Personalizing Recommendation in Micro-blog Social Networks and E-Commerce

Zhao Gang

Bachelor of Engineering

East China Normal University, China

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

[illegible]

Zhao Gang

11 November 2014

ACKNOWLEDGEMENTS

First and foremost I would like to thank my supervisors, Professor Mong Li Lee and Professor Wynne Hsu for their valuable guidance, continuous support, encouragement and freedom to pursue independent works 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 Kian-Lee Tan and Professor Min-Yen Kan, who have provided constructive feedback through GRP to this final thesis. 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 labmates in the Database Research Lab 2 for their supports and friendship especially during the many sleepless night rushing to complete experiments before conference deadline. I will never forget the days we together studying, discussion, playing and eating.

Last but not the least, I would like to thank my parents for their support for past 28 years. Without their encouragement and understanding, it would have been impossible for me to finish my Ph.D study.

TABLE OF CONTENTS

SUMMARY

Microblogs and e-commerce have emerged as two important applications of Web 2.0 technology. Service providers rely heavily on personalized recommender systems to drive sales and social interaction respectively. This thesis seeks to address the challenges of data sparsity and scalability in recommender systems, and proposes methods to improve the performance of personalized recommendation in microblog social systems and e-commerce.

We first examine how the Latent Dirichlet Allocation (LDA) to find latent clusters can be applied to improve user recommendation in microblogs. We utilize the follower-followee relationship and devise an LDA based method to discover communities among the users. These communities capture the hidden interests of users as they actively choose their followees. We apply the state-of-the-art matrix factorization approach on each community and generate the final top-k recommendation based on the recommendation lists obtained in each community. Extensive experiments on real world Twitter and Weibo data sets demonstrate that the proposed framework is scalable and effective in reducing the data sparsity of each community.

Next, we investigate the problem of product recommendation from the perspective that the value of a product for a user changes over time. We observe that the intervals between user purchases may influence a users purchase decision, and propose a framework that utilizes purchase intervals to improve the temporal diversity of the recommendations. Given the scale of users, products and purchase histories in any e-commerce website, it is necessary to efficiently compute the purchase interval between pairs of product for all users. We design an algorithm to compute purchase intervals from users' purchase histories, and incorporate the purchase intervals into a matrix factorization based method. We demonstrate on a real world e-commerce data set that the proposed approach improves the conversion rate, precision and recall, as well as achieve a significantly higher temporal diversity compared to traditional recommender systems.

Finally, we observe that users may have different preferences when purchasing different subsets of items, and the periods between purchases also vary from one user to another. We propose a framework that leverages on LDA to generate clusters that capture users hidden preferences for items as well as item time sensitivity before we apply matrix factorization on each cluster to personalize the recommendations. We introduce the notion of a cluster purchase interval factor which estimates the probability that users in a cluster will purchase an item. Experiment results indicate that our approach is scalable and significantly improves the conversion rate (by up to 10%) of state-of-the art product recommender methods.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the rapid development of Web 2.0 Internet technology, the interaction between people and Internet has been dramatically changed. Users share their photos online in Flickr[1], go to shopping online at Amazon[2], make new friends online on Facebook[3], write their daily updates online in Twitter[4] and watch the latest videos on YouTube[5], etc. The increasing online traffic has resulted in huge economic benefits and challenges for e-service providers, as well as serious information overload for online social network users. E-service providers are keen to invest in technologies to help users make decisions and increase satisfaction of users' online experiences.

Recommender systems have become a core technology to improve user experience in both e-commerce and social networks. A recommender system [72] aims to provide suggestions of items to satisfy users' interest, such as what products to buy, what books to read, what music to listen to, or what people to connect to.

[1]http://www.flickr.com/
[2]http://www.amazon.com/
[3]http://www.facebook.com/
[4]http://www.twitter.com/
[5]http://www.youtube.com/

1.1 Background

Recommender systems can be broadly classified into three types: (a) editorial recommendations, (b) top-k recommendations, and (c) personalized recommendations. Figure 1-1 shows examples of these different types of recommender systems employed in Google Play which recommends Apps to Android OS users. Editorial recommendations are typically made by experts in some specified areas, while top-k recommender systems capture statistics from users to determine the most popular item. However, these two types of recommender systems are not personalized to users. On the other hand, **personalized recommender systems** aim to provide users with recommendation based on their personal preference, and has attracted much attention from researchers in the information retrieval, data mining, machine learning and database communities.

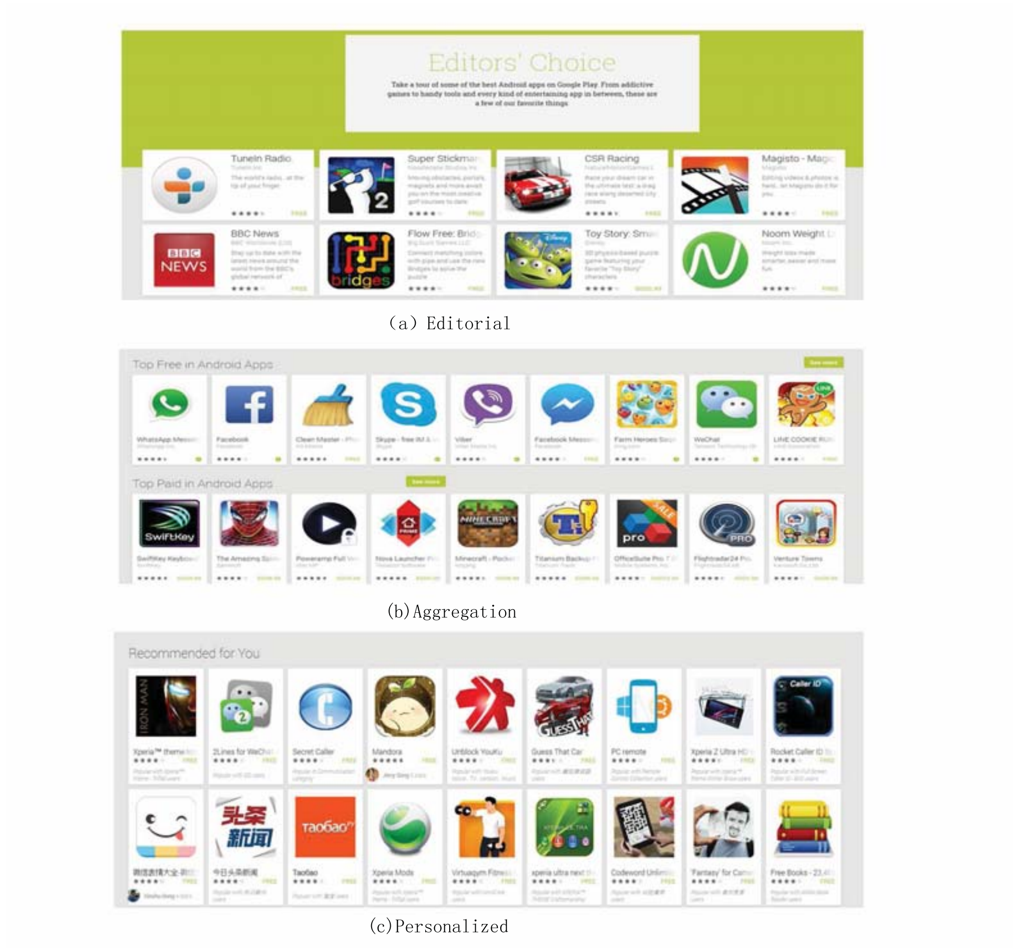


Figure 1-1: Different types of recommender systems

Figure 1-2 gives the general framework of a recommender system. It has the following main components:

- **Items**. Items are the objects that are recommended. Items are characterized by their value or utility. The value of an item indicates the preference from users. The main task of recommender systems is to estimate these item values using a range of properties and features of the items. For example, in a music recommender system, the genre (such as popular, rop, etc.), as well as the singer, and producer can be used to describe a song and to learn the utility of an item related to these features.

- **Users**. In order to personalize the recommendations, recommender systems exploit a range of information about the users' diverse characteristics, including their feedback or attitude to the items such as ratings, and personal particulars (age, salary and geographic information, etc.). Such user information is also known as user profile. Recommender systems utilize user profile to recommend to users items that are preferred by users who have similar profiles.

- **Events**. An event is a recorded interaction between a user and an item. An event typically has the format as $< user, item, feedback >$, which indicates that a $user$ gives a $feedback$ on an $item$. The feedback can be either explicit, e.g., ratings (1-5 stars) provided in the book recommender system, or implicit e.g, a user has observed or purchased an item. Another form of user interaction are tags that users give to items. For instance, in Delicious[1], users utilize tags or discriminative words[52] to describe URLs, e.g. ”job hunting”, or ”java development”.

The objective of a recommender system is to determine a ranked lists of items that are the most suitable products or services for a **target user** based on the user's preferences and constraints learned from user profile. The challenge is to achieve a high user acceptance rate on their recommendations.

[1]http://del.icio.us/

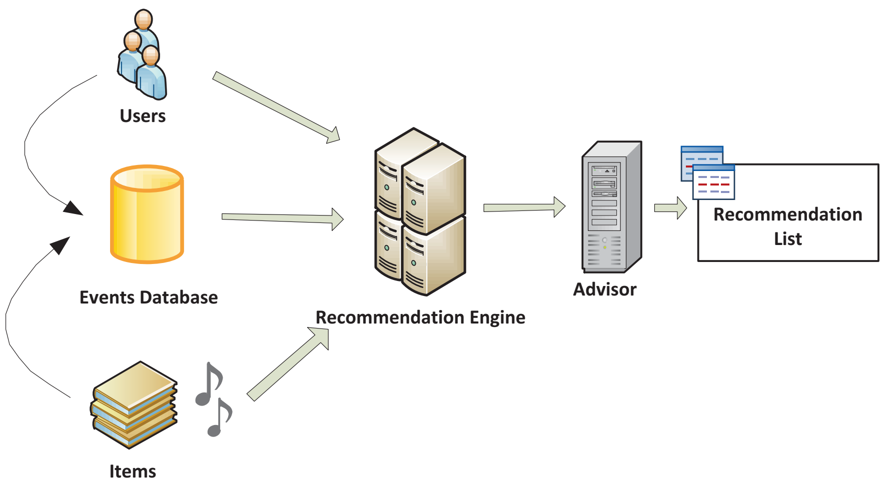


Figure 1-2: General framework of a recommender system

One of the powerful personalization technique is **collaborative filtering**. This method increases user acceptance towards recommendation (filtering) on the interests of a user by collecting preferences or information from many users (collaborating). The system users, e.g., a consumer in Amazon, provide feedback on their past purchase such as good, neutral or bad. Recommender systems record these feedback and construct models to learn what items may be interesting to the users in future. The theory underlying such recommendation systems is that individuals often rely on recommendations provided by peers in making decisions [58]. Recommender systems capture this behavior by leveraging on the recommendations suggested by a community of users to the target user. The rationale is that if a target user has agreed in the past with some users, then the other recommendations coming from these similar users should be relevant as well and are of interest to the target user.

Collaborative filtering techniques have been widely studied in information retrieval and knowledge management research communities. The current state-of-the-art collaborative filtering method is matrix factorization and its variants [47]. However, matrix factorization involves a computationally intensive learning process, and scalability becomes an issue given the huge number of users and items. Further, with limited user feedback on the wide variety of items, data sparsity continues to be a research challenge.

1.2 Motivation

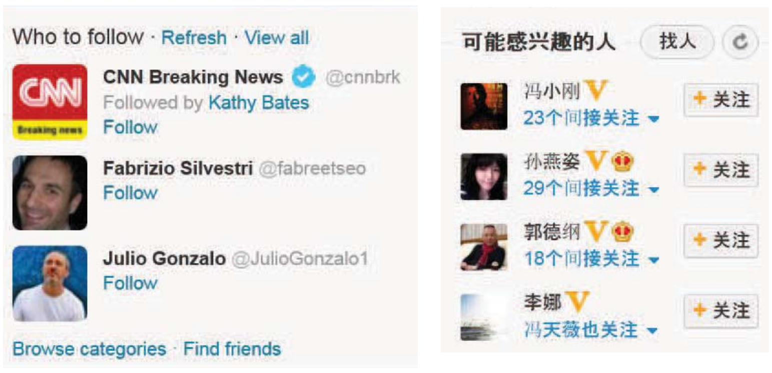
Microblogs and e-commerce have emerged as two important applications of Web 2.0 technology. The service providers rely heavily on personalized recommender systems to drive social interaction and sales respectively. The goal of this thesis is to develop efficient and effective methods for (a) user recommendation in microblogs, and (b) product recommendation in e-commerce systems. We will discuss their specific research challenges and briefly describe our proposed approaches to address them.

1.2.1 User Recommendation in Microblogs

One of the most successful Web 2.0 products is the social network platform, e.g. Facebook and Twitter, which facilitates and enhances relationships among users. The continued success of these social networks relies heavily on their abilities to recommend appropriate and relevant users to drive relationship creation.

One typical user recommendation task is for bi-directional friendship social systems, such as Facebook. The relationships in these systems are reciprocal and model the friendships in the real world. The most commonly employed user recommendation technique in bi-directional social systems is compute the number of overlapping friends, that is, these system will recommend friends who share the most number of friendship (links) with the target user. This makes sense since they assume that two users know each other if there is a link between them.

In contrast to the bi-directional relationships in Facebook, the relationships in Twitter-style social networks or microblogs are **uni-directional** and not necessarily reciprocal. The relationships in microblogs are of the **follower-followee** nature, e.g., the fans follow some super star, but the super star may not want to build friendship with all his/her fans. Figure 1-3 shows the screen shots of followee recommendations in Twitter and Weibo. If the user actually chooses one of the users from the list of recommended top-K users to follow, then we say that the recommendation is successful.


Who to follow · Refresh · View all
CNN Breaking News @cnnbrk
Followed by Kathy Bates
Follow
Fabrizio Silvestri @fabreetseo
Follow
Julio Gonzalo @JulioGonzalo1
Follow
Browse categories · Find friends
可能感兴趣的人
找人
冯小刚
23个间接关注
+关注
孙燕姿
29个间接关注
+关注
郭德纲
18个间接关注
+关注
李娜
冯天薇也关注
+关注

(a) Twitter (a) Weibo

Figure 1-3: Screen shots of followee recommender feature in Twitter and Weibo

Recommending who to follow in microblogs is a challenge because of the limited user profile information. Inferring user preferences from their tweets is also difficult as tweets are inherently noisy. Tweets are typically short (maximum 140 characters) and they are often peppered with acronyms and abbreviations.

The work in [33] investigates the use of combinations of tweet content and follower-followee relationships to recommend users to follow in Twitter. They found that follower-followee relationships are dominant features that capture the interest of users since users actively choose people they are interested in to follow. In this thesis, we examine how follower-followee relationships in Twitter-style social network can be utilized to discover communities and recommend users to follow within these communities. Forming communities for user recommendation in a uni-directional social network reduces data sparsity, and is scalable as the matrix factorization of each community (a subset of the original data set) can be carried out in parallel.

1.2.2 Product Recommendation in E-commerce

A report in [41] reveals that the sales volume of B2C (business-to-Consumer) in China market is about 47 billion RMB yuan (7.5 billion US dollar) in 2011, and is expected to reach 650 billion RMB yuan (103 billion US dollar) in 2013. E-service providers are keen to invest in technologies that help users make purchase decisions and increase the satisfaction of users' online shopping experiences. E-commerce recommender systems aim to produce a personalized list of recommendations that users may be interested to buy. Research [46] has shown that **temporal diversity** is an important facet of such systems, and even randomly changing the recommendation list can improve users' satisfaction with the recommendations [49].

Existing works build models to predict the rating or preference that a user would give to an item, and items with the highest predicted ratings are then recommended to the user [82, 30, 76, 57, 59, 65]. However, these models assume that the value of an item for a user does not change over time, and suffer from the problem of recommending the same or almost same products to users.

The works in [46, 70, 94] examine the temporal dynamics in recommendation systems. [70] consider the order of the items purchased and apply the Markov Chain theory to predict the next item that a user will purchase. [46, 94] design models to capture changes in user preferences for products over time due to external events such as new product offerings, seasonal changes or festive holidays (short-term bias) as well as long term interest. However, the temporal diversity of these works is not high for users who do not make purchases often and the same top-k item will be repeatedly recommended to these users.

Theories in economics and consumer behavior postulate that the value of certain products may change over time, especially if the user has recently purchased them. This is known as the Law of Diminishing Marginal Utility [8]. For example, a user is less likely to buy a second computer or mobile phone if s/he has recently bought one. In contrast, products such as milk, bread and eggs are likely to be purchased over and over again. Thus the value or marginal utility of a product for a user depends on his/her purchase history.

Recent works have applied these theories to recommender systems [51, 90]. The authors in [90] incorporate marginal utility into product recommender systems. They adapt the widely used Cobb-Douglas utility function [23] to model product-specific diminishing marginal return and user-specific basic utility to personalize recommendation. In this thesis, we propose a framework that incorporate purchase intervals for product recommendation. The model in our framework combines purchase interval information in users' purchase histories with marginal utility, and enables us to increase the temporal diversity of the recommended items.

Besides temporal diversity, studies on consumer behavior have shown that the underlying mechanisms governing user purchase behavior is very complex. A user is often interested in more than one subset of products, indicating his/her diverse purchase behavior. Two users may purchase the same product for different reasons, demonstrating the diverse characteristics of a product. In this thesis, we also develop a bi-cluster (i.e., a clustering method which can both capture user's preference and item similarity) based collaborating filtering method, and incorporate temporal information into the recommendation process. Our goal is to find user-item subgroups in the large user-item matrix that effectively capture the users' preferences for items as well as item time sensitivity to increase the **conversion rate**, i.e., the proportion of users who become buyers.

1.3 Contributions of Thesis

Although many recommender systems have been proposed in the literature and developed in real world systems to enhance users' experience in both microblogs and e-commerce, there still exists limitations as described above. This thesis seeks to address the challenges of data sparsity and scalability in recommender systems, and proposes methods to improve the performance of personalized recommendation in microblog social systems and e-commerce. Specifically, the contributions of this thesis are as follows:

- We examine how the Latent Dirichlet Allocation (LDA) [13] method can be used to find latent clusters to improve user recommendation in microblogs. We propose to utilize the follower-followee relationship and devise an LDA based method to discover communities among the users. These communities capture the hidden interests of users as they actively choose their followees. We apply the state-of-the-art matrix factorization approach on each community and generate the final top-k recommendation based on the recommendation lists obtained in each community. The advantages of the proposed framework are: (a) it learns the different user preferences from different communities; (b) the data sparsity of each community is reduced which improves the recommendation performance; (c) it is scalable as the matrix factorization of each community can be performed in parallel. These advantages are confirmed by extensive experiments on real world Twitter and Weibo data sets.

- We approach the problem of product recommendation from the perspective that the value of a product for a user changes over time. We observe that the intervals between user purchases may influence a users purchase decision, and propose a framework to utilize purchase intervals to improve the temporal diversity of the recommendations. Given the scale of users, products and purchase histories in any e-commerce website, it is necessary to efficiently compute the purchase interval between pairs of product for all users. We design an algorithm to compute the purchase intervals from the users' purchase histories, and describe how to incorporate purchase intervals into a matrix factorization based method. We demonstrate on a real world e-commerce data set that the proposed approach improves the conversion rate, precision and recall, as well as achieve a significantly higher temporal diversity compared to traditional recommender systems.

- We also observe that users may have different preferences when purchasing different subsets of items, and the periods between purchases also vary from one user to another. We propose a framework that leverages on LDA to generate clusters that capture the users hidden preferences for items as well as item time sensitivity before we apply matrix factorization on each cluster to personalize the recommendations. We introduce the notion of a cluster purchase interval factor which estimates the probability that users in a cluster will purchase an item. Experiment results indicate that our approach is scalable and significantly improves the conversion rate (by up to 10%) of state-of-the art product recommender methods. We also compare our approach with a non-LDA method to show that the improvement is not simply due to the use of purchase intervals.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows:

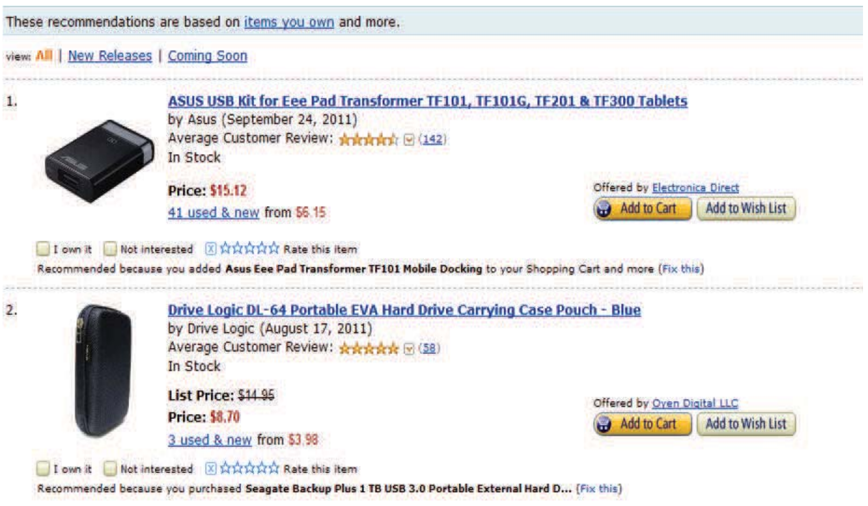
- Chapter 2 presents an comprehensive review on existing techniques in recommender systems, with a focus on techniques used in product recommender and user recommender systems.
- Chapter 3 describes our community-based approach that utilizes follower-followee relationships to find the hidden interests of users and improve user recommendation in microblogs.
- Chapter 4 introduces the purchase interval concept in e-commerce systems, and describes how to utilize the new feature to improve the accuracy and diversity of recommendation.
- Chapter 5 presents our probabilistic approach to generate latent clusters and use purchase intervals to refine the clusters to improve the performance of product recommendation.
- Finally, Chapter 6 concludes the thesis and discusses possible directions for further research.

CHAPTER 2

LITERATURE REVIEW

The manner in which people interact with Internet has changed significantly in the last two decades. The first revolutionary change are search engines such as Google and Baidu. However, search engines are passive as they retrieve items in response to users' queries, while recommender systems are proactive in pushing items that users are interested in. Research in personalized recommender systems emerged in the mid-1990s [35, 81], and they have become a core technology for e-service providers. Amazon is one of the pioneers in using recommendations to drive sales; 25% of their annual sales come from suggesting products to users by showing related books or personalized music recommendations. Figure 2-1 shows sample screen shots of the variety of recommender systems.

Recommender systems play an important role in social networks to help connect people online and promote social interactions. Similarly, recommender systems not only help identify what products to offer to an individual customer, but they also help to increase cross-sell by suggesting additional products to the customers and improve the consumer loyalty because consumers tend to return to the sites that best serve their needs [79, 19]. In this chapter, we will first review the state-of-the-art recommender tech-



These recommendations are based on items you own and more.

view: All | New Releases | Coming Soon

1. ASUS USB Kit for Eee Pad Transformer TF101, TF101G, TF201 & TF300 Tablets
by Asus (September 24, 2011)
Average Customer Review: ★★★★☆ (142)
In Stock

Price: $15.12
41 used & new from $6.15

Offered by Electronica Direct
Add to Cart | Add to Wish List

I own it | Not interested | ☆☆☆☆☆ Rate this item
Recommended because you added **Asus Eee Pad Transformer TF101 Mobile Docking** to your Shopping Cart and more (Fix this)

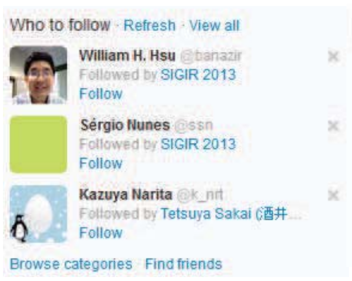
2. Drive Logic DL-64 Portable EVA Hard Drive Carrying Case Pouch - Blue
by Drive Logic (August 17, 2011)
Average Customer Review: ★★★★★ (58)
In Stock

List Price: ~~$14.95~~
Price: $8.70
3 used & new from $3.98

Offered by Oyen Digital LLC
Add to Cart | Add to Wish List

I own it | Not interested | ☆☆☆☆☆ Rate this item
Recommended because you purchased **Seagate Backup Plus 1 TB USB 3.0 Portable External Hard D...** (Fix this)

(a) Amazon recommends products to buy



Who to follow · Refresh · View all

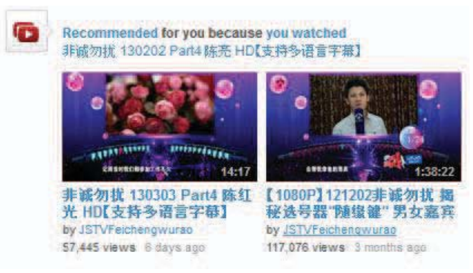
William H. Hsu @banazir
Followed by SIGIR 2013
Follow

Sérgio Nunes @ssn
Followed by SIGIR 2013
Follow

Kazuya Narita @k_nrt
Followed by Tetsuya Sakai (酒井...
Follow

Browse categories · Find friends

(b) Twitter recommends friends



Recommended for you because you watched
非诚勿扰 130202 Part4 陈亮 HD【支持多语言字幕】

14:17
非诚勿扰 130303 Part4 陈红光 HD【支持多语言字幕】
by JSTVFeichengwurao
57,445 views 6 days ago

1:38:22
【1080P】121202非诚勿扰 揭秘选号器"随缘键" 男女嘉宾
by JSTVFeichengwurao
117,076 views 3 months ago

(c) Youtube recommends videos

Figure 2-1: Example of Recommender Systems

niques, followed by the related works in user recommender and product recommender systems.

2.1 Recommendation Techniques

The techniques used in recommender systems can be broadly classified into content-based [82, 66, 33] and collaborative filtering [1, 30, 76, 57].

2.1.1 Content-based Filtering

Content-based filtering aims to find the items whose contents are similar to the items which are previously liked by the target user [61]. The first step in this class of techniques is to build an item vector for each item. Then given a target user, a set of items is generated from the events in the user's profile, and the target user vector based on these items' vectors is built. Finally, the similarity between an unseen item vector and the target user vector is computed.

Note that an item vector is a set of features with associated value or weight. For example, a movie can have features such as author, title and director. The cosine similarity between two vectors i and j is defined as:

$$sim(i, j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}|| \times ||\vec{j}||} \tag{2.1}$$

Content-based filtering has its roots in information retrieval research which automatically extracts content (or important words) from items [5, 9]. Therefore, content-based recommender systems are designed mostly to recommend text-based items, where the content in these systems is usually described with keywords. For instance, the work in [66] recommends website based on the web page contents, and [33] considers users' tweets content to recommend users to follow.

The classical method to weigh the features in the item vector is based on TF-IDF (Term Frequency-Inverse Document Frequency) [74], a metric often used in information retrieval. Term frequency is defined as:

$$TF_{t,d} = \frac{f_{t,d}}{\underset{z}{argmax}\ f_{z,d}} \tag{2.2}$$

where $f_{t,d}$ is the number of occurrence of a term t in a given document d. This is a normalized term frequency representation which considers term t against the maximum frequency over all keywords $f_{z,d}$.

However, some words that appear in a large number of documents are not useful in distinguishing between a relevant document and a non-relevant one. Therefore, the inverse document frequency is combined with TF. The IDF for a term t is defined as

$$IDF_t = log\frac{N}{n_t} \tag{2.3}$$

where N is the number of documents in a given corpus and n_t indicates the number of documents which term t occurred in.

Then the TF-IDF weight for term t in a document d is defined as

$$w_{t,d} = TF_{t,d} \times IDF_t \tag{2.4}$$

The content of a document d with k terms can be modeled with a vector as $d = (w_{1d}, ..., w_{kd})$. We call this the vector space model [75].

The user vector can be similarly formalized. For a target user u, we also use the k words to describe the user and define a vector $u = (w_{1u}, ..., w_{ku})$, where each value in the vector is the user preference. The preference can be learned from the user profile. There are variety of techniques to compute the user vector from the user's profile. For example, the works in [67, 66] use a Bayesian classifier to estimate the probability of user's preference.

Applications that use content-based filtering to make recommendations include news [11, 12, 2], movies [55] and books [61]. The works in [11, 12, 2] allow users to give positive or negative feedback on articles or authors. The user preference vector is based on a fixed number of topics, and all the news items are mapped to the same space, i.e., the same topics as the user preference vector. INTIMATE [55] recommends movies by using text categorization techniques to learn from movie synopses obtained from the IMDB, and LIBRA [61] implements a naive Bayes text categorization method for book recommendation.

However, content-based recommender systems have several limitations [6]:

- **Feature selection.** Content-based recommendation basically associate both users and items with a set of features, and compute the similarities between them to produce the recommendation list. Although this approach works well in extracting features from text documents such as web pages, it is difficult to automatically obtain a relevant set of features from items in domains such as multimedia.

- **Over-specialization.** Content-based recommendation aims to find the items have highest similarity with the items target user liked previously. In another words, the recommender can only push the items that highly match the user's profile, and the items are limited to those that have already been rated. For example, a programmer reading "JAVA" related news may be recommended news about "JAVA island". One solution to this problem is to introduce some randomness, e.g., [12] filters out items if they are either too different from user's preference, or too similar to some item the user has seen before.

2.1.2 Collaborative Filtering

Collaborative filtering addresses the over-specialization problem in content-based recommendation by using information about the user's past behavior and similar users to make suggestions. Collaborative filtering approaches can be categorized into memory-based and model-based methods [14]. Memory-based algorithms utilizes the entire data to make recommendations, while model-based algorithms use the data to learn or train a model which is subsequently used to make predictions.

Memory-based Collaborative Filtering

Memory-based CF algorithms [25, 62, 15, 30, 76, 92] find users that are similar to the target user and use their preferences to predict ratings for the target user. Cosine similarity and Pearson correlation [27] are two standard measures used to determine the similarity between users. Given a target user u_t, various neighborhood selection strategies have been proposed to obtain the set of similar users as U_t. These strategies include applying some threshold [81], finding top-k users with the highest similarity scores [71]. The work in [30] report that selecting the most similar users not only reduce the computational complexity but it also leads to better recommendation results compared to using all users.

The score of a target user u_t on an item i, denoted as $score(u_t, i)$, is estimated based on the rating $r_{u,i}$ assigned to item i by the users $u \in U_t$ who are similar to target user u_t. The score function can be a simple aggregation such as the average ratings of the similar users:

$$score(u_t, i) = \frac{1}{|U_t|} \sum_{u \in U_t} r_{u,i} \tag{2.5}$$

However, this basic scoring is not personalized for the target user. A commonly accepted approach is to use a weighted sum where ratings by users who are more similar to the target user contribute more towards the prediction of the item rating:

$$score(u_t, i) = \frac{\sum_{u \in U_t} sim(u, u_t) \times r_{u,i}}{\sum_{u \in U_t} sim(u, u_t)} \tag{2.6}$$

where $sim(u_t, u)$ is the similarity between target user u_t and user u who has previously rated item i.

Another commonly used scoring function is the adjusted weighted sum method which takes into account the fact that different users may use the rating scale differently.

$$score(u_t, i) = \overline{r_i} + \frac{\sum_{u \in U_t} sim(u, u_t) \times (r_{u,i} - \overline{r_u})}{\sum_{u \in U_t} sim(u, u_t)} \tag{2.7}$$

where $\overline{r_i}$ denotes the average rating of item i and $\overline{r_u}$ is the average of the all ratings made by user u previously.

The above three scoring functions are designed for recommender systems that utilizes explicit feedback (i.e., rating). The work in [96] propose the following scoring function for recommender systems that utilizes implicit feedback such as purchase history, watching habits and browsing activity to model user preferences:

$$score(u_t, i) = \sum_{u \in U_t} sim(u_t, u) \times b_{u,i} \tag{2.8}$$

where $b_{u,i} = 1$ if user u has observed item i and $b_{u,i} = 0$ otherwise.

The works in [76, 50] use the similarity between items instead of users to predict the score that a target user will give to an item. The preference of user u_t to item i can be obtained by computing the sum of the ratings given by u_t on items that are similar to i. Each rating is weighted by the similarity $sim(i, j)$ between items i and j.

$$score(u_t, i) = \frac{\sum\limits_{j \in ItemSet(u_t)} sim(i, j) \times r_{u,j}}{\sum\limits_{j \in ItemSet(u_t)} sim(i, j)} \tag{2.9}$$

where $ItemSet(u_t)$ is set of items rated previously by target user u_t.

A more sophisticated item similarity based approach is the slope one predictor [50] which takes into consideration the average difference between the ratings of one item and another for users who rated both:

$$score(u_t, i) = r_{u_t,j} + dev_{j,i} \tag{2.10}$$

where $dev_{j,i}$ is the average rating deviation of item i with respect to j, defined as:

$$dev_{j,i} = \frac{\sum_{u \in U(i,j)} r_{u,i} - r_{u,j}}{|U_{i,j}|} \tag{2.11}$$

where $U_{i,j}$ denotes the set of users who has previously rated both items i and j. If we can compute $dev_{j,i}$ which indicates that users tend to rate j approximately some rating (e.g., 1.5 stars) higher than the rating on i, then we predict the unknown rating of user u_t on i as $score(u_t, i) = r_{u_t,j} + dev_{j,i}$. Let $coItemSet(u_t, i)$ be the set of items that has been rated by target user u_t, and has been co-rated with item i by at least one user. Then we can vary item j in Equation 2.10 as follows:

$$score(u_t, i) = \frac{1}{|coItemSet(u_t, i)|} \sum_{j \in coItemSet(u_t,i)} r_{u_t,j} + dev_{j,i} \tag{2.12}$$

Equation 2.12 can be extended to incorporate user similarity into the term $dev_{j,i}$ and item similarity into the term $r_{u_t,j}$. The work in [93] combines it with a user-based CF algorithm to improve the prediction performance.

Model-based Collaborative Filtering

In contrast to memory-based CF, model-based CF algorithms [10, 38, 56, 45, 97, 42, 47] use the collection of user ratings on items to learn a model, which is then used to make rating predictions. Model-based recommender methods incorporate techniques from statistics, data mining and machine learning, including Singular Value Decomposition (SVD) [47], probabilistic Latent Semantic Analysis (pLSA) [37], Bayesian model [21], Latent Dirichlet Allocation (LDA) [13] and approaches for implicit feedback [42, 69].

Model-based CF algorithms takes as input the user-item-rating data in the form of a matrix R with M items and N users. Each element $r_{u,i}$ in R corresponds to the rating user u gives to item i, where $r_{u,i} = 0$ if u has not rated i before. Then the recommendation algorithm aims to determine the zero entries in R which are consider items unknown to users.

One challenge faced in building the model is the extremely sparse data set, typically $> 99\%$ in most of the applications. Many algorithms utilize feature selection to reduce the dimensionality in order to mitigate the sparsity problem. One of the most successful dimensionality reduction techniques is SVD [47].

Dimensionality reduction is achieved by introducing K hidden (latent) variables which try to capture the preferences of users and attributes of items. The original R is then approximated by the product of two matrices

$$R \approx W \cdot V \tag{2.13}$$

where W and V are matrices with dimensions $N \times K$ and $K \times M$ respectively, $K << M$ and $K << N$. This approach is also known as Matrix Factorization (MF) approach.

After obtaining the matrices W and V, each user is represented as a vector p_u which is a row vector of V, and each item is represented as a vector q_i which is a column vector of W. Then we can estimate the rating $\hat{r}_{u,i}$ that a user u will give to some unknown item

i by the dot product of p_u and q_i as follows:

$$\hat{r}_{u,i} = p_u \cdot q_i \tag{2.14}$$

Essentially, q_i is a vector which contains some latent features or characteristics of the item, while p_u is the vector that depicts the user's preference of corresponding characteristics of the item.

Various methods have been proposed to approximate W and V. Suppose $\hat{R}$ denotes the dot product of W and V. Then the simplest way [86] to learn $\hat{R}$ is by minimizing the Frobenius Norm of the matrix $R - \hat{R}$:

$$\hat{R} = \underset{R'}{argmin} ||R - R'|| \tag{2.15}$$

The work in [47, 64] shows that the accuracy can be improved through a regularized model to avoid the overfitting problem. Hence, to learn the vectors p_u and q_i, the algorithm minimizes the regularized squared error on the set of known ratings:

$$\underset{p*,q*}{argmin} \sum_{(u,i)\in D} (r_{u,i} - p_u q_i)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{2.16}$$

The above model based matrix factorization approach have become a dominant method in collaborative filtering recommender systems [47], as demonstrated in the Netflix competition.

Since collaborative filtering systems use other users' rating to make recommendations, they can deal with any kind of content and recommend any items, even the ones that are dissimilar to those seen in the past. However, collaborative filtering recommender systems have their own limitations:

Sparsity. In many real applications, the number of items and users are often exceedingly large (e.g., millions of items in Amazon). This causes the overlap between two users to be very small, or even none, which imposes a limit to the performance of collaborative filtering.

Cold start. When there is a new user or a new item, collaborative filtering is unable to provide recommendation since it does not have sufficient information.

Scalability. Model based collaborative filtering is computationally expensive, and scalability becomes an issue with the huge number of users and items.

2.1.3 Hybrid Recommendations

Given that both content-based and collaborative filtering techniques have their strengths and weaknesses, several works have proposed various ways to combine these techniques to improve the recommendation accuracy [3, 5, 6, 7, 12, 22, 24, 31, 59, 67, 68, 80, 83, 87, 88, 78]. One main issue that hybrid recommender systems aim to address is the cold-start problem [80, 1].

The most straightforward strategy is to implement the collaborative and content-based methods separately and combine their predictions by using a weighted average [22] or a voting scheme [67]. The work in [12] use the confidence criterion to switch between different recommendation techniques, while [87] propose to use the agreement between a user's past ratings and the recommendations of each technique to make the final recommendation.

Another strategy is to add content-based characteristics to collaborative models. The hybrid recommenders in [6, 67, 31, 59] are based on traditional collaborative filtering techniques, but maintain content-based profiles for users and items to calculate the similarity. [67] proposes a framework to add user profile into a collaborative filtering recommender, while [59] utilizes the user's ratings as a vector to calculate the content-based score.

Alternatively, one can also add collaborative characteristics to content-based models by using some dimension reduction technique on content-based profiles. For example, the work in [83] proposes to use latent semantic indexing to build a collaborative model on users' profile contents.

Other works have proposed unified models to incorporate both content-based and collaborative characteristics [7, 68, 80, 3, 24]. The work in [7] utilizes both content-based and collaborative characteristics in a rule-based classifier. [68, 80] propose probabilistic methods based on latent semantic analysis to integrate collaborative and content-based recommendations, while [3, 24] use Bayesian mixed-effects regression models with Markov chain Monte Carlo method for parameter estimation.

For hybrid recommendation systems, they can balance the advantages of both content-based and collaborative filtering techniques, however it is difficult to build a unified framework for different applications as different hybrid methods require different components. Moreover, the performance hybrid recommendation systems is bottle-necked by the basic techniques(i.e.,content-based or collaborative filtering techniques). In this thesis, we will only focus on research of improving performance for collaborative filtering recommender systems.

2.1.4 Cluster-based Collaborative Filtering

Clustering is often used as an intermediate process in collaborative filtering to obtain sub-groups for further analysis. There are three types of cluster-based CF models.

The first type is one-sided clustering which partitions either the users or the items into distinct groups. For example, Sarwar et al. [77] split the users into groups based on user similarity before applying memory based CF algorithm to make recommendation. O'Connor et al. [63] discover item clusters from user rating data. [88] cluster users and items separately using variants of k-means and Gibbs sampling. The drawback of the one-sided clustering method is that the inter-relationships that exist between users and items is ignored.

The second type is two-sided clustering CF models [29] which simultaneously obtain user and item neighbourhoods via co-clustering, and generate predictions based on the average ratings of the co-clusters. The drawback of both the one-sided and two-sided clustering approaches is that each user or item can belong to only one cluster.

The third type, bi-cluster CF models, allows overlaps among clusters. Bi-clustering was first proposed in biological data analysis [20, 54]. Symeonidis et al. [85] apply bi-clustering model on collaborative filtering to discover the inter-relationship between users and items.

The most recent work in Xu et al. [95] propose a multi-class co-clustering (MCoC) algorithm to find user-item subgroups (i.e., bi-cluster). The authors design a unified framework to incorporate the subgroups into collaborative filtering methods. MCoC works in two phases. The first phase maps all the users and items into a shared low-dimensional space. The second phase use existing clustering algorithms such as fuzzy c-means and k-means to find the user-item clusters from the low-dimensional space. However, this method is computationally expensive as the mapping to the low-dimensional shared space involves matrix manipulations with at least $(m + n)^2$ parameters, m is the number of users and n is the number of items.

2.2 User Recommender Systems

User recommendation has become an important recommendation task in social networks such as Facebook and Twitter. There has been much research on using recommender systems to help users connect with people online [40, 32, 26, 18]. These works are focused on more structured data and restricted domains such as co-authorship links [32], community membership in enterprise social network [18].

The work in [32] profile users by aggregating information from multiple sources in an enterprise and highlighting users who have contributed in similar ways, e.g., patent authorship, co-author papers or wikis. [18] propose algorithms that utilize content similarity and social network structure in user recommendation. The former is based on the intuition that ”if two users both post content on similar topics, then they might be interested in getting to know each other”, while the latter is based on the Friend-of-Friend hypothesis that ”if many of my friends consider someone a friend, then I might be interest to know that person too".

Recent work has examined methods for recommending users to follow in noisy unstructured micro-blogging data such as Twitter [33, 4]. The authors in [33] investigate both content-based approach (users' own tweets, their followers' tweets and followees' tweets) and collaborative filtering approach (users' ID, followers' ID and followees' ID) to profile users. User profiles are indexed and the information retrieval TF-IDF approach is used to rank and recommend users based on a target user profile. They find that the collaborative filtering approach are better at finding relevant followees for a user.

Collaborative filtering methods such as matrix factorization and its variants [47, 42, 69] have been applied to user recommendation with implicit feedback. [42] design a a matrix factorization method called $IF\text{-}MF$ for implicit feedback data sets. Each user-item (or user-user) pair is associated a confidence variable in the cost function, and each decision is assigned a weight in the learning process. [69] propose a probabilistic matrix factorization method ($BPR\text{-}MF$) for implicit feedback data sets. Unlike other matrix factorization approaches that take the unseen items as missing samples, $BPR\text{-}MF$ divides the unseen items into negative samples and missing samples. The training process takes the rank pair as input such as (u, i, j) which means user u prefer item i to j. This work has also been applied in KDD Cup 2012 [41] to predict which users a target user might follow in Tencent Weibo.

A topology-based algorithm is designed in [4] to search the follower/followee network for candidate users to recommend. This algorithm is based on the hypothesis that, for a target user u, the users followed by the followers of u's followees are candidates to recommend to u. This approach is a variant of the neighborhood item recommendation method [76] where a followee is equivalent to an item. However, the work in [47] shows that neighborhood approaches perform worse than matrix factorization approach.

2.3 Product Recommender Systems

Recommender systems are used in e-commerce websites to help customers find products to purchase. Most of the collaborative filtering methods reviewed in Section 2.1.2 can be employed in these product recommender systems. These include memory-based ones [30, 76, 57] and model-based methods [14, 60, 84, 39, 17]. However, these systems can be further enhanced by taking into account the temporal factor and temporal diversity which are important in e-commerce as users' preference tend to change over time.

The work in [70] combines the latent factor model and Markov chain model to predict the next basket of products that may be purchased by users. This method utilizes the order of products purchased as the temporal information. They construct a tensor which captures the probabilities of all product pairs $p(i|j)$ that indicates the probability of purchasing product i after purchasing product j and further use the factorizing method to estimate the unknown value in the tensor. The estimated values are used as evidence for the next basket recommendation.

The authors in [46] design a method to learn the temporal changes of users and items. They propose a time-aware factor model which distinguish the transient effects and long term patterns. They first bin the time dimension into small time slots, and then incorporate temporal factors such as user bias, item bias and user preference into the standard matrix factorization model. Similarly, recent works have developed systems to recommend the right products at the right time [91].

Unlike [46] that utilizes a factorization model, Liang et. al [94] propose a graph based method to capture the temporal factor. A Session-based Temporal Graph (STG) is employed to simultaneously learn the user's long and short preference on items. The score of each product is determined by both the long-term preference and short-term bias due to external events such as seasonal festivities. The long-term preference is defined by user similarity when they purchase the same products, while the short-term bias considers the product similarity over a short period of time. Both these preferences can be determined from their proposed STG.

The work in [51] highlights that product recommender systems in e-commerce differ from music or movie recommender systems as the former should take into account the utility of products in their ranking. The authors employ the utility and utility surplus theories from economics and marketing to improve the list of recommended product. The work in [90] propose to recommend product which maximize users' marginal utility [23], e.g, the marginal utility of a mobile phone drops after a user purchase and subsequent recommendations should not include similar phones but phone accessories instead and [89] models the joint probability of a user making a follow-up purchase of a particular product at a particular time to improve recommendation accuracy. [90] use the matrix factorization based model to learn the features of products and enhance the model with the Law of Diminishing Marginal Utility. In our work, we combine the notions of purchase intervals and utility surplus to obtain a model to increase the temporal diversity of products recommended.

2.4 Summary

In this chapter, we have reviewed existing works on recommender techniques which form the background of this thesis. We have also discussed related works on user recommender systems and product recommender systems, and have identified the following limitations in these works:

- Existing user recommendation approaches assume that user preference information are available to depict their interests. However, this is a challenge in microblogs due to limited user information and noisy tweets. The accuracy of user recommendation can be improved if we can form user communities to reduce data sparsity and discover the latent characteristics of communities instead of individual user.

- Existing product recommenders in e-commerce consider the order of items purchased by users to obtain a list of recommended items. The models aim to capture the preferences of users for items over time, including their long term interest and short term bias. However, they do not consider the time intervals between the items purchased, and assume that the value of an item for a user does not change over time. These factors can be utilized to improve the performance of e-commerce recommender as well as increase the temporal diversity of the recommended items.

- The state-of-the-art collaborative filtering method, matrix factorization, typically used in product recommenders, constrains each user to one preference vector, and sparsity remains a challenge for matrix factorization given the huge number of users and products. Since users may have different preferences when purchasing different subsets of items, and the periods between purchases also vary from one user to another, we could reduce sparsity and improve product recommendations with a bi-cluster based collaborating filtering method.

The next three chapters of this thesis will describe our proposed approaches to address the above limitations.

CHAPTER 3

USING LATENT COMMUNITIES FOR USER RECOMMENDATION IN MICROBLOGS

Advances in Web 2.0 technology has led to the rising popularity of many social network services. Microblogs such as Twitter allow users to post short text messages (tweets), and have become real time information sources as users follow one another. It is reported that there are over 500 million active users in Twitter, and user recommendation has become a key service to help users find people they might be interested in to follow. Unlike traditional user recommendation systems, user interest is not expressed explicitly in the form of ratings on items s/he likes. Instead, the profile of a Twitter user is given by the tweets s/he publishes and the structure of the follower-followee network.

In this chapter, we describe a community-based approach to user recommendation in Twitter-style social networks. We utilize the follower-followee relationships and employ an LDA-based method to discover hidden communities before applying matrix factorization on each of the communities. This work has been published in [100].

3.1 Motivation

Existing user recommendation approaches assume that user preference information such as ratings and purchase histories are available to depict their interests. However, this is a challenge in Twitter because of its limited user information. Inferring user preferences from their tweets is difficult as tweets are inherently noisy (short and peppered with acronyms and abbreviations). The work in [33] examines using combinations of tweet content and follower-followee relationships to recommend users to follow in Twitter. They found that follower-followee relationships are dominant features that capture the interest of users since users actively choose people they are interested in to follow.

Figure 3-1 shows a sample Twitter-style social network where the relationships are directional and not necessarily reciprocal. The directed edge $e(u, v)$ indicates that user u is following user v. Each user u has a set of followers F_u and a set of followees G_u. For example, we have $F_{u_1} = \{u_2, u_4, u_5\}$ and $G_{u_1} = \{u_2, u_3, u_4, u_6\}$. Note that we do not have the edge $e(u, v)$ where $u = v$ since a user does not follow him/herself.

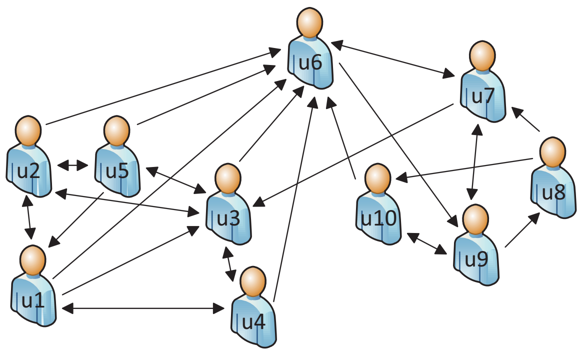
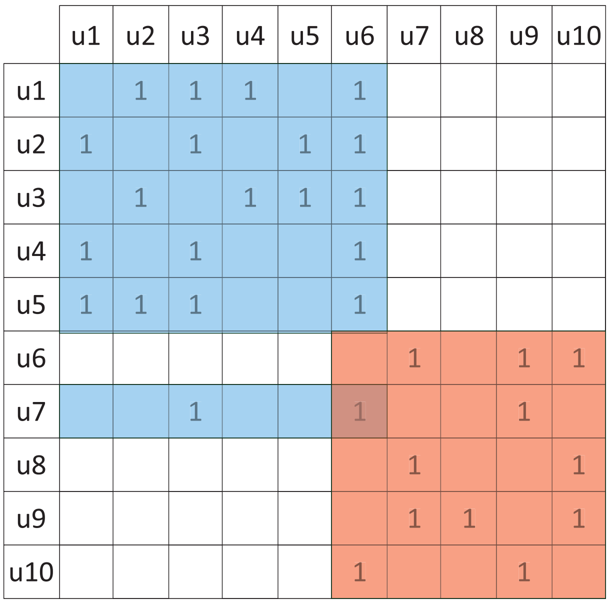


Figure 3-1: Example of a Uni-directional Social Network

Although the follow relationship among users seems disorganized and chaotic, communities exist in these social networks as a user follows another user based on his/her interests. Figure 3-2 gives the matrix representation of the follow relationships in Figure 3-1. The rows and columns denote user ids. An element at row i and column j with a value of 1 indicates that user u_i is a follower of user u_j. In other words, row i is the followee list G_{u_i} for user u_i and column j is the follower list F_{u_j} for user u_j.



	u1	u2	u3	u4	u5	u6	u7	u8	u9	u10
u1		1	1	1		1				
u2	1		1		1	1				
u3		1		1	1	1				
u4	1		1			1				
u5	1	1	1			1				
u6							1		1	1
u7			1			1			1	
u8							1			1
u9							1	1		1
u10						1			1	

Figure 3-2: Matrix Representation of the Network in Figure 3-1

By clustering or re-arranging the rows and columns in the matrix, we obtain two communities as indicated by the red and blue submatrice. We observe that:

1. A user may be a follower in more than one communities, indicating his/her multiple interests, e.g., pop music and sports. For example, user 7 is a follower in both the red and blue communities.

2. A user may be a followee in multiple communities, demonstrating his/her influence in these communities. For example, user 6 is a followee in both the red and blue communities.

3. A user may play different roles in different communities. For example, user 6 is both a followee and a follower in the red community. However, s/he is only a followee in the blue community.

The above observations motivate us to utilize a probabilistic approach that leverages both the follower and followee information of users to discover communities. The goal is to form communities of users with similar influence as well as interests. Then, applying state-of-the-art matrix factorization methods and its variants *IF-MF* [42] and *BPR-MF* [73] to each community will lead to better personalized recommendations.

Suppose we want to recommend users to u_{10} to follow. We observe that u_{10} is in the red community. If we apply matrix factorization on the red sub-matrix, we will recommend u_7 to u_{10}. However, if we apply matrix factorization on the entire matrix in Figure 3-2, we will recommend u_3 because u_{10} follows u_6 and u_9, and the majority of the u_6's followers also follow u_3. Our experiments demonstrate that by discovering communities in Twitter-style social network and recommending users to follow within these communities leads to significant improvement in conversion rate, precision and recall over performing matrix factorization on the original dataset (see Section 3.3.4).

Further, forming communities for user recommendation in a uni-directional social network reduces the sparsity in the matrix which is one of the most serious limitations of contemporay matrix factorization approaches. For example, the densities of the 2 sub-matrix in Figure 3-2 which correspond to the red and blue communities are increased to 48%, 58% respectively compared to the original density of 32%. The proposed approach is also scalable as the matrix factorization of each community (a subset of the original data set) can be performed in parallel (see Section 3.3.6).

In this work, we utilize the follower-followee relationships in Twitter-style social network and propose a two-step approach to recommend users to follow. We first employ an LDA-based method to discover the communities. Then we apply matrix factorization on each of the discovered communities, and provide two ways to combine the results obtained to recommend the top-k followees for a target user. Extensive experiments on two real world data sets, Twitter and Weibo, demonstrate that the proposed approach is scalable and improves the conversion rate by 20% compared to the state-of-the-art matrix factorization based recommendation algorithms [42, 69].

3.2 Proposed Framework

Our proposed framework comprises two main phases. The first phase utilizes an LDA-based method to determine the topic distribution of the users. Communities are formed by grouping users whose probability of a given topic is above some threshold. The second phase applies matrix factorization on each community to generate a list of candidate followees. We then combine these candidate lists to obtain the top-k users for a target user to follow. Before we describe the details of each phase, we summarize the symbols used in Table 3.1.

Table 3.1: Meanings of symbols used

Symbol	Meaning
u	A Twitter user
U	The set of all Twitter users
f	A follower
F	The set of all followers
g	A followee
G	The set of all followees
$e(f, g)$	A follow edge from f to g
E	The set of all edges $e(f, g)$ $f \in F, g \in G$
z	A topic
Z	The set of all topics
c	A community
C	The set of all community
$c.F$	The set of followers in community c
$c.G$	The set of followees in community c
$c.E$	The set of edges $e(f, g)$ in a community c, $f \in c.F, g \in c.G$

3.2.1 Discover Communities

LDA has been shown to be effective in document classification and recently, it has been applied to uni-directional social network such as Twitter to group users based on their follower relationship [16]. In this work, we propose to incorporate both the follower and followee relationships into the LDA model to discover communities. We map both followees and followers into the same space so that the communities obtained will link users based on their interests (followees) and influence (followers).

Let U be the set of users and E be the set of directed edges connecting the users in a social network. An edge $e(f, g) \in E$ implies that user f follows user g. Let $F \subset U$ be the set of followers and $G \subset U$ be the set of followees defined as:

$$\begin{aligned} F &= \{u \mid u \in U \wedge \exists g \in U \wedge \exists\, e(u,g) \in E\} \\ G &= \{u \mid u \in U \wedge \exists f \in U \wedge \exists\, e(f,u) \in E\} \end{aligned} \tag{3.1}$$

Just as one has a topic in mind when choosing a word for a document, likewise a user has an interest in mind when following another user in Twitter. Hence, each follower f can be regarded as a document consisting of a list of followees g. We denote $Pr(z|f)$ as the multinomial probability of topic z given a follower f, and $Pr(g|z)$ as the multinomial probability of a followee g given z.

Since a user u can be both a follower f and a followee g, s/he is associated with two documents d_f and d_g. The content of d_f is the list of followees of u, while the content of d_g is the list of followers of u, denoted as follows:

$$\begin{aligned} d_f &: \{u \mid u \in U \wedge \exists\, e(f,u) \in E\} \\ d_g &: \{u \mid u \in U \wedge \exists\, e(u,g) \in E\} \end{aligned} \tag{3.2}$$

Therefore our document corpus D is given by

$$D = \bigcup_{f \in F} d_f \;\cup\; \bigcup_{g \in G} d_g \tag{3.3}$$

We apply LDA on D to generate a pre-defined number of topics Z. Figure 3-3 depicts the graph model for this representation.

For each topic $z \in Z$, we form a community c such that the followers and followees in c, denoted as $c.F$ and $c.G$ respectively, are given by

$$\begin{aligned} c.F &= \{f \mid f \in F \wedge Pr(z|d_f) > \gamma\} \\ c.G &= \{g \mid g \in G \wedge Pr(z|d_g) > \gamma\} \end{aligned} \tag{3.4}$$

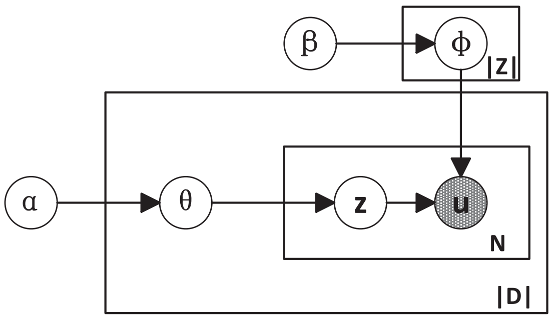


Figure 3-3: Graphical Model Representation

where γ is some threshold. The edges in c, denoted as $c.E$, represent the follower-followee relationships in c and is given by

$$c.E = \{e(f, g) \mid e(f, g) \in E \land f \in c.F \land g \in c.G\} \tag{3.5}$$

The output for this phase is a set of communities C where $|C| = |Z|$.

3.2.2 Recommend Followees

After discovering the communities, the next phase is to generate candidate followees from these communities for recommendation. The work in [42] adapted the state-of-the-art matrix factorization approach [47] to handle binarized user preference for items in implicit feedback data sets ($IF\text{-}MF$).

Here, we utilize the $IF\text{-}MF$ method by considering $f \in F$ as users and $g \in G$ as items and construct the matrix M in the model as follows. For each community $c \in C$, the matrix M has dimensions $|c.F| \times |c.G|$. Each entry $M[f, g]$ has a value of 1 if there is an edge $e(f, g) \in c.E$, otherwise $M[f, g] = 0$.

After matrix factorization, we obtain two matrices, namely $P^{|c.F|\times L}$ and $Q^{RD\times|c.G|}$, where $P^{|c.F|\times RD}$ denotes the mappings of followers in the reduced latent space of RD dimensions and $Q^{RD\times|c.G|}$ denotes the mappings of followees to the same reduced latent space. In other words, each follower f is associated with a vector $p_f \in P^{|c.F|\times RD}$, while each followee g is associated with a vector $q_g \in Q^{RD\times|c.G|}$.

Then for a follower f, we obtain the score that s/he will follow g in community c. This is given by the inner product of p_f and q_g as follows:

$$score(f, g, c) = \left\langle p_f, q_g \right\rangle \tag{3.6}$$

Since a target user f may belong to more than one community, s/he will have a different candidate followee recommendation list from each community. Here, we propose two ways to compute the final score that a target user $f \in F$ will follow $g \in G$ from these lists.

We can take the maximum score among the scores in the communities that both f and g belong to.

$$maxScore(f, g) = \underset{c \in C}{Max}(score(f, g, c)) \tag{3.7}$$

Alternatively, we can sum up the scores in all the communities that f and g appear in as follows:

$$sumScore(f, g) = \sum_{c \in C}(score(f, g, c) \times Pr(c|f)) \tag{3.8}$$

where $Pr(c|f)$ is the probability that f belongs to the community c.

Note that $Pr(c|f)$ is $Pr(z|d_f)$ in the LDA model where z is the latent topic corresponding to community c. Finally, we sort these scores for each follower f and output the top-K followees to recommend to f.

Algorithm 1 summarizes our proposed approach. We call our method Community-Based Matrix Factorization (*CB-MF*). The algorithm first obtain the set of followers and followees from the follower-followee relationships (lines 1-3). Then we obtain the document corpus and apply LDA to generate a pre-determined number of topics (lines 4-11). Lines 12 to 18 shows how to construct each community with its followers, followees and associated edges. Then we perform matrix factorization on each community (lines 19 to 24). Lines 25-28 aggregates the scores from each community and we obtain a ranked list of recommended followees for each follower.

Algorithm 1: *CB-MF* Algorithm

```
input : 1. Set of follower-followee relationships $E = \{e(f,g)\}$,
        2. Number of communities $N$,
        3. Number of latent factors $L$,
        4. Threshold $\gamma$
output: Ranked recommendation list
$F \leftarrow \{f \mid \exists e(f,g) \in E\}$;
$G \leftarrow \{g \mid \exists e(f,g) \in E\}$;
$U \leftarrow F \cup G$;
$D = \emptyset$;
foreach $f \in F$ do
    $d_f = \{u \mid u \in U \wedge \exists\, e(f,u) \in E\}$
    $D = D \cup \{d_f\}$;
end
foreach $g \in G$ do
    $d_g = \{u \mid u \in U \wedge \exists\, e(u,g) \in E\}$
    $D = D \cup \{d_g\}$;
end
$Z \leftarrow LDA(D, N)$;
$C = \emptyset$;
foreach $z \in Z$ do
    $c \leftarrow \emptyset$
    $c.F = \{f \mid f \in F \wedge Pr(z|d_f) > \gamma\}$;
    $c.G = \{g \mid g \in G \wedge Pr(z|d_g) > \gamma\}$;
    $c.E = \{e(f,g) \mid e(f,g) \in E \wedge f \in c.F \wedge g \in c.G\}$;
    $C = C \cup \{c\}$;
end
$R = \emptyset$;
foreach $c \in C$ do
    construct matrix $M_c$;
    $IF\text{-}MF(M_c, L)$;
    $R_c = \{score(f,g,c) \mid f \in c.F \wedge g \in c.G\}$
    $R = R \cup \{R_c\}$;
end
$Result = \emptyset$;
foreach $pair\ (f,g)$ do
    compute $sumScore(f,g)$ (or $maxScore$) according to Equation 7 (or 8);
end
Return the ranked lists of followees for each follower;
```

3.3 Experimental Study

In this section, we report the results of the extensive experiments we have carried out to evaluate both of the effectiveness and efficiency of our proposed *CB-MF* method. We compare the performance of our method with the following methods:

1. *TopPop*. This is a baseline algorithm which ranks users according to their number of followers and recommends the top-K most popular users to follow.

2. *FoF*. This is based on the Friend-of-Friend hypothesis, that is, if a particular person is followed by many followees of a target user, then s/he might be interested to follow this person too. In other words, we find the top-K most highly ranked followees of a target user's followees.

3. *NB-based* [4]. This is an implementation of the neighborhood based algorithm in [4]. Given a target user u and its set of followees G_u, we find the set of followers $F = \{u \mid \exists e(u, g) \in E \wedge \exists g \in G_u\}$. For each $f \in F$, we find the set of followees G_f and take the union. Then we find the top-K users with the most occurrences to recommend to u.

4. *LDA-based* [16]. This is an implementation of the LDA model described in [16] which map followers to documents and followees to words. Each followee g is scored using Equation 3.9 and we recommend the top-K followees with the highest score.

$$Pr(g|f) = \sum_{z \in Z} Pr\,(g|z)\,Pr\,(z|f) \tag{3.9}$$

5. *IF-MF* [42]. This is the state-of-the-art matrix factorization method for implicit feedback data sets.

6. *BPR-MF* [69]. This is a probabilistic matrix factorization method for implicit feedback data sets.

We implement the methods using Python. We code the LDA model according to [36], and use the C# implementation provided in [28] for the methods *BPR-MF* and *IF-MF*. All the experiments are carried out on an Intel(R) Core(TM) i7-2600 with 3.4 GHz, 8 GB RAM, 64 bit Microsoft Windows 7 operating system.

3.3.1 Experimental Data Sets

We use two real world Twitter-style data sets for our experiments. The first data set is the social network data used in [48] which is obtained from Twitter[1]. The second data set is the social network data which we crawled from Weibo[2], the biggest Chinese micro-blog system in China.

Table 3.2: Statistics of Twitter and Weibo data sets

Statistic	Twitter	Weibo
$\|F\|$	130,352	168,561
$\|G\|$	114,997	150,761
$\|U\|$	142,624	169,750
$\|E\|$	10,242,503	40,358,104
$\underset{g\in G}{Max}(\|E(*,g)\|)$	31,952	55,948
$\underset{f\in F}{Max}(\|E(f,*)\|)$	26,663	2,053
Sparsity	99.93%	99.84%

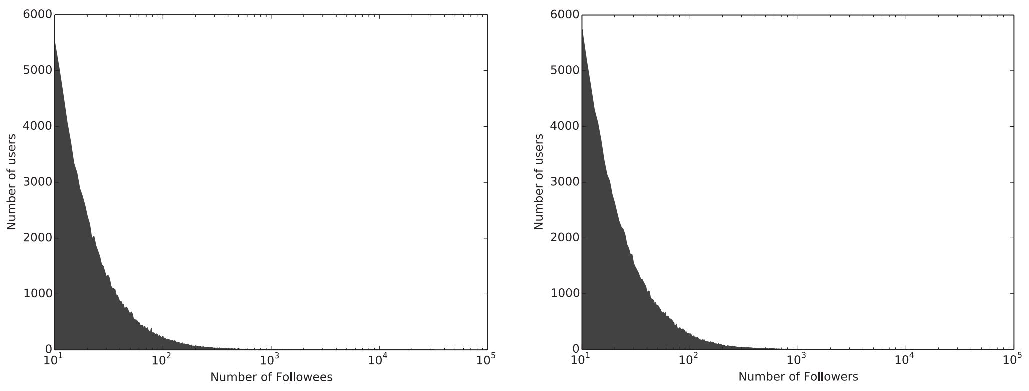


Figure 3-4: Characteristics of Twitter Dataset

[1]http://www.twitter.com
[2]http://www.weibo.com

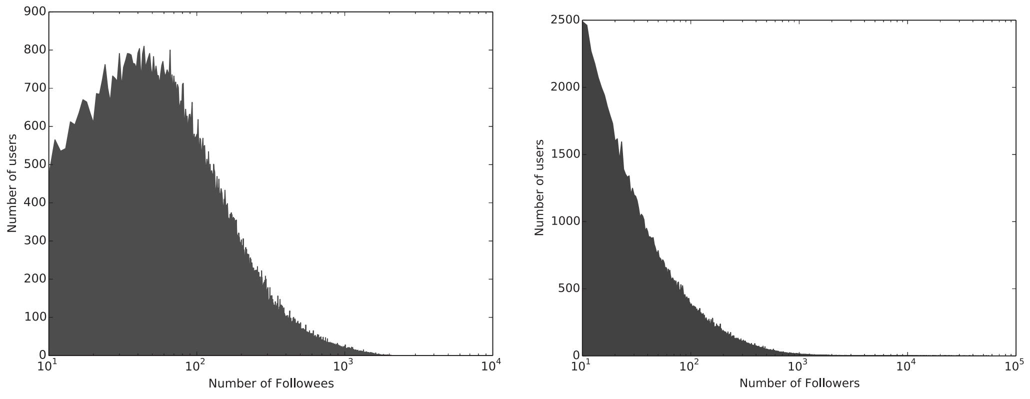


Figure 3-5: Characteristics of Weibo Dataset

We pre-process these data sets to anonymize the user ids and improve the data set density by removing users who have less 10 followers/followees. Table 3.2 gives the statistics of the two data sets after pre-processing.

Figures 3-4 and 3-5 show the characteristics of the Twitter and Weibo data sets respectively. The figures depict the number of users who have same number of followers or followees. As expected, both data sets have long tails, indicating that a small number of users have large number of followers or followees. For the Weibo data set, we see that more users have around 100 followees instead of 10 primarily because Weibo provide features such as batch following to encourage a user to have more followees. The difference in the number of followees in the two data sets is due to the different policies in Twitter and Weibo. Twitter allows users to have more followees as long as their number of followers increase. On the other hand, Weibo places a limit on the number of followees that a user can have (< 3000).

3.3.2 Evaluation Metrics

Our goal is to recommend top-k users for a target user to follow. For each follower, we randomly choose 10% followees s/he has followed as testing data, and keep the rest as training data. Our evaluation metrics include conversion rate, NDCG [43], precision, recall and F1 score.

Conversion rate is a commonly used metric in recommender systems to determine if a user has obtained at least one good recommendation. If L is the list of recommended k followees and L' is the list of k followees actually followed by the user, then the conversion rate is given by:

$$Conversion\ Rate = \begin{cases} 1 & \text{if } |L \cap L'| > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

We compare the conversion rates of the various algorithms by taking the average of values computed for each test user.

Normalized Discounted Cumulative Gain (NDCG) is a widely used metric for a ranked list. $NDCG_k$ is defined as:

$$NDCG_k = \frac{1}{IDCG_k} \times \sum_{i=1}^{k} \frac{2^{b_i - 1}}{log_2(i+1)} \tag{3.11}$$

where b_i is a binary value, 1 if the item at position i is hit item (i.e., item match) and 0 otherwise, $IDCG_k$ is the maximum $NDCG_k$ that corresponds to the optimal ranking list so that perfect NDCG can be 1.

The standard definitions for precision and recall are:

$$Recall = \frac{|L \cap L'|}{|L'|} \tag{3.12}$$

$$Precision = \frac{|L \cap L'|}{|L|} \tag{3.13}$$

We also report the *F1* score, which is the harmonic mean of precision and recall, defined as:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.14}$$

3.3.3 Sensitivity Experiments

We first examine how the various parameters affect the performance of our proposed *CB-MF* method. We fix the number of latent factors $LF = 16$, and vary the threshold γ

and number of communities N.

We measure the $F1$ score for k=3 using the two ways of combining the lists of candidate followees from each community (Equations 3.7 and 3.8). Tables 3.3 and 3.4 show the results for the Twitter and Weibo data sets respectively. We see that the $F1$ scores obtained by summing the weighted scores from the candidate lists ($F1_{sum}$) is higher compared to taking the maximum scores ($F1_{max}$). Further, a larger value for N improves the performance of $CB\text{-}MF$ on the larger Weibo data set.

Based on the results in Tables 3.3 and 3.4, we obtain the optimal parameter settings for the rest of the experiments. We use $\gamma = 0.02$, $N = 10$ for the Twitter data set, and $\gamma = 0.01$, $N = 15$ for the Weibo data set.

Table 3.3: Performance on Twitter for varying γ and N

	N=5		N=10		N=15		N=20	
γ	$F1_{sum}$	$F1_{max}$	$F1_{sum}$	$F1_{max}$	$F1_{sum}$	$F1_{max}$	$F1_{sum}$	$F1_{max}$
0.01	0.0695	0.0612	0.0725	0.0638	0.0735	0.0650	0.0637	0.0572
0.02	0.0722	0.0632	**0.0740**	0.0681	0.0708	0.0602	0.0649	0.0580
0.04	0.0682	0.0593	0.0692	0.0595	0.0690	0.0597	0.0650	0.0581
0.08	0.0657	0.0584	0.0690	0.0595	0.0652	0.0579	0.0593	0.0521

Table 3.4: Performance on Weibo for varying γ and N

	N=5		N=10		N=15		N=20	
γ	$F1_{sum}$	$F1_{max}$	$F1_{sum}$	$F1_{max}$	$F1_{sum}$	$F1_{max}$	$F1_{sum}$	$F1_{max}$
0.01	0.0385	0.0313	0.0436	0.0372	**0.0440**	0.0375	0.0410	0.0326
0.02	0.0377	0.0308	0.0428	0.0350	0.0423	0.0333	0.0418	0.0330
0.04	0.0359	0.0293	0.0348	0.0290	0.0402	0.0327	0.0401	0.0323
0.08	0.0298	0.0231	0.0351	0.0298	0.0343	0.0270	0.0360	0.0285

3.3.4 Comparative Experiments

Next, we compare the performance of the various user recommendation methods. We set the number of latent factors $LF = 16$ for the matrix factorization based methods ($BPR\text{-}MF$, $IF\text{-}MF$). Our $CB\text{-}MF$ calls $IF\text{-}MF$ for each community with the same LF setting.

Figures 3-6 and 3-7 show the Conversion Rate, Recall and Precision for the Twitter and Weibo data sets respectively. The NDCG for these two datasets are shown in Figure 3-8. The results indicate that the matrix factorization based methods (*BPR-MF*, *IF-MF* and *CB-MF*) outperform the methods that do not utilize matrix factorization (*TopPop*, *FoF*, *LDA-based* and *NB-based*).

Among the 3 matrix factorization based methods, the proposed *CB-MF* gives the best performance. All the methods perform better of Weibo compared to Twitter in terms of conversion rate. This is mainly because that the density of Weibo data set is higher then Twitter data set. For state-of-the-art matrix factorization approaches *IF-MF* and *BPR-MF*, *IF-MF* performs better than *BPR-MF* on both data sets. This is because *IF-MF* can better handle the data set sparsity.

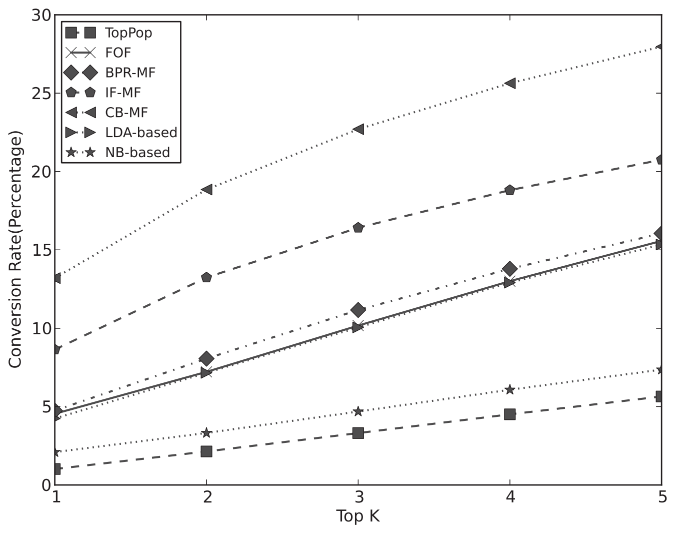
We also observe that *FoF* outperforms the *NB-based* algorithm. This is because the recommendations given by *NB-based* for a target user who follows popular users will be similar to the baseline *TopPop*. The *LDA-based* method is better than *TopPop*, *FoF* and *NB-based* mainly because it is able to discover and utilize the hidden characteristics of followees and followers for recommendation.

Overall, our proposed community-based approach improves the conversion rate in Weibo by about 15%, and leads to a significant 30% increase in the conversion rate for Twitter. This is because our approach applies matrix factorization on communities which have lower sparsity compared to the original data set. Figure 3-9 compares the sparsity of the original data sets and the communities obtained, clearly indicating that reducing data sparsity can help improve the effectiveness of user recommendation.

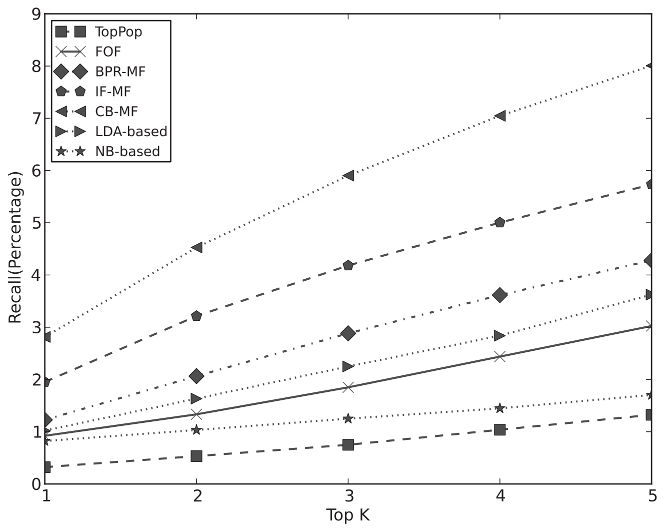
3.3.5 Comparison of Community Discovery Methods

We also examine the impact of using different community discovery methods on the conversion rate. We compare our approach to find communities with the following two methods:

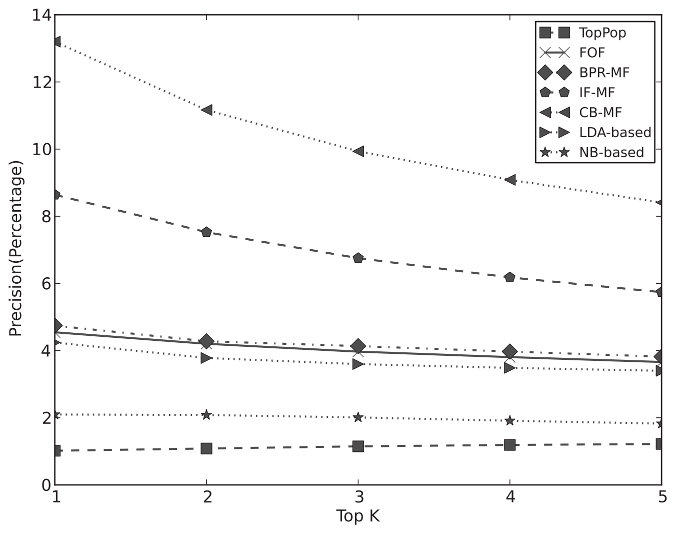
1. *LDA-Followee* [16]. This is an LDA-based model which utilizes only follower



(a) Conversion Rate

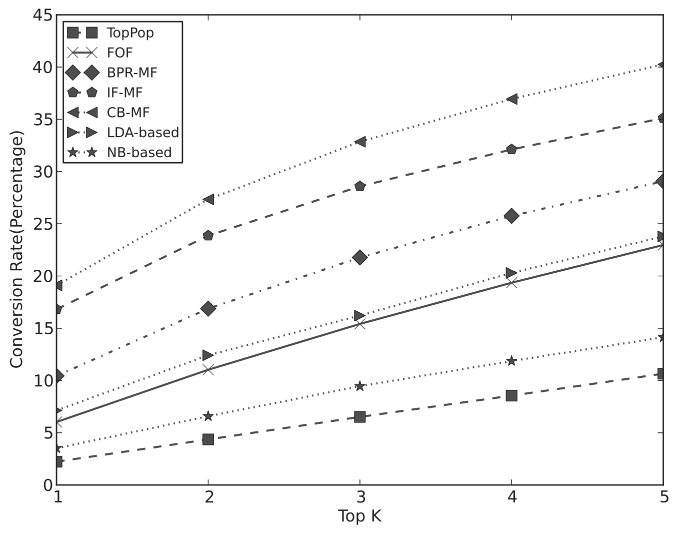


(b) Recall

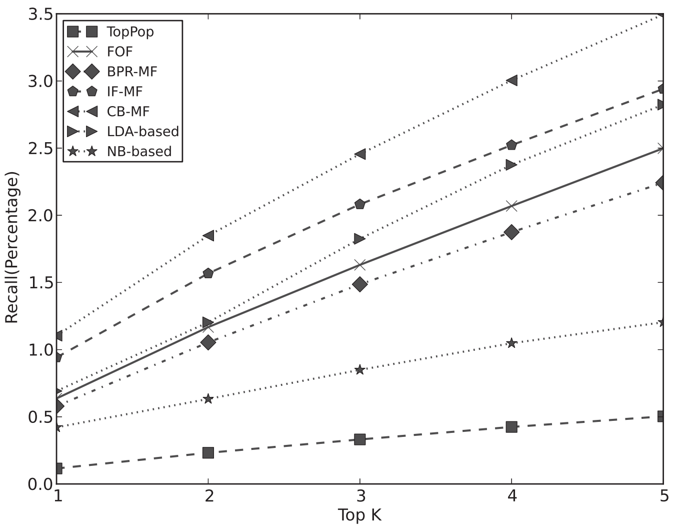


(c) Precision

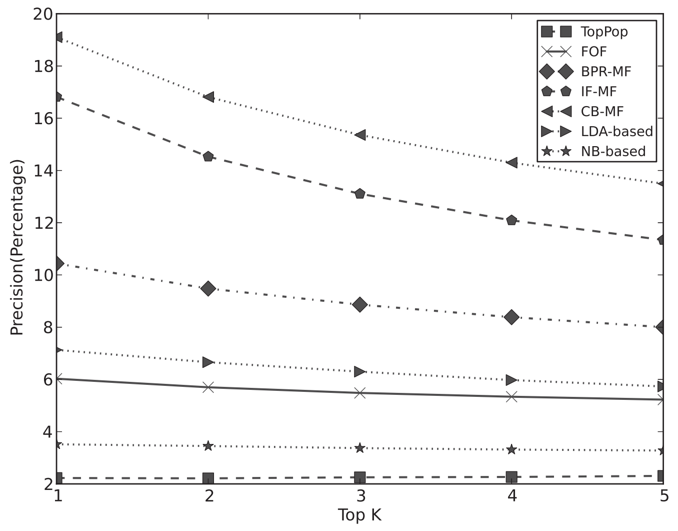
Figure 3-6: Comparative study on Twitter data set



(a) Conversion Rate

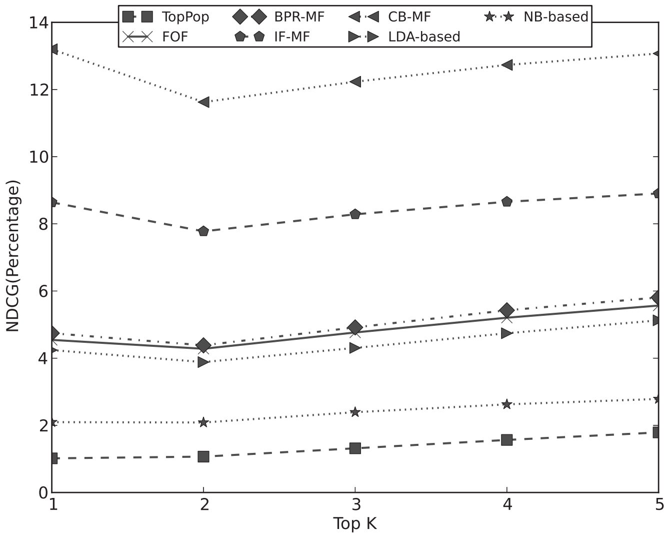


(b) Recall

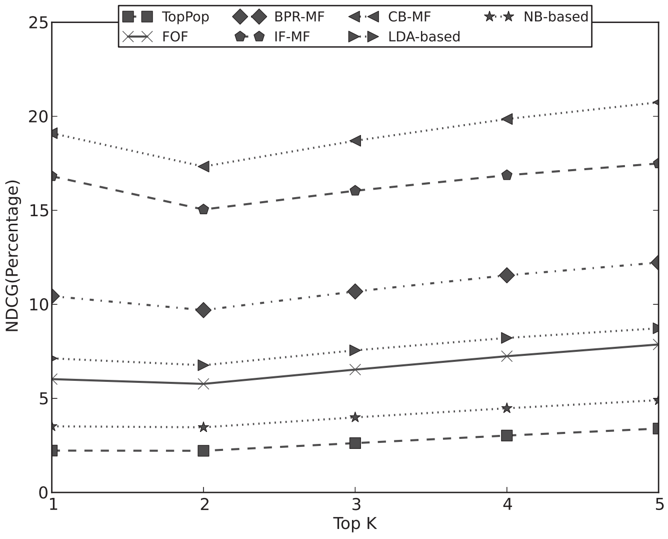


(c) Precision

Figure 3-7: Comparative study on Weibo data set

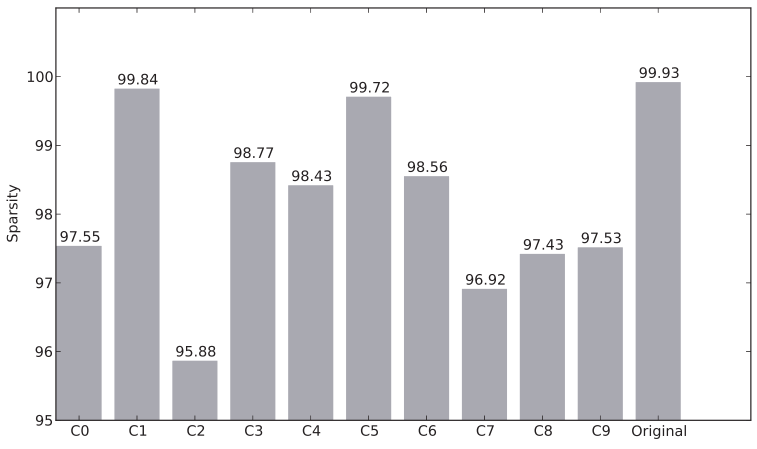


(a) Twitter

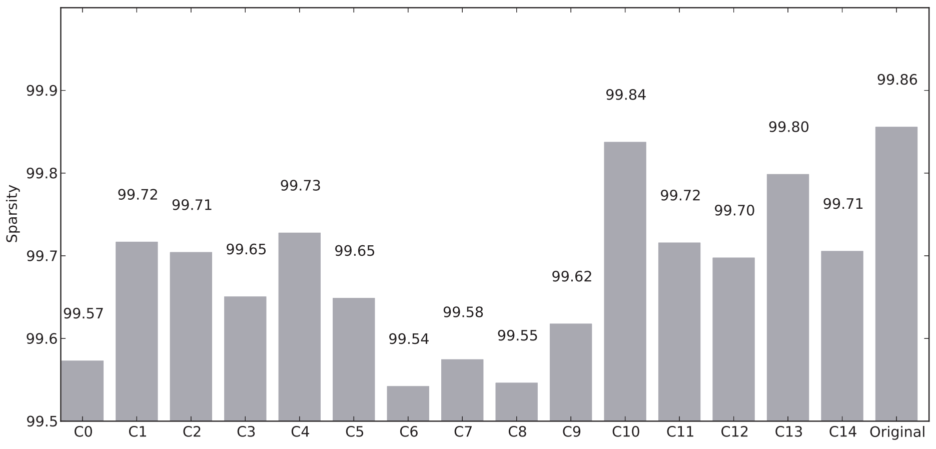


(b) Weibo

Figure 3-8: NDCG of the various methods



(a) Twitter Data Set



(b) Weibo Data Set

Figure 3-9: Sparsity of original dataset vs. discovered communities

relationships.

2. *MCoC* [95]. This is a multi-class co-clustering method to find user-item subgroups for item recommendation. We use this method to find follower-followee subgroups.

The *MCoC* code provided by the authors could not scale on the large Weibo data set. For the Twitter data set, we had to further improve the density by filtering out users who have less than 100 followers or followees. The resulting data set has 19305 followers and 16782 followees, and the data set sparsity is improved to 98.62%.

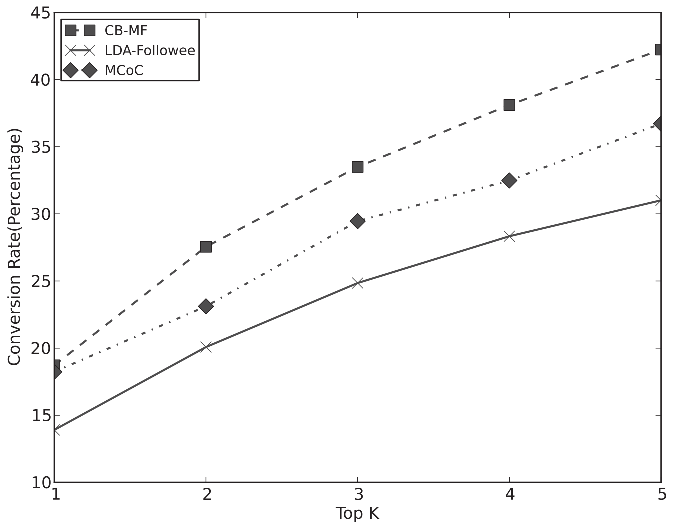
We apply the same matrix factorization approach *IF-MF* with $LF = 16$ on the communities obtained by the different methods. Figure 3-10 shows the results on both Twitter and Weibo data sets. We observe that our LDA-based model which utilizes both follower and followee relationship outperforms both *LDA-Followee* and *MCoC*, indicating that the communities obtained by our model are able to capture the user influence and interests.

3.3.6 Scalability Experiments

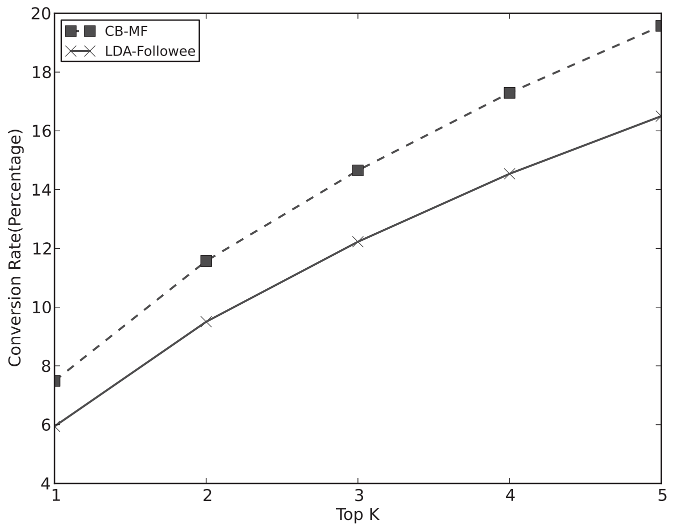
In this last set of experiments, we examine the scalability of the proposed approach. Matrix factorization is computationally expensive, especially when the number of latent factors increases. We propose that *CB-MF* can be an alternative form of parallelization for matrix factorization, and compare the performance of *CB-MF* and *IF-MF* on the larger Weibo data set.

Figure 3-11 shows the runtime and F1 scores as we vary the number of latent factors LF from 16 to 128. The run time of *CB-MF* is given by the time needed to discover communities and the maximum time obtained from running *IF-MF* on each of the community in parallel.

The results clearly demonstrate the effectiveness of the proposed community-based matrix factorization approach and its ability to scale. Although the F1 scores of both

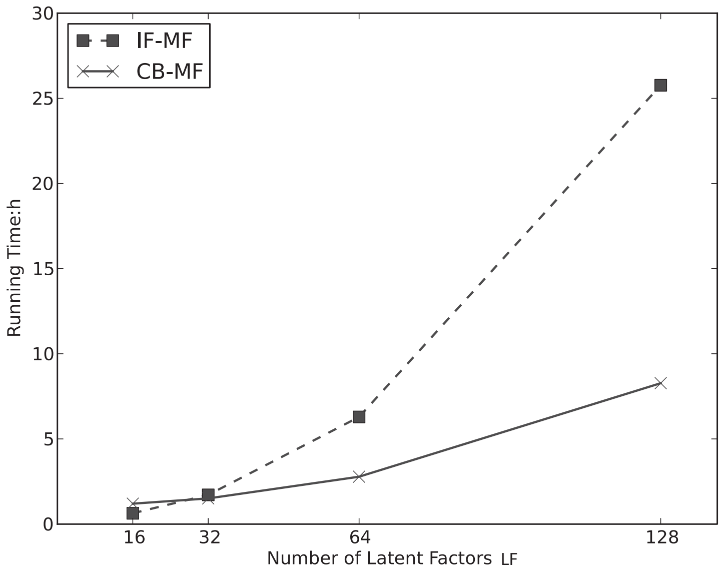


(a) Twitter Data Set

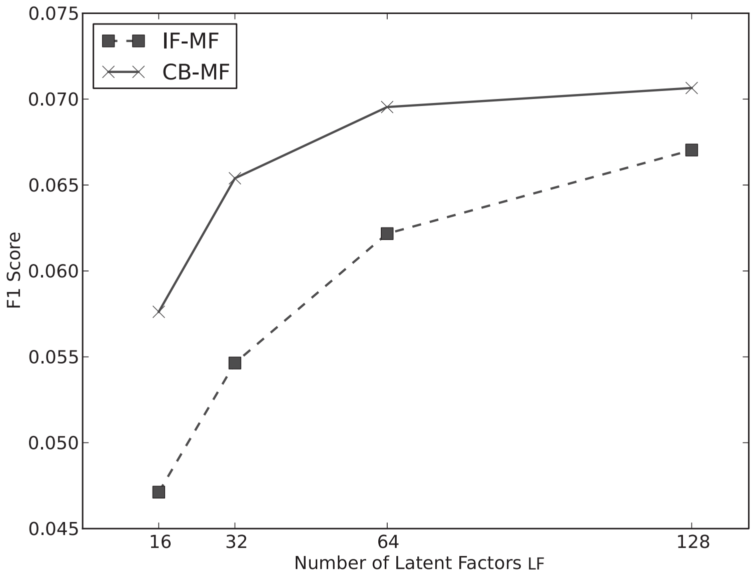


(b) Weibo Data Set

Figure 3-10: Effect of different community discovery methods on conversion rate



(a) Runtime



(b) F1 Score

Figure 3-11: Effect of *LF* on runtime and F1 (Weibo dataset)

methods increase with L, the running time for CB-MF remains reasonably stable while the run time for IF-MF grows significantly.

3.4 Summary

In this chapter, we have investigated how using both follower and followee relationships to discover communities can improve user recommendation in uni-directional social networks. We have introduced a two-phase approach where we first utilized the LDA model to discover communities, and then applied matrix factorization on each community found. We carried out extensive experiments to evaluate the performance of our approach on two real world microblog data sets, Twitter and Weibo. The results indicate that the proposed CB-MF method significantly outperforms state-of- the-art recommender algorithms. We have further shown that the community-based approach is a good alternative form of parallelization for matrix factorization.

CHAPTER 4

USING PURCHASE INTERVALS FOR PRODUCT RECOMMENDATION IN E-COMMERCE

The development of Web 2.0 technology has led to huge economic benefits and challenges for both e-commerce websites and online shoppers. One core technology to increase sales and consumers' satisfaction is the use of recommender systems to produce a list of recommendations that users may be interested to purchase. As reviewed in Chapter 2, research in recommender systems build models to characterize item (content-based filtering [82]) or users (collaborative filtering [76, 57]) or both (hybrid recommender systems [59, 65]) so as to predict the rating or preference that a user would give to an item. Items with the highest predicted ratings are then recommended to the user. However, these models assume that the value of an item for a user does not change over time.

In this chapter, we describe how we utilize purchase interval information to improve the performance of the recommender systems for e-commerce. This work has been published in [99].

4.1 Motivation

Existing product recommender systems typically consider the order of items purchased by users to obtain a list of recommended items. However, they do not consider the time interval between the products purchased. For example, there is often an interval of 2-3 months between the purchase of printer ink cartridges or refills. Thus, recommending appropriate ink cartridges one week before the user needs to replace the depleted ink cartridges would increase the likelihood of a purchase decision.

Figure 4-1 shows the purchase histories of 5 users for products i_1, i_2, i_3, i_4, i_5. We use H_i to denote the purchase history of user u_i. Suppose we want to recommend some products to user u_5 at time point 24. Since u_5 has previously bought products i_1 and i_3, the algorithms in [70, 94] would recommend product i_4 to u_5. This is because based on the purchase order and short-term bias, both users u_1 and u_3 buy i_4 after purchasing i_3. From the long term interest of users [94], we see that users who purchase product i_3 always purchase i_4. Further, suppose u_5 does not buy i_4 at time point 24. When s/he logs into the system at time points 19 and 21, the algorithms in [70, 94] will still recommend i_4 to u_5.

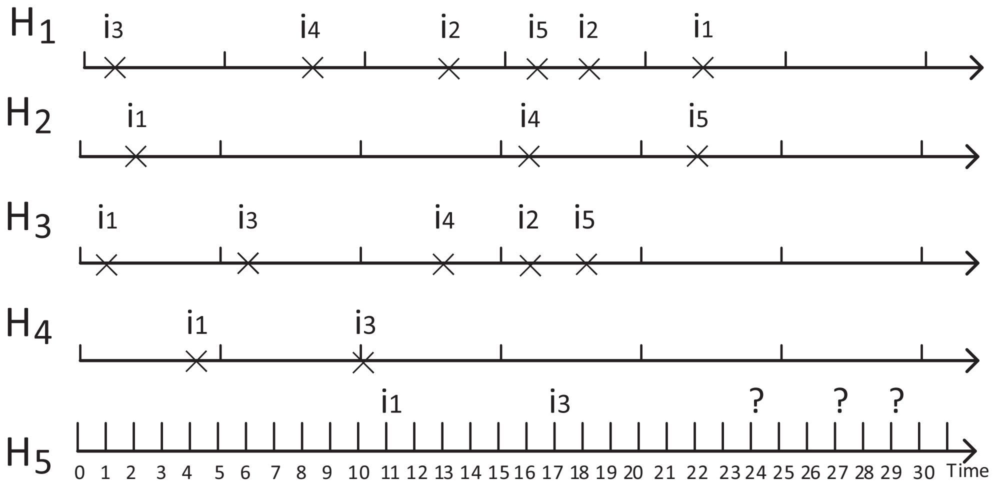


Figure 4-1: Example of Users' Purchase History

In this work, we advocate that the intervals between user purchases may influence a user's purchase decision. A quick survey reveals that users who purchase an iphone usually purchase apps one week later, after they have setup the new phone and become familiar with its use. Thus recommending apps to a new iphone user a day after is not likely to increase sales, whereas pushing apps 5-6 days later is probably more effective.

Consider again our example in Figure 4-1. For illustrative purposes, let us assume that the interval between each time point is a day. We observe that user u_2 buys product i_4 14 days after s/he has bought i_1, while user u_3 buys i_4 12 days after s/he has bought i_1. We could conclude that, on average, users are likely to purchase i_4 13 days after s/he has purchased i_1. Assuming that all the users u_1 to u_5 are similar, we can summarize the average purchase interval between product pairs from their purchase histories as shown in Table 4.1.

Table 4.1: Average Purchase Intervals

	i_1	i_2	i_3	i_4	i_5
i_1	-	15	5.7	13	18.5
i_2	4	-	-	-	2.5
i_3	-	11	-	7	13.5
i_4	14	3	-	-	6.3
i_5	6	2	-	-	-

Based on Table 1, if u_5 logs into the system at time point 24, then we will recommend product i_4 since s/he has bought i_1 13 days ago. However, if s/he logs into the system at time point 27, then we will recommend product i_2 to him/her because we see that users who bought i_3 tend to buy i_2 11 days later. Similarly, our approach will recommend i_5 to u_5 if s/he logs into the system at time point 29.

Theories in economics and consumer behavior postulate that the value of certain products may change over time, especially if the user has recently purchased them. This is known as the Law of Diminishing Marginal Utility [8]. For example, a user is less likely to buy a second computer or mobile phone if s/he has recently bought one. In contrast, products such as milk, bread and eggs are likely to be purchased over and over again. Thus the value or marginal utility of a product for a user depends on his/her purchase history. Recent works have applied these theories to recommender systems [51, 90]. The authors in [90] incorporate marginal utility into product recommender systems. They adapt the widely used Cobb-Douglas utility function [23] to model product-specific diminishing marginal return and user-specific basic utility to personalize recommendation.

Here, we propose a framework that incorporates purchase intervals for product recommendation. The model in our framework will combine purchase interval information in users' purchase histories with marginal utility. This enables us to increase the temporal diversity of the recommended items which is an important facet of product recommender systems [49]. Given the scale of users, products and purchase history in any e-commerce website, we design an algorithm to efficiently compute the purchase interval between pairs of product for all users. We evaluate our method on a real world data set obtained from an e-commerce B2C website Jingdong in China. The experimental results show that our approach improves the conversion rate, precision and recall of the state-of-the-art utility-based recommendation algorithm [90]. Further, we can achieve a significantly higher temporal diversity compared to traditional recommender systems.

The rest of the chapter is organized as follows. Section 2 gives the background of the concepts used in this work. Section 3 describes the proposed framework. Section 4 gives the results of the experimental study, and we conclude in Section 5.

4.2 Preliminaries

In this section, we provide the background for the economics concepts used in our proposed model. We review the notions of utility, utility surplus and the law of diminishing marginal utility and discuss how these concepts are adapted to model product-specific diminishing marginal return and user-specific utility for personalized recommendation.

4.2.1 Utility and Utility Surplus

Utility and Utility Surplus are two fundamental concepts from economics. *Utility* can be defined as a measure of the satisfaction obtained from the consumption or purchase of various goods and services. The utility of a product i, denoted as δ_i, can be modeled as the weighted sum of k characteristics or features of the product as follows:

$$\delta_i = \sum_k \beta^k * c_i^k \tag{4.1}$$

where c_i^k is the k^{th} feature of product i and β^k is its weight.

Utility surplus determines the excess utility one gets by purchasing a product. It is defined as the gain in the utility of a product minus the cost/price of the product as follows:

$$US(i) = \delta_i - \alpha \cdot price_i \tag{4.2}$$

where $price_i$ is the price of product i and α is the marginal net utility of money.

Equations 4.1 and 4.2 assume that users have the same preference for the features of a product. In reality, users may have different preferences for a product's features. In order to personalize product recommendations, we revise the above equations to allow user-specific preferences for product features as follows:

$$\delta_{u,i} = \sum_k \beta_u^k * c_i^k \tag{4.3}$$

$$US(u,i) = \delta_{u,i} - \alpha_u \cdot price_i \tag{4.4}$$

where β_u^k is the user u's preference for the k^{th} feature of product i and α_u is the sensitivity of user u to the product price.

Consumer behavior theory shows that a person usually makes a purchase decision depending on a product's utility surplus. Hence, our goal is to find the list of products that maximizes the utility surplus for users.

4.2.2 Law of Diminishing Returns

The Law Of Diminishing Marginal Utility states the marginal utility of product drops as the consumption of the product increases. Thus, each product will have a product-specific diminishing rate γ_i. [90] use the well-known Constant elasticity of substitution (CES) to capture this property. If a user has already purchased X_i quantity of product i, then then the marginal utility of buying an additional unit of i is $(X_i + 1)^{\gamma_i} - X_i^{\gamma_i}$.

By taking into consideration the Law Of Diminishing Marginal Utility, we can refine the utility surplus as follows:

$$US(X_i, i, u) = \delta_{u,i} \cdot ((X_i + 1)^{\gamma_i} - X_i^{\gamma_i}) - \alpha_u \cdot price_i \tag{4.5}$$

Note that in Equation 4.5, the utility surplus is decreased due to the diminishing effect.

4.3 Proposed Framework

The proposed framework incorporates the information of purchase intervals into the utility surplus model. It has three main phases:

- **Phase I: Generate Purchase Interval Cube.**

  This phase computes a purchase interval matrix M_u for each user from his/her purchase history. Each cell $M_u[i, j]$ stores the interval between the purchase of products i and j for user u. We obtain a purchase interval cube when we combine the purchase interval matrix of all the users. Section 4.3.1 gives the computational details of this phase.

- **Phase II: Build Model.**

  This phase uses the matrix factorization method [45] to learn the feature vectors of users and products from the training data. Then we can learn the optimal settings for the parameters in our purchase interval enhanced utility based model. We discuss how to incorporate the purchase interval information into the utility model in Equation 4.5 in Section 4.3.2 and estimate its parameters in Section 4.3.3.

- **Phase III: Recommendation.**

  When a user logs into the system at time t, we use our model to compute the utility surplus of each product at this time point t and rank them. A list of top K products that the user is most likely to purchase is recommended to the user.

4.3.1 Purchase Interval Cube

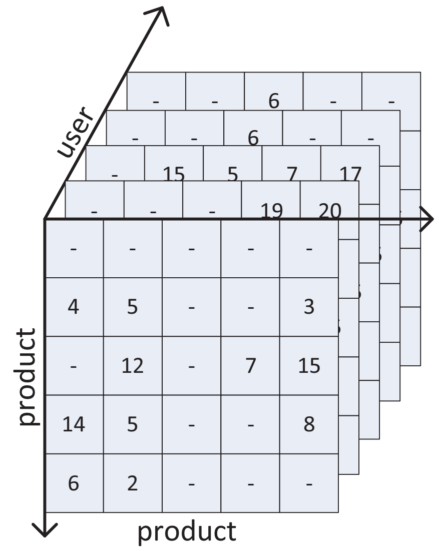


Figure 4-2: Purchase Interval Cube obtained from Figure 4-1. (Unit: day)

An e-commerce database D consists of a set of users U, product items I and purchase histories of users H. Each entry in the purchase history is a tuple $< u, i, t >$ which records the time t at which user u purchase item i.

If we want to capture the pattern of the purchase interval, we need first to know the purchase pattern for products pair $< i, j >$ for individual user u. For the computation, we not only consider the interval between each pair of products, but also the factors such as how many items purchased between the items and repurchased items. The computation

Algorithm 2: Generate Purchase Interval Cube

```
input  : Set of purchase histories $H = \{H_1, ..., H_n\}$,
         Window size $\omega$
output : Purchase interval cube $M = \{M_1, ..., M_n\}$
1  foreach $u \in U$ do
2      Initialize list $Q \leftarrow \emptyset$; /* to store entries $e$ */
3      /* e is a 3-tuple $< u, id, t >$ */
4      Initialize lists $R, R_w, R'_w \leftarrow \emptyset$; /* to store records $r$ */
5      /* $r$ is a 4-tuple $< i, j, x, c >$ where $i$ and $j$ are product ids, $x$ is the purchase interval
       between $i$ and $j$, and $c$ is the number of products purchased between $i$ and $j$.*/
6      foreach $e \in H_u$ do
7          /* remove records whose purchase interval exceeds $\omega$ */
8          while $(e.t$ - $e'.t) > \omega$ where $e'$ is the first entry in $Q$ do
9              remove $e'$ from $Q$;
10             remove all tuples $r$ from $R_w$ where $r.i = e'.id$;
11         end
12         /* update purchase interval and count of products */
13         Let $e'$ be the last entry in $Q$;
14         $x' \leftarrow e.t$ - $e'.t$;
15         if $x' > 0$ then
16             $c' \leftarrow 1$;
17         end
18         else
19             $c' \leftarrow 0$;
20         end
21         /* handle the case $e.id$ occur more than once in $\omega$ */
22         if $\exists\, e'$ in $Q$ s.t. $e'.id = e.id$ then
23             while $r.i <> e.id$ where $r$ is the first entry in $R_w$ do
24                 create $r' \leftarrow (r.i, e.id, r.x + x', r.c + c')$;
25                 add $r'$ to $R'_w$;
26                 remove $r$ from $R_w$;
27             end
28         end
29         /* create new interval records */
30         foreach $r \in R_w$ do
31             create $r' \leftarrow (r.i, e.id, r.x + x', r.c + c')$;
32             add $r'$ to $R'_w$;
33             add $r'$ to $R$;
34         end
35         $R_w \leftarrow R'_w$;
36         $R'_w \leftarrow \emptyset$;
37         if $\exists$ e' in $Q$ s.t. $e'.id = e.id$ then
38             remove e' from $Q$;
39             remove all tuples $r$ from $R_w$ where $r.i = e.id$ ;
40         end
41         create a tuple $r \leftarrow < e.id, e.id, 0, 0 >$;
42         add $r$ to list $R_w$;
43         push $e$ to queue $Q$;
44     end
45     Use Equation 4.6 to obtain $M_u$ from $R$;
46 end
```

and detail is discussed in this section. Let H_u be the purchase history of user u and I_u be the set of products purchased by a user u over a time window ω. For each pair of products $i,\ j \in I_u$, we want to find the average time interval when u would purchase j after purchasing i. Since u may purchase products i and j multiple times, we let T_i and T_j be the sets of time stamps at which u bought i and j respectively. For each $t_j \in T_j$, we find the $t_i \in T_i$ such that $t_j - t_i$ is the smallest and is less than ω. Let $\Phi = \{[t_{i_1}, t_{j_1}], [t_{i_2}, t_{j_2}], \cdots, [t_{i_n}, t_{j_n}]\}$ be the set containing such pairs of time stamps. Then the average purchase interval $d_{u,i,j}$ where a user u buys product j after buying product i can be determined as follows:

$$d_{u,i,j} = \frac{\sum_{[t_{i_r}, t_{j_r}] \in \Phi} t_{j_r} - t_{i_r} / (log_2(2 + count(t_{j_r}, t_{i_r}))}{\sum_{[t_{i_r}, t_{j_r}] \in \Phi} 1 / (log_2(2 + count(t_{j_r}, t_{i_r}))} \tag{4.6}$$

where the function $count(t_{i_r}, t_{j_r})$ returns the number of products purchased by the user between time stamps t_{i_r} and t_{j_r}.

Note that Equation 4.6 considers both the time interval between the purchase of the two products i and j, as well as the number of products bought in between i and j. We penalize the interval that has more purchases in between by taking the log.

Given the huge amount of user purchase histories and the large number of users and products, we need an efficient method to compute the purchase intervals between product pairs. Algorithm 2 gives the details of generating the purchase interval cube that captures the purchase interval between pairs of products for all users.

The algorithm takes as input the set of user purchase histories H and the window size ω to capture more accurate purchase intervals. Each entry e in the user purchase history is a 3-tuple consisting of the user id u, product id id and a time stamp t. The output is a purchase interval cube M that stores the purchase intervals of products for all users. We denote the purchase history and purchase interval matrix for each user u as H_u and M_u respectively.

For each user, we slide a window of size ω over his/her purchase history H_u and use a list Q to maintain the purchases which occur in the current window. The list R

stores tuples containing product pairs and their smallest purchase intervals. Each entry in R is a 4-tuple $< i, j, x, c >$ where i and j are product ids, x is the purchase interval between i and j, and c is the number of products purchased between i and j. In order to reduce computation, we use two lists R_w and R'_w to temporarily store the interval pairs corresponding to the purchase entries in Q. These lists are initialized in lines 1-4.

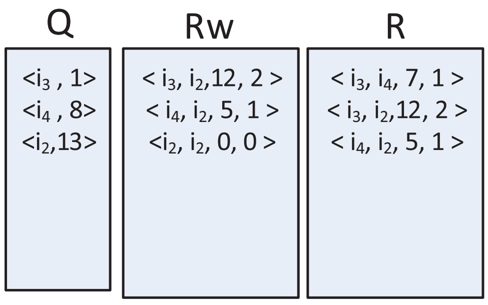
For each entry e in the user purchase history, we compare the timestamps between e and the oldest entry e' in Q. If the purchase interval exceeds the window size ω, we remove e' from Q and all the records that has the same product id as e' from the list R_w, that is, we do not need to consider this pair of products $e'.id$ and $e.id$ further (Lines 8-10).

Based on the latest entry e' in Q and the records in R_w, we update the interval and count of products purchased between $r.i$ and $e.id$ (Lines 12-33). If a user has purchased the product $e.id$ more than once in the window ω, then we remove records whose time intervals are not the smallest (Lines 19-23 and 31-33). Otherwise, we generate new interval records r' and store them in both R and R'_w (Lines 25-28). Note that we only need to update the interval records in R_w with the time difference between e and the latest entry e' in Q (line 12-17). Lines 34-36 creates an interval record r for each entry e and inserts e into Q.

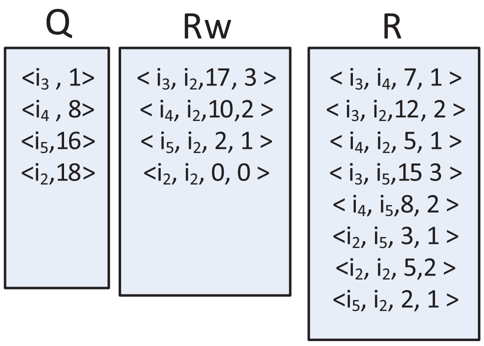
Finally, R stores the list of purchase intervals between product pairs for a user u. We apply Equation 4.6 to compute $d_{u,i,j}$. Each cell $M_u[i, j]$ stores the value of $d_{u,i,j}$ (Line 37).

The purchase interval matrix of all users $u \in U$ will form a purchase interval cube M. Figure 4-2 shows the purchase interval cube obtained for our example in Figure 4-1.

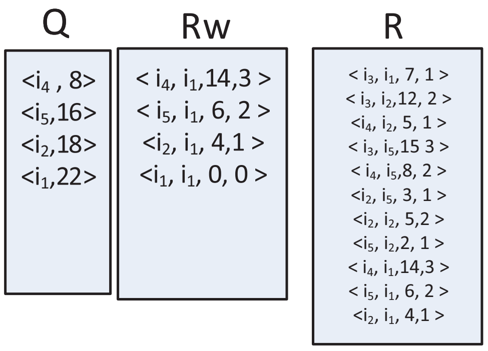
Let us illustrate Algorithm 2 with user u_1's purchase history in Figure 4-1. Suppose ω is set to 20. The first entry i_5 is $< u_1, i_3, 1 >$. Since all the lists are initially empty, we create a record $r < i_3, i_3, 0, 0 >$ and add it to R_w. We also insert i_3 into Q. When we process the second entry $< u_1, i_4, 8 >$, we compare its timestamp with that of the first entry $< u_1, i_3, 1 >$ in Q to check whether their purchase interval exceeds ω. Since it does not, we will compute the interval $x' = 8$ and $c' = 1$ and generate a new record



(a) Lists obtained after $< u_1, i_2, 13 >$



(b) Lists obtained after $< u_1, i_2, 18 >$



(c) Lists obtained after $< u_1, i_1, 22 >$

Figure 4-3: Example to illustrate Algorithm 1

$< i_3, i_4, 7, 1 >$. We insert this record into R and update R_w. R_w now has two records, $< i_3, i_4, 7, 1 >$ and $< i_4, i_4, 0, 0 >$. The third entry in H_{u_1} is similarly processed. Figure 4-3(a) shows the lists Q, R_w and R obtained.

When we process the entry $< u_1, i_2, 16 >$ in H_{u_1}, we find that there is another entry for product i_2 in Q. We create new records $< i_2, i_2, 5, 2 >$ and $< i_5, i_2, 2, 1 >$ for the product pairs (i_2, i_2) and (i_5, i_2). We also remove the older entry $< u_1, i_2, 13 >$ from Q and insert $< u_1, i_2, 18 >$ into Q. Figure 4-3(b) shows the updated lists.

When we process the last entry $< u_1, i_1, 22 >$, we find that its timestamp and that of the first entry $< u_1, i_3, 1 >$ in Q exceeds the window size ω. Hence, we remove $< u_1, i_3, 1 >$ from Q and the corresponding records $< i_3, i_2, 17, 3 >$ from R_w. We only need to create 3 new records for R (see Figure 4-3(c)).

Algorithm 1 only needs to scan the user purchase history once. It has a complexity of $O(|U| \times m \times n)$ where m is the maximum number of purchase entries in the window size ω and $n = max(|H_u|)$.

In order to ensure that we only consider frequently purchased product pairs, we will do a post-processing of the purchase interval cube M obtained from the user history as follows. For each product pair i and j, we count the number of users u such that $M_u[i, j]$ is nonzero. If the number of users that purchase this pair of products is less than some threshold, then we will discard this pair of products and set $M_u[i, j]$ to a null value.

4.3.2 Utility Model with Purchase Intervals

Due to the sparsity problem, it is impossible to generate all the intervals between pairs of products from a user's own purchase history. We address this by using the purchase history of similar users to estimate the purchase interval information. We denote the similarity between user u and u' as $sim(u, u')$. Then the average purchase interval between two products i and j for user u can be estimated as follows:

$$d'_{u,i,j} = \frac{\sum_{u' \in U} d_{u',i,j} * (1 + sim(u, u'))}{\sum_{u' \in U} (1 + sim(u, u'))} \tag{4.7}$$

The similarity between users can be obtained by using some existing latent factor based methods. In our experiments, we use the degree of overlap in the users' purchase histories as a measure of their similarity.

Having obtained the average purchase interval between product pairs for each user, the next step is to incorporate the interval factor into the utility plus model. Suppose a user u has already purchased a set of products I_u and s/he logs into the system at time t. Before we recommend some product j, we want to consider the effect of the interval between a product $i \in I_u$ and j. We model this purchase interval factor $PI(u, i, j)$ as follows:

$$PI(u, i, j) = \frac{1}{log_2(|t - t_i - d'_{u,i,j}| + 2)} \tag{4.8}$$

Equation 4.8 reduces the effect of purchase intervals by the distance between $(t - t_i)$ and $d_{u,i,j}$. For example, suppose we have estimated that u typically buys product j $d'_{u,i,j}$ days after buying product i, then at time point t we will more confidence to recommend product j to u if $|(t - t_i)| = d'_{u,i,j}$.

We can now modify the utility surplus function to include the purchase interval factor as follows:

$$\begin{aligned} US^+(X_i, i, u) = & \qquad (4.9) \\ & \delta_{u,i} \cdot (X_i + 1)^{\gamma_i} - X_i^{\gamma_i}) \cdot (1 + max_{j \in H_u} PI(u, j, i))^{\mu_i} \\ & -\alpha_u \cdot price_i \end{aligned}$$

where $\delta_{u,i}$ is the basic utility of product i to user u and α_u is the sensitivity of u to the product price p_i. The term $((X_i + 1)^{\gamma_i} - X_i^{\gamma_i})$ considers the return rate γ_i for product i, while the term $(1 + max_{j \in H_u} PI(u, j, i))^{\mu_i}$ accounts for the purchase interval factor. Note that we find the product pair (j, i) that has the largest purchase interval effect and add an product-specific parameter μ_i to tune the effect of each product.

4.3.3 Parameter Estimation

In this section, we discuss how we estimate the parameters $\delta_{u,i}$, γ_i, μ_i and α_u in Equation 4.9. Recall that

$$\delta_{u,i} = \sum_k \beta_u^k * c_i^k$$

We use the matrix factorization method [45] to learn the user preference for the product features. Given a $|U| \times |I|$ user-product matrix A, each entry $a_{u,i}$ in A can be estimated as

$$\hat{a}_{u,i} = \left\langle q_i^T, p_u \right\rangle \tag{4.10}$$

With this, we replace β_u^k and c_i^k in Equation 4.10 with p_u and q_i respectively. We have $\delta_{u,i} = \hat{A}_{u,i}$ This allows us to use matrix factorization to estimate the utility of a product to a user.

Similar to [90], we define the joint probability of these parameters as:

$$\begin{aligned} JP &= \prod_u Pr(p_u) \prod_i Pr(q_i) \prod_u Pr(\alpha_u) \prod_i Pr(\gamma_i) \prod_i Pr(\mu_i) \\ & \prod_{u,i,t} Pr(r_{u,i,t}|US^+(X_i, i, u)) \end{aligned} \tag{4.11}$$

where $Pr(r_{u,i,t}|US^+(X_i, i, u))$ denotes the conditional probability of a user u making the decision to purchase product i at time t given the utility surplus value $US^+(X_i, i, u)$ at time t.

Here, we define

$$Pr(r_{u,i,t}|US^+(X_i, i, u)) = \frac{1}{1 + e^{-r_{u,i,t} \cdot US^+(X_i,i,u)}} \tag{4.12}$$

where $r_{u,i,t} = 1$ if user u purchase i at time t. Otherwise $r_{u,i,t} = -1$. Note that a higher utility surplus value indicates that the user is more likely to purchase the product i.

The parameters can be estimated by maximizing the joint probability. We assume Gaussian priors on all the model parameters $N(\varphi, \frac{1}{\lambda})$. For p_u and q_i, we have $\varphi_{p_u} =$

$\varphi_{q_i} = 0$ and variance λ_1. The mean and variance for α, γ and μ are $\varphi_\alpha, \varphi_\gamma$, φ_μ and $\lambda_\alpha, \lambda_\gamma$, λ_μ respectively. Then maximizing the joint probability is equivalent to minimizing the negative log likelihood as follows:

$$\begin{aligned}
(p_u, q_i, \alpha_u, \gamma_i, \mu_i) &= argmin[-logJP] \\
&= argmin\frac{1}{2}\lambda_1 \sum_u \|p_u\|^2 + \frac{1}{2}\lambda_1 \sum_i \|q_i\|^2 + \\
&\frac{1}{2}\lambda_\alpha \sum_u (\alpha_u - \varphi_\alpha)^2 + \frac{1}{2}\lambda_\gamma \sum_i (\gamma_i - \varphi_\gamma)^2 + \\
&\frac{1}{2}\lambda_\mu \sum_i (\mu_i - \varphi_\mu)^2 + \sum_{u,i,t} log(1 + e^{-r_{u,i,t} \cdot US(X_i,i,u)})
\end{aligned}$$

We use the stochastic gradient descent method to find the optimal values for the parameters. Our learning algorithm updates the parameters by using the following first order derivatives:

$$\begin{aligned}
p_u &= p_u - \theta_1 \cdot \frac{\partial(-logJP)}{\partial p_u} \\
q_i &= q_i - \theta_1 \cdot \frac{\partial(-logJP)}{\partial q_i} \\
\alpha_u &= \alpha_u - \theta_2 \cdot \frac{\partial(-logJP)}{\partial \alpha_u} \\
\gamma_i &= \gamma_i - \theta_3 \cdot \frac{\partial(-logLJP)}{\partial \gamma_i} \\
\mu_i &= \mu_i - \theta_4 \cdot \frac{\partial(-logJP)}{\partial \mu_i}
\end{aligned} \tag{4.13}$$

At each iteration, the learning rate is controlled by θ_1, θ_2, θ_3 and θ_4. The values of θ_1, θ_2, θ_3, θ_4, λ_1, λ_α, λ_γ and λ_μ can be set by cross-validation.

4.4 Experimental Study

In this section, we report the results of the experiments we have carried out to evaluate the effectiveness of our proposed approach. We call our method $PIMF$. We also compare the performance of our method with three algorithms:

1. $TopPop$. This is a baseline algorithm which recommends the top-K most popular products to users;

2. MF [45]. Matrix Factorization is a widely used recommendation algorithm based on latent factors and matrix factorization.

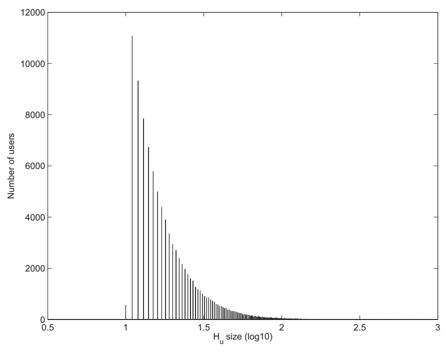
3. $UTMF$ [90]. $UTMF$ is the state-of-the-art recommendation algorithm for e-commerce which incorporates marginal net utility and the law of diminishing returns into matrix factorization approach.

Since MF, $UTMF$ and $PIMF$ all have a learning phase to set the optimal values for their parameters, we use the first 90% of each user's purchase history for training and the remaining 10% for testing. We associate each user u and each product i with a feature vector p_u and q_i respectively. We set the dimension of the latent factor is 50. We determine use 50 here is because we want to experiment within a reasonable running time. Although the performance for all MF-based methods will typically improve as the number of latent factors increases, however, the runtime will also increase significantly.

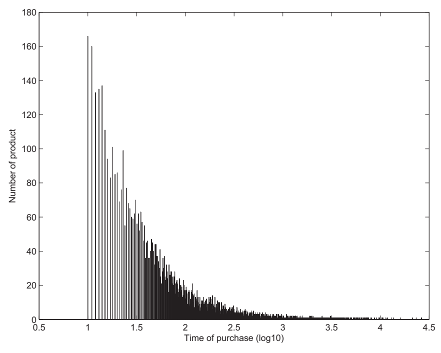
We use both positive and negative training points. Each purchase record is a positive point. Given that the density of e-commerce data set is usually much lower than that of movie or music rating dataset [34], we randomly sample 0.1% of the missing entries and set them to 0 to train the MF model. For $UTMF$ and $PIMF$, we randomly sample 1% from the missing entries as negative points.

4.4.1 Experiment Dataset

We collect a dataset from Jingdong which is one of the biggest B2C e-commerce websites in China. We crawled products that belong to the electronic category. The dataset consists of 197,025 users, 98,302 products and 2,610,279 purchase records from January 2010 to November 2011. The density of the dataset is 0.013%.



(a) # Distinct Products



(b) Product Popularity

Figure 4-4: Characteristics of Jingdong Dataset

We pre-process the dataset to remove products that have less than 10 purchase records, as well as users who have purchased less than 10 products. The clean dataset D has 96,882 users, 6,921 products and 2,302,066 purchase records. The density of this dataset is now 0.34%. We sort the purchase records according to the users to form their purchase history. Records in each user purchase history are ordered by their time stamps. Figure 4-4 shows the distribution of the number of distinct number of products bought by users and the popularity of the products. As expected, there is a long tail in the products purchased.

4.4.2 Evaluation Metrics

Our evaluation metrics include temporal diversity, conversion rate, precision and recall. Temporal diversity measures the differences between two lists of recommendations when a user logs into the system at different times. For example, suppose a user is given a set of 5 recommended products at time t. If he logs into the system later and only 1 of the 5 recommendations is different, then the diversity between the two lists is 1/5 = 0.2. [49] derives the diversity between two lists L_t and $L_{t'}$ from their set theoretic difference as follows:

$$L_{t'} \setminus L_t = \{x \in L_{t'} | x \notin L_t\} \tag{4.14}$$

$$diversity(L_t, L_{t'}, K) = \frac{|L_{t'} \setminus L_t|}{K} \tag{4.15}$$

where K is the number of products in each list. If the two lists are exactly the same, then diversity is 0.

Conversion rate is a commonly used metric in e-commerce to determine if a user has obtained at least one good recommendation. If the user purchased at least one product from the recommended top K list, then we say that the user has switched from a browser to a buyer. If L is the list of recommended products and L' is the list of products actually purchased by the user, then the conversion rate is given by:

$$conversion\ rate@K = \begin{cases} 1 & \text{if } |L \cap L'| > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

We compare the conversion rate of the various algorithms by taking the average of values computed for each test user.

The standard definitions for precision and recall are as follows:

$$recall@K = \frac{|L \cap L'|}{|L'|} \tag{4.17}$$

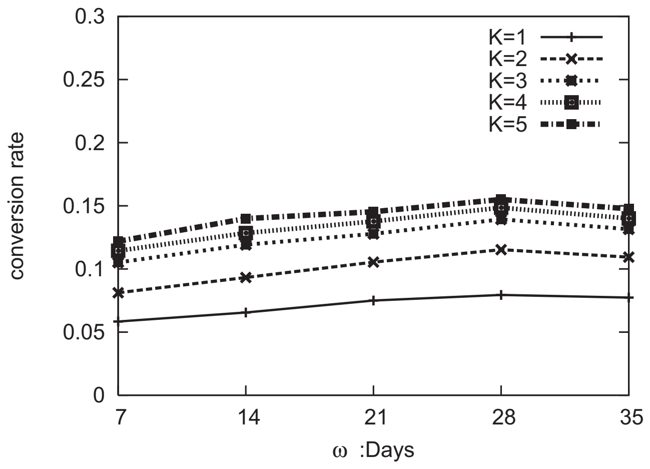
$$precision@K = \frac{|L \cap L'|}{K} \tag{4.18}$$

4.4.3 Results and Analysis

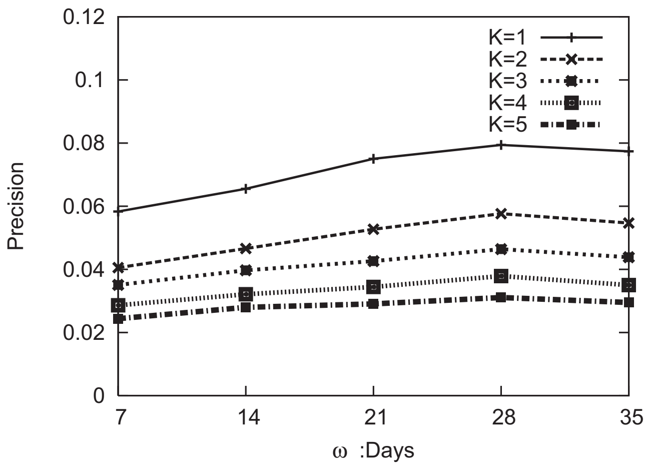
We first examine the effect of the window size ω on the conversion rate, recall and precision of our proposed algorithm *PIMF*. We set ω to 7, 14, 21, 28 and 35 days. This implies that the purchase intervals generated are less than or equals to ω. Table 4.2 shows the accumulated density of the purchase interval matrix generated by Algorithm 1 for the different values of ω. We observe that the density increases as we consider larger purchase intervals.

Table 4.2: Effect of ω on the Density of Purchase Interval Matrix

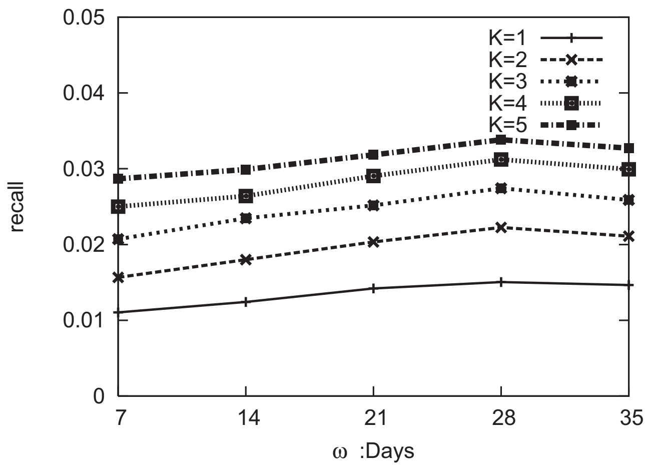
ω	7	14	21	28	35
Density	0.41%	0.59%	0.65%	0.74%	0.83%



(a) Conversion Rate

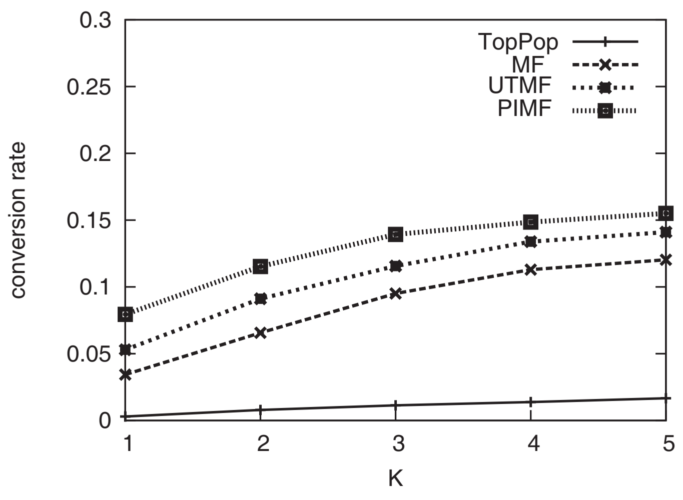


(b) Precision

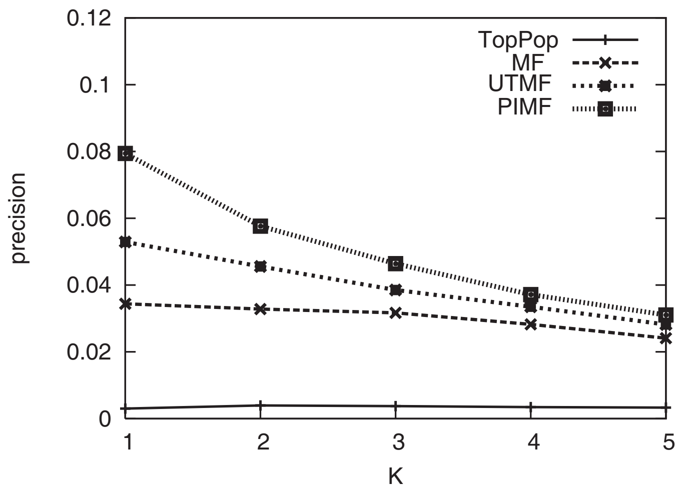


(c) Recall

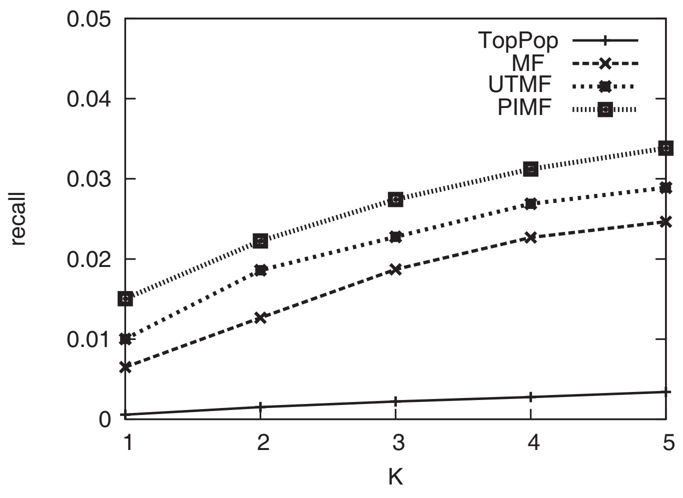
Figure 4-5: Effect of Window Size ω on $PIMF$



(a) Conversion Rate



(b) Precision



(c) Recall

Figure 4-6: Comparative Study

Figure 4-5 gives the results of this experiment for different values of K. We see that the conversion rate increases as K increases. Similar observations are made for the precision and recall graphs. The best result is obtained when $\omega = 28$. We use this window size for the rest of the experiments.

Next, we compare the performance of our algorithm $PIMF$ to $TopPop$, MF and $UTMF$. Figure 4-6 shows the results. Since $TopPop$ recommends the most popular products and is not personalized, it has the lowest conversion rate compared to the personalized MF-based methods. When we compare $PIMF$ with MF and $UTMF$, we see that using purchase intervals significantly improves the conversion rate, recall and precision of the recommendations.

We give an example of the results obtained. Table 4.3 gives the products corresponding to the product IDs. Table 4.4 shows a sample of the purchase intervals between the products 229040002 (DVD-ROM) and 24101000e (Wireless Router) computed from the dataset. We observe that users typically purchase a DVD-ROM about 25 days after purchasing a Wireless Router.

Table 4.3: Sample Products

Product ID	Product Title
22306000e	Galaxy GT430 512MB DDR5 Graphics Card
229040002	LITEON 18X DVD-ROM
23d010004	PHILIPS SWA1938/93-5 Internet Cable 5M
23d02000e	CHOSERL Q505 VGA Cable 1.5M
24101000e	TP-LINK WR340G+ 54M Wireless Router

Table 4.5 shows a sample of user $U_{10370829}$'s purchase history in the training data. The first entry in the testing data for this user is $< 229040002, 2010-07-16 >$ indicating that s/he purchases the product 229040002 (DVD-ROM) at time $2010-07-16$. Upon examining the list of products recommended by the various algorithms, we find that our method $PIMF$ also recommends this product 229040002 (DVD-ROM) to the user, whereas both MF and $UTMF$ recommend the product 23d010004 (Internet Cable). From the user purchase history data, we find that users either purchase the prod-

Table 4.4: Sample Purchase Intervals (in days)

Product ID	Product ID	Interval
24101000e	229040002	22
24101000e	229040002	25
24101000e	229040002	25
24101000e	229040002	27
24101000e	229040002	27
24101000e	229040002	29
24101000e	229040002	22
24101000e	229040002	23
24101000e	229040002	26
24101000e	229040002	27

ucts 24101000e (Wireless Router) and 23d010004 (Internet Cable) together, or purchase Internet Cable 1-2 days after buying Wireless Router. Since MF and $UTMF$ do not consider the purchase intervals, both of them recommend the product Internet Cable to user $U_{10370829}$.

Table 4.5: Sample of User $U_{10370829}$'s Purchase History in Training Data

Product ID	Purchase time
23d02000e	2010-03-19
23201000d	2010-03-30
23d100002	2010-04-20
23d060003	2010-04-26
233020011	2010-05-25
24101000e	2010-06-19
22306000e	2010-06-26

4.4.4 Temporal Diversity

We also investigate the temporal diversity of the various algorithms. We obtain the first recommendation list L at the time stamp of the first entry in the testing data. Then we partition the testing data into two and put the first partition into the training data. We re-run the algorithms on this larger training data and obtain a second recommendation list L' at the time stamp of the first entry in the second partition.

Figure 4-7 shows the temporal diversity result for the top-5 and top-10 recommended

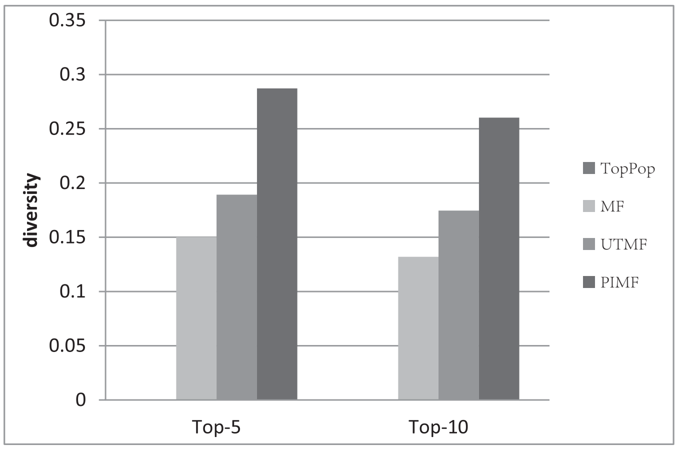


Figure 4-7: Top-5 and Top-10 Temporal Diversity for $TopPop$, MF, $UTMF$, $PIMF$

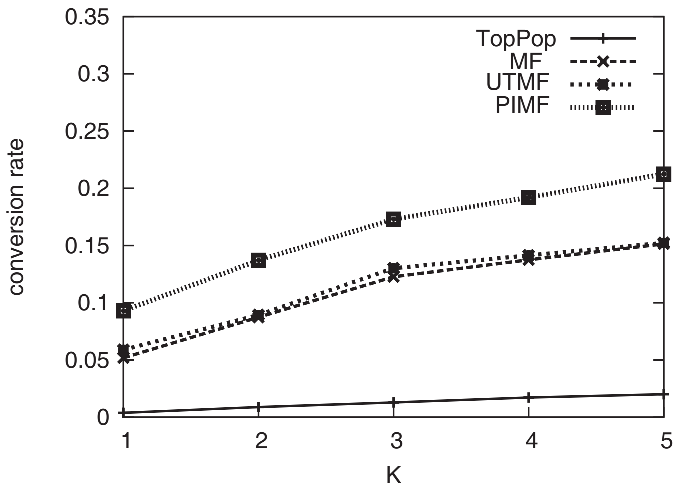
items. The temporal diversity of $TopPop$ is 0 because both its recommendation lists contains the same popular products. Our proposed method $PIMF$ has the highest temporal diversity compared to MF and $UTMF$ since the products recommended depends on the time at which the user logs into the system.

Based on the results, we envisage that our proposed approach would be useful in email-based or message-based marketing applications. At different time points, when we want to push some products to consumers, we can determine the products that would be most attractive to consumers at that time point instead of only products that consumers may potentially like.

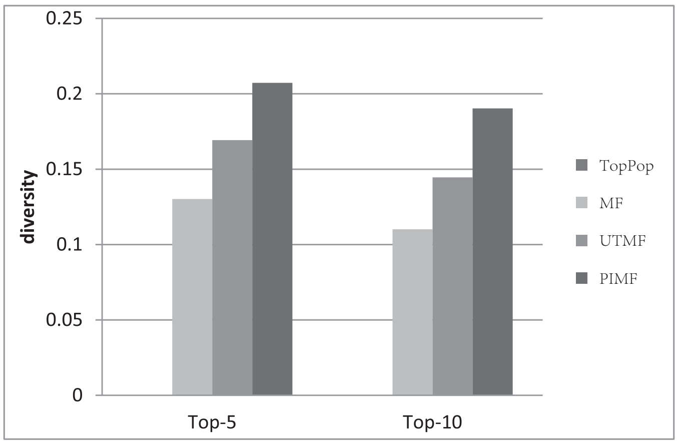
4.4.5 Effect of Taxonomy

Finally, we study the effect of taxonomy on the various algorithms. Based on the original e-commerce website, products are classified under various levels of category, e.g., both the products *"DELL V3400R-426S 14 inch laptop"* and *"DELL V3400R-426S 13.3 inch laptop"* fall under the same brand *"Dell"* which is under the product type *"Laptops"*. The product type *"Laptops"* is under *"Computers"* which is under *"Computers & Software"*. Thus, we could consider both models of the Dell laptops as the same product type.

In this experiment, we replace specific product models by their product type and



(a) Conversion Rate



(b) Temporal Diversity

Figure 4-8: Effect of Taxonomy

brand, that is, we generalize *"DELL V3400R-426S 14 inch laptop"* and *"DELL V3400R-426S 13.3 inch laptop"* to a more general product called *"DELL Laptops"*. Then we re-generate the purchase interval cube and learn a new model for recommendation. Note that we drop the product price term from Equation 4.9 since we would not have the price of the generalized product.

Figure 4-8 gives the conversion rate and temporal diversity of the algorithms on this dataset. We observe that all the methods show improvement in the conversion rate. The temporal diversity of all the algorithms decrease slightly because the total number of distinct products is reduced when we merge the individual product models of the same brand to their product type. The gap between $PIMF$ and the other algorithms widens as the density of the purchase interval cube increases with the use of taxonomy.

4.5 Summary

In this chapter, we have described how purchase intervals are able to increase the temporal diversity and conversion rate in product recommender systems. We have designed a model that combines purchase interval information in users purchase history with marginal utility. We have also developed an efficient algorithm to generate a purchase interval cube by scanning users' purchase history once. Experimental results on the real world Jingdong e-commerce dataset demonstrate that the proposed approach improves the precision, recall, conversion rate of existing algorithms. It also significantly improves temporal diversity which is essential for product recommendation systems.

CHAPTER 5

UTILIZING PURCHASE INTERVALS IN LATENT CLUSTERS FOR PRODUCT RECOMMENDATION

Online shopping service providers aim to provide users with quality list of recommended items that will enhance user satisfaction and loyalty. The dominant collaborative filtering method, matrix factorization, typically used in product recommenders, depicts each user as one preference vector, and sparsity remains a challenge for matrix factorization given the huge number of users and products. In practice, we observe that users may have different preferences when purchasing different subsets of items, and the periods between purchases also vary from one user to another.

In this chapter, we describe a probabilistic approach to learn latent clusters in the large user-item matrix, and incorporate temporal information into the recommendation process. The clusters obtained capture users' hidden preferences for items as well as item time sensitivity. This work has been published in [98]

5.1 Motivation

A widely adopted approach for product recommendation is collaborative filtering which associates a user with a group of similar users based on their preferences over *all* the items, and recommends to the user items which the group likes. The assumption is that similar users will have similar preferences on all the items. The state-of-the-art collaborative filtering method, matrix factorization [47], reduces the dimension of the high dimensional user-item matrix by finding latent vectors for users and items. However, matrix factorization constrains each user to one preference vector, and sparsity remains a challenge for matrix factorization given the huge number of users and products [53].

Studies on consumer behavior have shown that the underlying mechanisms governing user purchase behavior is very complex. A user is often interested in more than one subset of products, indicating his/her diverse purchase behavior. Two users may purchase the same product for different reasons, demonstrating the diverse characteristics of a product. The work in [95] shows that two users with similar preferences on a subset of items may have vastly different preferences on another subset of items, and propose to co-cluster both users and items into multiple subgroups to improve recommendation accuracy. The authors develop an approach called MCoC which simultaneously clusters users and items into smaller subgroups before applying any recommendation algorithm.

The clustering step in MCoC has two steps. First, the high dimensional $M \times N$ user-item matrix is transformed to a $(M + N) \times d$ matrix, where $d << min\{M, N\}$. This transformation maps both users and items into the same latent space, and each dimension can be seen as a latent feature for a user or a item. Next, a soft clustering fuzzy c-means is applied on the lower dimensional latent space to obtain clusters of users and items.

Inspired by the MCoC framework, we want to examine whether the purchase interval information described in Chapter 4 can be utilized to find clusters of users with similar purchase behaviors. We first employ tensor decomposition [44] to reduce the $M \times N \times N$ purchase interval cube to a $M \times d$ matrix, before applying fuzzy c-means to find clusters on the lower dimension matrix. Figure 5-1 shows the purchase interval data for 4 users and 5 items. Table 5.1 gives the results after applying tensor decomposition and fuzzy c-means on the purchase interval cube obtained from the matrices in Figure 5-1. Suppose we set the number of clusters to 2 for the fuzzy c-means algorithm, then we will obtain the cluster membership for each user as shown in the last column in Table 5.1. If we set the membership threshold as 0.3, then we will cluster the users into two clusters $c_1 = \{u_1, u_2\}$ and $c_2 = \{u_2, u_3, u_4\}$.

Figure 5-1: Example Purchase Interval Matrices for Users (Unit: Day)

	i_1	i_2	i_3	i_4	i_5
i_1		4		6	
i_2	2		2		
i_3					6
i_4	5		4		
i_5			2	2	

(a) User u_1

	i_1	i_2	i_3	i_4	i_5
i_1		3	5	6	
i_2				3	
i_3					6
i_4	6				
i_5			3		

(b) User u_2

	i_1	i_2	i_3	i_4	i_5
i_1			4		2
i_2				3	4
i_3	6		3		
i_4					4
i_5	5				

(c) User u_3

	i_1	i_2	i_3	i_4	i_5
i_1			5		
i_2				4	3
i_3	5		3		
i_4		2			3
i_5	4				

(d) User u_4

Table 5.1: Tensor Decomposition & Clustering Result

	Latent Space			Membership	
	d=1	d=2	d=3	c=1	c=2
u_1	1.49167494	1.06868527	1.68747388	0.96	0.04
u_2	1.78127964	1.74024785	0.46215857	0.42	0.58
u_3	0.83263382	1.30810371	2.18367095	0.28	0.72
u_4	0.91476136	2.34232086	0.81208335	0.03	0.97

We implement the above Purchase Interval Clustering method (*PIC*) and compare it with MCoC on the Jingdong e-commerce dataset. We apply matrix factorization on the clusters obtained by *PIC* and *MCoC* respectively. We find that the clusters obtained by *PIC* did not yield better results than those obtained by *MCoC* (see Section 4.3.5). Further investigation reveals that the density[1] of the purchase interval cube (0.00092%)

[1]We define the density as the number of non-empty cells in the matrix/cube divided by the size of matrix/cube.

is worse than the user purchase data or user-item matrix (0.21%). Based on the initial experimental results, we purpose to develop a clustering method which not only utilize user purchase data, but also incorporate purchase interval information to improve the cluster quality.

In this chapter, we describe a bi-cluster based collaborating filtering method, and incorporate temporal information into the recommendation process. Our goal is to find user-item subgroups in the large user-item matrix that effectively capture the users' preferences for items as well as item time sensitivity. Item time sensitivity determines the relevance of an item at a given time stamp. In particular, we adopt a probabilistic approach to discover user-item clusters, and refine the clusters by utilizing user purchase intervals to find the most relevant items for the given time stamp. Then we apply matrix factorization on each of these clusters to personalize the recommendations.

Our approach leverages on the Latent Dirichlet Allocation (LDA) [13] method to capture the hidden aspects of user interests and distribution of products purchased. Just as one has some criteria in mind when making some purchase, a user may be interested in the latest IT gadgets such as mobile phone regardless of the price, while another user tends to splurge on the latest fashion and is more careful about the cost of IT devices. Thus, if we are able to identify the shared interests behind users purchasing activities, we could significantly improve the quality of recommendations in e-commerce sites like Amazon and Taobao.

We create "documents" that contain the ids of items bought by a user previously and utilize latent Dirichlet allocation to generate latent groups. We introduce the notion of a cluster purchase interval factor which estimates the probability that users in a cluster will purchase an item. Experiment results on a real world e-commerce data set demonstrate that our approach significantly improves the conversion rate (by up to 10%), as well as the precision and recall of state-of-the-art product recommender methods. We also compare our approach with other clustering methods to show that the good performance is not simply because of the use of purchase intervals.

5.2 Proposed Approach

Our proposed framework has three main phases. The first phase utilizes an LDA-based method to generate latent user-item clusters. The second phase refines the clusters by incorporating information on the user purchasing intervals. The last phase performs matrix factorization on each cluster and combines the top K items from the clusters that the user occurs in to obtain the final list of recommended items.

We describe the details of each phase in the following subsections.

5.2.1 Generate Latent Clusters

A topic model is a statistical model developed for discovering hidden topics from a collection of documents. The assumption is that every document is a mixture of topics, and words in a document describes these hidden topics.

Latent Dirichlet Allocation (LDA) [13] has become a well-established method for modeling the topic distribution of a set of documents D. Similar to Probabilistic Latent Semantic Indexing ($PLSI$) [37], each document in the LDA model is represented as a mixture of a fixed numbers of topics Z, with topic z having a probability $Pr(z|d)$ in document d. Each topic is a probability distribution over a finite vocabulary of words W, with word w having probability $Pr(w|z)$ in topic z.

Given the parameters α and β where α is a vector of dimension $|Z|$ and β is a vector of dimension $|W|$, the document generation process is as follows:

1. Choose the number of topics.
2. Choose $\theta \sim Dir(\alpha)$
3. For each word w_n
    - Choose a topic $z_n \sim Multinomial(\theta)$
    - Choose word w_n from $Pr(w_n|z_n, \beta)$

LDA has been shown to be effective in document classification and recently, it has been successfully applied to user recommendations in microblogs [100]. In this work, we propose to use the LDA model to identify the hidden interests behind user purchasing activities and generate latent user-item clusters.

Each user u can be regarded as a document consisting of a list of items i s/he has bought before. We denote $Pr(z|u)$ as the multinomial probability of topic z given a user f, and $Pr(i|z)$ as the multinomial probability of a item i given z. We use d_u to denote a user document, and the content of d_u is the list of item purchased by u. Therefore our document corpus D is given by

$$D = \bigcup_{u \in U} d_u \tag{5.1}$$

We apply LDA on D to generate a pre-defined number of topics Z. Figure 5-2 depicts the graph model for this representation.

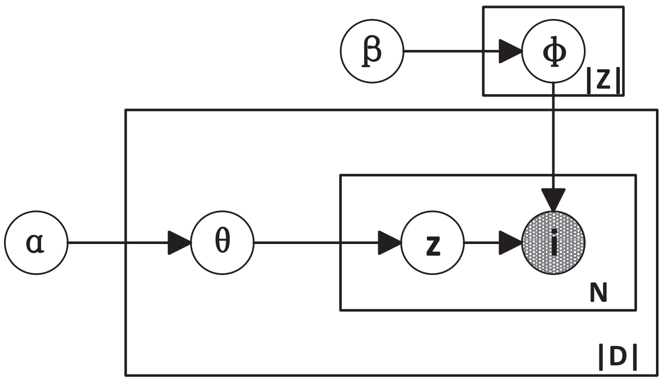


Figure 5-2: Graph Model Representation

For each topic $z \in Z$, we form a cluster c such that the users in c, denoted as $c.U$ is given by

$$c.U = \{u \mid u \in U \wedge Pr(z|d_u) > \gamma\} \tag{5.2}$$

where γ is some threshold.

The set of clusters C generated by LDA are latent clusters, where $C = Z$ with estimates $Pr(c|d_u)$ and $Pr(i|c)$. $Pr(c|d_u)$ indicates the likelihood of a user u belongs to a cluster, and thus can be used to cluster users into latent groups. $Pr(i|c)$ indicates the importance of an item i to a cluster c, and thus can be used to estimate the likelihood of users in c purchasing i. We use these estimate to compute the cluster purchase interval factor in the cluster refinement phase.

Algorithm 3 shows the details of this phase. The input is a dataset H containing the purchase histories of users, the number of clusters N and threshold γ. The output is a set of latent clusters C generated by LDA, where $C = Z$ with estimates $Pr(c|d_u)$ and $Pr(i|c)$. Each cluster $c \in C$ is associated with a set of users $c.U$ whose $Pr(c|d_u) > \gamma$.

Algorithm 3: Generate Latent Clusters

```
   input : 1. Set of purchase records $H = \{< u, i, t >\}$,
           2. Number of clusters $N$,
           3. Threshold $\gamma$
   output: $C, Pr(c|d_u), Pr(i|c)$
 1 $D = \emptyset$;
 2 foreach $u \in U$ do
 3     $d_u = \{i \mid i \in I \land \exists < u, i, t > \in H\}$
 4     $D = D \cup \{d_u\}$;
 5 end
 6 $Z \leftarrow LDA(D, N)$;
 7 foreach $z \in Z$ do
 8     $c \leftarrow \emptyset$
 9     $c.U = \{u \mid u \in U \land Pr(z|d_u) > \gamma\}$;
10     $C = C \cup \{c\}$;
11 end
```

5.2.2 Refine Latent Clusters

Before we use the clusters obtained in the previous step to make recommendations at some time point T, we want to refine the set of items $c.I$ in each cluster $c \in C$ such that users in c have a high probability to purchase the items at time t.

We normalize the purchase interval factor of each user in Equation 3.8 as follows:

$$NPI(u, i, j, t) = \frac{PI(u, i, j, t)}{\sum_{i' \in I} PI(u, i', j, t)} \tag{5.3}$$

Then the probability that the users in a cluster c will purchase an item j at time t after purchasing i is given by:

$$CPI(c, j, t) = Pr(j|c) * \sum_{u \in c.U} Pr(c|d_u) * (\sum_{i \in I} (1 + NPI(u, i, j, t)) * Pr(i|c)) \tag{5.4}$$

We call CPI the cluster-level purchase interval factor at t. The formula for CPI comprises of three terms. The first term $Pr(j|c)$ considers the importance of the item j to the cluster, while the second term $Pr(c|d_u)$ weights the user's interest to the cluster. The last term $(1 + NPI(u, i, j, t)) * Pr(i|c)$ in the equation measures the purchase interval effect of item i in purchase history to item j.

We can rank the items in each cluster according to their CPI values. Then we put items with the highest $CPI(c, i)$ values into $c.I$ for each $c \in C$

Note that the number of items to be placed in each cluster should be proportionate to the number of users in that cluster, and is given by

$$\tau * |I| * |c.U|/|U|$$

where τ is a tuning factor dependent on the dataset and is obtained experimentally.

Users who have not purchased any item $i \in c.I$ are removed from $c.U$. Let H be the set of user purchase records. Then the subset of purchase records in a cluster c, denoted as $c.H$, is given by:

$$c.H = \{< u, i, t > \ | \ < u, i, t > \in H \wedge u \in c.U \wedge i \in c.I\} \tag{5.5}$$

5.2.3 Recommend Items

After refining the clusters, the next phase is to compute candidate items from these clusters for recommendation. We utilize the $PIMF$ collaborating filtering method on each cluster to accomplish this.

Given a target user u, we obtain the score that s/he will purchase the item i in cluster c as $score(u, i, c)$. Since u may occur in more than one latent clusters, we need to combine the different lists of candidate items recommended in each cluster. Here, we sum up the scores of all the clusters that u and i appears in as follows:

$$sumScore(u, i) = \sum_{c \in C} (score(u, i, c) \times Pr(c|d_u)) \tag{5.6}$$

where $Pr(c|d_u)$ is the weight of user u's interest in cluster c.

Finally, we sort the scores for each user u and output the top-K items to recommend to u.

Algorithm 4 shows the details of our proposed approach. We call our method c-$PIMF$ for cluster-based Purchase Interval Matrix Factorization. The input is the set of latent clusters obtained from Algorithm 3. Lines 1 to 2 computes the normalized purchase interval factor at time T for each user. Lines 3 to 7 computes the cluster level purchase interval factor CPI and refines the clusters with items that have the high CPI values. Then we perform $PIMF$ on each cluster (lines 9-12). Lines 14-18 aggregates the scores from each cluster and we obtain a ranked list of recommended products for each user.

Let us illustrate our approach with the sample user purchase histories in Figure 5-3. If we let $N = 2$, then Algorithm 3 will generate two clusters $c_1 = \{u_1, u_3, u_4\}$ and $c_2 = \{u_1, u_2, u_4\}$.

Suppose our target user is u_4, and we want to make recommendations at time point $t = 24$. Table 5.2 shows the CPI values of the items computed for each cluster. If we want the top three items from each cluster, then we will put items i_2, i_4, i_5 in cluster c_1

Algorithm 4: *c-PIMF* Algorithm

```
input  : 1. Set of clusters $C$
         2. Set of purchase records $H = \{< u, i, t >\}$
         3. Number of latent factors $LF$
         4. Tuning factor $\tau$
         4. Time $T$
output: Ranked recommendation list
use Equation 3 to compute purchase interval factor $PI$ for each user $u \in U$;
use Equation 7 to compute the normalized purchase interval factor $NPI$ at time $T$
for each user;
foreach $c \in C$ do
    Use Equation 5.4 to compute the cluster purchase interval $CPI$ ;
    Rank items according to their $CPI$ values;
    $c.I$ = top $(\tau * |c.U|/|U| * |I|)$ items;
    $c.H = \{< u, i, t > \mid < u, i, t > \in H \wedge u \in c.U \wedge i \in c.I\}$;
end
$R = \emptyset$;
foreach $c \in C$ do
    $PIMF(c, LF)$;
    $R_c = \{< u, i, score(u, i, c) > \mid u \in c.U \wedge i \in c.I\}$;
    $R = R \cup \{R_c\}$;
end
$Result = \emptyset$;
foreach $pair$ $(u, i)$ do
    Use the scores in $R$ to compute $sumScore(u, i)$ according to Equation 5.6;
    $Result = Result \cup < u, i, sumScore(u, i) >$;
end
foreach $u \in U$ do
    Return the ranked list of items in $Result$;
end
```

since these items have the highest CPI values. Similarly, we will put items i_2, i_3, i_5 into cluster c_2. For each cluster, we apply $PIMF$ and finally recommend item i_2 to u_4.

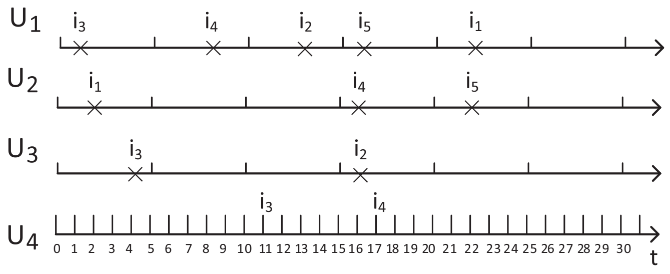


Figure 5-3: Example of Users' Purchase History

On the other hand, suppose we want to make recommendations to u_4 at time point $t = 27$. Table 5.3 shows the CPI values obtained for each cluster. Based on the top three CPI values for each cluster, we have $c_1.I = \{i_1, i_4, i_5\}$ and $c_2.I = \{i_1, i_2, i_3\}$. Item i_5 will be recommended to u_4 after applying $PIMF$ on each cluster.

Table 5.2: CPI Values at $t = 24$

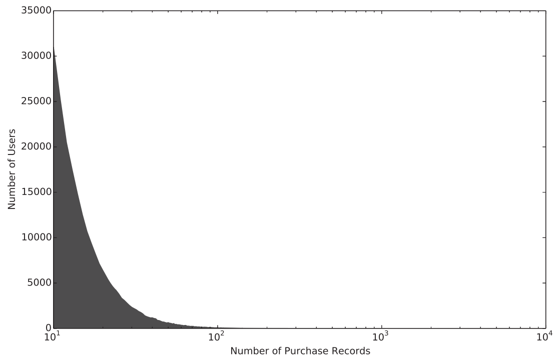
Item	Cluster c_1	Cluster c_2
i_1	0.391	0.296
i_2	0.414	0.407
i_3	0.368	0.426
i_4	0.575	0.352
i_5	0.598	0.370

Table 5.3: CPI Values at $t = 27$

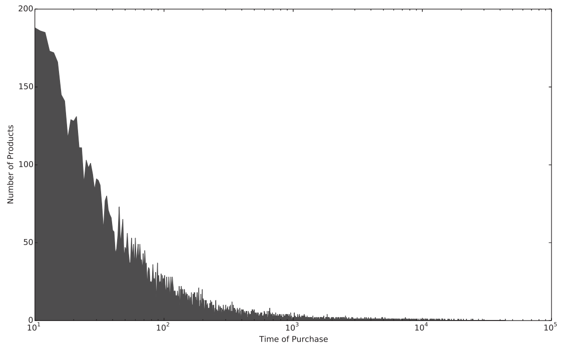
Item	Cluster c_1	Cluster c_2
i_1	0.399	0.340
i_2	0.378	0.376
i_3	0.357	0.394
i_4	0.483	0.304
i_5	0.483	0.376

5.3 Experimental study

In this section, we report the results of the extensive experiments we have carried out to evaluate both of the effectiveness and efficiency of our proposed c-$PIMF$ method. We



(a) # Distinct Products



(b) Product Popularity

Figure 5-4: Characteristics of Dataset

compare the performance of our method with the following methods:

1. $TopPop$. This is a baseline algorithm which ranks product items according to their popularity among users and recommends the top-K most popular product items to a target user.

2. $IF\text{-}MF$ [42]. This is the state-of-the-art matrix factorization method for implicit feedback data sets. We set the number of latent factor to be 16 for $IF\text{-}MF$.

3. $PIMF$ As proposed in Chapter 4. This is the state-of-the-art temporal-based matrix factorization method which utilizes purchase interval information for product recommender systems. Since $PIMF$ has a parameter ω which specifies the time window to compute the user purchase intervals, we run a set of experiments to obtain the optimal value of $\omega = 35$ that yields the best performance for $PIMF$.

We implement the methods using Python. We code the LDA model according to [36], and use the C# implementation provided in [28] for the method $IF\text{-}MF$. All the experiments are carried out on an Intel(R) Core(TM) i7-2600 with 3.4 GHz, 8 GB RAM, 64 bit Microsoft Windows 7 operating system.

5.3.1 Experimental Data Set

The experimental dataset is collected from Jingdong which is one of the biggest B2C e-commerce websites in China. In order to observe significant performance distinction among different approaches, we re-crawled a larger dataset than that was used in Chapter 4. We crawled products that belong to the electronic category. The dataset consists of 3,775,069 users, 12,316 products and 12,784,961 purchase records from January 2011 to November 2013. The density of the data set is 0.027%.

We pre-process the dataset to remove products that have less than 10 purchase records, as well as users who have purchased less than 10 products. The processed dataset has 239,468 users, 10,775 products and 5,328,887 purchase records, with a density of 0.21%. We sort the purchase records according to the users to form their purchase history. Records in each user purchase history are ordered by their time stamps. For each user's purchase history, we use the first 90% of the records as training dataset and the remaining 10% as the test data set.

Figure 5-4 shows the characteristics of the dataset. Figure 5-4(a) depicts the distribution of distinct products purchased by users, and Figure 5-4(a) gives the product popularity over time. Both figures show long tails in the distribution.

5.3.2 Evaluation Metrics

We use the conversion rate, precision and recall as our evaluation metrics. Conversion rate is a standard metric in e-commerce to determine if a user has obtained at least one good recommendation. If the user purchased at least one product from the recommended list of top K items, then we say that the user has switched from a browser to a buyer. If

L is the list of recommended products and L' is the list of products actually purchased by the user, then the conversion rate is given by:

$$conversion\ rate@K = \begin{cases} 1 & \text{if } |L \cap L'| > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

We compare the conversion rate of the various methods by taking the average of values computed for each test user.

Precision and recall are defined as follows:

$$recall@K = \frac{|L \cap L'|}{|L'|} \tag{5.8}$$

$$precision@K = \frac{|L \cap L'|}{K} \tag{5.9}$$

We also report the $F1$ score, which is the harmonic mean of precision and recall, given by:

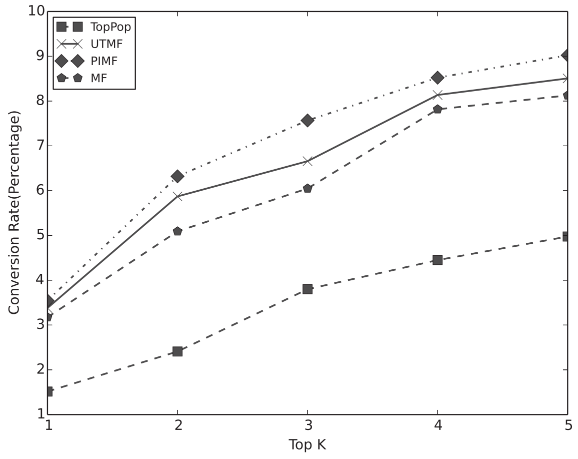
$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.10}$$

5.3.3 Preliminary Experiment

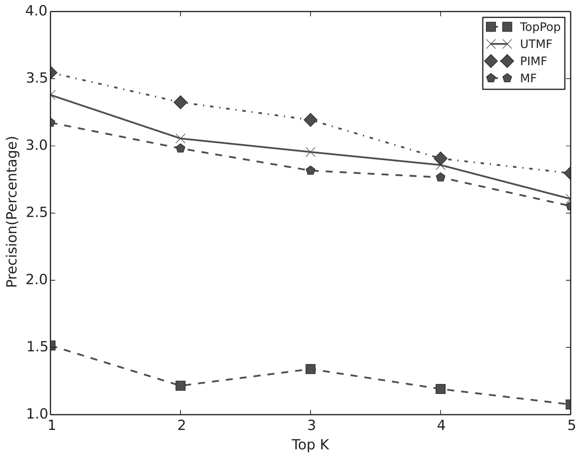
We repeat the comparative experiment in Chapter 4 on the new larger data set. The results in Figure 5-5 shows similar trend. We observe that the method $TopPop$ shows improved performance in this larger data set. This is because of the change in consumers' purchase patterns as Jingdong has become more popular recently and more consumers actually buys the most popular item recommended.

5.3.4 Sensitivity Experiments

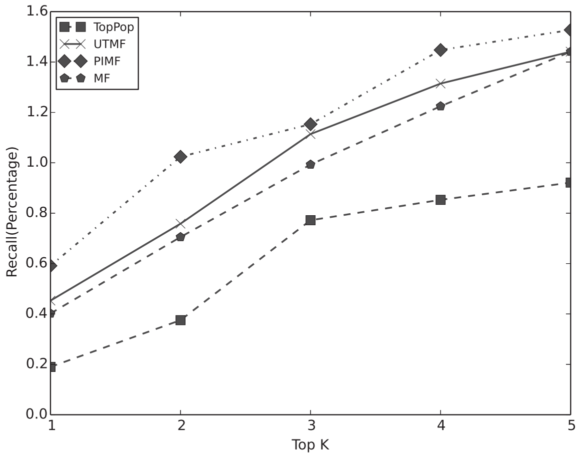
We first examine how the parameters γ, N and LF affect the performance of our proposed $c\text{-}PIMF$ approach. We fix the number of latent factor at $LF = 16$, and vary the threshold



(a) Conversion Rate



(b) Precision



(c) Recall

Figure 5-5: Preliminary Experiment Study

γ and the number of clusters N.

We use $F1$ score to measure the performance of the recommendations. Table 5.4 shows the results for $K = 3$. We see that the $F1$ score increases as the number of clusters increase from 3 to 6, indicating that the additional clusters are able to capture users' preferences for different subsets of items. The $F1$ score drops when $N = 7$ due to information loss as purchase records of users who have low probabilities of purchasing items are removed from clusters.

Based on the results, we use $\gamma = 0.02$ and $N = 6$ for *c-PIMF* in the subsequent experiments.

Table 5.4: Effect of γ and N on *c-PIMF*

γ	N=3	N=4	N=5	N=6	N=7
0.01	0.0122	0.0165	0.0174	0.0191	0.0179
0.02	0.0159	0.0163	0.0177	**0.021**	0.0173
0.04	0.0148	0.0166	0.0169	0.0192	0.0168
0.08	0.0144	0.0152	0.0161	0.0185	0.0163

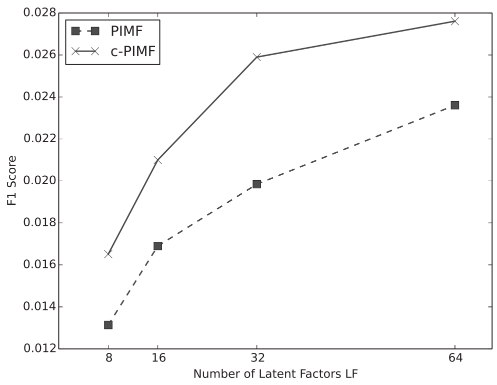


Figure 5-6: Effect of varying latent factor LF

Figure 5-6 shows the $F1$ scores for $PIMF$ and *c-PIMF* as we vary the number of latent factors LF from 8 to 64. We observe that although the performance of both methods improve as LF increases, the proposed *c-PIMF* still outperforms $PIMF$. We also use $LF = 16$ for both $PIMF$ and *c-PIMF* for the rest of our experiments.

5.3.5 Comparative Experiments

Next, we compare the performance of $c\text{-}PIMF$ with the baseline method $TopPop$, and state-of-the-art matrix factorization methods $IF\text{-}MF$ and $PIMF$. Figure 5-7 shows the conversion rate, precision and recall of the various approaches. The results indicate significant improvement achieved by $c\text{-}PIMF$.

The baseline method $TopPop$ has the worst performance as it only recommends the most popular products and it is not personalized. Although the $IF\text{-}MF$ method personalized the recommendations for users by considering their latent preferences, it does not capture the temporal information.

Our proposed method $c\text{-}PIMF$ outperforms $PIMF$ with a 10% improvement in conversion rate as it is able to capture users' latent preferences for different subset of items.

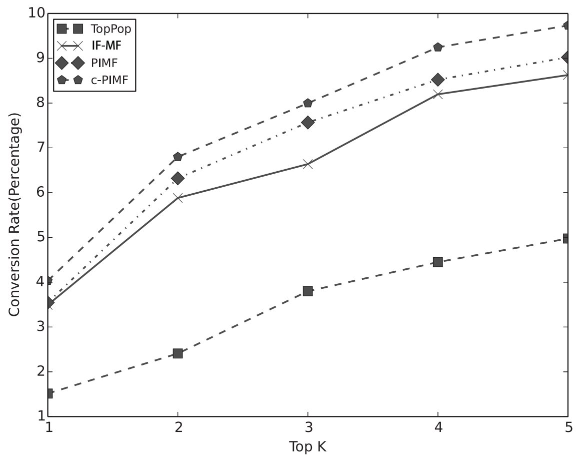
5.3.6 Analysis of Clustering Methods

We also analyze the impact of clustering methods on the performance since our proposed method applies matrix factorization on clusters which have lower sparsity compared to the original data set. We compare our approach to find clusters with the following methods:

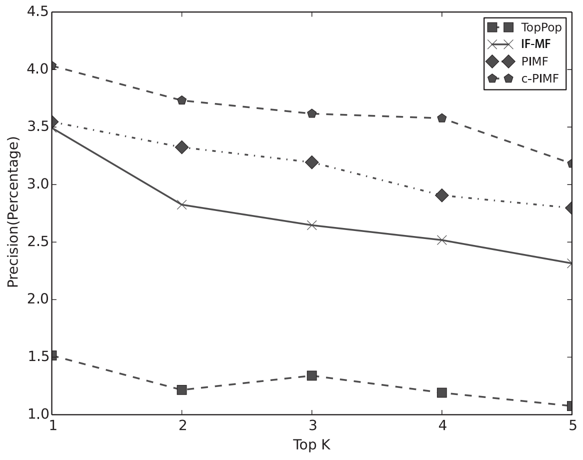
1. *MCoC* [95]. This is the state-of-the-art multi-class clustering method that finds user-item subgroups for item recommendation.

2. *PIC* (Purchase Interval Clustering). This method utilizes the purchase interval cube to cluster users. We assign the top $\tau * |I| * |c.U|/|U|$ items with highest frequency to the corresponding cluster.

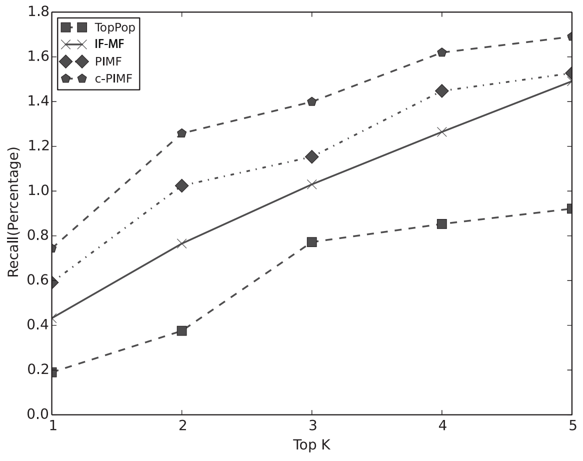
3. *cLDA*. This is a variant of our approach which only employs LDA to generate the clusters and does not incorporate purchase interval factor to refine the clusters. Thus for each cluster, the top $\tau * |I| * |c.U|/|U|$ items with highest $Pr(i|c)$ values will be assigned to corresponding cluster.



(a) Conversion Rate

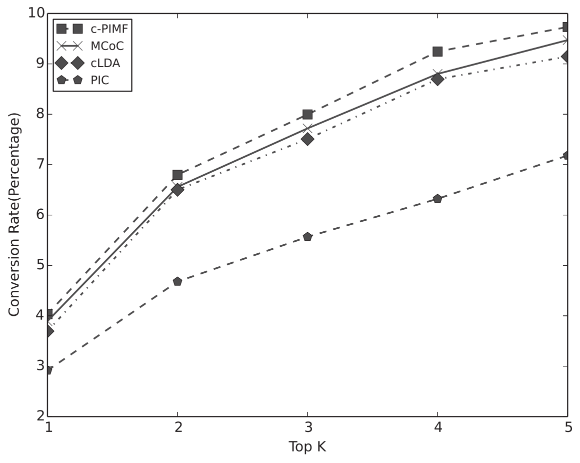


(b) Precision

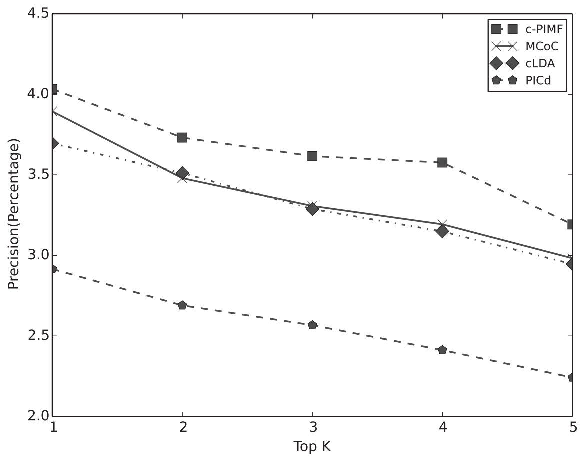


(c) Recall

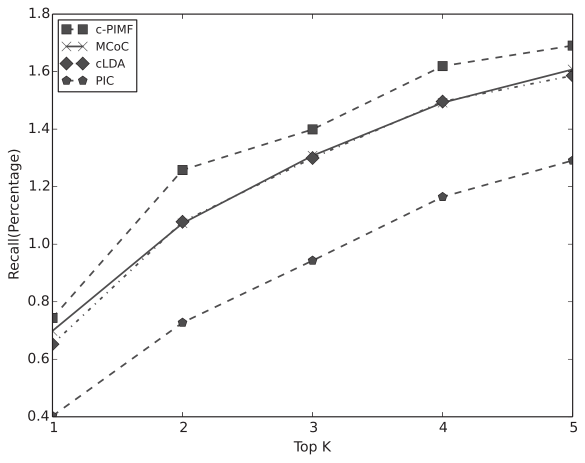
Figure 5-7: Comparative experiments



(a) Conversion Rate

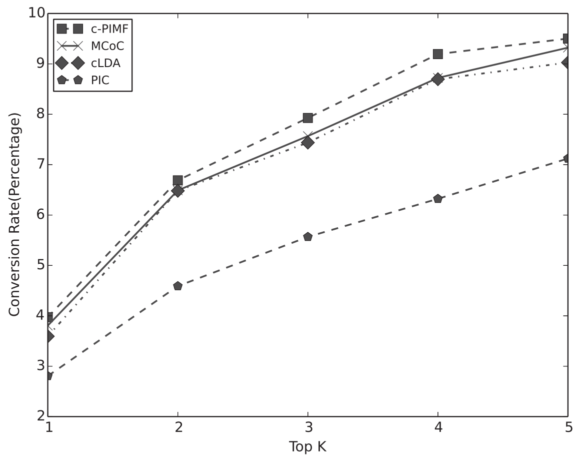


(b) Precision

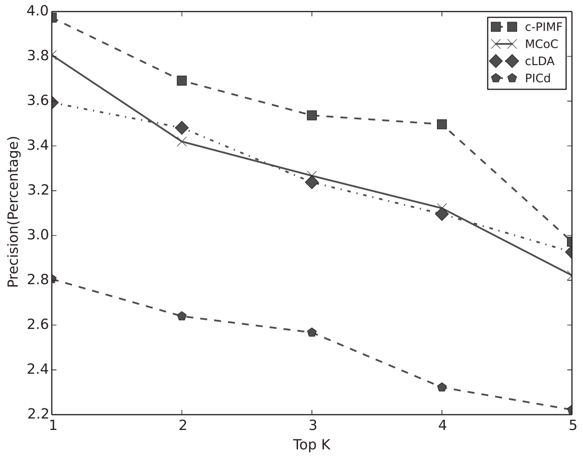


(c) Recall

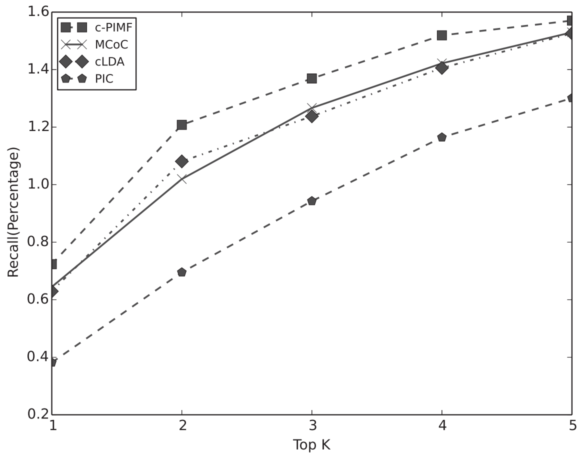
Figure 5-8: Comparison of clustering methods using PIMF



(a) Conversion Rate

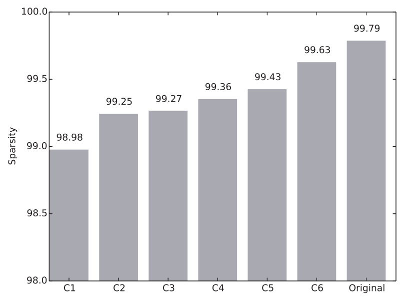


(b) Precision

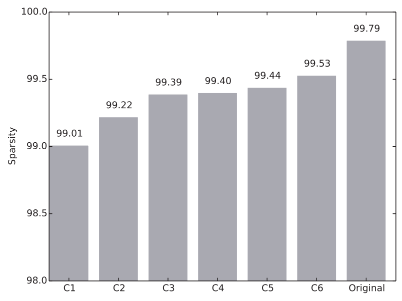


(c) Recall

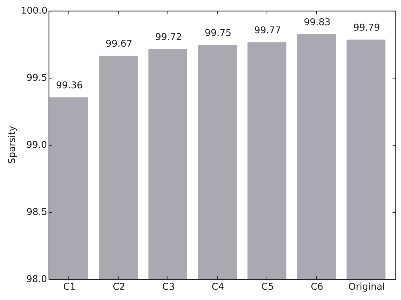
Figure 5-9: Comparison of clustering methods using MF



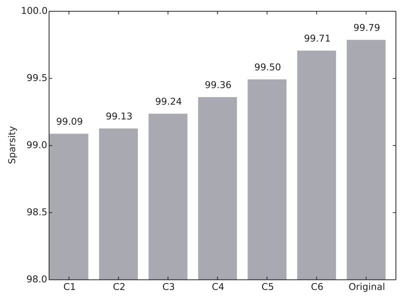
(a) c-PIMF



(b) MCoC



(c) PIC



(d) cLDA

Figure 5-10: Sparsity of original data set vs. discovered clusters for different clustering methods

We respectively apply the matrix factorization approach $IF\text{-}MF$, and the purchase interval based $PIMF$ with $LF = 16$ on the clusters obtained by the different methods. Figure 5-8 and Figure 5-9 show the experimental results for comparison for different clustering methods. We observe that our approach that captures both the latent preferences of users as well as the item time sensitivity yields the best performance. Moreover, we can see that the pure purchase interval based clustering method (PIC) does not perform well indicating that purchase interval information cannot be used as key feature to cluster user purchase behavior.

Figure 5-10 compares the sparsity of the original data sets and the clusters obtained by the various methods. The sparsity of a cluster is defined as

$$sparsity = 1 - \frac{|c.H|}{|c.U| * |c.I|} \tag{5.11}$$

While all the clustering methods reduce data sparsity, we see that the sparsity of the clusters generated by $c\text{-}PIMF$ is generally lower that obtained by $MCoC$, PIC and $cLDA$. The sparsity of the clusters obtained by PIC remains high, showing that purely clustering users based on their purchase interval information is not effective.

5.3.7 Analysis of Latent Groups

In order to further understand why $c\text{-}PIMF$ works best, we examine the latent groups discovered by our approach. Table 5.5 shows the items purchased by a subset of the users in two latent groups.

We observe that users in latent group 1 have mainly purchased mobile devices such as iPad minis and laptop models, as well as related accessories such as mouse and keyboard. On the other hand, the users in latent group 2 bought DIY PC items such as CPUs and monitors, and PC related accessories such as harddisk and cables. Note that the items monitor and router occur in both latent groups since such items are commonly used in both mobile devices and PCs. We see that our approach can effectively cluster items with their latent features.

5.4 Summary

In this chapter, we have developed a probabilistic approach to discover latent clusters from a large user-item matrix. The goal is to capture the hidden preferences and interests of users in each cluster as well as item time sensitivity. We have introduced the notion of a cluster-level purchase interval factor to indicate the likelihood that users in a cluster will purchase an item. We utilized this factor to refine the latent clusters before applying matrix factorization approach on each cluster.

We have carried out extensive experiments to evaluate the performance of our approach on a real e-commerce data set. In order to show that our approach gives good performance not because of the use of purchase intervals, we have also compared our ap-

Table 5.5: Sample Latent Groups of Users and Items Purchased

Latent Group 1	
User Id	*Sample Purchase History*
2472	Logitech M185 wireless mouse, TP-LINK 300M wireless route EDIFIER K800 Earphone, SAMSUNG 21.5’ Monitor, EPSON LQ-630K Printer 360 Geek WiFi 2,Apple iPad mini 7.9’,ThinkPad X230i 12.5 laptop
6325	360 Geek WiFi 2, Apple MacBook Pro 13.3,MacBook Pro Screen Protector SAMSUNG 21.5’ Monitor, Apple iPad mini 7.9’,EPSON LQ-630K Printer Kingston 16G USB flash disk, Hagibis MacBook HDMI Cable
10298	EDIFIER K800 Earphone,Apple iPad mini 7.9’ , SAMSUNG SSD 120G Kingston DDR3 4G, Kingshare data cable, DEEPCOOL Laptop Cooler EDIFIER Multimedia Speaker,HYUNDAI keyboard and mouse
41024	Apple MacBook Air, Apple iPad mini 7.9’, Acer D101E Projector MacBook Air Screen Protector,TRNFA 12bit Calculator,ARITA DVD R EDIFIER Multimedia Speaker, TP-LINK 300M wireless router
73092	Kingston DDR3 4G, HP 14.0’ Laptop, DEEPCOOL Laptop Cooler EPSON LQ-630K Printer, HP 802 black cartridge, EDIFIER Multimedia Speaker Logitech M185 wireless mouse, Kingston 16G USB flash disk

Latent Group 2	
User Id	*Sample Purchase History*
392	GIGABYTE Mainboard, Kingshare data cable, CoolerMaster U3 Computer Case HYUNDAI keyboard and mouse, DELL UltraSharp Monitor, Internet Cable, Antec 450W VP 450P power supply, EDIFIER Multimedia Speaker
1098	Kingston DDR3 4G, SAMSUNG 21.5’ Monitor, Intel CORE i3-3220 CPU Logitech M185 wireless mouse, GIGABYTE Mainboard, EPSON LQ-630K Printer TP-LINK 300M wireless router, Antec 450W VP 450P power supply
11524	Internet Cable, SAMSUNG SSD 120G, Apple iPad mini 7.9’ Acer G206HQL b 19.5’ Monitor, Kingston 16G USB flash disk Logitech MK260 Wireless Keyboard Suit, MAXSUN 1G 128bit graphics card
30297	Intel CORE i3-3220 CPU, ARITA DVD R, UniFly Webcam, ORICO audio card, Acer D101E Projector, Internet Cable Logitech MK260 Wireless Keyboard Suit,Seagate 500G 7200r Hard disk
71026	Intel CORE i3-3220 CPU, GIGABYTE Mainboard, ORICO audio card Internet Cable, NZXT Computer Case, Seagate 1T 7200r Hard disk Antec 450W VP 450P power supply,360 Geek WiFi 2

proach with a non-probabilistic technique that also employs the same purchase interval information. The results have demonstrated that the proposed $c\text{-}PIMF$ method significantly outperforms state-of-the-art recommender methods, and is useful in providing more accurate recommendations and clusterings for e-commerce systems. We further find that it may not possible to use only purchase interval to cluster users behavior, hence it is a good idea to use it as additional feature to generate the clusters.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Microblog social networks and e-commerce have becomes two important applications of Web 2.0 technology. Recommender systems play a key role in driving sales and social interactions in these applications. In this thesis, we have developed novel methods to personalize and improve the performance of user and product recommender systems.

In user recommender system, we have focused on improving user acceptance of ”friendship” in Twitter style micro-blog social networks. In this work, we investigated using both follower and followee relationships to discover communities to improve user recommendation in uni-directional social networks. We introduced a two-phase framework where we first utilized the LDA model to discover communities, and then applied matrix factorization on each community found. We carried out extensive experiments to evaluate the performance of our approach on two real world uni-directional social network data sets, Twitter and Weibo. The results indicated that the proposed method significantly outperformed the state-of-the-art user recommender algorithms. We further showed that the community-based approach is a good alternative form of parallelization for matrix factorization.

In product recommender systems, we have proposed a framework that utilizes purchase intervals to improve the temporal diversity of recommended items. Existing works have primarily considered the order of items purchased by users, and not the time intervals between the products purchased. We have designed a model that combines purchase interval information in users' purchase history with marginal utility and the Law of Diminishing Returns. We also devised an efficient algorithm to generate a purchase interval cube by scanning users' purchase history once.

We have further designed a LDA based approach to discover latent clusters in the large user-item matrix and incorporate temporal information into the recommendation process. We introduced the notion of a cluster purchase interval factor which estimates the probability that users in a cluster will purchase an item. Extensive experiments on a real world data set obtained from an e-commerce B2C website Jingdong in China demonstrate that the proposed methods are able to improve the precision, recall, conversion rate of the state-of-the-art product recommendation algorithms.

6.2 Future Work

There are several directions that require further investigations. We list two major directions for future work.

- **Parallelization.** Big data is now a very hot topic in both industry and academia. Scalability remains a challenge for recommender systems. One possibility is to using parallel frameworks such as MapReduce to increase the scalability of our proposed algorithms.

- **Unified subgroup framework for matrix factorization.** We have shown that it is possible to employ LDA based method utilizing some data characteristics such as purchase interval factor and follower-followee relationships, to discover meaningful clusters from e-commerce and social network data respectively. After obtaining the clusters, state-of-the-art matrix factorization approaches can be applied to each cluster. The advantages are lower sparsity and smaller data set for each cluster. Hence, this approach can both improve the effectiveness and efficiency of recommender systems. Therefore, an interesting direction is to investigate how we can develop a unified framework that can discover clusters for matrix factorization.

- **Hybrid recommendation systems.** For product recommendation, it would be interesting to study how purchase intervals compares with sequential patterns, and how to incorporate purchase interval with other temporal features such as sequential pattens. For user recommendation, although the user information is usually limited and tweets are noisy in microblog social networks, it would be still interesting to see how the proposed algorithm can be combined with user preference and content features.

BIBLIOGRAPHY

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] Jae-wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Yeon Syn. Open user profiles for adaptive news systems: help or harm? In *Proceedings of the 16th International Conference on World Wide Web*, pages 11–20, 2007.

[3] Asim Ansari, Skander Essegaier, and Rajeev Kohli. Internet recommendation systems. *Journal of Marketing research*, 37(3):363–375, 2000.

[4] Marcelo G Armentano, Daniela L Godoy, and Analía A Amandi. A topology-based approach for followees recommendation in twitter. In *Proceedings of 9th International Workshop on Intelligent Techniques for Web Personalization & Recommendation*, pages 22–30, 2011.

[5] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*. ACM press, New York, 1999.

[6] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[7] Chumki Basu, Haym Hirsh, William Cohen, et al. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 714–720, 1998.

[8] William Baumol and Alan Blinder. *Microeconomics: Principles and policy*. Cengage Learning, 2011.

[9] Nicholas J Belkin and W Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[10] Daniel Billsus and Michael J Pazzani. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning*, pages 46–54, 1998.

[11] Daniel Billsus and Michael J Pazzani. A hybrid user model for news story classification. *CISM International Centre for Mechanical Sciences*, pages 99–108, 1999.

[12] Daniel Billsus and Michael J Pazzani. User modeling for adaptive news access. *User Modeling and User-adapted Interaction*, 10(2):147–180, 2000.

[13] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[14] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14 Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufmann Publishers, 1998.

[15] Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web*, 5(1):1–33, 2011.

[16] Youngchul Cha and Junghoo Cho. Social-network analysis using topic models. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 565–574, 2012.

[17] Sonny Han Seng Chee, Jiawei Han, and Ke Wang. Rectree: An efficient collaborative filtering method. In *Data Warehousing and Knowledge Discovery*, pages 141–151. Springer, 2001.

[18] Jilin Chen, Werner Geyer, Casey Dugan, Michael Muller, and Ido Guy. Make new friends, but keep the old: recommending people on social networking sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 201–210, 2009.

[19] Pei-Yu Chen, Shin-yi Wu, and Jungsun Yoon. The impact of online recommendations and consumer feedback on sales. *Proceedings of the 25th International Conference on Information Systems*, pages 711–724, 2004.

[20] Yizong Cheng and George M Church. Biclustering of expression data. In *ISMB*, volume 8, pages 93–103, 2000.

[21] Yung-Hsin Chien and Edward I George. A bayesian model for collaborative filtering. In *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics*. Morgan Kaufman Publishers, 1999.

[22] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, 1999.

[23] Charles W Cobb and Paul H Douglas. A theory of production. *The American Economic Review*, pages 139–165, 1928.

[24] Michelle Keim Condliff, David D Lewis, David Madigan, and Christian Posse. Bayesian mixed-effects models for recommender systems. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 23–30, 1999.

[25] Joaquin Delgado and Naohiro Ishii. Memory-based weighted majority prediction. In *SIGIR Workshop on Recommender System*, 1999.

[26] Jill Freyne, Michal Jacovi, Ido Guy, and Werner Geyer. Increasing engagement through early recommender intervention. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 85–92, 2009.

[27] MH Fulekar. *Bioinformatics: Applications in life and environmental sciences*. Springer, 2009.

[28] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems*, pages 305–308, 2011.

[29] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *5th IEEE International Conference on Data Mining*, 2005.

[30] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[31] Nathaniel Good, J Ben Schafer, Joseph A Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 16th National Conference on Artificial Intelligence*, pages 439–446, 1999.

[32] Ido Guy, Inbal Ronen, and Eric Wilcox. Do you know?: recommending people to invite into your social network. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, pages 77–86, 2009.

[33] John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pages 199–206, 2010.

[34] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[35] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 194–201, 1995.

[36] Matthew D Hoffman, David M Blei, and Francis R. Bach. Online learning for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 23:856–864, 2010.

[37] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

[38] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 259–266, 2003.

[39] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[40] William H Hsu, Andrew L King, Martin SR Paradesi, Tejaswi Pydimarri, and Tim Weninger. Collaborative and structural recommendation of friends using weblog-based social network analysis. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 55–60, 2006.

[41] http://news.imeigu.com/a/1315461895947.html. Market share and sales growth situation of jingdong mall., September 2011.

[42] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 8th IEEE International Conference on Data Mining*, pages 263–272, 2008.

[43] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[44] Tamara G. Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *Proceedings of the 8th IEEE International Conference on Data Mining*. Computer Society, 2008.

[45] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[46] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[47] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Journal of Computer*, 42(8):30–37, August 2009.

[48] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, pages 591–600, 2010.

[49] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 210–217, 2010.

[50] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of the SIAM Data Mining Conference*, pages 1–5, 2005.

[51] Beibei Li, Anindya Ghose, and Panagiotis G Ipeirotis. Towards a theory model for product search. In *Proceedings of the 20th International Conference on World Wide Web*, pages 327–336, 2011.

[52] Xin Li, Lei Guo, and Yihong Eric Zhao. Tag-based social interest discovery. In *Proceedings of the 17th International Conference on World Wide Web*, pages 675–684, 2008.

[53] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. *Physics Reports*, 519(1):1–49, 2012.

[54] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 1(1):24–45, 2004.

[55] Harry Mak, Irena Koprinska, and Josiah Poon. Intimate: A web-based movie recommender using text categorization. In *Proceedings of IEEE/WIC International Conference on Web Intelligence*, pages 602–605, 2003.

[56] Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In *Proceedings of Conference on Neural Information Processing Systems*, 2003.

[57] Matthew R McLaughlin and Jonathan L Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–336, 2004.

[58] Frank McSherry and Ilya Mironov. Differentially private recommender systems: building privacy into the net. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 627–636, 2009.

[59] Prem Melville, Raymond J Mooney, and Ramadass. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *National Conference on Artificial intelligence*, pages 187–192, 2002.

[60] Koji Miyahara and Michael J Pazzani. Collaborative filtering with the simple bayesian classifier. In *PRICAI Topics in Artificial Intelligence*, pages 679–689. Springer, 2000.

[61] Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the 5th ACM Conference on Digital libraries*, pages 195–204, 2000.

[62] Atsuyoshi Nakamura and Naoki Abe. Collaborative filtering using weighted majority prediction algorithms. In *Proceedings of the 15th International Conference on Machine Learning*, pages 395–403, 1998.

[63] Mark OConnor and Jon Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, 1999.

[64] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, pages 5–8, 2007.

[65] Dmitry Pavlov and David M Pennock. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *Proceedings of Neural Information Processing Systems*, volume 2, pages 1441–1448, 2002.

[66] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[67] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5):393–408, 1999.

[68] Alexandrin Popescul, David M Pennock, and Steve Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 437–444. Morgan Kaufmann Publishers Inc., 2001.

[69] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[70] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 811–820, 2010.

[71] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[72] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[73] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning*, pages 880–887, 2008.

[74] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[75] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[76] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, 2001.

[77] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the 5th International Conference on Computer and Information Technology*, volume 1, 2002.

[78] J Ben Schafer. Dynamiclens: A dynamic user-interface for a meta-recommendation system. *Beyond Personalization*, pages 72–76, 2005.

[79] J Ben Schafer, Joseph A Konstan, and John Riedl. E-commerce recommendation applications. *Applications of Data Mining to Electronic Commerce*, pages 115–153, 2001.

[80] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260, 2002.

[81] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 210–217. ACM Press/Addison-Wesley, 1995.

[82] Luo Si and Rong Jin. Flexible mixture model for collaborative filtering. In *Proceedings of International Conference on Machine Learning*, volume 3, pages 704–711, 2003.

[83] Ian Soboroff and Charles Nicholas. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI*, pages 86–91, 1999.

[84] Xiaoyuan Su and Taghi M Khoshgoftaar. Collaborative filtering for multi-class data using belief nets algorithms. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pages 497–504, 2006.

[85] Panagiotis Symeonidis, Alexandros Nanopoulos, Apostolos Papadopoulos, and Yannis Manolopoulos. Nearest-biclusters collaborative filtering. *Information Retrieval*, 11(1):51–75, 2008.

[86] Gabor Takacs, Istvan Pilaszy, Bottyan Nemeth, and Domonkos Tikk. On the gravity recommendation system. In *Proceedings of KDD Cup and Workshop*, 2007.

[87] Thomas Tran and Robin Cohen. Hybrid recommender systems for electronic commerce. In *Proceedings Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press*, 2000.

[88] Lyle H Ungar and Dean P Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*, volume 1, 1998.

[89] Jian Wang, Badrul Sarwar, and Neel Sundaresan. Utilizing related products for post-purchase recommendation in e-commerce. In *Proceedings of the 5th ACM Conference on Recommender Systems*, pages 329–332, 2011.

[90] Jian Wang and Yi Zhang. Utilizing marginal net utility for recommendation in e-commerce. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1003–1012, 2011.

[91] Jian Wang and Yi Zhang. Opportunity model for e-commerce recommendation: right product; right time. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 303–312, 2013.

[92] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 501–508, 2006.

[93] Pu Wang and HongWu Ye. A personalized recommendation algorithm combining slope one scheme and user based collaborative filtering. In *International Conference onIndustrial and Information Systems*, pages 152–154, 2009.

[94] 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. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 723–732, 2010.

[95] Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st International Conference on World Wide Web*, pages 21–30, 2012.

[96] Wei Zeng, Ming-Sheng Shang, Qian-Ming Zhang, Linyuan Lü, and Tao Zhou. Can dissimilar users contribute to accuracy and diversity of personalized recommendation? *International Journal of Modern Physics C*, 21(10):1217–1227, 2010.

[97] Yi Zhang and Jonathan Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 47–54, 2007.

[98] Gang Zhao, Mong Li Lee, and Wynne Hsu. Utilizing purchase intervals in latent clusters for product recommendation. In *Proceedings of Workshop on Social Network Mining and Analysis Social Network Study for Business, Consumer and Social Insights*, pages 28–36, 2014.

[99] Gang Zhao, Mong Li Lee, Wynne Hsu, and Wei Chen. Increasing temporal diversity with purchase intervals. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 165–174, 2012.

[100] Gang Zhao, Mong Li Lee, Wynne Hsu, Wei Chen, and Haoji Hu. Community-based user recommendation in uni-directional social networks. In *Proceedings of the 22nd ACM International Conference on information & Knowledge Management*, pages 189–198, 2013.