Given a target domain/task, it is an important research question on how to find related source/auxiliary domains/tasks to ensure positive transfer.

Third, besides accuracy and transparency, diversity, serendipity and trust are also important factors in improving the users' acceptance. For example, the recommenders may always recommend popular movies such as Avatar to users, this not good if the user has already seen the recommendation before. User wants novel recommendation and not the items he/she already knows. Increasing the diversity and serendipity of recommendation is an important research direction.

Finally, the availability of big data presents many exciting opportunities to develop algorithms and to build scalable and robust recommender systems that can adapt and learn from bulk quantities of dynamic real-world data in a life-long learning manner.



---

## BIBLIOGRAPHY

- [1] Nonnegative tucker decomposition. In *CVPR*. IEEE Computer Society, 2007.
- [2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.
- [3] Charu C. Aggarwal, Joel L. Wolf, Kun-Lung Wu, and Philip S. Yu. Horting hatches an egg: a new graph-theoretic approach to collaborative filtering. *KDD*, pages 201–212, 1999.
- [4] Xavier Amatriain, Neal Lathia, Josep M. Pujol, Haewoon Kwak, and Nuria Oliver. The wisdom of the few: a collaborative filtering approach based on expert opinions from the web. In *SIGIR*, pages 532–539. ACM, 2009.
- [5] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, pages 66–72, March 1997.
- [6] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: using social and content-based information in recommendation. *AAAI/IAAI*, pages 714–720, 1998.
- [7] Michael W. Berry, Zlatko Drmac, and Elizabeth R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Rev.*, 41(2):335–362, June 1999.
- [8] Mustafa Bilgic and Raymond J. Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Proceedings of Beyond Personalization 2005: A Workshop on the Next Stage of Recommender Systems Research at the 2005 International Conference on Intelligent User Interfaces*, 2005.
- [9] Daniel Billsus and Michael J. Pazzani. A personal news agent that talks, learns and explains. In *Proceedings of the Third International Conference on Autonomous Agents*, pages 268–275. ACM Press, 1999.
- [10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

- [11] Toine Bogers. *Recommender Systems for Social Bookmarking*. PhD thesis, Tilburg University, December 2009.
- [12] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *UAI*, pages 43–52, 1998.
- [13] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.
- [14] Bin Cao, Nathan Nan Liu, and Qiang Yang. Transfer learning for collective link prediction in multiple heterogenous domains. *ICML*, 2010.
- [15] G Casella and E I George. Explaining the Gibbs sampler. *The American Statistician*, 46:167–174, 1992.
- [16] Maurice Coyle and Barry Smyth. (web search)shared: Social aspects of a collaborative, community-based search network. In *Proceedings of the 5th international conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, AH, pages 103–112, 2008.
- [17] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.
- [18] Chrysanthos Dellarocas. The digitization of word of mouth: Promise and challenges of online feedback mechanisms. *Manage. Sci.*, 49(10):1407–1424, October 2003.
- [19] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22:143–177, 2004.
- [20] Jennifer Ann Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, College Park, MD, USA, 2005.
- [21] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, 4(2):133–151, July 2001.
- [22] J.L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. *CSCW*, pages 241–250, 2000.
- [23] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. *SIGIR*, pages 230–237, 1999.
- [24] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.
- [25] Thomas Hofmann. Probabilistic latent semantic indexing. *SIGIR*, pages 50–57, 1999.

- [26] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. *IJCAI*, pages 688–693, 1999.
- [27] Mohsen Jamali and Martin Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. *KDD*, pages 397–406, 2009.
- [28] Mohsen Jamali and Martin Ester. A transitivity aware matrix factorization model for recommendation in social networks. *IJCAI*, pages 2644–2649, 2011.
- [29] Rong Jin, Joyce Y. Chai, and Luo Si. An automatic weighting scheme for collaborative filtering. *SIGIR*, pages 337–344, 2004.
- [30] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *CIKM*, pages 247–254, 2001.
- [31] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. *RecSys*, pages 165–172, 2011.
- [32] Tamara G. Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *ICDM*, pages 363–372, 2008.
- [33] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.
- [34] Yehuda Koren. Collaborative filtering with temporal dynamics. *KDD’09*, pages 447–456, 2009.
- [35] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.
- [36] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.
- [37] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562. MIT Press, 2000.
- [38] Wee Sun Lee. Collaborative learning and recommender systems. *ICML ’01*, pages 314–321, 2001.
- [39] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate?: cross-domain collaborative filtering for sparsity reduction. *IJCAI*, pages 2052–2057, 2009.
- [40] Bin Li, Qiang Yang, and Xiangyang Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. *ICML*, pages 617–624, 2009.
- [41] Tzu-Kuo Huang, Jeff Schneider, Jaime G. Carbonell, Liang Xiong, Xi Chen. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of SIAM Data Mining*, 2010.

- [42] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. volume 7, pages 76 – 80, jan/feb 2003.
- [43] Chao Liu, Hung-chih Yang, Jinliang Fan, Li-Wei He, and Yi-Min Wang. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. WWW, pages 681–690, 2010.
- [44] Hao Ma, Irwin King, and Michael R. Lyu. Learning to recommend with social trust ensemble. SIGIR, pages 203–210, 2009.
- [45] Hao Ma, Irwin King, and Michael R. Lyu. Learning to recommend with explicit and implicit social relations. *ACM Trans. Intell. Syst. Technol.*, 2(3):29:1–29:19, May 2011.
- [46] Hao Ma, Michael R. Lyu, and Irwin King. Learning to recommend with trust and distrust relationships. RecSys, pages 189–196, 2009.
- [47] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. WSDM, pages 287–296, 2011.
- [48] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. RecSys, pages 17–24, 2007.
- [49] Raymond J. Mooney, Paul N. Bennett, and Lorie Roy. Book recommending using text categorization with extracted information. In *RECOMMENDER SYSTEMS. PAPERS FROM 1998 WORKSHOP*, pages 49–54. AAAI Press, 1998.
- [50] Raymond J. Mooney and Lorie Roy. Content-based book recommending using learning for text categorization. In *In Proceedings Of 5th ACM Conference On Digital Libraries*, pages 195–204. ACM Press, 1999.
- [51] Cataldo Musto. Enhanced vector space models for content-based recommender systems. RecSys, pages 361–364, 2010.
- [52] Weike Pan, Nathan Nan Liu, Evan Wei Xiang, and Qiang Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *IJCAI*, pages 2318–2323. IJCAI/AAAI, 2011.
- [53] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. In *Proceedings of the 31st international conference on Very large data bases, VLDB*, 2005.
- [54] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Statistics*, 2007(2):2–5.
- [55] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Mach. Learn.*, 27(3):313–331, June 1997.
- [56] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, December 1999.

- [57] Ian Porteous, Evgeniy Bart, and Max Welling. Multi-hdp: A non parametric bayesian model for tensor factorization. *AAAI*, pages 1487–1490, 2008.
- [58] Naren Ramakrishnan, Benjamin J. Keller, Batul J. Mirza, Ananth Y. Grama, and George Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, November 2001.
- [59] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. *KDD*, pages 727–736, 2009.
- [60] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Collaborative Work Conference*, pages 175–186, 1994.
- [61] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An open architecture for collaborative filtering of netnews. pages 175–186. *ACM Press*, 1994.
- [62] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, March 1997.
- [63] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. *Advances in Neural Information Processing Systems*, pages 1257–1264, 2008.
- [64] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. *ICML*, pages 880–887, 2008.
- [65] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, (5):513–523, 1988.
- [66] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.
- [67] J. Ben Schafer, Joseph A. Konstan, and John Riedl. E-commerce recommendation applications. *Data Min. Knowl. Discov.*, 5(1-2):115–153, January 2001.
- [68] Shilad Sen, Jesse Vig, and John Riedl. Tagommenders: connecting users to items through tags. In *WWW*, pages 671–680, May 2009.
- [69] Yelong Shen and Ruoming Jin. Learning personal + social latent factor model for social recommendation. *KDD*, pages 1303–1311, 2012.
- [70] Yue Shi, Martha Larson, and Alan Hanjalic. Tags as bridges between domains: improving recommendation with tag-induced cross-domain collaborative filtering. *UMAP*, pages 305–316, 2011.
- [71] Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. *KDD*, pages 650–658, 2008.



- [72] Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.
- [73] Xiaoyuan Su and Taghi M. Khoshgoftaar. Collaborative filtering for multi-class data using belief nets algorithms. *ICTAI*, pages 497–504, 2006.
- [74] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis. *TKDE*, 22:179–192, February 2010.
- [75] Michael E. Tipping and Chris M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622, 1999.
- [76] Karen H. L. Tso-sutter, Ro Balby Marinho, and Lars Schmidt-thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *In Proceedings of the 2nd ACM Symposium on Applied Computing*. ACM, 1995.
- [77] Lyle Ungar, Dean Foster, Ellen Andre, Star Wars, Fred Star Wars, Dean Star Wars, and Jason Hiver Whispers. Clustering methods for collaborative filtering. AAAI Press, 1998.
- [78] Vishvas Vasuki, Nagarajan Natarajan, Zhengdong Lu, Berkant Savas, and Inderjit Dhillon. Scalable affiliation recommendation using auxiliary networks. *ACM Trans. Intell. Syst. Technol.*, pages 3:1–3:20, 2011.
- [79] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. *SIGIR*, pages 501–508, 2006.
- [80] Quan Wang, Jun Xu, Hang Li, and Nick Craswell. Regularized latent semantic indexing: A new approach to large-scale topic modeling. *ACM Trans. Inf. Syst.*, 31(1):5:1–5:44, January 2013.
- [81] Chen Wei, Wynne Hsu, and Mong Li Lee. A unified framework for recommendations based on quaternary semantic analysis. *SIGIR*, pages 1023–1032, 2011.
- [82] Chen Wei, Wynne Hsu, and Mong Li Lee. Making recommendations from multiple domains. *KDD*, pages 892–900, 2013.
- [83] Chen Wei, Wynne Hsu, and Mong Li Lee. Modeling users receptiveness over time for recommendation. *SIGIR*, pages 373–382, 2013.
- [84] Chen Wei, Wynne Hsu, and Mong Li Lee. Tagcloud-based explanation with feedback for recommender systems. *SIGIR*, pages 945–948, 2013.
- [85] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long- and short-term preference fusion. *KDD*, pages 723–732, 2010.

- [86] Mao Ye, Xingjie Liu, and Wang-Chien Lee. Exploring social influence for recommendation: a generative model approach. *SIGIR*, pages 671–680, 2012.
- [87] Kai Yu, Shenghuo Zhu, John Lafferty, and Yihong Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. *SIGIR*, pages 211–218, 2009.
- [88] Shiwan Zhao, Nan Du, Andreas Nauerz, Xiatian Zhang, Quan Yuan, and Rongyao Fu. Improved recommendation based on collaborative tagging behaviors. *IUI*, pages 413–416, 2008.
- [89] Tom Chao Zhou, Hao Ma, Irwin King, and Michael R. Lyu. Tagrec: Leveraging tagging wisdom for recommendation. *CSE*, pages 194–199, 2009.