4-21-2014

Next Generation of Recommender Systems: Algorithms and Applications

Lei Li
lli003@cs.fiu.edu

DOI: 10.25148/etd.FI14071110

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

NEXT GENERATION OF RECOMMENDER SYSTEMS: ALGORITHMS AND APPLICATIONS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Lei Li

2014

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Lei Li, and entitled Next Generation of Recommender Systems: Algorithms and Applications, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Shu-Ching Chen

Jainendra K Navlakha

Bogdan Carbunar

Debra VanderMeer

Tao Li, Major Professor

Date of Defense: April 21, 2014

The dissertation of Lei Li is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2014

ACKNOWLEDGMENTS

It is more difficult to comprehensively identify people who assisted me in getting to this stage of my Ph.D. study than actually writing this dissertation. Hence, what follows is certainly incomplete in terms of acknowledging the help of everyone.

During my Ph.D. study, my most obvious benefactor is my advisor, Dr. Tao Li, an excellent researcher in the community of Data Mining. I am very fortunate to work under the supervision of Dr. Tao Li. Without his inimitable guidance and support through these years, I would be poor for not building my extensive research work and sharpening my critical thinking and writing skills. Additionally, I am very grateful to my committee members for the valuable instruction in helping me complete this dissertation, including Dr. Shu-Ching Chen, Dr. Jainendra K Navlakha, Dr. Bogdan Carbunar and Dr. Debra VanderMeer. Further, I would like to thank University Graduate School for awarding me the Presidential Fellowship in 2009-2011.

I would like to thank all my colleagues in Knowledge Discovery and Research Group (KDRG) for making me enhance my knowledge and pushing me to interesting research directions. Special thanks must go to: Dr. Dingding Wang for her masterful knowledge and commendable guidance on helping me achieve my first research experience; Jingxuan Li and Li Zheng for their great willingness to collaborate with me on various research topics; Chao Shen for his valuable discussion on multiple research directions; Yexi Jiang and Chunqiu Zeng for their reliable services in providing helpful suggestions. I would also like to extend my gratitude to all my internship mentors and coauthors of my research work, including Dr. Wei Peng, Dr. Tong Sun, Dr. Yingnan Zhu, Dr. Chen Lin and Dr. Wenxing Hong.

ABSTRACT OF THE DISSERTATION

NEXT GENERATION OF RECOMMENDER SYSTEMS: ALGORITHMS AND APPLICATIONS

by

Lei Li

Florida International University, 2014

Miami, Florida

Professor Tao Li, Major Professor

Personalized recommender systems aim to assist users in retrieving and accessing interesting items by automatically acquiring user preferences from the historical data and matching items with the preferences. In the last decade, recommendation services have gained great attention due to the problem of information overload. However, despite recent advances of personalization techniques, several critical issues in modern recommender systems have not been well studied. These issues include: (1) understanding the accessing patterns of users (i.e., how to effectively model users' accessing behaviors); (2) understanding the relations between users and other objects (i.e., how to comprehensively assess the complex correlations between users and entities in recommender systems); and (3) understanding the interest change of users (i.e., how to adaptively capture users' preference drift over time). To meet the needs of users in modern recommender systems, it is imperative to provide solutions to address the aforementioned issues and apply the solutions to real-world applications.

The major goal of this dissertation is to provide integrated recommendation approaches to tackle the challenges of the current generation of recommender systems. In particular, three user-oriented aspects of recommendation techniques were studied, including understanding accessing patterns, understanding complex relations and understanding temporal dynamics. To this end, we made three research contributions.

First, we presented various personalized user profiling algorithms to capture click behaviors of users from both coarse- and fine-grained granularities; second, we proposed graph-based recommendation models to describe the complex correlations in a recommender system; third, we studied temporal recommendation approaches in order to capture the preference changes of users, by considering both long-term and short-term user profiles. In addition, a versatile recommendation framework was proposed, in which the proposed recommendation techniques were seamlessly integrated. Different evaluation criteria were implemented in this framework for evaluating recommendation techniques in real-world recommendation applications.

In summary, the frequent changes of user interests and item repository lead to a series of user-centric challenges that are not well addressed in the current generation of recommender systems. My work proposed reasonable solutions to these challenges and provided insights on how to address these challenges using a simple yet effective recommendation framework.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the extensive development of web technologies, an increasing number of online services are becoming popular, aiming to provide different types of information based on online users' information need. Representative examples of such services include Netflix and Youtube for watching movies and videos, Google News and Yahoo! News for reading news articles, Amazon and Ebay for online shopping, Last.fm and Spotify for listening music, etc. These services provide a huge amount of interesting information and generates a myriad of user behavior data as online users access the available resources. Under the constraint of low latency for users, it is extremely difficult for these services to promptly present the right information as the volume of data goes exponentially. Then, the question of intelligent information management turns on its head, and has led to an unprecedented acceleration of automatic techniques, e.g., web search, to assist users in finding interesting data.

In general, search-based online services harness millions or even billions of data (or say, items). Given a user's text query on some information, the search can be achieved by indexing items according to their content or the linkage with each other, which often refers to as *information retrieval* [MRS08]. A lot of elaborate search algorithms have been proposed to handle the large-scale retrieval problems, e.g., PageRank [PBMW99]. Search-based information retrieval requires users' efforts of inputting the search query; however in most scenarios, it is difficult for users to come up with queries as they may not know what information they are interested in. For example, what web pages offer users the information that they want? Are there some recent songs that are similar to a user's favorite playlist played in the morning? How to assist users in finding some high-quality research papers to enrich their understanding on the state-of-the-art of a particular field? The vast amount of information and

the diversity of resources render data filtering, exploration and representation more difficult. To provide better user experience in using online services, more intelligent algorithms are expected to avoid the intervention or efforts of users. This is where personalized recommender systems have seen their largest use cases on the web.

Personalized recommender systems aim to assist users in retrieving and accessing interesting items by automatically acquiring user preferences from the historical data and matching items with the preferences. In the last decade, personalized recommendation services have gained great attention and are becoming increasingly prevalent, as the problem of information overload is intensified by the great advance of Internet technologies. Boosted by the Netflix Prize competition of 2006[1], there has been much work done in both academia and the industry on designing and developing new solutions to recommender systems in recent years. Examples of such applications include recommending news at Google News, products at Amazon.com, movies by Netflix, jobs and talents by LinkedIn, users and communities on social media, etc. Benefited from personalization techniques, these successful services provide great convenience for users enjoying recommendations, and meanwhile, bring the enterprises notable economic benefits.

In the following, we initially discuss functions of recommender systems from the perspectives of both service providers and users, and then briefly introduce existing solutions to the problem of personalized recommendation (including content filtering, collaborative filtering and hybrid approaches). Next, we present the motivations of my research work from three distinct yet interleaved user-oriented aspects, and summarize the major contributions of my dissertation.

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

1.1 Recap: Functions of A Recommender System

Recommender systems (RSs) are software applications and techniques providing information suggestions to a myriad of online users [RV97, RRS11]. The suggestions depend on specific decision-making purposes, e.g., what items to buy, what music to listen to, etc. It is necessary to distinguish the functions that a RS can provide with respect to the roles on behalf of the service provider and online users of the RS. For instance, an online shopping system often offers promotions on particular commodities in order to gain more economic interest and increase the number of visitors, whereas a user's motivation for accessing the shopping website is to find a suitable product for daily use rather than promotion items.

From the perspective of a service provider, there are various reasons to exploit the recommendation technology, listed as follows:

- Increase the conversion rate. The conversion rate is defined as the number of users that accept the recommendation and consume an item, compared to the number of visitors that just browse through the information [RRS11]. As a commercial RS, the key function is to sell items as many as possible to maximize the economic benefits. Non-commercial applications have similar goals, as they are expecting to maximize the daily visits of the websites.
- Show more diverse items. A RS enables the user to select unpopular items from the item repository by providing a precise recommendation. Such a strategy allows the system to collect interactions of the items in the long tail [BHS06]. For example, a movie recommender aims to recommend all the movies in the catalogue, but not just the most popular ones.
- Improve user satisfaction. A well-designed RS can improve the usage experience of users. The user will find the recommendation relevant to his/her preference

and enjoy using the system. The combination of back-end recommendation strategies and front-end usable interfaces will definitely increase the user's satisfaction, and in turn will help improve the system usage.

- Increase user fidelity. A loyal user to a RS enables the system to gain more information about the user, as it keeps collecting the user's interactions, e.g., ratings or clicks of items. Consequently, the user model can be gradually improved in a long run based on user feedbacks, and provide more customized recommendation to match the user's preference.

Comparatively, when a user is using a RS, he/she may have different expectations, depending on the user's individual preference, such as:

- Find some good items. The standard format of recommendation is to provide a list of items to users, along with predictions of how much the user would like them, e.g., 1 to 5 star ratings. The item list is often associated with a particular ranking of items in accordance with the user's preference on different types of items. This is the primary function of most recommender systems.
- Recommend a sequence. Sometimes a user may expect a sequence of items from a RS, as he/she may click them one by one. Typical applications involve recommending a TV series, i.e., a series of TV programs; or a music track that contains a sequence of songs. The user's satisfaction can be extended by accepting a sequence of items, rather than a single recommendation.
- Improve user profile. The preference of an individual user is likely to evolve over time as he/she interacts with the system. Such an evolution might be triggered by different types of events, e.g., promotional activities of online shopping websites, or major sporting events reported by news media, etc. A RS should have the capability of capturing the evolution of user interest.

The aforementioned functions of a RS are quite diverse, and also the role of a RS differs in the perspectives of service providers and users. As a matter of fact, a RS must balance the needs of both service providers and users and provide a service that is beneficial to both. To this end, an extensive exploration of a range of different techniques has been conducted in the last decade. In the following section, these techniques will be introduced for better understanding the problem of personalized recommendation.

1.2 Existing Solutions to Personalization

A typical personalization algorithm often involves modeling user preferences by analyzing the historical consumption data and then retrieving interesting items based on user profiles. However, for many reasons, user preferences towards items are difficult to guess, and therefore there is a considerable variance of personalization techniques in assessing users' personal tastes. Typical approaches to personalized recommendation services can be divided into two categories: content filtering and collaborative filtering. Meanwhile, there are some hybrid solutions that integrate the advantages of these two filtering techniques in order to obtain more reasonable recommendation results.

1.2.1 Content Filtering

The principle of a content-based RS is to suggest the items that are similar to the items that the user liked in the past in terms of the item content. In general, a RS acquires users' consumption history and constructs appropriate user models based on the content of items, to indicate which (type of) items the user likes. The content, such as descriptions, keywords, categories, etc., provides useful information about the

item, and evidences of users' preferences. The major operation of a content-based RS is to match the user's model (profile) with the content of unknown items with respect to a target user [DGM08, LdGS11, PB07].

In the past decades, various techniques have been proposed to tackle the problem of content-based recommendation. These techniques include probabilistic models [LDP10, PB97], nearest neighbor based methods [BPC00], classification-based models [Joa98, WQF07], etc. One advantage of content-based recommendation techniques is that the modeling process is purely based on the content of items, and hence does not require a large number of users to be involved. This is because the matching between a user's profile and an item is often achieved by comparing the similarity/relevance of these two, without the intervention of other users. Further, new items can be recommended to users as long as they have enough content. Hence, content-based RSs do not suffer from the *item cold-start* problem [SPUP02].

Recommender systems based on content filtering are easy to implement; however, in some scenarios, simply representing the user's profile information by item content is insufficient to capture the exact preference/interest of the user. Firstly, the item content, i.e., the metadata, might not be specified completely, especially in the systems where the recommendation functionality is not the major service. Such an incompleteness renders user profiling inaccurate. In addition, the representation of user profiles, e.g., vector space model [MRS08], cannot effectively capture the correlations among items that a user has accessed before. Further, a content-based RS cannot easily adapt to the changes of user interest, as the strategy is to suggest items extremely similar to the ones that a user consumed. Finally, content filtering suffers from the *new user* problem, i.e., if a user has only a few consumption activities, the recommendation model cannot accurately express his/her preference until enough consumptions are collected.

1.2.2 Collaborative Filtering

Collaborative filtering (CF) analyzes accessing behaviors of users similar to the target user and then recommends items based on collaborative activities [DDGR07, HKTR04, Hof04, KBV09, Kor10, RIS+94, SKKR01]. The basic idea of a CF system is that if users have similar accessing behaviors in the past, they will also prefer similar items in the future. Based on this assumption, given a target user A, the recommendation for A can be obtained by selecting items that have been consumed by the neighbors of A but not accessed by A. Collaborative filtering can be further categorized to memory-based filtering and model-based filtering.

Memory-based algorithms provide predictions for users based on their past ratings. In general, the prediction is computed as a weighted average of the ratings given by other neighbors where the weight is proportional to the similarity between the target user and the neighbor [DDGR07]. Typical similarity metrics involve the cosine similarity [BHK98] and the Pearson correlation coefficient [RIS+94]. Memory-based methods are easy to understand and require little efforts for training; however, it is quite challenging to make this type of algorithms more scalable as the process needs to calculate all pairwise similarities of users or items.

Comparatively, model-based algorithms try to create user models based on their past ratings, and predict the ratings on unknown items using these models. In practice, model-based algorithms capture various user interests by utilizing latent factors or explicitly classifying users to multiple clusters, such as latent semantic indexing [SKKR00], probabilistic latent semantic indexing [Hof04], multiplicative factor model [MZ04], latent Dirichlet allocation [WB11], etc.

Collaborative filtering systems can efficiently capture users' behaviors in case where overlap in historical consumption across users is relatively high and the content universe is almost static [SKR99]; however, in many web-based applications, the

content universe undergoes frequent changes, with content popularity changing over time as well [LCLS10]. Moreover, many online users do not have enough historical consumption records, which is known as the so-called *cold-start* problem [SPUP02]. These issues render collaborative filtering ineffective.

1.2.3 Hybrid Solutions

As mentioned previously, both content filtering and collaborative filtering have their advantages as well as drawbacks and problems. A common practice for obtaining more robust solutions is to combine the two types of methods to eliminate the drawbacks, which often refers to as *hybrid* approach. For instance, a latent factor model (a CF model) can be enriched by absorbing more content features of both users and items, and feature-based matrix factorization [CZL+11] can be applied to obtain the latent factors. Such an integration enables the CF-based RSs to tackle the problem of *item cold-start*, as it takes into account the rich content of items. Many hybrid methods that integrate the above two categories of algorithms have been developed [BS97, DLS07, MMN02, PPL01], in which the defects of one type of methods can be commonly alleviated by the other type of methods [Bur02].

1.3 Motivation: User-Oriented Aspects

In essence, a recommender system is a user-oriented service that aims to attract more users by providing high-quality recommendation results. Thus, the design of recommendation strategies should follow a user-centric paradigm, that is, to consider various potential user requirements when constructing user profiles. The major component of traditional recommendation approaches is to compare user profiles with available items using specific similarity measures. Nonetheless, as the scale of item reposi-

tory increases and the interest of users changes promptly, it is insufficient to provide reasonable recommendation by purely calculating the similarity. To accommodate the algorithm itself to user requirements, several other criteria have been studied, such as item diversity [ZH08], item coverage [MSDR04], serendipity [MMO08], novelty [WXLN07], etc., and a lot of recommendation approaches have been proposed by utilizing these criteria. However, despite extensive recent advances of personalization techniques, several critical issues in modern recommender systems have not been well explored in previous studies, including multi-granularity user behaviors, complex user relations and temporal dynamics of user preferences. In the following, detailed discussions of these issues are provided.

Multi-granularity user behaviors: User behaviors (e.g., click behavior) are generally considered as a set of events or modeled as a binary variable [HKV08, LPP07, SPUP02], which refers to as the analysis of *coarse-grained behavior*. The profiles of users and the recommendation are achieved by analyzing the set of activities of users. However, the detailed behavioral activities (e.g., the click sequence of items) are often being ignored, which refers to as *fine-grained behavior*. Taking a news recommender as an example, a user may always click the first news article recommended to him/her for detailed reading, and then go through the titles of news in the remaining list. Such an accessing pattern, if well captured, would be very helpful for the system to construct high-quality user profiles, and consequently provide meaningful recommendation results.

Complex user relations: A user's decision on choosing an item may be influenced by the underlying correlations between users and other objects within the recommender system. The objects, in general in recommender systems, involve users, items, and even aspects within items or item hierarchies [BTC+10, SZF07]. As in the previous example, a news reader may read news articles describing a specific topic

(e.g., sports or politics) or containing a specific named entity (e.g., a celebrity or an organization) [GDH04, LCGF10]. Objects in recommender systems are correlated with each other, and in turn affect the user's preference. It would be beneficial to the recommendation if these relations are comprehensively understood.

Temporal dynamics of user preferences: Personal user preference might evolve over time [CCYX09, KS00, MLDO07], i.e., the user's interest may be dominated by the recent events (e.g., promotion of sales) or life experiences (e.g., giving birth to a baby). Again in a news recommender, a user interested in sports-related news topics may read articles about "Cycling World Championships", and after several days, he/she may prefer to read news of "Baseball World Cup". Such an interest drift might happen very frequently, or is incidental in terms of the user's consumption history, depending on the characteristics of different users. Previous research often formalizes interest drift as a temporal factor; however, such a dynamics scheme may only reflect the current status of user preference, and hence lose an overview of a user's interest.

To summarize, the issues of the current generation of recommender systems include: (1) understanding the accessing patterns of users (i.e., how to effectively and efficiently model users' clicking behaviors into user profiles); (2) understanding the relations between users and other objects (i.e., how to comprehensively assess the complex correlations between users and entities involved in recommender systems); and (3) understanding the interest change of users (i.e., how to adaptively capture users' preference drift over time). Previous research and industrial efforts towards these aspects are not satisfactory. Therefore, the current generation of recommender systems still requires further improvements to make recommendation approaches more effective [AT05] in capturing users' personal preferences and suggesting interesting information to them.

1.4 Contribution of this Dissertation

My dissertation follows the stream of research that covers the aforementioned issues of existing recommender systems. In this work, three interleaved aspects of recommendation techniques were carefully explored, including understanding accessing patterns of users, understanding complex relations within recommender systems and understanding temporal dynamics of user interests. In particular, the contributions of this dissertation are as follows.

- Modeling: In this research, various approaches related to the aforementioned three aspects were explored. In modern recommender systems, these aspects being studied play vital roles in capturing the exact preference of users and performing reasonable and meaningful recommendation. Specifically, I have designed and developed:

  - Personalized user profiling algorithms to capture accessing patterns of users, e.g., the clicking sequence that a user may behave in his/her consumption history. Such patterns are valuable for building high-quality user profiles; however, existing approaches treat a user's history as a set of click events, and ignore the relative sequence of clicks, and hence lose the detailed information. Along this direction, several work has been published [LWL+11, LPK+13] to capture both coarse-grained and fine-grained user behaviors.

  - Graph-based recommendation models to catch the complex relations within a recommender system. It is intuitive to model the complex relations linked to users using graphs, by which the weighted linkage among different elements can be naturally represented. However, the high-order relations in a recommender system have not been well explored in existing literatures.

Along this direction, a series of ideas have been implemented and published [LL12, LL13, ZLHL13] to utilize graph-based methods in handling the relationships within a recommender system.

  - Temporal approaches to integrate both long-term and short-term user profiles for recommendation. User preferences often evolve over time, and therefore it is imperative to capture such changes when performing user profiling and recommendation. Nevertheless, the interest drift is generally modeled as a temporal variable, which may cause the loss of an overview on user preferences. In this research direction, there are a list of publications during my Ph.D. study [HLL12, LZL11, LHL12] to model the temporal change of user preferences.

- Framework: A versatile recommendation framework that seamlessly integrates the proposed recommendation techniques was designed. The framework includes:

  - User profiling that effectively models both coarse- and fine-grained user behaviors, complex element relations linked to users and temporal dynamics of user interests;

  - Recommendation that fully exploits user profiles and provides elegant selecting and ranking solutions to user personalization;

  - Evaluation by taking into account various recommendation evaluation criteria, including prediction accuracy, ranking measures, item space coverage, item diversity, user reciprocity, algorithm consistency, etc.

- Applications: The proposed framework was applied to different real-world recommendation applications, including personalized news recommender systems and personalized online recruiting systems. Specifically, for news recommenda-

tion, the framework operates on the input of abundant news content, i.e., substantial content features are utilized to construct user profiles; comparatively in online recruiting systems, the framework exploits multiple information sources that characterize user preferences, including users' demographics, crawled information and accessing histories. Along this direction, one paper has been published in SIGKDD Industrial Track [HLLP13], which utilizes the proposed framework to build a real-world application.

1.5 Organization of this Dissertation

To facilitate the understanding and reading of this dissertation, an overview of the material presented in this dissertation is given as follows.

We start by introducing the preliminaries of the topic of personalized recommendation in Chapter 2, including the general notations, formal definitions of basic recommendation techniques and various evaluation criteria used in this dissertation. We then highlight the state-of-the-art that are relevant to the aforementioned three research issues.

Next, in Chapter 3, we study the problem of understanding user behaviors within recommender systems. In this direction, two approaches are proposed to harness user behaviors in the perspective of coarse- and fine-grained granularities. Specifically, we consider the collective user behaviors towards a user's personal interest and a community's topic in the environment of social media, and propose a generative recommendation model to capture such behaviors in a coarse-grained fashion. In addition, we analyze the decreasing reading interest of users in a news recommender, and propose a fine-grained recommendation model that integrates various characteristics when recommending items to users.

We then proceed to consider the problem of modeling the complex relations linked to users in a recommender system. Chapter 4 begins this focus with general discussion of how graphs can be used to formalize such a problem. Then we propose two graph-based approaches to the problem of personalized recommendation. Particularly, a bipartite graph based method is presented to capture the complex reciprocity of two different sets of users in the applications of user recommendation, and a hypergraph based method is proposed to model the complex relations among different types of elements in a news recommender system.

The last technical contribution is Chapter 5, which studies the problem of temporal dynamics of user interest, i.e., a user's preference may change over time. Along this stream of research, we propose an integrated approach that considers both long-term and short-term user profiles simultaneously when modeling user interest changes. We also explore the possibility of utilizing domain taxonomies to improve the accuracy of capturing the evolution of user preferences.

In Chapter 6 we propose a comprehensive recommendation framework that integrates the profiling and recommendation techniques proposed in the previous three chapters, and then apply this framework to a representative recommendation application – online recruiting.

Finally in Chapter 7 we conclude the dissertation by discussing possible extensions of the proposed approaches in addressing the three research challenges.

CHAPTER 2

PRELIMINARIES AND RELATED WORK

This chapter is organized as follows. We first introduce the notations used in this dissertation, and then formally discuss the basic filtering techniques: content filtering and collaborative filtering in Section 2.1. Afterwards, we describe the evaluation metrics for recommender systems based on different quality criteria in Section 2.2. Finally, we highlight existing literatures that are related to the topics studied in this dissertation in Section 2.3.

2.1 Notations and Basic Techniques

We formally define the recommendation problem by following the definition of [AT05]. In a recommender system, let $U = \{u_1, \ldots, u_n\}$ be the set of users, and $I = \{i_1, \ldots, i_m\}$ be the set of items in the item repository. Assume there is a utility function $\hat{r} : U \times I \rightarrow S$ measuring the utility $\hat{r}_{u,i}$ of item i to user u, hence this function returns a totally ordered ranked list S consisting of nonnegative integers of utility scores. In most recommender systems, e.g., movie recommenders or music recommenders, the utility of an item i to a user u is typically represented by a *rating* value, indicating how much u likes i, e.g., 1 to 5 stars. Then, the recommendation problem can be formulated as selecting an unknown item $i'_u \in I$ for each user $u \in U$ that maximizes the utility function $\hat{r}$, i.e.,

$$\forall u \in U, \quad i'_u = \arg\max_{i \in I} \hat{r}_{u,i}. \tag{2.1}$$

For the scenario of recommending a list of items to a user, we can iteratively select the item that maximizes the utility function in Eq.(2.1) until the number of selected items reaches to the budget. The utility function can serve to *predicting* a user's rating towards a particular item, and hence $\hat{r}$ can be regarded as the prediction

function, i.e., $\hat{r}_{u,i}$ denotes the predicted rating value of user u to item i. In terms of prediction, we call user u as the *target user* and item i as the *target item*. In some scenarios, the prediction function produces binary rating values, e.g., 0 (like) and 1 (dislike). The typical goal of recommender systems is to build a predictive model that can estimate the prediction function $\hat{r}$.

2.1.1 Content Filtering: Revisit

Given a target user u, content filtering approaches first build the profile p_u of u using the historical content information accessed by u, and then calculate the similarity between p_u with the content of items. In this sense, content filtering is similar to information retrieval, as the task involves identifying items that match the user's preference and filtering out unrelated items [HSS01].

The first question of content filtering is how to represent the content of both user profiles and items. This, in general, depends on the specific domain of recommendation and the type of content information available. For example, in a movie recommender, the content of a movie often consists of actors, directors, genre, etc., which can be regarded as structured content; comparatively, in a news recommender, the content of a news article is unstructured text, and the representation cannot be simply derived. In the following, we assume that the item content is unstructured text including a set of words/terms.

A simplest format of representing a text document is to use a binary vector, each entry of which indicates whether the corresponding word or phrase appears in the text document or not. However, this approach ignores the number of times that a word appears in the document, and hence cannot capture the importance of the word. To this end, term frequency based methods have gained popularity in information retrieval community [MRS08], e.g., the TF-IDF (term frequency-inverse document

frequency) encoding format. Given a word w from a document d, the TF-IDF value of w in d is calculated as

$$TFIDF(w) = TF(w) \times IDF(w) = \frac{d(w)}{|d|} \times \log \frac{N}{n(w)}, \tag{2.2}$$

where $d(w)$ is the number of occurrences of w in d, and $|d|$ is the total number of words in d. N is the total number of documents, and $n(w)$ is the number of documents that contain the word w. The first component $TF(w)$ defines the importance of w with respect to d, and the second component $IDF(w)$ denotes the discriminating evidence of w with respect to all the documents.

Once the representation of items is fixed, we can utilize nearest neighbor based methods to identify content recommendations. Given a predictive function $\hat{r}$, a target user u and a target item i, the prediction of $\hat{r}_{u,i}$ can be achieved by first identifying the k most similar items to i that u have assigned a rating, and then aggregating the ratings of these items as the prediction value. In general, the similarity between the TF-IDF vectors can be calculated using the cosine similarity metric [MRS08].

2.1.2 Collaborative Filtering: Revisit

In the scheme of collaborative filtering, the prediction value $\hat{r}_{u,i}$ is calculated by aggregating the preferences/ratings of a large set of users, or say, the wisdom of the crowd. Let us take a user-based collaborative filtering as an example. Given a target user u and a target item i, to obtain the prediction value $\hat{r}_{u,i}$, the collaborative filtering engine works as follows:

1. Compute user relevance: The user relevance denotes the taste similarity of two users in terms of the accessed items. By adopting standard relevance metrics, the relevances between the target user u and all other users can be calculated, and can then serve as the basis for selecting neighbors of u. Examples of rele-

vance metrics include Pearson correlation coefficient and adjusted cosine similarity [MRS08]. Pearson correlation coefficient (`Pearson`) is defined as the linear dependence between two users' accessing histories, a and b, i.e.,

$$sim(a,b) = \frac{\sum_{i \in I}(r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\sum_{i \in I}(r_{a,i} - \bar{r}_a)^2}\sqrt{\sum_{i \in I}(r_{b,i} - \bar{r}_b)^2}}, \tag{2.3}$$

where $r_{a,i}$ is the rating of item i given by user a, and $\bar{r}_a$ is the average rating over all the items rated by user a. `Pearson` ranges in $[-1, 1]$, where negative values indicate low similarities and positive values represent high similarities. Adjusted cosine similarity (`AdjCosine`) factors out the average rating behavior when computing the similarity between two items i and j, i.e.,

$$sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_u)^2}}. \tag{2.4}$$

`AdjCosine` is similar to `Pearson` in terms of the returned values.

2. Identify user neighbors: The next step is to select k neighbors of u based on the relevance score, and these neighbors have provided a rating for the target item i. In practice, it is crucial to decide an appropriate k for a nearest neighbor collaborative filtering algorithm [HKBR99]. If k is chosen too large, then it is possible to involve a lot of noise data due to the uncertainty of neighborhood users' interests; otherwise, if k is chosen too small, then the predicted rating value might be biased on small number of neighbors.

3. Aggregate neighbor ratings: Once the k neighbors of u are identified, the predicted rating value can be obtained by aggregating the nearest neighbors' ratings for the target item i. The aggregation often follows a weighted combination of the ratings scores by the neighbors [RIS+94], i.e.,

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{u' \in U} sim(u,u') * (r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in U} sim(u,u')}, \tag{2.5}$$

where $sim(u, u')$ aims to measure the similarity between the target user and one of the neighbors $u' \in U$, and can be regarded as a weight of deciding the final prediction value.

The above procedures are used to predict the rating of an item, and they can be repeated for all items that have not been rated by the target user. A top-n recommendation list can be obtained by selecting the top n ranked items in terms of the predicted rating score. This is the typical process of a memory-based collaborative filtering method.

2.2 Evaluation Metrics

It is a challenging task to evaluate the performance of a recommender system, and it often consists of three different levels of experiments, i.e., *offline experiments*, *user studies* and *online evaluation*. Offline experiments are purely based on the static data without requiring interaction with real users, and hence are easy to conduct. User studies are often held in a controlled environment by asking a small group of subjects to use the system and report their experience. Online evaluation is often performed among a large pool of real users who are unaware of the experiment, which is the closest to reality [SG11]. In this dissertation, most of the work employs offline experiments as the evaluation scheme, and hence we mainly introduce the common metrics used in offline experiments.

An offline experiment is often conducted on a pre-collected data set that includes user behaviors on a set of items. The data set serves as the basis of simulating the behaviors of users that interact with a real system. Thus, a common assumption here is that the user behavior observed in the data set should be similar to the one of the deployed recommender system. The goal of offline experiments is to filter out inap-

propriate methods. To this end, the parameter tuning process is generally involved in the experiments. To evaluate the performance of a recommendation algorithm, a range of properties need to be considered. In the following, the set of metrics used in this dissertation are introduced from 4 different perspectives.

2.2.1 Measuring Usage Prediction

Some recommendation applications do not require users to provide ratings on items. For example, in a news recommender, the feedback of users is simply a click/non-click of the recommended news article. Hence, the major function of such systems is to predict whether the user will click an item or not.

In an offline environment, the data used in the experiments is often split into two different sets: *training data* and *testing data*. The training data is utilized to learn the recommendation model or optimize the model parameters, whereas the testing data is adopted to evaluate the learned recommendation model. A recommendation algorithm will generate a prediction for a target user. Compared with the ground truth within the testing data, we may have four possible outcomes, as shown in Table 2.1.

Table 2.1: Possible results of a recommendation of an item to a user.

	Recommended	Not recommended
Clicked	True-Positive (TP)	False-Negative (FN)
Not clicked	False-Positive (FP)	True-Negative (TN)

A reasonable recommendation algorithm should have more true positives, and less false positives and false negatives. We can count the number of instances in the experiments that fall into each category in the table and compute the following quantities:

$$\textbf{Precision} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fp}}, \quad \textbf{Recall} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fn}}. \tag{2.6}$$

These two metrics have a trade-off with each other: while increasing the number of recommendation list typically improves *recall*, it is also likely to reduce *precision*. Hence in the offline experiments, we often consider both metrics simultaneously by calculating the metric of *F1-score*, i.e.,

$$\textbf{F1-Score} = \frac{2 * \textbf{Precision} * \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}. \tag{2.7}$$

In addition, in some recommendation applications, the number of recommendations that can be presented to the user is predefined. For instance, in a newsfeed recommender system, the recommended feed has a budget of 5 or 10 recent news articles. Another example is the online advertising system, e.g., ads presented by Google search engine, where the number of the available ad positions is restricted. In these systems, a useful measure of evaluating the accuracy of recommendation is **Precision at N**, meaning how accurate a recommendation list of size N is.

2.2.2 Measuring Ranking Prediction

In most scenarios, the recommendation result will be represented as a list of items, imposing a certain browsing order. For example, in a news recommender system, the application will show a list of popular news articles to the user once the user logins. The system puts more focus on predicting the ranking of items, rather than the rating scores.

A typical scheme of evaluating the ranking quality is to assume that the utility of a list of recommendations is additive [SG11], calculated as the sum of utilities of all the recommended items in the list. The utility of each recommendation is the utility of the item discounted by a factor related to its position in the list. Typically, the utility is defined as the likelihood that a user will observe a recommendation at

position j in the list. As the user goes through the recommendation list, the utility will decrease while the position j increases.

To account for the positional discount of the utility, a metric, called *Normalized Cumulative Discounted Gain* (**NDCG**) [JK02] is widely used in the filed of information retrieval, in which the positions are discounted logarithmically. In general, we can assume that each user u will have a utility g_{ui} from being recommended an item i, then the averaged Discounted Cumulative Gain (DCG) for a list of J items is defined as

$$\mathrm{DCG} = \frac{1}{|U|} \sum_{u \in U} \sum_{j=1}^{J} \frac{g_{ui_j}}{\max(1, \log_b j)}, \tag{2.8}$$

where the logarithm base is typically set to 2 to ensure all positions are discounted. Then **NDCG** is the normalized version of DCG, defined as

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\mathrm{DCG}^*}, \tag{2.9}$$

where DCG^* is the ideal DCG. In such a setting, a recommendation list that places interesting items with high utility close to the beginning of the list, will therefore be preferred to the list that place these interesting items down the list. The reason is straightforward: these interesting items down the list might not be observed by the user, and hence may not generate any utility for the system [SG11].

2.2.3 Measuring Item Diversity

In content-based recommender systems, the matching function of user profiles and item content is a heuristic to capture the user's subjective opinion on how well an item agrees with the information need. In some cases, the best matched items obtained from the matching function may not correspond to the most desirable items from the user's point of view. This may lead to an unsatisfactory system due to the limited variety of items. For this reason, many practical systems incorporate *diversity* when

recommending items, to allow the recommended list to contain more variance in terms of the item content.

Assume the item content is available, a typical *diversity* metric is defined as the *aggregate* or *average* dissimilarity of all pairs of items in the recommendation list [ZH08]. Specifically, for a given *distance* function $d(\cdot, \cdot)$, the diversity of the recommended list $\mathcal{S}$ is defined as

$$f_d(\mathcal{S}) = \frac{2}{p(p-1)} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}, j \neq i} (d(i, j)) \tag{2.10}$$

where $|\mathcal{S}| = p$, and the dissimilarity of an item pair is represented as $d(i, j)$, which is symmetric, i.e., $d(i, j) = d(j, i)$. The dissimilarity, or say, distance, of two items is application-dependent, which may correspond to a distance between feature vectors under a mapping of items into a feature space. For example, in a news recommender system, the feature vector could be term/word-based vector, each entry of which represents the importance of the corresponding term/word, e.g., the TF-IDF score. The distance measure on such a vector could be the cosine similarity.

2.2.4 Measuring User Vitality

In the applications of people-to-people recommendation, the success of a recommendation is not purely based on the satisfaction of a single user, but the preferences of both parties being involved, which is different from traditional user-item recommender systems. We refer this type of recommendation as *reciprocal recommendation*. In a reciprocal recommender system, the vitality of a user is an important feature. It defines how active the user is, e.g., how often the user sends messages to other users. By explicitly considering the vitality for recommendation, a vital user can improve the engagement of other passive users, which renders the reciprocal network more healthy and energetic. To measure how active that the users within the recommended list are, we define the *set vitality* measurement as the *average activeness* of all the users

in the list. Specifically, given a recommended user set $\mathcal{S}$, associated with each user's interactive activities $u_{i\leftrightarrow}$, the *set vitality* of $\mathcal{S}$ is calculated as

$$f(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{u_i \in \mathcal{S}} \frac{1}{|V_{u_i}|} \sum_{v_j} \frac{|u_{i \to j}|}{|u_{i \leftrightarrow j}|}, \tag{2.11}$$

where V_{u_i} is the set of users that have been clicked or contacted by u_i, $|u_{i \to j}|$ denotes the number of clicks on v_j by u_i and messages that u_i has sent to v_j and $|u_{i \leftrightarrow j}|$ represents the total number of interactions between u_i and v_j. This measure is able to evaluate the vitality of users in the recommended user list.

2.3 Related Work

In the following, we highlight the research efforts that are related to the three questions being addressed in this dissertation. In particular, Section 2.3.1 describes the existing solutions of modeling user accessing behaviors in recommender systems; Section 2.3.2 presents graph-based approaches that have been developed to capture the complex relations for recommendation; and Section 2.3.3 reviews the temporal methods devised to model the preference dynamics of users along with time.

2.3.1 User Behaviors

In general, user behavior data, i.e., the consumption history of users (including the clicks that a user has made), is recorded as in the log. A lot of research work focuses on analyzing user click logs, associated with user information and item content, to infer user preferences. Broadly speaking, there are two strategies to achieve this, including content filtering and collaborative filtering.

The content filtering approach creates a profile for each user or item to characterize its property [KBV09]. A user's profile may involve the user's demographic information

or pre-obtained user preferences. According to users' behavioral data, the profiling techniques are able to find a mapping between users and items. For instance, [LDP10] develops a Bayesian framework for predicting users' current interests from the activities of that particular user and the trending topics demonstrated in the activity of all the users (obtained from user logs). Related work also includes [ISY06, ISY07], contributing to the behavior modeling by different types of probabilistic approaches.

An alternative to content filtering analyzes users' behavioral data to find similar users without requiring the creation of an explicit profile, which is known as collaborative filtering. This type of methods captures the relationship between users and interdependencies among items, and in turn identifies new matchings among users and items. In general, collaborative filtering can be divided into two categories: neighborhood methods and latent factor models.

Neighborhood methods focus on calculating the similarities between users (known as user-based), or alternatively, between items (known as item-based). The neighborhood strategy has been significantly explored in the past decade. A lot of research work has been published along this direction [DI99, DK04, LSY03, NA98, SKKR01], including user-based and item-based recommendation. User-based recommender systems [KMM$^+$97] evaluate the interest of a user over an item by analyzing the behaviors of other users over this item, i.e., they have similar rating patterns, or say, their rating patterns over items correlate with each other. Comparatively, in item-oriented filtering approaches [DK04, SKKR01], a user's preference over an item is calculated based on ratings of similar items by the same user. Here "similar" means that the items have similar ratings with respect to the target item.

In contrast to neighborhood-based recommender systems, which utilize the historical data directly in the prediction process, latent factor models use the ratings or click behaviors to learn a predictive model. The basic idea is to model the user-

item relationships using latent factors, which can represent latent characteristics of users and items in the recommender systems, e.g., groups that users or items belong to. By trained using the available historical data, the predictive model can be used to infer the ratings of users over new items. In recent years, various latent factor models have been presented to provide solutions for recommendation, including Latent Semantic Analysis [Hof03], Latent Dirichlet Allocation [AC10], Maximum Entropy [ZK12], Support Vector Machines [GFMG06] and Singular Value Decomposition [BKV07, Kor08, TPNT09], etc.

The aforementioned approaches focus on analyzing users' historical clicks or ratings and treating them as a set. Recommenders using these approaches are able to capture users' general preference over items, but may fail to obtain the detailed accessing patterns, e.g., what is the accessing sequence/pattern of users, how long does a user may read through a news article he/she prefers, etc. Without such information, a recommender system cannot be well tailed for a user's personalization. Recently, researchers in recommendation domain start to put focus on recommending an accessing sequence of items for users. For example, in a news recommender system, a piece of news will be presented right after the news content that a user is reading, which enables the user to read through the related articles. Hence, besides tracking user click behaviors, other information is also valuable to build high-quality user profiles. As pointed out in [ACW12], the quality of recommendations can be improved by exploiting different types of "post-read" engagement signals like sharing, commenting, printing and e-mailing article links.

2.3.2 Complex Relations

Recommender systems often involve complex relations among different elements, e.g., users, items, metadata of items, etc. Such complex relations are the key factor that

dominates the decision of users. Therefore, it is imperative to consider these relations when modeling users' behaviors and formalizing user profiles. To this end, a lot of research work has been proposed, following the stream that models the complex relations linked to users. For example, a method of constructing context-aware and multi-applicable preference models is proposed in [OKMA07], which utilizes Bayesian networks to capture the relationships between users and other related elements.

The most common approach to analyzing complex element relations in recommender systems is graph-based method. Naturally in a graph (e.g., a bipartite graph), users and items can be represented as nodes, and their relations can be modeled as edges. Additional elements can also be put into the graph as new types of nodes or edges. Graph-related features in a recommender system highlight the interactions between users and items. The topological properties of nodes can be considered as an indication of users preferring items, and hence be used in recommendation [GP07]. In addition, the graph structure can be used to formalize affinity measures between users and items for cross recommendation. The bipartite graph can also be transformed to a unipartite user/item graph to simplify the graph structure [ZRMZ07].

Graph-related features can also be used in the learning-based paradigm [LC12]. As an example, [MKR03] emphasizes reachability via a graph-based algorithm within the implicit graph structure underlying a recommender system. [Yaj06] uses a Laplacian kernel to capture the positional relations among nodes on the graph and builds one-class SVM models for each user to recommend items that are closer to their previously accessed items. Similar work includes [KSKÇ13, WMYL06, WTZ10], in which graphs are utilized to model the relations of users accessing items.

For more complex relations rather than user $\times$ item, hypergraphs are typically employed to provide comprehensive modeling. Hypergraph can help resolve general learning problems, e.g., clustering [GV00], classification [SJY08, WK07] and

embedding [ZHS07]. Hypergraph learning has been explored in various machine learning areas, such as gene expression classification [HTKK08, THK09], image retrieval [DY08, HLZM, WHZ10], document analysis [LL11], etc. In the domain of recommendation, hypergraph modeling is also utilized to capture the complex relations within a recommender system. For example, [BTC+10] uses hypergraph to model various music-related objects (e.g., users, songs, artists, albums, tracks, tags, etc.) and relations within the domain of music recommendation and considers music recommendation as a ranking problem on this hypergraph.

2.3.3 Temporal Dynamics

In most recommender systems, the information needs of users over items are temporally dependent [BDN11], i.e., they oscillate and evolve over time. The temporal changes of user preferences often result from different factors. For example, in an online shopping recommender, a user's interest changes when new products or services emerge on the website. Additional reasons related to this include specific holidays, seasonal changes or even a change in the family structure (e.g., giving birth to a baby). To capture such an interest drift in recommender systems, a couple of research work has been proposed. The most representative one is [Kor10], in which the time changing behavior is tracked throughout the life span of the data. They argue that the temporal dynamics is able to exploit the relevant components of all data instances, while discarding only what is modeled as being irrelevant.

In existing research efforts, the temporal dynamics of user interest is handled in two different ways. One way is to model the temporal information into a temporal variable, as part of the predictive model. When the consumption data of users are feed into the system, the temporal variable will change accordingly. As pointed out in [Kor10], the temporal change of user interests may depend on multiple factors,

some are fundamental while others are more circumstantial. For example, in a movie recommender system, a user may change his/her preferred genre or become interested in a director. In addition, the user's rating pattern might change, e.g., he/she assigns a "3" star to a movie to indicate neutral preference, but later he/she may indicate dissatisfaction by the same "3" star. Such factors can be modeled into a predictor $b_i(t)$, which represents the estimation for the user's rating to the item i at time t.

Along this direction, several researchers investigate how the model of collaborative filtering will change as items are rated over time. For example, [LHCA10] explores the time diversity of the top-N recommendations in a recommender system, and shows that how different characteristics of user rating patterns affect the diversity of results. Similarly, in [LPP08], a recommendation approach that constructs pseudo rating data from the implicit feedback is proposed, and the temporal information is incorporated into the pseudo rating for dynamics modeling. Additional related work include [AT11, KM03, LHC09], most of which adopt the strategy of modeling temporal information as a temporal factor/variable.

Another line following the stream of this research introduces the concepts of long-term and short-term profiles [DL05], and proposes different weighting schemes to integrate both types of profiles for temporal profile modeling. Time is an important aspect in capturing temporal dynamics of user interest. Several time-evolving models [STF06, SFPY07] introduce time as a universal dimension shared by all users. However in recommender systems, the time dimension should be a local effect and should not be compared cross all users arbitrarily [XYZ+10]. According to this intuition, Xiang et al. [XYZ+10] proposes a session-based temporal graph that simultaneously captures users' long-term and short-term preferences over time, and recommends items via random walking on the temporal graph. Another example is [RN07], in which a critique-based mobile recommendation method is proposed, where the

long-term preferences are collected by mining past interactions and by letting users explicitly define a set of stable preferences. A type of critique is also introduced to let users express additional session-specific preferences.

2.4 Concluding Remarks

This chapter first introduces the notations and formal procedures of content filtering and collaborative filtering. Then, the evaluation metrics used in this dissertation is presented, in accordance with different quality criteria. Finally, the state-of-the-art related to the topics studied in this dissertation is discussed.

CHAPTER 3

UNDERSTANDING BEHAVIORS

In this chapter, we focus on the problem of understanding user behaviors in recommender systems from both coarse-grained and fine-grained perspectives. To begin with, we describe the research objective along this direction. We then propose two representative approaches published during my Ph.D. study, one exploring coarse-grained user behaviors, and the other focusing on fine-grained accessing patterns. The chapter concludes with the discussion of potential application scenarios of the proposed approaches.

3.1 Research Objective and Contributions

User behavior data is the primary input of user profiling and recommendation algorithms, and hence it plays the key role to model users' personalized interest in recommender systems. In order to effectively analyze user behaviors and obtain important user-oriented preference, users' historical behaviors in a recommender system are often modeled from both coarse-grained and fine-grained perspectives. Specifically, the goal of understanding user behaviors includes:

- From a coarse-grained perspective, user behaviors will be treated as a set of user clicks or user ratings, and user preferences will be learned based on such input associated with the detailed content of items. Most research efforts of modeling user preferences fall into this direction. The essence of the approaches is to obtain the binary preference or numeric scores of users towards items.
- From a fine-grained perspective, detailed user accessing history will be analyzed in a session-based granularity, e.g., in a user session, what specific items are accessed by the user, how many clicks a user performs, what is the customized

click sequence of users, etc. Such information is valuable for capturing the fine-grained preference of users, and in turn can provide guidance to personalized recommendation.

In this dissertation, I explored the ways of modeling user behaviors from these two perspectives, and proposed the corresponding solutions to each perspective. In particular, a method, named FRec [LPK+13], is presented to model the coarse-grained granularity of user behaviors, in which a user's behavior is decomposed into two parts, one reflecting the user's personal preference whereas the other contributing to the enrichment of community-based topics. The major contribution of this work is two-fold:

- A topic modeling approach to distinguishing community interests from personal interests by using a Bernoulli variable to control the distribution from which a word is drawn;
- A principled recommendation framework that is capable of recommending topic-related influential users and topic-cohesive interactive communities given a user profile.

This work is studied under the environment of social media, and it can be easily extended to other application scenarios. For example, a community could be a research community that focuses on a specific research area, e.g., context-aware recommendation; and a user can be a researcher who is active in different research areas and develops innovative ideas. Users may have their own research interests, as well as collaborating with other researchers. The coarse-grained behaviors of users involve publishing research papers at different research communities. Hence, our proposed method can be applied to such a scenario.

To reason on fine-grained user behaviors, a method, named SCENE [LWL+11], is presented to capture the changing pattern of a user's accessing behavior, where the

decreasing property of the user's interest is formulated using a principled click model. The major contribution of this work is two-fold:

- A novel two-level representation for recommendation results, where the first level contains general topics summarized from item clusters similar to the user's profile, and the second level includes representative items within each item cluster;
- A principled framework for item selection. We observe that the interestingness of items with respect to a user could be regressive, e.g., user interests towards items in a news recommender system, and based on this "submodularity" property, we model the item selection problem as a budgeted maximum coverage problem [KMN99], which is more realistic than independently selecting items.

This work explores the detailed click behaviors of users, and the derived recommendation model is suitable to the scenario that a user goes through a list of recommended results. Hence, it can be applied to a lot of recommendation applications, e.g., recommending a list of news articles, movies, music tracks, etc. In the following, these two approaches are introduced in details and are discussed in depth.

3.2 Modeling User- and Community-Oriented Behaviors

In a recommender system, users behave diversely in terms of their personal interests; however, sometimes users interact with each other and exhibit similar user behavioral patterns, and consequently show a collective group/community behavior. The collective user behaviors (coarse-grained) contribute to both personal interests and community topics in an implicit way.

To illustrate this, we take the application of social media as an example. In social media, a community is often formed by a collection of users with social connections

as well as similar topic preferences. Taking online marketing campaign as an example, marketers not only target individuals with certain interest, but also hope the marketing messages could be cascaded to large audience sharing similar interests. In such a scenario, one critical issue of utilizing social media data is how to precisely identify users' personal interest and the interest of communities which these users are connected to or frequently interact with. Thus it is very important to capture both user-oriented and community-oriented topics.

Automated discovery of topics and communities has received widespread attention in academia and has been addressed differently in previous works. A common approach is to use generative Bayesian models to capture the correlations among users, communities and topics. However, prior approaches cannot make a distinction between user-oriented and community-oriented topics. Taking a query "campaign + economy" as an example, the task is to identify users and communities that are interested in US presidential campaign and also often discuss the topic of economy related to the campaign. "campaign" is discussed by a lot of people as it is relevant to the presidential selection, whereas "economy" often appears in users' general posts and may not be related to "campaign". In this case,

- if we only consider user-oriented topics, the recommended users identified to be interested in the query are not necessarily connected to the communities focusing on the query-related topics. Targeting these users will not guarantee the marketing messages to further cascade in the social network. In addition, the extreme versatility of users interest, informal writing, and spam in the social network make it difficult to infer communities interests with reasonable perplexity.
- if we only consider community-oriented topics based on posts by all the users in the communities, the fine-grained topic interest of each individual user is

difficult to model due to the coarse community-oriented topic structure. Also, detecting topics in an indiscriminate way will result in a lot of noise since all the user-generated content will contribute to the community topics. Therefore, we cannot identify the source from which "economy" is originated.

The advantage of modeling user-oriented and community-oriented topics simultaneously is that it could identify high-quality community topics by sampling the topic for each word from either the community topic-word distribution or the user topic-word distribution. Thus the noises induced by a wide variety of user interests that could contaminate the community topics can be naturally mitigated.

In this work, we identify the latent relationships among social objects, i.e. users and communities, by distinguishing a user's interest from interests of communities. We propose a generative topic model to capture both types of interests as topics in a parameter universe with a mechanism that identifies the association of interests to either a given user or a given community. Our proposed model makes use of the communities derived from the social links of users to avoid the expensive computation of combining the community discovering process with the topic modeling process. We further provide a novel Framework of Recommendation, named FRec, based upon the derived relationships, which is able to recommend topic-related influential users and topic-cohesive interactive communities for a given user's profile.

3.2.1 Background and Prior Approaches

User recommendation, often referred to as friendship recommendation or link prediction, focuses on recommending users to a target user based on diverse criteria. From a network perspective, user recommendation refers to finding missing edges in a user network. Typical approaches to solving this problem often utilize the network structure and node connections, e.g., proximity measures that are based on network topo-

logical features [LNK07], supervised learning methods [AHCSZ06], relational learning methods [PU03], etc.

In social media, the content generated by users, e.g., user relationships or posts, is a valuable information source to model users' preference. Recently, several methods have been proposed to resolve user recommendation in social media by employing latent Dirichlet allocation (LDA) alike topic models [PG11]. These efforts, however, only consider interest similarity, and ignore the interaction of users, which is essential for expanding social network. In our work, we try to recommend users with influence abilities, given the fact that these users can help enrich the interactions among users. In addition, our model can distinguish users' personal interests from the topics discussed within communities.

Automated community discovery has been well studied by researchers. One direction in community discovery involves using the social linking structure among users to identify communities, e.g., min-cut based partitioning, centrality-based and Clique percolation methods [For10, POM09]. However, they did not take into account the content generated by users in social network, which might result in the irrationality of the identified communities.

Another direction in community discovery is to incorporate content analysis into the discovery process. Probabilistic models are often employed to capture the topics being discussed by users and within communities [SCFS12, XZWY12, YJCZ09, ZML+06], which assume all the content generated by a user will contribute to the community detection. In reality, however, an online user often posts his/her personal information, e.g., moods and activities, which might not be related to any community. Comparatively, our model distinguishes community-oriented topics from users' personal topics within the content, which is more reasonable in modeling the topic interests of users.

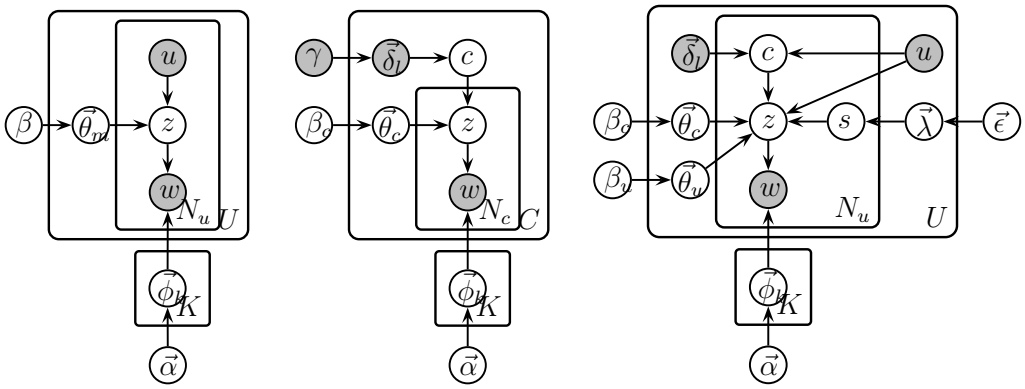
Given the detected communities, a further step for online community management is community recommendation. [CZC08] proposes a collaborative filtering method for personalized community recommendation, by considering multiple types of co-occurrences in social data, e.g., semantic and user information. [CCL+09] uses association rule mining to discover associations between sets of communities that are shared across many users, and LDA [BNJ03] to model user-community co-occurrences via latent aspects. Both works performed experimental evaluation on Orkut data set. However, they cannot distinguish community topics from users' interest.

3.2.2 User-Community-Topic Model

In this section, we first discuss two basic topic models used for tracking topic interests of online users or online communities. Based on the discussion, we propose User-Community-Topic model to resolve the issues in the two basic models. We then describe how to learn the hyper-parameters using Gibbs sampling.

Discussion on Topic Models

Fig. 3.1(a) shows the graphical model for what we refer to as the "user-topic model" (UT). UT aims to capture the correlation between users and topics. The generation of a document (containing all the posts of a user) is considered as a mixture of topics. Each topic corresponds to a multinomial distribution over the vocabulary. Based on the learned posterior probability, each user's preference of using words and involvement in topics can be discovered. However, in most cases, users might have diverse interests over topics. By using UT model, the obtained posterior probability of a user over a specific topic might be affected by the general topic interests of this user. In addition, users in social media often share common interests over topics, which cannot be captured in UT model.



(a) UT Model. (b) CT Model. (c) UCT Model.

Figure 3.1: Plate diagram for three topic models.

Another model is called "community-topic model" (CT) (as shown in Fig. 3.1(b)), where the generation of a document is affected by both the topic factors and the community factors in a hierarchical manner. In CT model, we treat all the posts within a community as a document. The difference from UT is the community factor c, by which topics within a document would be affected. One major problem of the CT model is that user posts in a community could include various topics, rendering the community document highly inconsistent. Sampling for all the words in a community document would result in uncontrolled generalization error for inference due to the noisy feature of social media data. In addition, there is no way to capture a specific user's interests using CT model, since no user factor is involved.

The Proposed Model

Our goal is to model the relations among users, topics and communities on social media. Taking Twitter as an example, we have the tweets posted by users and the follower-followee relations of users; however, we do not have the explicit community membership of users. We perform community discovery on the users' friendship network, and allow a user to belong to multiple communities.

To achieve this, we employ the algorithm introduced in [YYT06] to obtain the community memberships of users. We therefore assume there is a community factor c that captures the user-community memberships with respect to user u. Also, within each community, users might discuss different topics, and hence we have a topic factor z that characterizes the topic-community relations. For topic mixture and term mixture, we give them Dirichlet priors; for community mixture, we use the distribution derived by analyzing the follower-followee relations. In this way, we are not concerned with the relations between the community and the user, but focus more on the relations between the community and the topic (i.e., $p(z|c)$), and the relations between the user and the topic (i.e., $p(z|u)$). Table 3.1 lists the notations used in our model.

Table 3.1: Notations for quantities in the model.

	Descriptions
U	the user set in the community data.
V	the dictionary of texts in the community data.
L	the number of communities predefined.
N_u	the term set of texts posted by user u.
$\vec{\alpha}$	Dirichlet prior hyperparameter (known) on the term distribution.
$\vec{\beta}$	Dirichlet prior hyperparameter (known) on the mixture topic distribution.
$\vec{\gamma}$	Prior hyperparameter (known) on the mixture community distribution.
$\vec{\epsilon}$	Prior hyperparameter on the binary mixture.
$\vec{\phi_k}$	$p(t\|z=k)$, the mixture component of topic k.
$\vec{\theta_m}$	$p(z\|u=m)$, the topic mixture proportion for user m.
$\vec{\delta_l}$	$p(u\|c=l)$, the user proportion for community l. (observed)
$\vec{\lambda}$	binary mixture for word generation.
c	the community mixture.
u	mixture indicator that chooses a user from a community.
z	mixture indicator that chooses the topic for the term from a user.
w	term indicator for the word from a user.
s	binary factor for word generation.

We denote our proposed topic model as "user-community-topic model" (UCT in Fig. 3.1(c)). We add a latent Bernoulli variable s (a binary factor) to indicate whether

a word is related to a user itself or to a community. In particular, s takes value 0 if the word w is generated via the user-topic route, value 1 if the word is generated from the community-topic route. The variable s in our model acts as a switch: if $s = 0$, words are sampled from a user-specific multinomial $\vec{\theta}_u$, whereas if $s = 1$, words are sampled from a community-specific multinomial $\vec{\theta}_c$ (with different symmetric Dirichlet priors parameterized by β_u and β_c). s is sampled from a document-specific Bernoulli distribution $\vec{\lambda}$, which in turn has a prior ϵ. The joint probability of the UCT model can be written as:

$$\begin{aligned} & p(w, z, u, c, s, \phi_k, \theta_u, \theta_c, \delta_l, \lambda | \vec{\alpha}, \vec{\beta}_c, \vec{\beta}_u, \vec{\epsilon}) \\ & = p(w|z, \phi_k) p(z|u, c, s, \theta_u, \theta_c) p(c|u, \delta_l) \\ & \cdot p(s|\lambda) p(\lambda|\vec{\epsilon}) p(\phi_k|\vec{\alpha}) p(\theta_u|\vec{\beta}_u) p(\theta_c|\vec{\beta}_c), \end{aligned}$$

where $p(z|u, c, s, \theta_u, \theta_c) = p(z|u, s = 0, \theta_u)$ (where $s = 0$), and $p(z|u, c, s, \theta_u, \theta_c) = p(z|c, s = 1, \theta_c)$ (where $s = 0$). Here $p(z|u, s = 0, \theta_u)$ is the probability of a user-specific topic, whereas $p(z|c, s = 1, \theta_c)$ is the probability of a community-specific topic. Given the graphical model described in Fig. 3.1(c), the generative scheme is shown in Alg. 3.1.

Gibbs Updates

To estimate the model, we use the collapsed Gibbs sampling [GS04]. For our UCT model, we are interested in the latent user-topic portions $\vec{\theta}_u$, the latent community-topic portions $\vec{\theta}_c$, the topic-word distributions $\vec{\phi}_k$ and the topic index assignments for each word z_i. Also in the learning process, the value of s will be generated based on a Bernoulli distribution and be updated through the Gibbs sampling for each word. $\vec{\theta}_u$, $\vec{\theta}_c$ and $\vec{\phi}_k$ can be calculated using just the topic index assignments z_i, i.e., $\mathbf{z}$ is a sufficient statistic for the three distributions. Therefore, we can integrate out the multinomial parameters and simply sample z_i and s_i.

Algorithm 3.1 Generative scheme of UCT model.

for each topic $z \in (1, \cdots, K)$ **do**
  Sample $\phi_k \sim Dir(\cdot|\vec{\alpha})$
end for
for each user $u \in (1, \cdots, U)$, **do**
  Sample $\lambda_u \sim Beta(\cdot|\vec{\epsilon})$
  for each word $w \in (1, \cdots, N_u)$, **do**
    Sample $s \sim Bern(\cdot|\vec{\lambda}_u)$
    Choose a community assignment $c_u \sim Mult(\cdot|\vec{\delta}_l)$
    if(s==0): **then**
      Choose a topic assignment $z \sim Mult(\cdot|\vec{\theta}_u)$
    else
      Choose a topic assignment $z \sim Mult(\cdot|\vec{\theta}_c)$
    end if
    Choose a term $w \sim Mult(\cdot|\vec{\phi}_k, z)$
  end for
end for

The collapsed Gibbs sampler needs to compute the probability of a topic z being assigned to a word w_i, given all other topic assignments to all other words, with respect to a specific value of s (0 or 1). Similarly, it needs to calculate the probability of s being assigned to a word w_i, given all other s assignments to all other words. Let $\mathbf{z}_{-i}$ denote all topic allocation except for z_i and $\mathbf{s}_{-i}$ represent all s assignments except for s_i. The probabilities that we need to update include: (1) $p(s_i = 0|\mathbf{s_{-i}}, w_i, z_i, u_i, c_i)$, (2) $p(s_i = 1|\mathbf{s_{-i}}, w_i, z_i, u_i, c_i)$, (3) $p(z_i|\mathbf{z_{-i}}, w_i, s_i = 0, u_i, c_i)$, and (4) $p(z_i|\mathbf{z_{-i}}, w_i, s_i = 1, u_i, c_i)$. The derivations of the updates for these probabilities are described in Table 3.2.

We analyze the computational complexity of Gibbs sampling in the proposed UCT model. As discussed above, in Gibbs sampling, we need to compute the posterior probability $p(z_i|\mathbf{z_{-i}}, w_i, s_i, u_i, c_i)$ for user-word pairs $(U \times V)$ and community-word pairs $(C \times V)$, where V is the total number of words. Each $p(z_i|\mathbf{z_{-i}}, w_i, s_i, u_i, c_i)$ consists of K topics, and requires a constant number of operations, resulting in $O(V \cdot K \cdot U)$, assuming $U >> C$, for a single sampling.

Table 3.2: Gibbs updates for UCT model.

$$p(s_i = 1|\mathbf{s_{-i}}, w, z, u, c) \propto \frac{p(s_i = 1, \mathbf{s_{-i}}, w, z, u, c)}{p(\mathbf{s_{-i}}, w, z, u, c)} \propto p(s_i = 1|z_i, c_i) = p(z_i|s_i = 1, c_i) \cdot p(s_i = 1|u_i)$$
$$\propto \frac{n_{z_i,c_i,s_i=1} + \beta_c(z_i)}{\sum_{z_i} n_{z_i,c_i,s_i=1} + \sum_{z_i} \beta_c(z_i) - 1} \cdot \frac{n_{s_i=1,u_i=u} + \epsilon_{s=1}}{\sum_{s_i} n_{s_i=1,u_i=u} + \epsilon_{s=0} + \epsilon_{s=1} - 1}.$$
$$p(s_i = 0|\mathbf{s_{-i}}, w, z, u, c) \propto \frac{p(s_i = 0, \mathbf{s_{-i}}, w, z, u, c)}{p(\mathbf{s_{-i}}, w, z, u, c)} \propto p(s_i = 0|z_i, u_i) = p(z_i|s_i = 0, u_i) \cdot p(s_i = 0|u_i)$$
$$\propto \frac{n_{z_i,u_i,s_i=0} + \beta_u(z_i)}{\sum_{z_i} n_{z_i,u_i,s_i=0} + \sum_{z_i} \beta_u(z_i) - 1} \cdot \frac{n_{s_i=0,u_i=u} + \epsilon_{s=0}}{\sum_{s_i} n_{s_i=0,u_i=u} + \epsilon_{s=0} + \epsilon_{s=1} - 1}.$$
$$p(z_i|\mathbf{z_{-i}}, w, s_i = 0, u, c) \propto \frac{p(\mathbf{z_i}, w, s_i = 0, u, c)}{p(\mathbf{z_{-i}}, w, s_i = 0, u, c)} \propto p(z_i, s_i = 0, w_i, u_i, c_i) = p(w_i|z_i) \cdot p(z_i|s_i = 0, u_i)$$
$$\propto \frac{n_{w_i,z_i} + \alpha(w_i)}{\sum_V n_{w_i,z_i} + \sum_V \alpha(w_i) - 1} \cdot \frac{n_{z_i,u_i,s_i=0} + \beta_u(z_i)}{\sum_{z_i} n_{z_i,u_i,s_i=0} + \sum_{z_i} \beta_u(z_i) - 1}.$$
$$p(z_i|\mathbf{z_{-i}}, w, s_i = 1, u, c) \propto \frac{p(\mathbf{z_i}, w, s_i = 1, u, c)}{p(\mathbf{z_{-i}}, w, s_i = 1, u, c)} \propto p(z_i, s_i = 1, w_i, u_i, c_i) = p(w_i|z_i) \cdot p(z_i|s_i = 1, c_i)$$
$$\propto \frac{n_{w_i,z_i} + \alpha(w_i)}{\sum_V n_{w_i,z_i} + \sum_V \alpha(w_i) - 1} \cdot \frac{n_{z_i,c_i,s_i=1} + \beta_c(z_i)}{\sum_{z_i} n_{z_i,c_i,s_i=1} + \sum_{z_i} \beta_c(z_i) - 1}.$$

3.2.3 Recommendation Strategies

In our work, we try to recommend a list of users with relevant topic interests and cohesive discussions. The target user can select some of the recommended users as friends, and then start to involve the discussion among these users. Our recommendation framework, FRec, provides various recommendation mechanisms based on our user-community-topic model. We also consider the user influence with respect to a topic. For each topic in the topic list, we can use the derived probabilities $p(u|z)$ as the initialization of the PageRank algorithm [LSMW98], and run PageRank on the friendship network to obtain the influence scores of users towards a specific topic z. Then the topic-relevant user influence can be denoted as $R(u|z)$. We setup a threshold (0.01) for $p(u|z)$ to filter out low probabilities.

User-to-User Recommendation

Given a target user $\hat{u}$, we can rank other users based on $p(u_i|\hat{u})$, and then select top ranked ones as $\hat{u}$'s recommendation. $p(u_i|\hat{u})$ can be calculated using Eq.(3.2).

$$\begin{aligned} p(u_i|\hat{u}) &= \frac{\sum_z \sum_c p(u_i \hat{u} c z)}{p(\hat{u})} \\ &\propto p(u_i) \sum_z \sum_c p(z|\hat{u}) p(z|u_i, s=0) p(z|c, s=1) p(c|u_i) p(c|\hat{u}) p(c) \\ &\propto p(u_i) \sum_z \left(p(z|\hat{u}) p(z|u_i, s=0) \sum_c p(z|c, s=1) p(c|u_i) p(c|\hat{u}) p(c) \right). \end{aligned} \quad (3.2)$$

Here $p(z|\hat{u})$ is the probability of topics given a test user $\hat{u}$, which can be obtained by extending Gibbs iterations over the test users after the hyper-parameters are learned. Note that in Eq.(3.2), we consider both user-based topics ($p(z|u_i, s=0)$) and community-based topics ($p(z|c, s=1)$). The user-based topics often include a user's personal interest. To make the recommendation more community-oriented, we can focus on community-based topics by removing the user-based component. The recommendation can be refined as

$$p(u_i|\hat{u}) \propto p(u_i) \sum_z \left(\frac{p(z|\hat{u})}{p(z)} \sum_c p(z|c, s=1) p(c|u_i) p(c|\hat{u}) p(c) \right). \quad (3.3)$$

By integrating the user influence into $p(u_i|\hat{u})$, we can have

$$p(u_i|\hat{u}) \propto p(u_i) \times \sum_z \left(\frac{p(z|\hat{u}) R(u_i|z)}{p(z)} \sum_c p(z|c, s=1) p(c|u_i) p(c|\hat{u}) p(c) \right). \quad (3.4)$$

In this strategy, the user-to-user relations residing in the friendship network are not considered. In order to make the recommendation more reasonable, we incorporate the neighborhood similarity between u_i and the target user $\hat{u}$ into the recommendation. The neighborhood similarity can be calculated as

$$sim(u_i, \hat{u}) = \frac{|neighborhood(u_i) \cap neighborhood(\hat{u})|}{|neighborhood(u_i) \cup neighborhood(\hat{u})|},$$

where $neighborhood(\cdot)$ denotes all the neighbors of the user. By integrating $sim(u_i, \hat{u})$ into Eq.(3.4), we have

$$\tilde{p}(u_i|\hat{u}) \propto p(u_i) \cdot sim(u_i, \hat{u}) \times \sum_z \left(\frac{p(z|\hat{u})R(u_i|z)}{p(z)} \sum_c p(z|c, s=1)p(c|u_i)p(c|\hat{u})p(c) \right). \tag{3.5}$$

User-to-Community Recommendation

Given a target user $\hat{u}$, we can also recommend communities to $\hat{u}$ based on the derived correlations among users, topics and communities. Given a community c, we can measure the relevance between $\hat{u}$ and c by

$$\begin{aligned} p(c|\hat{u}) = \frac{\sum_z p(c, \hat{u}, z)}{p(\hat{u})} &\propto \sum_z \frac{p(z|\hat{u}, s=0)p(z|c, s=1)p(c)}{p(z)} \\ &\propto p(c) \sum_z \frac{p(z|\hat{u}, s=0)p(z|c, s=1)}{p(z)}. \end{aligned} \tag{3.6}$$

A community with more influential users is likely to be more interactive, i.e., it may involve more activities of sharing information and discussing topics. Therefore, we consider the user influence for community recommendation. By integrating the user influence into $p(c|\hat{u})$, we have

$$\tilde{p}(c|\hat{u}) \propto p(c) \sum_z \frac{p(z|\hat{u}, s=0)p(z|c, s=1) \cdot \left(\sum_{u_j \in c} R(u_j|z) \right)}{p(z)}. \tag{3.7}$$

3.2.4 Empirical Evaluation

Real-World Data

The data set used in the experiment is a collection of tweets related to "presidential campaigns" between Barack Obama and Mitt Romney, ranging from March 1st, 2012 to May 31st, 2012. We crawled the tweets through Twitter Streaming API by feeding a list of keywords related to the campaign (e.g., campaign, Obama, Romney, economy, etc.) into the API request. We then crawled the follower relationships of each

user within the tweets data set. Due to the property of microblogging services, the crawled tweets might contain a lot of noise, which would hinder the topic modeling. Therefore, we did a series of preprocessing to alleviate the negative impact of noise data, including: (1) removing short tweets (with the word count less than 10); (2) removing tweets with hashtags more than 3; (3) removing tweets whose author has no more than 5 tweets; and (4) removing usernames (starting with "@") and URLs. After preprocessing, the tweets data contain 133,465 users, 5,558,763 mutual-following relationships and 5,079,994 tweets.

Comparison of Topic Models

For topic modeling, we concatenate the tweets of each user in the data set as a document. We process the tweets data by removing stopwords, tokenizing and stemming using MALLET. We also calculate the TF-IDF score of each word and then select the top ranked 10,000 words as features. After processing, the total number of word tokens in the tweets data is 6,643,278. We compare UCT model with two baselines: (1) CCF (Combinational Collaborative Filtering) [CZC08], which combines the bag-of-users and bag-of-words models to capture the relations among topics, communities and users within the network; and (2) TUCM (Topic User Community Model) [SCFS12], which assumes that a user's membership in a community is conditioned on its social relationship, the type of interaction and the shared information with other members. We also include the models shown in Fig. 3.1(a) (UT) and Fig. 3.1(b) (CT) in the comparison.

For all the models, we empirically set the number of communities as 500. We set the hyper-parameters to the following values [RZGSS04]: $\alpha = 0.01$, $\beta = \beta_u = \beta_c = 0.01$ and $\epsilon = 0.3$. We run 200 iterations of Gibbs sampling for training and extend the chain with 100 iterations over the test set.

Perplexity Comparison: We compare the predictive performance of our proposed UCT model with other baselines by computing the perplexity of unseen words in test documents. We calculate the averaged perplexity for 10-fold cross validation on the tweets data set. The lower the perplexity, the better the performance of the model [RZCG+10]. As is depicted in Fig. 3.2, the predictive performance of two basic

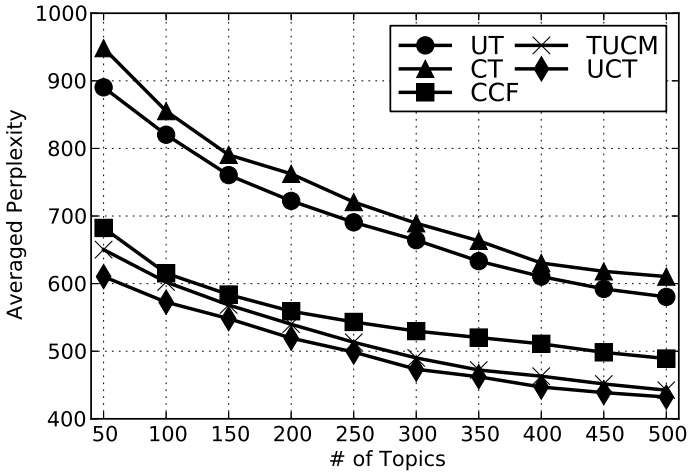
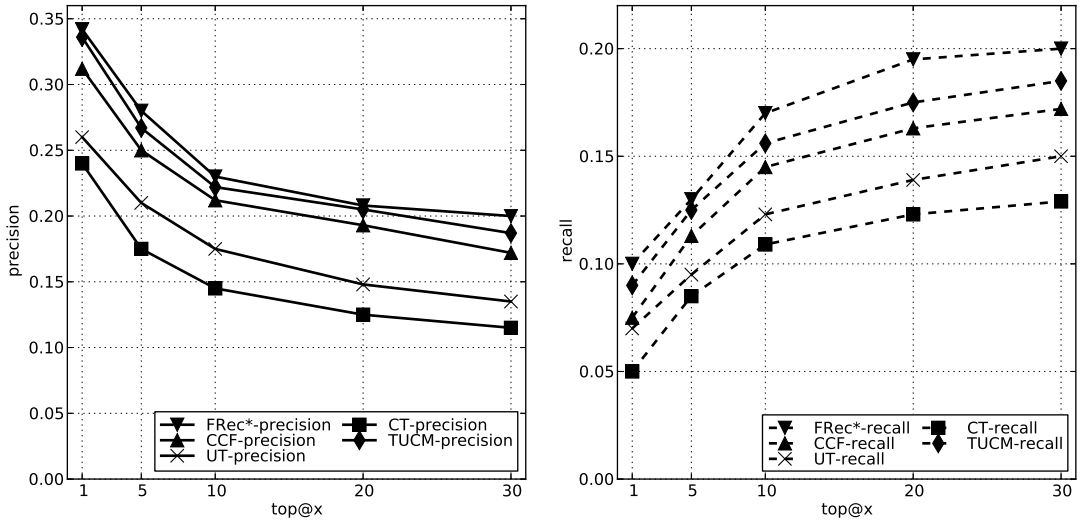


Figure 3.2: Perplexity evaluation.

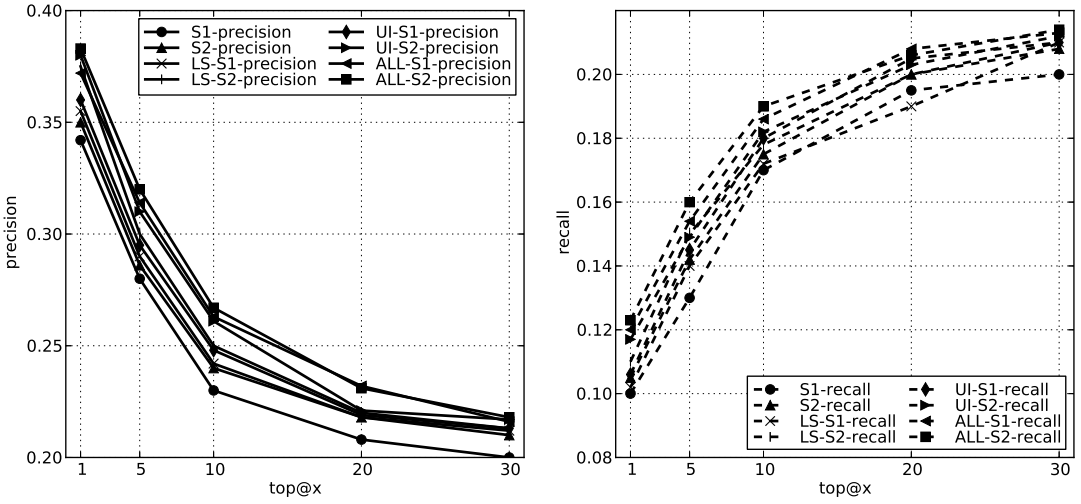
models (UT and CT) are not comparable with the other three topic models. The reason is straightforward: in both models, only one aspect (either u or c) is considered, which violates the characteristics of the data, since in social media, people post information not only for their own purpose, but also expecting to interact with each other. CCF combines the word factor and the user factor to capture the correlation between users and communities, and TUCM takes into account the type of interactions between users. These two models achieves better predictive performance compared with UT and CT. Our model distinguishes community-oriented topics and user-oriented topics. Such a distinction indeed exists in most real-world scenarios, i.e., a user has his/her personal topic interests, and is also often involved in the discussion within a specific community. In the recommendation experiments, we set the number of topics as 500 for all the models.

User Profile based Recommendation

To evaluate the user-profile based recommendation, we employ the leave-one-out or leave-n-out strategy as described in [CCL+09]. *precision* and *recall* are used to measure the recommendation effectiveness. *Precision* is calculated at a given cut-off rank, considering only the topmost results recommended by the approach, e.g., top@10, top@20 or top@30. We limit the size of our recommendation list to at most 30, and calculate the corresponding precision and recall values.



(a) Comparison of Topic Models.



(b) Effect of Different Components.

Figure 3.3: Comparison for user recommendation.

Recommending Users: We compare the user recommendation strategy introduced in §3.2.3 with several topic model based recommendation approaches: (1) UT: The recommendation can be achieved using the strategy similar to Eq.(3.2), by removing the components related to c; (2) CT: By considering the identified community membership, we can select a list of top ranked users, based on $p(u|c)$, from the community that the target user belongs to; (3) CCF: This method provides user recommendation by calculating the user similarity introduced in [CZC08]; (4) TUCM: The recommendation can be achieved using the strategy similar to Eq.(3.2).

Our goal is to select a list of users whose topic interests are close to the target user. By removing the user-oriented components from Eq.(3.2), we can make the recommendation results more community oriented, as defined in Eq.(3.3). Note that in Eq.(3.3), we consider the community information of both the target user and the recommended user. To this end, we randomly select 2,000 users from the user repository as the test set and randomly delete a set of links of each test user: (1) $S1$: removing 20% links; and (2) $S2$: removing 2% links. We conduct experiments based on these two setups. Fig. 3.3 shows the results for these users. For comparisons with topic models and link prediction methods, the experiments use setup $S1$; To evaluate the effect of different components in Eq.(3.5), we use setup $S1$ and $S2$.

From Fig. 3.3(a), we observe that our proposed framework FRec achieves the best recommendation performance in terms of *precision* and *recall*. Simply using topics (UT in Fig. 3.3(a)) cannot guarantee high-quality recommendation results. For example, two users might share similar interests but they do not have connections in the social graph.

In Fig. 3.3(b), we evaluate how user influence (UI) and users' local similarity (LS) affect the recommendation performance. We compare the basic model of FRec, the model with UI, the model with LS and the model with UI and LS for two different

settings $S1$ and $S2$. Based on the comparison, we observer that: (1) User influence component and local similarity component slightly improved the performance of user recommendation. Intuitively, a user will prefer to make friends with influential people, since through these people he/she can reach more friends. Also, a user will be likely to interact with friends-of-friends. (2) The user recommendation has more accurate results if more social links of users are reserved. This is primarily because social links can help identify the underlying communities and then enrich the recommendation model through the user-community relations.

Recommending Communities: For community recommendation, we treat the communities identified from the module of community detection as the ground truth. We randomly sample 2,000 users from the user repository and recommend communities for these users. The comparison includes: (1) FRec: The basic strategy described in Eq.(3.6); (2) FRec-s1: removing the factor of $p(z|u, s = 0)$ from Eq.(3.6), i.e., only considering the community-oriented topics for recommendation; (3) FRec-IN: the strategy described in Eq.(3.7); and (4) FRec-IN-s1: removing the factor of $p(z|u, s = 0)$ from Eq.(3.7), i.e., considering user influence and the community-oriented topic factor. We also compare FRec with several recommendation approaches, including CCF and TUCM as introduced previously. These two approaches use the inferred probabilities of $p(z|u)$ and $p(z|c)$ for community recommendation. We report the comparison in Fig. 3.4.

As observed in Fig. 3.4, the model of FRec-IN-s1 has the best performance against other baselines, which explains that users in social media would like to interact with influential users, and prefer to share information that is often discussed within a community, i.e., by a group of people. The community-oriented topic factor $p(z|c, s = 1)$ has superior power over user-oriented topic factor $p(z|u, s = 0)$ in dominating the results of community recommendation.

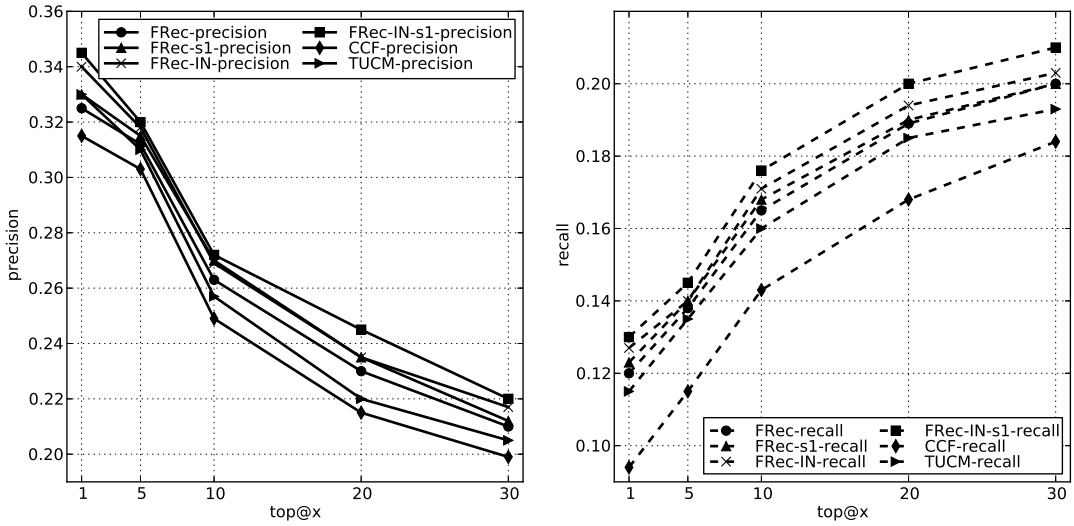


Figure 3.4: Community recommendation result.

3.2.5 Summary of FRec

We have introduced a generative graphical model, User-Community-Topic model (UCT), for capturing user-oriented topics and community-oriented topics simultaneously in social media data. Based on the model inference, we further proposed a novel recommendation framework, FRec. Given a user's profile, FRec is able to recommend a list of topic-related influential users or a list of topic-cohesive interactive communities. Extensive evaluation on a dataset obtained from Twitter has demonstrated the effectiveness of FRec compared with other probabilistic model based recommendation methods. The proposed framework can be easily extended to the case that recommends users and communities based on a set of keywords. In addition, it can be seamlessly integrated into real-life networks.

3.3 Modeling Decreasing Property of User Interest

Personalized recommendation is oriented from exploring the relations between items in item repository and the user's profile. In my previous work [LWL+11], I focus

on news personalization and try to model user behaviors in news recommender systems. What is proposed in [LWL+11] is essentially a hybrid recommendation method; however, different from prior approaches, we provide a two-level recommendation hierarchy, where the first level shows a brief summary for each topic category the user might prefer, and the second level gives a specific list of news articles similar to the user's reading interest.

Another significant contribution of this work is that the submodularity hidden in different dimensions of items motivates us to incorporate this property into our solution to the second level of the representation. The primary goal of this motivation is to explore the fine-grained user behavior in a more principled way. By considering the decreasing property of user interests towards items, such behaviors can be explicitly modeled using a submodularity function [NWF78].

3.3.1 Background

Web-based news reading services, like Google News and Yahoo! News, have become increasingly prevalent as the Internet provides fast access to news articles from various information sources around the world. With the gigantic amount of news articles, a key issue of online news services is how to help users find interesting articles that match the users' preference as much as possible, by making use of both news content and user information. This refers to the problem of personalized news recommendation.

Despite a few recent advances, personalized news recommendation remains challenging for at least three reasons. First, the scalability of most news recommendation services calls for fast speed of computation; Second, news item candidates are not independent in most scenarios, i.e., browsing one news item may affect the subsequent news reading; Third, the popularity and recency of news articles change dramatically

over time, which differentiates news items from other web objects, such as products and movies, rendering traditional recommendation methods inefficient.

In our work, to address the issues mentioned above, we propose SCENE, a *SC*alable two-stage p*E*rsonalized *N*ews r*E*commendation system with a two-level representation, where the first level contains various topics relevant to users' preference, and the second level includes specific news articles. In our system, we explore the intrinsic relation between users and news articles, along with the special properties (e.g., popularity and recency) of news items when recommending to individual users. Also, the system is capable of efficiently dealing with large scale news corpus.

Specifically, SCENE consists of three major components – *Newly-Published News Articles Clustering*, *User Profile Construction* and *Personalized News Items Recommendation*. For news articles clustering, we initially partition newly-published news articles into small groups by making use of *Locality Sensitive Hashing* [GIM99], and then hierarchically separate these groups into intermediate clusters, each of which is summarized using probabilistic language models (e.g., *Probabilistic Latent Semantic Indexing* (PLSI) [Hof99] and *Latent Dirichlet Allocation* (LDA) [BNJ03]). For personalization, the user's profile is constructed in three different yet related dimensions – news topic distribution, similar access patterns and news entity preference. Based on the generated topic distribution, we sequentially select news clusters similar to the profile of a given user as the first level of the result representation. In each news cluster, the submodularity hidden in different dimensions of news articles motivates us to incorporate this property into our solution to the second level of the representation. Extensive empirical experiments on a collection of news articles obtained from various news websites demonstrate the efficacy of our approach, in terms of the accuracy of selected top ranking news items and the diversity of the recommended news list.

3.3.2 Introduction to Submodularity

Let E be a finite set and f be a real valued nondecreasing function defined on the subsets of E that satisfies

$$f(T \cup \{\varsigma\}) - f(T) \leqslant f(S \cup \{\varsigma\}) - f(S), \tag{3.8}$$

where $S \subseteq T$, S and T are two subsets of E, and $\varsigma \in E \backslash T$. Such a function f is called a **submodular** function [NWF78]. Intuitively, by adding one element to a larger set T, the value increment of f can never be larger than that by adding one element to a smaller set S. This intuitive diminishing property exists in different areas, i.e., in social network, adding one new friend cannot increase more social influence for a more social group than for a less social group.

The budgeted maximum coverage problem is then described as: given a set of elements E where each element is associated with an influence and a cost defined over a domain of these elements and a budget B, the goal is to find out a subset of E which has the largest possible influence while the total cost does not exceed B. This problem is NP-hard [KMN99]. However, [KMN99] proposed a greedy algorithm which picks up the element that increases the largest possible influence within the cost limit each time and it guarantees the influence of the result subset is $(1 - 1/e)$-approximation. Submodularity resides in each "pick up" step. A key observation is that submodular functions are closed under nonnegative linear combinations [LKG+07].

3.3.3 Submodularity Model for Recommendation

In a particular news group, most of news articles concentrate on similar or even the same topic, with minor difference on major aspects of the corresponding topic. For example, given a news group talking about a popular movie "*Inception*", one piece of news may focus on the actor cast of this movie, while another may describe the

high-end techniques used in this movie. Typically, a news reader is interested in some specific aspects of the given topic, but not all of them. Based on this intuition, our news selection strategy can be described as follows (note that $\mathcal{N}$ denotes the original news group, $\mathcal{S}$ represents the selected news set, and ς is the news item being selected). After selecting ς,

- $\mathcal{S}$ should be similar to the general topic in $\mathcal{N} \setminus \mathcal{S}$;
- The topic diversity should not deviate much in $\mathcal{S}$;
- $\mathcal{S}$ should provide more satisfaction to the given user's reading preference.

Per the above strategy, we define a quality function f to evaluate the current selected news set $\mathcal{S}$ over the whole news group $\mathcal{N}$ as

$$\begin{aligned} f(\mathcal{S}) =& \frac{1}{|\mathcal{N} \setminus \mathcal{S}| \cdot |\mathcal{S}|} \sum_{n_1 \in \mathcal{N} \setminus \mathcal{S}} \sum_{n_2 \in \mathcal{S}} sim(n_1, n_2) \\ &+ \frac{1}{\binom{|\mathcal{S}|}{2}} \sum_{\substack{n_1, n_2 \in \mathcal{S} \\ n_1 \neq n_2}} -sim(n_1, n_2) + \frac{1}{|\mathcal{S}|} \sum_{n_1 \in \mathcal{S}} sim(u, n_1), \end{aligned} \tag{3.9}$$

where n_1 and n_2 denote news items, u represents the given user, and $sim(\cdot, \cdot)$ represents the similarity between two profiles, either the user profile or the news profile.

In Eq.(3.9), three components are involved, corresponding to the news selection strategy we list above. The first one aims to evaluate the quality of how representative that the selected news set $\mathcal{S}$ is over the original news set; the second one provides a perspective on how diverse that the topics underlying the selected news articles are; and the last component gives us the evidence that how much the user's preference is satisfied by the selected news set $\mathcal{S}$. $f(\mathcal{S})$ balances the contributions of different components. Note that all these three components are naturally submodular functions. Based on the linear invariability of the submodular function, $f(\mathcal{S})$ is also a

submodular function. Suppose ς is the candidate news article, the quality increase is therefore represented as follows:

$$I(\varsigma) = f(\mathcal{S} \cup \{\varsigma\}) - f(\mathcal{S}). \tag{3.10}$$

The goal is to select a list of news articles which provide the largest possible quality increase within the budget, where the budget can be regarded as the maximum number of recommended items in each news group. Hence, personalized news recommendation is transformed to the budgeted maximum coverage problem.

In each news group, a greedy algorithm is employed to solve the budgeted maximum coverage problem, by sequentially selecting the news article providing the largest quality increase based on the selected news set until the budget is reached. To integrate recommended news items from different news groups into the final recommendation list, we select top ranking items within each group, where the number of items selected in one group is proportional to the interest weight of the user on the corresponding topic category.

The proposed method models personalized news recommendation as a budgeted maximum coverage problem, i.e., the selection of one news item will influence the selection of the following news items. From this perspective, our work is similar to [LCLS10], in which personalized recommendation of news articles is modeled as a contextual bandit problem, where a learning algorithm sequentially selects articles to serve users based on contextual information about users and articles, while simultaneously adapting its article-selection strategy based on user-click feedback to maximize total user clicks. Our work is orthogonal to theirs in terms of news articles selection, since they focus on the long-term effect of recommendation, whereas our concern is located on single-session recommendation.

The submodularity-based news selection strategy provides us a diverse news recommendation list within each topic category. In addition, multiple topic categories

are recommended to individual users, by which the diversity of the final recommendation result is explicitly enriched to a great extent.

3.3.4 Empirical Evaluation

To evaluate the proposed method, we gather news articles along with users' access history from two popular news websites – *Bloomberg* and *Thewrap*. Both websites contain multiple news topic categories, such as sports, movies and politics. We gather the news data for 9 categories on purpose, where the data collection ranges from Aug 15th, 2010 to Nov 16th, 2010. In order to embody the role of similar access patterns in users' profiles, we preprocess the data by removing news articles that are rarely accessed (i.e., the accessed frequency is less than 10 times per day) and by storing users with frequent online reading behaviors (i.e., users who read news articles every day and read more than 10 pieces of news each day). After preprocessing, 112,380 news items are stored, with 4,630 users, each day in average with 1,221 news articles.

In order to verify the effectiveness of our proposed news selection strategy, we provide detailed comparison between ours and the general greedy selection strategy simply based on pairwise similarities. Also, we implement a recommender system that models the recommendation as a contextual bandit problem [LCLS10], as the comparison base. For each approach, we randomly select 100 users to provide recommendations for them. We plot the precision and recall pair for each user on top @10, @20, and @30 news items recommended to these users. Figure 3.5 shows the comparison results. From Figure 3.5, we observe that besides the higher precision and recall, the performance distribution of SCENE is more compact than the other methods, which demonstrates the stability of our proposed news selection strategy.

In the above experiments, all the users are equally treated as the experimental subject. In reality, users with different news access patterns, such as different read-

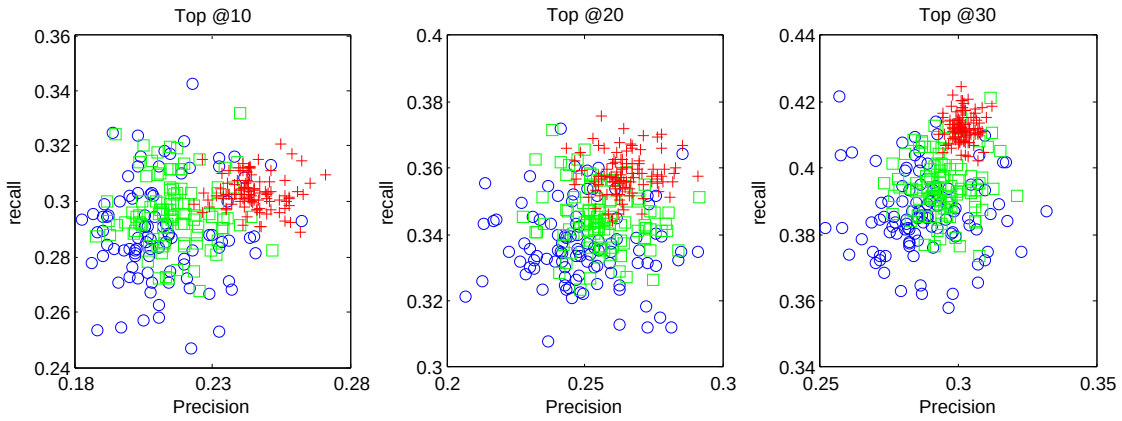


Figure 3.5: Precision-recall plot for different news selection strategies. Remark: ◯ represents users using the general greedy-based recommender system; □ denotes users using the bandit-based recommender system; and + represents users using SCENE.

ing frequency every day, may have distinct patterns of news topic preference, and therefore the dynamic interest on news articles may vary a lot. In addition, many news recommendation systems cannot address the so-called "cold-start" problem. In order to verify the performance of our proposed algorithm on different user groups, we separate the selected users into three groups based on their reading habits. Suppose a user reads N news articles per day, then the three groups are: (i) $N \leq 10$; (ii) $10 < N \leq 50$; (iii) $N > 50$. We apply different algorithms on these three users groups with top @10, top @20 and top @30 recommended news, and record the F1-score respectively. Here, the comparison base includes two existing approaches: [DDGR07] (Goo) and [LDP10] (ClickB). The former is a collaborative filtering based method, whereas the latter is a content-based method. Figure 3.6 shows the comparison results. From the comparison, we observe that our system SCENE can achieve a reasonable recommendation result when it is subject to the "cold-start" problem. The reason is that besides considering similar access patterns when recommending news articles to individual users, we also measure the importance of news content and named entities preferred by news readers.

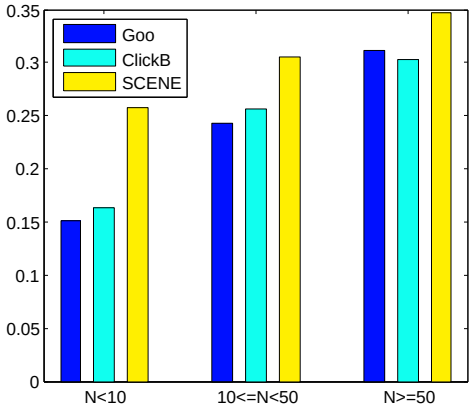


Figure 3.6: Comparison on F-score of different algorithms for three distinct user groups.

To evaluate how diverse our recommendation result is, we compare the set diversity described in [ZH08] between the results of SCENE and other recommender systems. The news set diversity is defined as the *average dissimilarity* of all pairs of news items in the recommendation list, as introduced in Section 2.2. For diversity evaluation, we choose [DDGR07] (Goo), [LDP10] (ClickB), [CP09] (Bilinear) and [LCLS10] (Bandit) as the comparison baselines. We employ the same experiment setup, to compare the diversities of recommendation lists with different cardinalities. Table 3.3 shows the averaged diversity result for 10 time ranges.

Table 3.3: Diversity evaluation on the result list.

Methods	Top @10	Top @20	Top @30
Goo	0.4101	0.3074	0.1105
ClickB	0.4329	0.3128	0.1562
Bilinear	0.4234	0.2517	0.0933
Bandit	0.5056	0.4126	0.2925
SCENE	**0.6930**	**0.6671**	**0.6059**

From the result, we observe that: (i) The diversity decreases as the recommendation news list enlarges. It is straightforward that when more news articles are selected, the topic distribution of the news list becomes closer to the user's reading interest, and therefore the selected news items are more similar. (ii) The diversity

of the recommendation list provided by the baseline methods drops dramatically as the list size increases, since they did not take the diversity into account. (iii) SCENE outperforms the others very significantly, and since we intentionally consider the requirement of news readers, the diversity decreases very smoothly when we recommend more news items to individual users.

3.3.5 Summary of SCENE

In summary, the contribution of this work is three-fold:

- A novel two-level representation: Unlike prior approaches simply providing a list of news items, our system generates a two-level representation, where the first level contains general topics *summarized* from news clusters similar to the user's profile, and the second level includes *representative* items within each cluster. Such a representation can help users easily navigate their preferred articles.
- A principled framework for news selection: We observe that the interestingness of news articles with respect to a user could be *regressive*, and based on this "submodularity" property, we model the news selection problems as a budgeted maximum coverage problem [KMN99], which is more realistic than independently selecting news items. The proposed framework achieves a good balance between the *novelty* and *diversity* of the recommendation result.
- Multi-factor user profile construction: We explore the feasibility of incorporating various properties of news articles – *news content*, *access patterns* and *named entities* – into the construction of user profiles. It is with great benefit of using such enriched profile to capture the exact interest of users.

3.4 Concluding Remarks

This chapter focuses on the problem of understanding user behaviors in recommender systems from both coarse-grained and fine-grained perspectives. In particular, we propose two novel approaches to catch multi-granularity user behaviors:

- In [LPK+13], we introduce a generative graphical model, User-Community-Topic model (UCT), for capturing user-oriented topics and community-oriented topics simultaneously in social media data. Based on the model inference, we further proposed a novel recommendation framework, FRec. Given a user's profile, FRec is able to recommend a list of topic-related influential users or a list of topic-cohesive interactive communities. The proposed framework can be easily extended to the case that recommends users and communities based on a set of keywords. In addition, it can be seamlessly integrated into real-life social networks.
- In [LWL+11], we propose a two-stage approach to tackle personalized news recommendation. We explore the intra relations among news articles, along with different characteristics of news items, including news content, similar access patterns and named entities preferred by users. Our system supports efficient clustering on newly-published news articles, as well as high quality of recommendation results. Extensive evaluation has demonstrated the efficacy and efficiency of SCENE.

These two approaches are able to capture the coarse-grained and fine-grained user behaviors, respectively, and they can be easily extended to other application scenarios. Specifically, FRec is applicable to the scenarios that involve both users and communities, e.g., in social media or research communities. A user's post contributes to both his/her personal preference and the community topic simultaneously, and

hence serves as the basis for inferring the User-Community-Topic model. In this case, the content generated by users is considered as a set of activities without specific order information. Comparatively, SCENE presents a generalized version of modeling user interests, and can be applied to the applications that provide a list of recommended items.

CHAPTER 4

UNDERSTANDING RELATIONS

In this chapter, we start by introducing the research objective of understanding complex relations within recommender systems. To understand complex relations, we focus on using graph-based paradigms to model the relations between users and other objects in a recommender system. Two representative approaches along this direction have been published during my Ph.D. study. In the following, the details of these approaches will be presented, associated with the applicable scenarios for each algorithm.

4.1 Research Objective and Contributions

In most recommender systems, the relation of user $\times$ item is not the unique relation. A myriad of relations might be contained in users' behavioral data. For example, in a news recommender, a user may have some correlations with other users, in a sense that they share similar accessing patterns. The user may also prefer specific topics and named entities rather than accepting all of them. In other domains, such as music recommendation, such a situation also holds. For instance, the songs, associated with artists, tracks, genres, etc., may form a natural taxonomy. These complex relations are valuable for building high-quality user profiles.

In this direction, we comprehensively explore the complex relations within a recommender system using different graph-based methods, based on the types of relations hidden in the recommender system. Specifically,

- In general recommender systems, the relations often involve the binary link between users and items. Such systems could be modeled using bipartite graph, where one set of nodes represent users, and the other set of nodes denote items.

The linkage between user nodes and item nodes typically represents the clicks by users on items. Hence, in [LL12], we model user relations by virtue of a bipartite graph, and reason on various special characteristics on such a graph in a user-to-user recommender system.

- Some recommender systems include more complex relations, where users and items have various properties. For example, in a music recommender system, items (songs) may have a natural taxonomy representing the relations among songs, artists, tracks, albums, etc. Users in such a system may have preference over songs (the lower level of taxonomy) and artists (the higher level of taxonomy). General graphs in which each edge contains only two nodes cannot handle the modeling over such systems. A more natural solution is to use the hypergraph. In [LL13], we formulate the implicit relations in a recommender system using hypergraph models, and conduct graph transductive inference to obtain the recommendation result.

In the following, we will discuss the details of the proposed algorithms, along with the application scenarios.

4.2 Bipartite Modeling with Domain Characteristics

In recent years, a special class of recommender systems – reciprocal recommenders – have emerged, and are tailored for applications that focus on recommending people to people, in which the preferences of both parties involved in the recommendation need to be satisfied. For instance, in an online recruiting system, a job seeker would search jobs that match his/her preference, e.g., the special skills and the salary; and a recruiter might seek suitable candidates to fulfil the job requirement. Other

illustrative examples of reciprocal recommenders include online dating services, online mentoring systems, customer-to-customer marketplaces, etc.

4.2.1 Background and Challenges

Obviously, the major challenge of reciprocal recommender systems is how to satisfy the needs of both users in a recommended match. This requires modeling *the bilateral relations between users* by considering the double-sided preferences. However, simply considering the bilateral relations is *insufficient* in the reciprocal community. In practice, reciprocal recommenders, such as online dating and online recruiting, possess special characteristics differentiating them from traditional user×item recommender systems. Figure 4.1 illustrates some challenges in these systems. We summarize the challenges as follows:

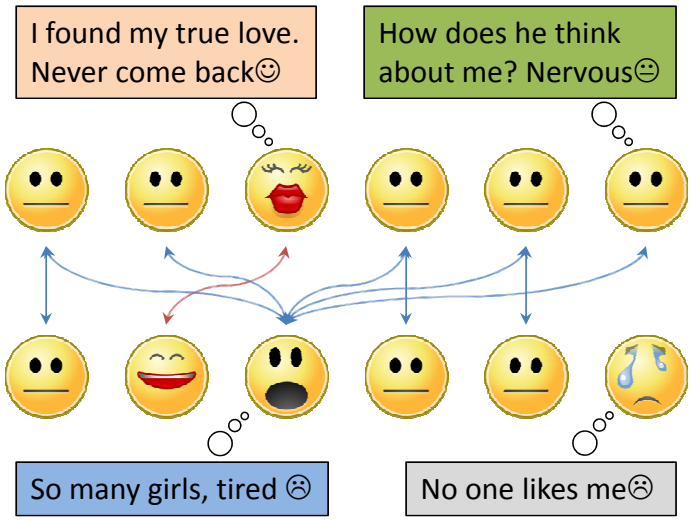


Figure 4.1: A toy example in an online dating system.

- *Reciprocity*: The success of a match *depends on the double-sided preference*, not solely on the user who receives the recommendation. This is the key feature of a reciprocal recommender.

- *Limitedness*: In traditional recommenders, an item can be preferred by a great amount of users. However, in reciprocal recommenders, people have *limited availability* towards other people, e.g., a boy cannot date with ten girls simultaneously.
- *Passiveness*: In reciprocal recommenders, a lot of users with limited engagement activities passively receive messages from other users. To maintain a vibrant community and further attract more users, it is imperative to consider the *passiveness* of users.
- *Sparsity*: Users in reciprocal communities would probably not return to the system if they find their preference. Therefore, different from traditional recommenders where users often have a long consumption history, the *data sparsity* issue in a reciprocal recommender needs more careful consideration.

The aforementioned challenges are essential to a successful reciprocal recommender system. Previous studies either focus on handling the main reciprocity of the recommender, or delve into a specific issue that exists in reciprocal recommenders. Little research work has been proposed to address the challenges in a principled and unified manner. In my previous work [LL12], we model the bilateral relations of users as a bipartite graph that maintains both *local* and *global* utilities in a reciprocal community. The *local* utility captures users' mutual preferences by considering *reciprocity*, *limitedness* and *passiveness*, whereas the *global* utility manages the overall quality of the entire reciprocal network in order to resolve the *sparsity* problem. The bipartite graph is constructed based on the users' self-descriptive and preference features, and then is refined by users' interactive activities.

Given a specialized community represented by a bipartite subgraph, our goal is to recommend for each user an attractive list of users from the other user set. By considering the characteristics of the reciprocal recommendation, we employ the bipartite

graph inference to obtain the recommendation result. Besides the relevance between users, we have the interactive activities, e.g., messages and chatting in online dating, and adding favorite jobs and sending interviews to applicants in online recruiting. We also take into account the availability of the users. We refine the bipartite subgraph based on these information. After refinement, each vertex has its vertex attribute, i.e., the availability, and also has two sets of edges, including the relevance edges and the activity edges. The relevance edge is undirected, and the activity edge is directed, indicating which vertex is the initiator of the activity.

A natural question is why we do not consider the activity information when partitioning the bipartite graph. Within the reciprocal community, not all the users have sufficient activities; if we incorporate the activities into the partitioning process, the generated results might isolate users with few activities from more active users, which may render the recommendation for these inactive users not reasonable.

4.2.2 Bipartite Graph Preliminaries

Formally in our problem setting, a bipartite graph $\mathbb{G}^* = (\mathcal{U}, \mathcal{V}, E_r, E_a, \mathbf{w}_r, \mathbf{w}_a)$ consists of two sets of vertices, $\mathcal{U}$ and $\mathcal{V}$, and two sets of edges, $E_r, E_a \subseteq \mathcal{U} \times \mathcal{V}$. Each edge in E_r is an undirected pair of nodes weighted by $rel(u, v)$, i.e., $\mathbf{w}_r : \mathcal{U} \times \mathcal{V} \to \mathbb{R}_r$. Each edge in E_a is an ordered pair of nodes $[u, v]$ representing the activity connection from u to v, weighted by the ratio of the activities toward the end node and all the activities of the initial node, i.e., $\mathbf{w}_a : \mathcal{U} \times \mathcal{V} \to \mathbb{R}_a$. Given a vertex v in $\mathbb{G}^*$ (either $v \in \mathcal{U}$ or $v \in \mathcal{V}$), the in-degree $p(v)$ and out-degree $q(v)$ are defined as

$$p(v) = \sum_{\{u|[u,v]\in E_a\}} w_a(u, v), \quad q(v) = \sum_{\{u|[v,u]\in E_a\}} w_a(v, u).$$

Let $\mathcal{H}(\mathbb{G}^*)$ denote the space of functions $f : \mathcal{U}, \mathcal{V} \to \mathbb{R}$, which assigns a real value $f(v)$ to each vertex v.

4.2.3 Inference on Bipartite Graph

Given a bipartite graph representing the relevance structure of the reciprocal network, a simple solution to the recommendation is to select top relevant users that directly link to a given user as the recommended result. However, the specific properties (availability and vitality) of users in a reciprocal community would be ignored if we follow such a simple paradigm. For example, a job seeker u has been recommended to multiple job recruiters; if we recommend u to a new job recruiter v, then u will have little chance to respond v, even if they are relevant in some sense. In such a situation, it would be more reasonable to recommend for v other job seekers who are similar to u, which is essentially collaborative filtering. Yet, both directional and non-directional information on the bipartite graph cannot be easily incorporated into traditional collaborative filtering algorithms. A much more natural solution to this problem is to perform graph inference on the bipartite graph to obtain the recommendation list.

Our inference paradigm is motivated by [ZSH05], in which the graph inference is performed on a directed bipartite graph to solve the problem of classification. The problem setting in their method is similar to ours. However, they only consider the directional information within the bipartite graph, i.e., the in-degree and out-degree of nodes, but fail to consider the rationality of the connectivity between nodes, i.e., why the two nodes are connected with each other, which is essential in the problem of reciprocal recommendation. In our work, we explicitly model the rationality of the connectivity of nodes as the relevance between users, by which the connectivity can be naturally explained, and the final recommendation result is more reasonable and explainable.

Formally, if two distinct vertices u_1 and u_2 in $\mathcal{U}$ are co-linked by vertex v in $\mathcal{V}$, it indicates that the properties of both u_1 and u_2 are likely to be similar, e.g., both job seekers are similar in profile-wise since they are all preferred by the same job

recruiter. The co-linkage strength induced by v between u_1 and u_2 can be measured by

$$c_v(u_1, u_2) = w_r(u_1, v) \cdot w_r(u_2, v) \cdot \frac{w_a(v, u_1) w_a(v, u_2)}{q(v)}. \tag{4.1}$$

With such a similarity measure, we not only consider the interactive activities of users ($w_a(v, u_1)$ and $w_a(v, u_2)$), but also emphasize the relevance between users ($w_r(u_1, v)$ and $w_r(u_2, v)$). It can be naturally understood in the context of online dating. If two boys are simultaneously contacted by a girl, then it indicates that both boys have similar characteristics that are preferred by the girl. Moreover, the more girls contact both boys, the more significant the similarity. A natural question arising in this context is why the similarity measure is further normalized by out-degree of v. It can be easily interpreted if we use the previous example. A girl who sends messages to a lot of boys may not have clear preference on what boy characteristics, and therefore the induced similarity of two boys by this girl is not significant.

In Eq.(4.1), we penalize the influence of active users by normalizing the similarity score using $q(v)$. It should be clarified that the vitality of the community cannot be reflected by flooded messages without definite purposes. The way we formalize the similarity considers the importance of dedicated users, e.g., if a girl is interested in a boy, she will not send too many messages to other boys but focus on building the relationship with this boy. Such an observation is beneficial to construct a vibrant reciprocal community.

Let f denote a function defined on one vertex set $\mathcal{U}$. Then the inference cost of function f can be measured by the following functional:

$$\Omega_{\mathcal{U}}(f) = \frac{1}{2} \sum_{u_1, u_2 \in \mathcal{U}} \sum_{v \in \mathcal{V}} \frac{1}{\tau(v)} c_v(u_1, u_2) \left(\frac{f(u_1)}{\sqrt{p(u_1)}} - \frac{f(u_2)}{\sqrt{(p(u_2))}} \right)^2. \tag{4.2}$$

In Eq.(4.2), we penalize large differences in function values for vertices in $\mathcal{U}$. Notice that the function values are normalized by in-degrees of the corresponding vertices.

In the context of online dating, the explanation is similar to the one given before. Many girls will prefer a handsome and successful man, which does not mean that these girls have similar preferences over the characteristics of the man. However, if two girls are sending messages to a boy without handsome appearance and strong background, it is likely to express a common interest of both girls. We also consider the availability of users using the reverse of $\tau(u)$. If the user has been recommended to other users many times within a time range, e.g., one week, then the possibility of this user being recommended to the target user should be small.

Similarly, the inference cost of function f on the vertex set $\mathcal{V}$ can be measured by:

$$\Omega_{\mathcal{V}}(f) = \frac{1}{2} \sum_{v_1, v_2 \in \mathcal{V}} \sum_{u \in \mathcal{U}} \frac{1}{\tau(u)} c_u(v_1, v_2) \left(\frac{f(v_1)}{\sqrt{p(v_1)}} - \frac{f(v_2)}{\sqrt{(p(v_2))}} \right)^2 . \tag{4.3}$$

Convexly combining together the two cost functionals Eq.(4.2) and Eq.(4.3), we can obtain an inference cost measure of function f over the bipartite graph $\mathbb{G}^*$:

$$\Omega_\gamma(f) = \gamma \cdot \Omega_{\mathcal{U}}(f) + (1 - \gamma) \cdot \Omega_{\mathcal{V}}(f), \quad \text{s.t. } 0 \leq \gamma \leq 1, \tag{4.4}$$

where the parameter γ indicates the relative importance between $\Omega_{\mathcal{U}}(f)$ and $\Omega_{\mathcal{V}}(f)$.

4.2.4 Recommendation by Regularization

$\Omega_\gamma(f)$ captures the inference cost of labeling nodes in a bipartite graph. For recommendation, the intuitive idea is to minimize the inference cost, since we want to find the set of users closely relevant yet not recommended to the target user, by making use of the co-linkage of nodes. Besides the inference, we have additional information about users in reciprocal communities, i.e., the interactive activities, which can be regarded as a user's engagement profile for recommendation. Formally, given a user in $u \in \mathcal{U}$, we can define a function y in $\mathcal{H}(\mathbb{G}^*)$ in which $y(v \in \mathcal{V}) = 1$ if vertex v has interaction with u, or 0 if v has never interacted with u. Then the recommendation

problem can be regarded as the problem of finding a function f, which infers new vertices for u while reproducing the target function y to a sufficient degree of accuracy. A formalization of this idea leads to the following optimization problem:

$$f^* = \arg\min_{f \in \mathcal{H}(\mathbb{G}^*)} \{\Omega_\gamma(f) + \mu \|f - y\|^2\}, \tag{4.5}$$

where $\mu > 0$ is the regularization parameter. The first component measures the inference cost of function f, and the second component indicates the closeness of f with respect to the given function y. The trade-off between these two competitive terms is captured by μ. The solution of the optimization problem, Eq.(4.5), can be found in [ZBL+04].

After obtaining the result of $f^*(u)$ for user u, we can take sign$f^*(u)$ to select the vertices in $\mathcal{V}$ whose labels are 1, and then rank the selected users based on the mutual relevance of users, i.e., $rel(u, v)$. The final recommendation result is obtained by selecting the top ranked ones without considering the users who have already interacted with u and the users whose availabilities exceed the availability budget b. Further, if the target user is a vital user, then the recommended list will be ranked via the vitality of users in an ascend order; otherwise, the list will be ranked in a descend order of the vitality. In this way, the engagement of passive users is possible to be improved to some extent.

4.2.5 Discussion

The reciprocal recommendation framework we propose is quite general. In this section, we show the connections and differences between our framework and various existing methods for reciprocal recommendation. The methods discussed in this section include gradient boosted decision trees (GBDTs) [DMAY10], reciprocal recom-

mender for online dating (RECON) [PRC+10] and content-collaborative reciprocal recommender (CCR) [AKY+11].

GBDTs and MEET: GBDTs considers the relevance between the query and the candidate by integrating the matching attributes and post-presentation (activities) features into a unified feature vector. Based on this vector, GBDTs calculates the relevance score between two single users. MEET considers a more general problem – recommendation, in which a user might have no definite preference on the information and therefore no explicit query is specified. Also the post-presentation features might not be available to new users. Moreover, MEET formalizes the reciprocal community as a dynamic network, whereas in GBDTs, the users are treated individually.

RECON and MEET: RECON calculates the compatibility scores between users from different sets by considering the self-description attributes and the activities. However, it fails to consider the other special characteristics of reciprocal recommendation, e.g., the *sparsity*, the *limitedness* and the *passiveness*. In such sense, RECON can be regarded as a special case of our proposed generalized framework.

CCR and MEET: CCR computes the users' similarity based on the content of user profiles, and then performs recommendation from collaborative-wise. It considers the a single-step diffusion of "like" and "dislike" of users towards other users. Such a diffusion is also incorporated into MEET by the inference on the refined bipartite graph. Therefore, CCR can also be treated as a special case of MEET.

4.2.6 Empirical Evaluation

Real-World Data: To evaluate the effectiveness of the proposed method, we use two different real-world datasets, described as follows.

Online Dating Data: This data set is collected from a dating web site from Oct, 2008 to Mar, 2011. We calculate the user relevance based on the new feature space,

and set $\mathcal{U}$ as the male set and $\mathcal{V}$ as the female set. The statistics of this data set is depicted in Table 4.1(a).

Table 4.1: Statistics of two data sets.

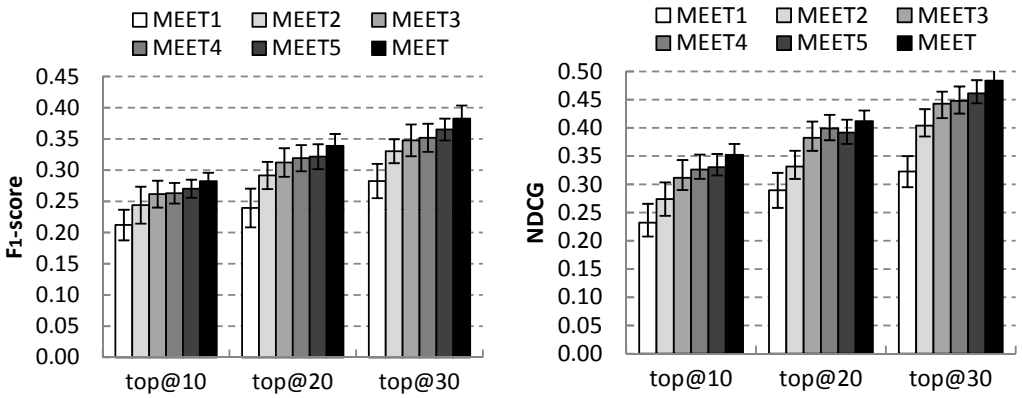
(a) Online Dating		(b) Online Recruiting	
Male (u)	344,552	Job Seekers (u)	199,999
Female (v)	203,843	Recruiters (v)	46,629
# of $\mathcal{F}_u^s$	528	# of $\mathcal{F}_u^s$	860
# of $\mathcal{F}_u^p$	506	# of $\mathcal{F}_u^p$	928
# of $\mathcal{F}_v^s$	506	# of $\mathcal{F}_v^s$	928
# of $\mathcal{F}_v^p$	528	# of $\mathcal{F}_v^p$	860
Activities	8,599,013	Activities	664,943

Online Recruiting Data: We collected the profiles and activities for users of a job searching service web site from Jan, 2008 to Oct, 2011. We set $\mathcal{U}$ as the job seeker set and $\mathcal{V}$ as the recruiter set. We use the same strategy to process this data set. The statistics of the data after processing is described in Table 4.1(b).

Effects of Reciprocal Properties

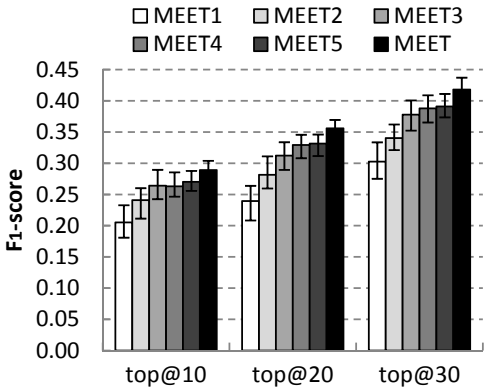
In our generalized framework, we comprehensively consider the special properties of reciprocal recommendation, i.e., *reciprocity*, *limitedness*, *passiveness* and *sparsity*. To examine the influence of different properties on the recommendation results, we evaluate several alternatives of MEET as follows:

- MEET^1: Do not consider *reciprocity*, *passiveness*, *limitedness* and *sparsity*, i.e., to drop the term $\frac{1}{\tau(v)}c_v(u_1, u_2)$ in Eq.(4.2) and Eq.(4.3) and do not perform feature weight learning process;
- MEET^2: In Eq.(4.2) and Eq.(4.3), only consider the relevance between users, i.e., to drop the term $\frac{1}{\tau(v)}$ and the term $\frac{w_a(v,u_1)w_a(v,u_2)}{q(v)}$ in Eq.(4.1);
- MEET^3: In Eq.(4.1), do not consider the passiveness of users, i.e., to drop the term $\frac{w_a(v,u_1)w_a(v,u_2)}{q(v)}$;

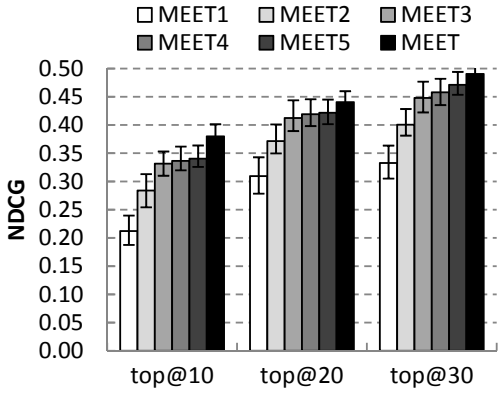


(a) F_1 on online dating.

(b) NDCG on online dating.



(c) F_1 on recruiting.



(d) NDCG on recruiting.

Figure 4.2: Performance comparison of different alternatives of MEET.

- MEET^4: In Eq.(4.2) and Eq.(4.3), do not consider the availability of users, i.e., to drop the term $\frac{1}{\tau(v)}$;
- MEET^5: Do not perform the feature weight learning process.

We compare these alternatives with the comprehensive MEET in terms of F_1-score and NDCG. Figure 4.2 shows the comparison results on two datasets.

It is evident that the generalized model MEET significantly outperforms the alternatives from both accuracy and ranking perspective. The reason behind this is quite straightforward: in the generalized model, the special characteristics of reciprocal community are well captured, rendering the recommendation results derived from such unified model more reasonable. Besides this, we observe that: (1) The *reciprocity*

is the dominant aspect in the reciprocal network, since only considering the relevance between users can significantly improve the quality of the recommendation results; and (2) The *limitedness*, *passiveness* and *sparsity* are also important properties of the reciprocal community, by which the performance of MEET can achieve slight improvement.

Comparison with Existing Methods

Our proposed framework is designated to reciprocal recommendation, which cannot be easily tackled by traditional collaborative filtering approaches. To verify this claim, we choose two recently published collaborative filtering methods [HKV08, LHZC10] as our baselines. [HKV08] (CFIF for short) proposed treating the data as indication of positive and negative preference associated with vastly varying confidence levels, which is a pure collaborative filtering approach. [LHZC10] (OCCF for short) exploited the rich user information available in community-based interactive information systems, and incorporated user information into modeling the recommendation. For this method, we use the neighborhood model as the baseline. We also implement GBDTs [DMAY10], RECON [PRC$^+$10] and CCR [AKY$^+$11] for comparison. We use F_1-score and NDCG to compare these algorithms with MEET for both online dating and online recruiting datasets. The feature set used in the baselines are identical to the one in our proposed method, and also the parameters in the baselines are optimally tuned.

The results are shown in Table 4.2. It is evident that MEET significantly outperforms the baselines on both F_1-score and NDCG. The two collaborative filtering based methods cannot effectively handle the reciprocal task. We investigated the recommendation results of both methods and found that users in most recommended matches are relevant. However, there are two reasons that both users in a match have

few or even no interactions: (1) The recommended user has been recommended to multiple users, and therefore he/she has limited availability; and (2) Both users are not vital, and hence they do not contact with each other. The three reciprocal methods being compared can slightly improve the recommendation performance; however, they only focus on different aspects of the reciprocal community. Instead, MEET provides a comprehensive overview of the reciprocal network, and therefore achieves the best.

Table 4.2: Comparison with existing methods. (The bold font indicates the best performance. * indicates the statistical significance at $p < 0.01$.)

Methods	Online Dating						online recruiting					
	top@10		top@20		top@30		top@10		top@20		top@30	
	F_1	NDCG	F_1	NDCG	F_1	NDCG	F_1	NDCG	F_1	NDCG	F_1	NDCG
CFIF	0.2307	0.3069	0.2918	0.3534	0.3206	0.4417	0.2301	0.3174	0.3121	0.3813	0.3481	0.4036
OCCF	0.2415	0.3134	0.3059	0.3670	0.3385	0.4498	0.2485	0.3320	0.3219	0.3929	0.3569	0.4127
GBDTs	0.2607	0.3146	0.3106	0.3881	0.3475	0.4662	0.2567	0.3592	0.3304	0.4131	0.3718	0.4432
RECON	0.2523	0.3221	0.3027	0.3672	0.3503	0.4530	0.2604	0.3608	0.3247	0.4025	0.3839	0.4507
CCR	0.2309	0.3250	0.3389	0.3724	0.3428	0.4621	0.2431	0.3745	**0.3573**	0.3987	0.3912	0.4729
MEET	**0.2823***	**0.3521***	0.3390	**0.4118***	**0.3826***	**0.4836***	**0.2890***	**0.3799**	0.3560	**0.4405***	**0.4181***	**0.4904***

Vitality Evaluation

The vitality of a user is an important feature within the reciprocal community. It defines how active the user is, e.g., how often the user sends messages to other users. By explicitly considering the vitality for recommendation, a vital user improve the engagement of other passive users, which renders the reciprocal network more healthy and energetic. To measure how active that users within the recommended list are, we define the *set vitality* measurement as the *average activeness* of all the users in the list, as introduced in Section 2.2.

The recommended user list provided by MEET exhibits a great vitality . Such vitality is resulted from the sparkle that we intentionally consider the *passiveness* of users in reciprocal community. We assume that the passive user can be spurred by the active user, and formalize the activeness of users in Eq.(4.1). Such activeness is

beneficial to construct an energetic reciprocal network, in which users are willing to proactively contact with other users, and therefore improve the vitality of the entire network.

To evaluate the vitality of the recommended results, we use the *set vitality* measurement defined in Eq.(2.11), and compare MEET with GBDTs, RECON and CCR. These three methods consider the interactive activities of users from different perspectives. We also compare MEET with an alternative MEET^3 that does not consider the passiveness of users. Note that the recommended list are ranked based on the activeness of users. The ranking quality of the recommended list has been verified in the previous section, and therefore we put our concern on the overall vitality of the list. We report the comparison results in Table 4.3.

Table 4.3: Evaluation on the set vitality of the recommended results. (The bold font indicates the best performance. * indicates the statistical significance at $p < 0.01$.)

	top@10	top@20	top@30
GBDTs	0.3513	0.3306	0.3027
RECON	0.3629	0.3430	0.3085
CCR	0.3731	0.3325	0.2964
MEET^3	0.3515	0.3276	0.2909
MEET	**0.4639***	**0.4517***	**0.4425***

From the comparison, we observe that the set vitality of results from different methods varies a lot. Since our framework explicitly formalizes the interaction between users into Eq.(4.1), it achieves the best performance. An interesting phenomenon is that when the number of recommended results increases, most methods show a decreasing trend in terms of the *set vitality*. Take CCR as an example for further analysis. CCR generates the results by considering the users who have interactions with users that are similar to the target user (based on the profile). The user ranking is based on the reciprocal interests. When recommending more users to the target user, the reciprocal interests of users with lower rankings will decrease

significantly, and therefore the set vitality of the recommended list deceases. Comparatively, in our framework, we prefer dedicated users, i.e., the users who have a lot of interactions with several other users but do not send flooded messages. Based on this intuition, MEET outperforms other candidates.

4.2.7 Summary of MEET

In this work, we study the problem of reciprocal recommendation. We comprehensively investigate the special properties of a reciprocal community, including reciprocity, limitedness, passiveness and sparsity. We propose a generalized reciprocal framework, MEET, in which the aforementioned properties are seamlessly integrated. Specifically, MEET first constructs a bipartite graph based on the mutual relevance of users, and then performs graph inference on the resulted subgraphs to obtain the recommendation list for individuals. The inference model formalizes the properties of the reciprocal network and elegantly casts the recommendation as an optimization problem.

4.3 Complex Relations in a Recommender

A key step in news recommender systems is to build the readers' preference profiles based on their historical consumption, i.e., the reading history. Traditionally, user profiling is conducted by extracting representative elements (e.g., words or phrases) from the reading history or selecting similar access patterns. However, users' historical consumption may contain a gigantic amount of element correlations, e.g., a group of users like the same topic, which cannot be well captured by the aforementioned profiling paradigms.

Further, online readers tend to prefer some named entities in news articles, e.g., when the event happened, where it happened, who were involved, etc. These types of entities can attract online readers' interest since they contain concise information about the news article itself. Therefore, named entities would be valuable to model users' preferences. However, few research efforts have been reported on utilizing named entities for user profiling. In [GDH04] and [LWL+11], named entities extracted from news articles are represented as an entity vector, and then the similarity based on such a vector is calculated for retrieving relevant news items. Such a representation might cause the information loss of news access data, e.g., what type of entities is preferred by a group of users, and therefore render user profiling less effective.

In [LL13], to address the aforementioned issues, we propose a novel news personalization algorithm by mining the implicit relations among users, news articles, topics and named entities. Motivated by [BTC+10], we use a unified hypergraph to model multi-type objects and implicit relations in news reading community. A hypergraph is a generalization of the ordinary graph, in which the edges, called *hyperedges*, are arbitrary non-empty subsets of the vertex set [ABB06]. However, due to the special properties of news articles (e.g., textual content, implicit relation and short shelf life), a straightforward extension of hypergraph modeling on music community cannot be directly applied to news recommendation. Instead, we first partition the hypergraph into multiple fine-grained ones, and further model personalized news recommendation as a ranking problem on fine-grained hypergraphs to recommend news articles.

4.3.1 Hypergraph Preliminaries

We follow the definitions in [BTC+10, ZHS07] to describe hypergraph preliminaries. We denote $\mathbb{G}(V, E, w)$ as a hypergraph, where V is a finite set of vertices, E is a family of hyperedges on V, and w is a weight function, $w : E \to \mathbb{R}$. Each hyperedge $e \in E$

contains a list of vertices that belong to V. The degree of a hyperedge e is defined by $\delta(e) = |e|$, i.e., the number of vertices in e. The degree $d(v)$ of a vertex v is defined by $d(v) = \sum_{v \in e} w(e)$, where $w(e)$ is the weight of the hyperedge e. We say that there is a *hyperpath* between vertices v_1 and v_k if there is an alternative sequence of distinct vertices and hyperedges $v_1, e_1, v_2, e_2, \cdots, e_{k-1}, v_k$, such that $\{v_i, v_{i+1}\} \subseteq e_i$ for $1 \leq i \leq k-1$. We formulate a vertex-hyperedge incidence matrix $\mathbf{H} \in \mathbb{R}^{|V| \times |E|}$ in which each entry $h(v, e)$ is 1 if $v \in e$ and 0 otherwise. Then we have $d(v) = \sum_{e \in E} w(e)h(v, e)$, and $\delta(e) = \sum_{v \in V} h(v, e)$. Let $\mathbf{D}_v$ and $\mathbf{D}_e$ denote the diagonal matrices containing the vertex and hyperedge degrees respectively, and $\mathbf{W}$ the diagonal matrix $|E| \times |E|$ containing the weights of hyperedges.

Data Model

In news reading community, multiple types of resources are often available for analysis, including users, news articles, representative terms, named entities, etc. Let $\mathbb{U}$ denote the user pool, and $\mathbb{N}$ denote the set of news articles, both of which are the major elements being considered in recommender systems. Further, let $\mathbb{T}^t$ denote the representative terms, or say, topics, and $\mathbb{T}^e$ be the set of named entities involved in the entire news corpus. Notice that in reality, the pools of these four types of resources would be enlarged as there are always news events happening everyday with different topic categories and distinct named entities. In our data model, we name these resources as *Media Objects*. To facilitate the reading, we list some notations in Table 4.4.

Besides, several relations among objects are implicitly embedded among *media objects*. For example, two users $u_1, u_2 \in \mathbb{U}$ are the fans of NBA star *LeBron James* ($t_i^e \in \mathbb{T}^e$), which is a named entity appearing in news articles of sports event ($t_j^t \in \mathbb{T}^t$). Then there is an implicit relation among these media objects. In our data model, we

Table 4.4: Notations in our data model.

$\mathbb{U}$	the user set.	u_i	a particular user.
$\mathbb{N}$	the article set.	n_i	a particular article.
$\mathbb{T}^t$	the topic set.	t_k^t	the k-th topic.
$\mathbb{T}^e$	the entity set.	t_k^e	the k-th entity.
n_i^k	the k nearest neighbors of an article i.		
α	the importance factor of content similarities.		
$E^{\mathbb{UNT}^t}$	the set of user-article-topic hyperedges.		
$E^{\mathbb{UNT}^e}$	the set of user-article-entity hyperedges.		
$E^{\mathbb{UUN}}$	the set of user-user-article hyperedges.		
$E^{\mathbb{UUT}^t}$	the set of user-user-topic hyperedges.		
$E^{\mathbb{UUT}^e}$	the set of user-user-entity hyperedges.		
$E^{\mathbb{NNT}^t}$	the set of article-article-topic hyperedges.		
$E^{\mathbb{NNT}^e}$	the set of article-article-entity hyperedges.		
$E^{\mathbb{N}^k}$	the set of k-nearest-articles hyperedges.		

formalize a hypergraph $\mathbb{G}$ that contains 7 different implicit relations with different objects and 1 implicit relation that considers the similarity graph of news articles. It is natural that the edges can be generalized from a pairwise co-occurrence, e.g., an edge incident on a news item and all of its readers. However, the generated incidence matrix would become much denser. For simplicity, we only consider the hyperedges with three vertices. Our algorithm can be extended to hyperedges with arbitrary number of vertices. Specifically, the edges contain:

- $E^{\mathbb{UNT}^t}$: A user reads a news article that describes an event, or a topic. Typically we assign the weight of this hyperedge to be 1. Here we assume that a user would only navigate a news item once.
- $E^{\mathbb{UNT}^e}$: A user reads a news article that embraces a named entity. Similar to $E^{\mathbb{UNT}^t}$, we assign the hyperedge weight as 1.
- $E^{\mathbb{UUN}}$: Two users might read the same news article. We assign the hyperedge weight to be 1.

- $E^{\mathbb{U}\mathbb{U}\mathbb{T}^t}$: Two users might read news articles with the same topic. The weight $w(e^{u_i u_j t_k^t})$ for this relation is set to be the frequency that both users, u_i and u_j, read articles with the same topic t_k^t, i.e.,

$$w(e^{u_i u_j t_k^t}) = |\{(u_i, u_j, t_k^t) | u_i \in \mathbb{U}, u_j \in \mathbb{U}, t_k^t \in \mathbb{T}^t\}|. \tag{4.6}$$

We normalize the weight as

$$w(e^{u_i u_j t_k^t}) = \frac{w(e^{u_i u_j t_k^t})}{\sqrt{\sum_{l=1}^{|\mathbb{U}|} w(e^{u_l u_j t_k^t})} \sqrt{\sum_{m=1}^{|\mathbb{U}|} w(e^{u_i u_m t_k^t})}}. \tag{4.7}$$

The above heuristic normalization aims to penalize the abnormal news readers with dense reading activities. Moreover, in order to treat different types of relations equally, we further normalize the weight as follows:

$$w(e^{u_i u_j t_k^t}) = \frac{w(e^{u_i u_j t_k^t})}{ave(w(e^{u_i u_j \mathbb{T}^t}))}, \tag{4.8}$$

where $ave(w(e^{u_i u_j \mathbb{T}^t}))$ is the average of normalized weights for users u_i and u_j on different topics.

- $E^{\mathbb{U}\mathbb{U}\mathbb{T}^e}$: Two users might read news articles containing the same named entity. The weight $w(e^{u_i u_j t_k^e})$ for this relation is set to be the frequency that both users, u_i and u_j, read articles containing the same entity t_k^e. The weight normalization is similar to Eq.(4.7) and Eq.(4.8).

- $E^{\mathbb{N}\mathbb{N}\mathbb{T}^t}$: Two news articles might describe the same or similar topic. We assign the hyperedge weight as 1.

- $E^{\mathbb{N}\mathbb{N}\mathbb{T}^e}$: Two news articles might contain the same entity. We assign the hyperedge weight to be 1.

- $E^{\mathbb{N}^k}$: In our data model, we also consider the similarity of news articles. We construct a k nearest neighbor (k-NN) news graph based on content-based item similarities. In the hypergraph, a hyperedge of this type is composed of the top

k articles similar to the target news item and the target item itself. The weight $w(e^{n_i^k})$ is the averaged similarity between the target news item and the ones similar to the target, i.e.,

$$w(e^{n_i^k}) = \frac{1}{k}\sum_{j=1}^{k} sim(n_i, n_j), \tag{4.9}$$

where $sim(n_i, n_j)$ is the similarity between two news articles. In our work, this similarity is calculated using the cosine similarity by considering the content features of news items. We introduce a parameter α to control the relative importance of content-based similarities in the unified hypergraph model. Then, the hyperedge weight is given by

$$w(e^{n_i^k}) = \alpha * w(e^{n_i^k}). \tag{4.10}$$

Finally, the unified hypergraph of news reading community is composed of 4 types of media objects as vertices and 8 types of object relations as hyperedges. Figure 4.3 summarizes the aforementioned media objects and relations. By employing the unified hypergraph model, we can effectively capture the high-order relations among various types of media objects without loss of any important information.

Based on the data model introduced above, we can derive the vertex-hyperedge incidence matrix **H** (as described in Table 4.5) and also the weight matrix **W**. The size of both matrices depends on the cardinality of different element sets involved in the matrices, and they are all sparse matrices.

Table 4.5: The incidence matrix H of the unified hypergraph.

	$E^{\mathbb{UNT}^t}$	$E^{\mathbb{UNT}^e}$	$E^{\mathbb{UUN}}$	$E^{\mathbb{UUT}^t}$	$E^{\mathbb{UUT}^e}$	$E^{\mathbb{NNT}^t}$	$E^{\mathbb{NNT}^e}$	$E^{\mathbb{N}^k}$
$\mathbb{U}$	$\mathbb{U}E^{\mathbb{UNT}^t}$	$\mathbb{U}E^{\mathbb{UNT}^e}$	$\mathbb{U}E^{\mathbb{UUN}}$	$\mathbb{U}E^{\mathbb{UUT}^t}$	$\mathbb{U}E^{\mathbb{UUT}^e}$	**0**	**0**	**0**
$\mathbb{N}$	$\mathbb{N}E^{\mathbb{UNT}^t}$	$\mathbb{N}E^{\mathbb{UNT}^e}$	$\mathbb{N}E^{\mathbb{UUN}}$	**0**	**0**	$\mathbb{N}E^{\mathbb{NNT}^t}$	$\mathbb{N}E^{\mathbb{NNT}^e}$	$\mathbb{N}E^{\mathbb{N}^k}$
$\mathbb{T}^t$	$\mathbb{T}^tE^{\mathbb{UNT}^t}$	**0**	**0**	$\mathbb{T}^tE^{\mathbb{UUT}^t}$	**0**	$\mathbb{T}^tE^{\mathbb{NNT}^t}$	**0**	**0**
$\mathbb{T}^e$	**0**	$\mathbb{T}^eE^{\mathbb{UNT}^e}$	**0**	**0**	$\mathbb{T}^eE^{\mathbb{UUT}^e}$	**0**	$\mathbb{T}^eE^{\mathbb{NNT}^e}$	**0**

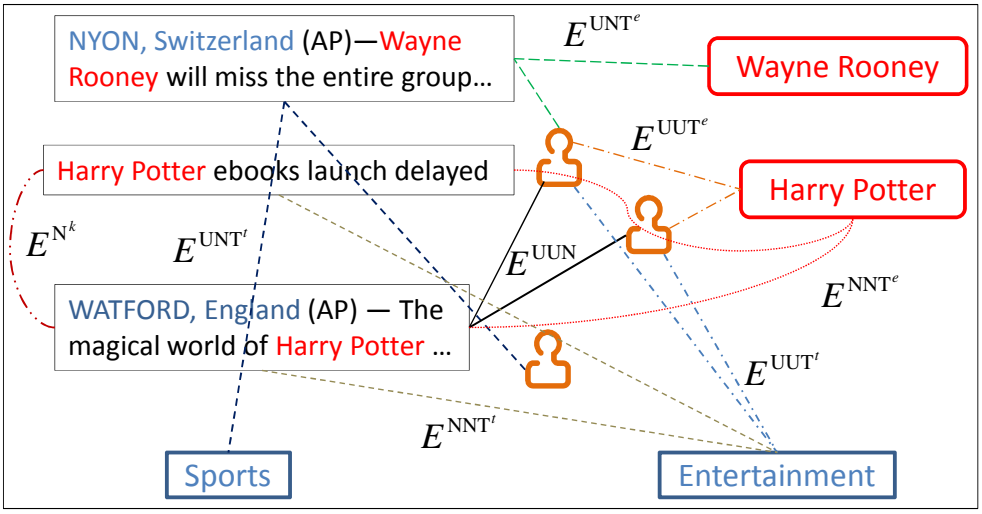


Figure 4.3: An illustrative example of data model in news reading community.

4.3.2 Recommendation via Hypergraph Inference

Hypergraph partitioning provides us a list of disjoint news capsules, in which the users' reading preferences are encapsulated, including topics, named entities and similar users. In the following, we present our approach in which we model the recommendation as a sub-hypergraph ranking problem.

Formally, given a capsule (or a sub-hypergraph) $\mathcal{C}$, a set of newly-published news articles $\mathcal{S}$ and a target user u within $\mathcal{C}$, we first link articles in $\mathcal{S}$ onto $\mathcal{C}$. To do so, we extract topics and named entities from $\mathcal{S}$ and then compare these objects with the objects in $\mathcal{C}$. In this way, we can not only connect new articles to $\mathcal{C}$, but also add new objects into $\mathcal{C}$. Next, we reconstruct the unified hypergraph based on the enriched $\mathcal{C}$ and get the vertex-hyperedge incidence matrix $\mathbf{H}^{\mathcal{C}}$ and the weight matrix $\mathbf{W}^{\mathcal{C}}$. Since the scale of $\mathcal{C}$ is supposed to be much smaller than the entire hypergraph, the reconstruction would be more efficient. Then the vertex degree matrix $\mathbf{D}_v^{\mathcal{C}}$ and the hyperedge degree matrix $\mathbf{D}_e^{\mathcal{C}}$ are computed based on $\mathbf{H}^{\mathcal{C}}$ and $\mathbf{W}^{\mathcal{C}}$. In the following, we discuss how to perform ranking on a sub-hypergraph by using similar idea of [BTC+10].

We define the cost function of $\mathbf{f}$ as follows:

$$\begin{aligned} Q(\mathbf{f}) =& \frac{1}{2}\sum_{i,j=1}^{|V|}\sum_{e\in E}\frac{1}{\delta(e)}\sum_{\{v_i,v_j\}\subseteq e} w(e)\left\|\frac{f_i}{\sqrt{d(v_i)}}-\frac{f_j}{\sqrt{d(v_j)}}\right\|^2 \\ &+\mu\sum_{i=1}^{|V|}||f_i-y_i||^2, \end{aligned} \tag{4.11}$$

where $\mu > 0$ is the regularization factor. To achieve the optimal ranking result, we need to minimize $Q(\mathbf{f})$:

$$\mathbf{f}^* = \arg\min_{\mathbf{f}} Q(\mathbf{f}). \tag{4.12}$$

For inference, we need to smooth the process as much as possible under the constraint that vertices that are contained by many common hyperedges should have similar ranking scores. As an illustrative example, if two news articles have been accessed by many common users, then both articles will have similar ranking scores. The smoothness can be achieved by minimizing the first term in Eq.(4.11). We also need to minimize the difference between the obtained ranking scores and the pre-given scores to guarantee that the result will not deviate much from the truth, i.e., to minimize the second term in Eq.(4.11). After a series of mathematical derivation by [BTC+10], we can obtain the optimal $\mathbf{f}^*$ as

$$\mathbf{f}^* = (\mathbf{I}-\gamma\mathbf{A})^{-1}\mathbf{y}, \tag{4.13}$$

where $\mathbf{A} = (\mathbf{D}_v^{\mathcal{C}})^{-1/2}\mathbf{H}^{\mathcal{C}}\mathbf{W}^{\mathcal{C}}(\mathbf{D}_e^{\mathcal{C}})^{-1}(\mathbf{H}^{\mathcal{C}})^T(\mathbf{D}_v^{\mathcal{C}})^{-1/2}$. Notice that under the constraint of $\mathcal{C}$, the matrix $\mathbf{I}-\gamma\mathbf{A}$ is highly sparse, and therefore the inverse of $\mathbf{I}-\gamma\mathbf{A}$ can be efficiently calculated. $\mathbf{y}$ corresponds a query vector given a user, each entry of which can be either 1 or 0, indicating what topics and entities are preferred by the user. After performing ranking on the sub-hypergraph, we can choose top ranked news articles as the recommendation list.

Discussion: An interesting setup to approximately satisfy the *diversity* requirement is to predefine query scores over different topics $\mathbb{T}^t$ and named entities $\mathbb{T}^e$ within

the profile of the target user. Specifically, we can analyze the user's profile and choose the topics and named entities that are ranked high in terms of the accumulated score of the edge weights. For these topics and named entities, the corresponding values in $\mathbf{y}$ can be set to 1 for the query purpose, whereas for other topics and named entities, the values can be set to 0. For example, given a user q with preference over the topic "Basketball" and the named entity "LeBron James", we can specify 1 value for the corresponding two entries in $\mathbf{y}_q$. In addition, we can set 1 value for entries that are related to the user's preference, e.g., "NBA" and "Competitive Sports", which can be obtained using some simple text mining techniques. In this way, the ranking result can have more diverse content in terms of topics and named entities that are distributed over the target user's profile and related preferences. Therefore the setup of the query vector $\mathbf{y}$ fosters the diversity in the results.

4.3.3 Transductive Inference on Hypergraph

Our proposed framework is capable of handling the so-called *cold-start* problem, especially for new users. In this subsection, we introduce the strategy of how we can tackle the user cold-start problem. Given a new user q without enough reading history, traditional recommender systems fail to construct a comprehensive user profile due to the data sparsity. Comparatively in our framework, we initially embrace this new user q into a specific capsule (by extracting topics and named entities of the limited consumption history of q and linking them to the capsule). Taking q into the construction of the query vector $\mathbf{y}$, we perform transductive inference on the new capsule to derive the vertices related to q's preference, and finally provide the recommendation list for q.

Specifically, given a news capsule $\mathcal{C} = (V^{\mathcal{C}}, E^{\mathcal{C}}, w^{\mathcal{C}})$, in which the vertices in a subset $S \subset V^{\mathcal{C}}$ have labels in $L = \{1, -1\}$ predefined by the new user q according

to q's reading history, our goal is to predict the labels of the remaining unlabeled vertices. Note that for news recommendation, the labels $\{1, -1\}$ indicate whether the user is interested in the corresponding element or not. On the one hand, the inference function should be as *smooth* as possible, i.e., we should assign the same label to all vertices contained in the same hyperedge; moreover, vertices lying on a densely linked sub-hypergraph are likely to have the same label. Thus we define a function

$$\Omega(\mathbf{f}) = \frac{1}{2} \sum_{e \in E} \sum_{\{u,v\} \subseteq e} \frac{w(e)}{\delta(e)} \left\| \frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right\|^2, \tag{4.14}$$

which sums the weighted variation of a function on each edge of the capsule. On the other hand, the initial label assignment should be changed as little as possible. Let $\mathbf{y}$ denote the initial label vector, in which the assignments are defined by $y(v) = 1$ or -1 if vertex v has been labeled as positive or negative respectively, and 0 if it is unlabeled. Then we consider the following optimization problem [ZHS05]

$$\arg\min_{\mathbf{f} \in \mathbb{R}^{|V|}} \{\Omega(\mathbf{f}) + \mu ||\mathbf{f} - \mathbf{y}||^2\}, \tag{4.15}$$

where $\mu > 0$ is the parameter specifying the tradeoff between the two components. The optimization problem defined in Eq.(4.15) is similar to the one in Eq.(4.11). The difference here is that for transudctive inference, our goal is to derive the labels of the unlabeled vertices, whereas for ranking, we try to derive the complete importance order of the vertices. Due to the space limit, we omit the detailed procedure for sovling Eq.(4.15). In this way, even if a user does not have enough reading history, we can still get enough labeled news items that the user might prefer. Hereby, for new users, we can recommend a list of unordered news articles.

4.3.4 Empirical Evaluation

Real-World Data

The data used in our experiment is obtained from multiple news reading portals, ranging from Aug 15th, 2010 to Nov 16th, 2010 [LWL+11], which includes news articles and users' access histories. It contains 10 news topic categories, such as sports, movies, politics, etc. We preprocess the data by removing news articles that are rarely accessed (i.e., the accessed frequency is less than 1 per day) and by storing users with frequent online reading behaviors (i.e., users who read news articles everyday and read more than 1 piece of news each day). By preprocessing, some unexpected noise can be removed to ensure the quality of the generated hypergraph. We perform LDA on news articles to extract representative words from each news category, as the topics of the data model. Here each "topic" is represented as a bag of words when constructing the unified hypergraph. For named entities, we use NLP tools, e.g., GATE [CMBT02], to perform information extraction. The media objects and relations contained in this data collection are summarized in Table 4.6(a) and 4.6(b), respectively. Note that the number of nearest neighbors, k, in the news similarity graph is not fixed, and therefore the number of hyperedges varies for $E^{\mathbb{N}^k}$.

Table 4.6: Statistics of our news collection.

(a) Objects

Elements	Count
Users	3,280
Articles	58,873
Topics	10
Entities	121,617

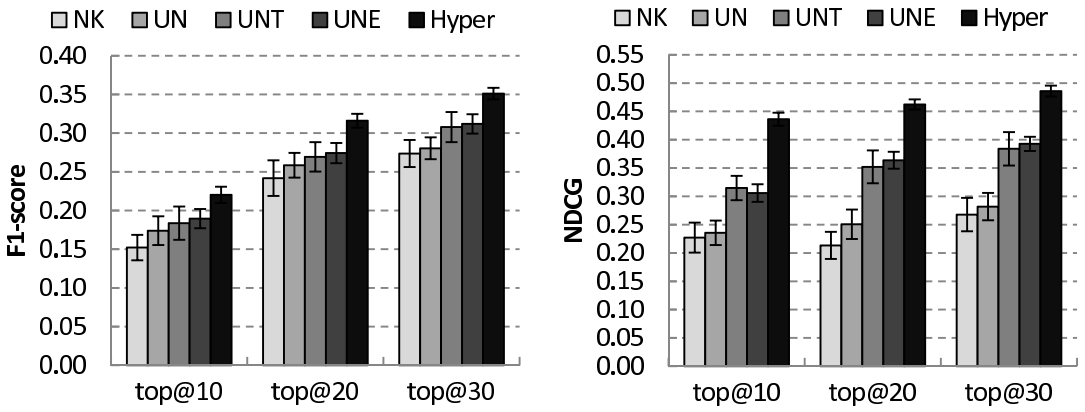
(b) Relations

Relations	Count	Relations	Count
$E^{\mathbb{UNT}^t}$	501,239	$E^{\mathbb{UUT}^e}$	402,918
$E^{\mathbb{UNT}^e}$	672,348	$E^{\mathbb{NNT}^t}$	52,136
$E^{\mathbb{UUN}}$	307,652	$E^{\mathbb{NNT}^e}$	176,431
$E^{\mathbb{UUT}^t}$	43,785	$E^{\mathbb{N}^k}$	-

Construction of Hypergraph

In our proposed data model, we integrate 8 different hyperedges or object relations into the construction of news hypergraph. Such a hypergraph provides an elegant representation of the news data, encapsulating multiple correlations among different media objects. To evaluate the effect of such a representation in personalized news recommendation, we consider different combinations of hyperedges for hypergraph construction: (1) $E^{\mathbb{N}^k}$ (NK): consider news similarities based on users' profiles, which is essentially a content-based method; (2) $E^{\mathbb{UUN}}$ (UN): construct the hypergraph using only users' co-visit information, which can be regarded as an example of collaborative filtering; (3) $E^{\mathbb{UNT}^t}$, $E^{\mathbb{UUT}^t}$ and $E^{\mathbb{NNT}^t}$ (UNT): build hypergraph by considering the topics in news collection and users' access patterns in a hybrid way; and (4) $E^{\mathbb{UNT}^e}$, $E^{\mathbb{UUT}^e}$, $E^{\mathbb{NNT}^e}$ (UNE): construct the hypergraph using the entity information, i.e., considering users' preference on specific named entities.

We compare NDCG, and the results are shown in Figure 4.4, in which Figure 4.4(a) demonstrates the performance difference of multiple hypergraph constructions in terms of F1-score, whereas Figure 4.4(b) shows the ranking performance in terms of NDCG.



(a) F1-score comparison. (b) NDCG comparison

Figure 4.4: Performance comparison of different hypergraph constructions.

It is evident that the unified hypergraph model significantly outperforms other constructions of hypergraph from both accuracy and ranking perspective. The reason behind this is straightforward: in our unified model, high-order correlations among different media objects are well-captured, which extensively enrich a user's reading preference and hence make the recommended result more accurate. Besides this, we observe that: (i) The performance of hybrid constructions (i.e., UNT and UNE) is better than uni-edge construction (i.e., NK and UN); and (ii) the result of UNE is comparable with UNT, which means that in real-world news recommender systems, users pay equal or more attention on named entities they prefer, not just the topics that news articles are reporting.

Comparison with Other Methods

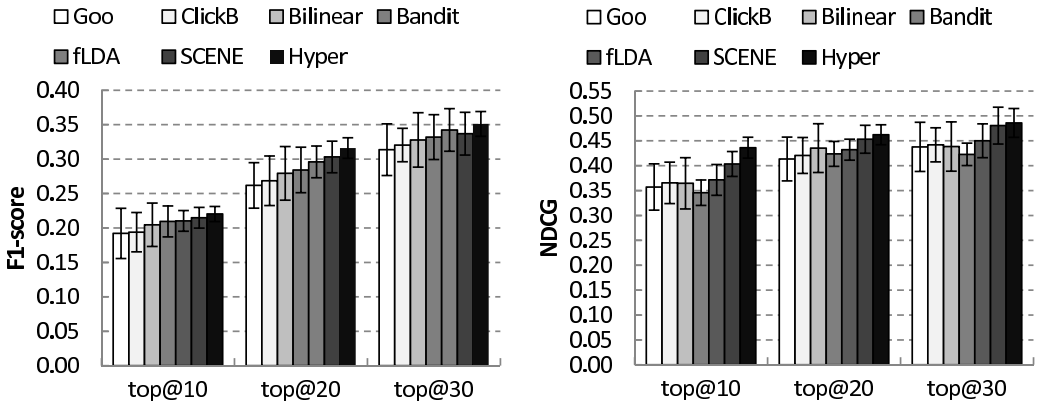
We also implement several recently published methods: Goo [DDGR07], ClickB [LDP10], Bilinear [CP09], Bandit [LCLS10], fLDA [AC10], and SCENE [LWL+11] for comparison. The details of these algorithms are described as follows:

- Goo: The method is essentially a collaborative filtering approach, in which MinHash clustering, PLSI and covisitation counts are taken into account for a unified recommendation paradigm.
- ClickB: In this approach, the profiles of user's news interests are built based on their past click behavior, and then a Bayesian framework for predicting users' current news interests.
- Bilinear: This method maintains profiles of content of interest based on temporal characteristics of the content, e.g., popularity and freshness, and also maintains profiles of users including historical activities. The recommendation is achieved via a feature-based machine learning approach.

- Bandit: The method models recommendation as a contextual bandit problem, in which a learning algorithm sequentially selects articles to serve users based on contextual information about users and articles, while simultaneously adapting its selection strategy based on user-click feedback.
- fLDA: The method regularizes both user and item factors simultaneously through user features and the bag of words associated with each item. It is essentially a hybrid filtering method.
- SCENE: The method assumes the interestingness of news articles with respect to a user could be regressive, and uses the "submodularity" property to model the news selection problem.

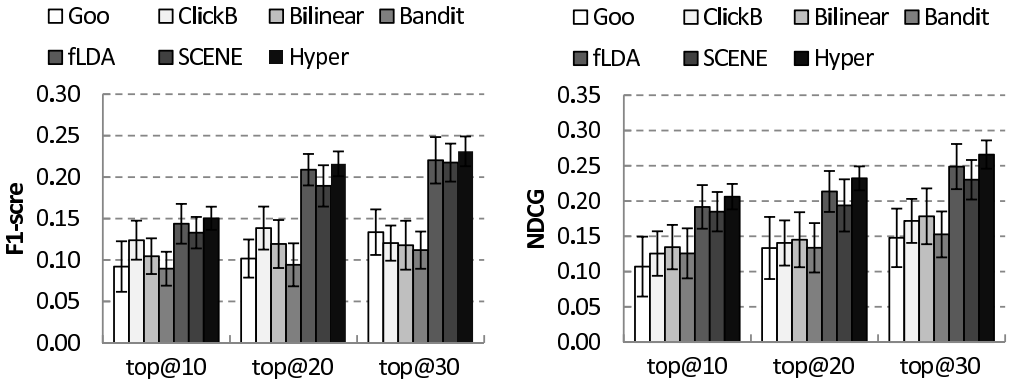
Note that the parameters of these baseline methods are optimally tuned in our experiment to ensure the fair comparison. We use F1-score and NDCG to compare these baselines with *Hyper*. The results are shown in Figure 4.5(a) and 4.5(b). It is evident that our proposed method significantly outperforms other candidates on both metrics.

The results of fLDA and SCENE are comparable to ours. In fLDA, each word in an item is associated with a discrete latent factor often referred to as the topic of the word; item topics are obtained by averaging topics across all words in an item. Therefore, fLDA considers more fine-grained granularity of topics, and consequently obtains reasonable recommendation results. SCENE explicitly takes into account the "submodularity" within users' reading behaviors for recommendation. As we observe, the performance of Goo and ClickB is relatively poor. This is because the recommended lists of both methods are heavily determined by users' co-visiting histories; however, the news data used in the experiments contains a great portion of relatively



(a) F1-score comparison.

(b) NDCG comparison



(c) F1-score comparison.

(d) NDCG comparison

Figure 4.5: Performance comparison for regular recommendation and cold-start user recommendation. Remark: (a) and (b) are comparisons of different algorithms on averaged metrics; (c) and (d) are comparisons of different algorithms for the cold-start problem of new users.

new users[1], i.e., users who read less than 5 news articles per day. Hence, the *cold-start* problem of new users in the dataset causes the poor performance of both methods.

Handling Cold-start

In traditional recommendation methods, the *cold-start* problem cannot be well handled due to the data sparsity. To resolve it, we perform transductive inference on the unified news capsule for new users. The intuition is straightforward: we borrow the concept of inference by utilizing a small set of labeled data to infer the labels of unlabeled data on the hypergraph. Notice that in such a case, we do not focus on the ranking of news articles, but pay more attention on the interestedness of items, i.e., whether the items are preferred by the user or not.

Specifically, we randomly choose 100 users who read news articles less than 5 per day, and then recommend news articles (top@10, top@20 and top@30) for these users. For evaluation purpose, we compare our method with Goo, ClickB, Bilinear, Bandit, fLDA and SCENE. Figure 4.5(c) and 4.5(d) shows the comparison results. As shown in Figure 4.5(c), the *cold-start* problem can be elegantly alleviated by using our proposed method, *Hyper*. The explanation is straightforward: in *Hyper*, we explicitly model the recommendation for new users as transductive inference on a specified news capsule, and meanwhile we consider high-order correlations among different media objects, which significantly complement the data sparsity of new users when performing recommendation. In Figure 4.5(d) we also observe that although we do not intentionally take the ranking into account, our method surprisingly outperforms the others in terms of NDCG. The reason is that we can provide more news articles in the recommendation list that match a user's reading interest, and thus the quality of ranking is improved consequently. The result of fLDA on cold-start handling is

[1]In our dataset, the number of new users is around 400, compared with the total number of users, 3,280.

comparable to our performance. For cold-start users, fLDA gives more weight to the prior mean (predicted by user and/or item features) in estimating the factor, and therefore selects news items that are quite relevant to the prior.

Diversity Evaluation

The recommended news list provided by our algorithm shows a great diversity on topic aspects. Such diversity is originated from the sparkle of the query vector $\mathbf{y}$, by which we can not only specify the target user, but also provide topics and named entities that the user prefers. To evaluate how diverse our recommendation result is, we compare the set diversity [ZH08] among the results of our method and other recommender systems. The set diversity is defined as the *average dissimilarity* of all pairs of news items in the resulted list. Specifically, given a news set $\mathcal{S}$, the *average dissimilarity* of $\mathcal{S}$, $f_d(\mathcal{S})$, is defined as

$$f_d(\mathcal{S}) = \frac{2}{p(p-1)} \sum_{s_i \in \mathcal{S}} \sum_{s_j \in \mathcal{S}, s_j \neq s_i} (1 - sim(s_i, s_j)), \tag{4.16}$$

where $|\mathcal{S}| = p$, and the dissimilarity of a news pair is $1 - sim(s_i, s_j)$, in which $sim(s_i, s_j)$ denotes the content similarity between the news item s_i and s_j, and it is calculated using the cosine similarity.

For diversity evaluation, we choose the aforementioned methods (Goo, ClickB, Bilinear, Bandit, fLDA and SCENE) as our comparison baselines. We employ the experiment setup similar to the previous section, and then compare the diversities of recommended lists with different cardinalities (top@10, top@20 and top@30). Table 4.7 shows the averaged diversity for 100 randomly selected users.

From the result, we observe that (i) The diversity decreases as the recommended news list enlarges. It is straightforward that when more news articles are selected, the topic distribution of the news list becomes closer to the user's reading interest, and therefore the selected news items are more similar. (ii) The diversity of the

Table 4.7: Diversity evaluation on the result list. (The bold font indicates the best performance. * indicates the statistical significance at $p < 0.01$ w.r.t. the randomness of selected users.)

Methods	top@10	top@20	top@30
Goo	0.4101	0.3074	0.1105
ClickB	0.4329	0.3128	0.1562
Bilinear	0.4234	0.2517	0.0933
Bandit	0.5056	0.4126	0.2925
fLDA	0.4726	0.3981	0.2705
SCENE	**0.6537***	0.5771	0.4859
Ours	0.6323	**0.5987***	**0.5072***

recommendation list provided by the baseline methods drops dramatically as the list size increases, since they did not take the diversity into account. (iii) Our proposed method outperforms the others very significantly, except SCENE. SCENE explicitly selects different news items solely from topic-wise for recommendation, and hence the diversity of the result from SCENE is higher more or less. In our work, we consider the interest of news readers by specifying different topics and named entities in the query vector $\mathbf{y}$. The diversity decreases very smoothly when we recommend more news items to individual users, compared with other rivals.

4.3.5 Summary of Hypergraph Modeling

In this work, we propose to use hypergraph learning methods to deal with the issues existed in news recommenders. We first represent news data as a unified hypergraph, in which various correlations among different media objects (e.g., users, news articles, topics and named entities) are integrated into an information capsule. We then decompose the recommendation problem as two subproblems: partitioning and ranking, where the former aims to separate the entire hypergraph into multiple groups, or subcapsules, and the latter is designed to select a list of news articles from a specific capsule and recommend them to a target user (the user is regarded as a query).

The experiments on a news dataset collected from multiple news service websites demonstrate the efficacy of our proposed method.

4.4 Concluding Remarks

This chapter focuses on the problem of understanding complex relations in recommender systems using graph-based approaches, e.g., bipartite graph and hypergraph. There are several ways to extend the algorithms presented in this chapter:

- Bilateral reciprocal recommendation discussed in our work might not cover all possible reciprocal recommendation tasks in a broader perspective. For example, friend recommendations on Facebook and colleague recommendations on LinkedIn exhibit different characteristics, since the recommendation activities on these two platforms might involve multiple parties instead of two. In the future, we plan to expand our reciprocal framework to tackle more reciprocal tasks.
- In the hypergraph model, we simplify the profiling problem by only analyzing a user's reading history (as the profile) without accessing any other auxiliary information. In reality, a news reader might have other information, e.g., demographics, locations, and other social and behavioral patterns. We believe such information can be easily incorporated into our proposed framework, e.g., by encapsuling the user relations within close locations or the same social community into the data model. We can delve into extending our framework along this direction. In addition, although we intentionally partition the hypergraph into multiple small ones to expedite the procedure, the scalability of our proposed framework is not quite satisfactory. We plan to use distributed environment,

e.g. Hadoop, to accelerate the partitioning and recommendation procedure in our future work.

CHAPTER 5

UNDERSTANDING DYNAMICS

In this chapter, we start by introducing the problem of interest dynamics in a recommender system, and then discuss two previous work [LZL11, LHL12] published during my Ph.D. study along this direction. In the following, I will present the details of these two approaches, and discuss how the algorithms can incorporate temporal information when modeling user profiles.

5.1 Research Objective and Contributions

In this research, the problem of temporal dynamics of user interest in recommender systems is studied. In general, a user's preference over items is not static; it might change over time, depending on different types of events, e.g., promotions of a product, holidays, or even the change of a family structure (for example, giving birth to a baby). Based on this intuition, a user's behavioral data might be different from his/her previous consumption history. To capture such changes in recommender systems and consequently provide more reasonable recommendation results, it is imperative to design elegant profiling approaches that explicitly consider the dynamics of user interests. Specifically, the goal of understanding interest dynamics includes:

- Capturing the dynamics of user interest. Due to the properties of different recommender systems, it is possible that a user's fine-grained preference would evolve over time while his/her long-term interest remains stable. If the recommendation technique cannot capture such pattern, the recommended result might not be preferred by users, and hence the system might lose the trust of users.

- Expanding the interest of users by dynamics modeling. In some cases, a user might not actively change his/her interests as he/she accesses different items in a recommender system. Therefore, the user's interest might be static in a long run. If the system cannot recommend "somewhat novel" items to users, the user's inclination towards the system might decrease.

In my previous work, I study the temporal dynamics of user interests in news recommendation [LZL11] and product recommendation [LHL12]. Specifically, in news recommendation, a user's interest evolves over time, depending on casual events happening everyday. The interest can be modeled as a weighted combination of user profiles within consecutive time windows. In contrast in product recommendation, a user's shopping behavior might exhibit a fixed pattern, e.g., customers who have a baby will show similar shopping patterns. We integrate the product taxonomy into the profiling process, and consequently provide recommendation based on the properties of different shopping stages. In the following, I will discuss these two algorithms in more details.

5.2 Dynamics in User Interests

User profiling is an important step for solving the problem of personalized recommendation. Traditional user profiling techniques often construct profiles of users based on static historical data accessed by users. However, due to the properties of different recommender systems, it is possible that a user's fine-grained preference would evolve over time while his/her long-term interest remains stable. Therefore, it is imperative to reason on such preference evaluation for user profiling in recommender systems. Besides, in content-based recommender systems, a user's preference tends to be stable due to the mechanism of selecting similar content-wise items with respect to the

user's profile. To activate users' reading motivations, a successful recommender needs to introduce "somewhat novel" items to users.

5.2.1 Background and Contributions

A key issue of content-based recommender systems is how to construct the user's profile based on his/her own consumption history, named as "user profiling". To handle this issue, most content-based recommender systems take into account a user's accessing history as a whole, and summarize the history to be the user's profile [BP99, KC03]. Research efforts have also been paid to consider the context of a user when recommending items. For example, [JNT10] postulated that a user's preference for particular news articles depends not only on the topic and on propositional contents, but also on the user's current context. [LDP10] developed a Bayesian framework for predicting users' current news interests from the activities of that particular user and the news trends demonstrated in the activity of all users. However, they failed to consider the evolution of users' interest. In reality, the general topics that the user is interested in would be relatively stable or vary slightly in a long-term perspective, whereas the content accessed by the user might change along different short-term perspectives. Taking online news recommendation as an example, Fig. 5.1 illustrates the scenario that a user's interest changes over time.

As is shown, the user's general interest of news articles is sports-related; however, he/she might prefer news articles related to "Cycling World Championships 2011" from t_3 to t_2, and then read newsfeeds that describe "Baseball World Cup 2011" from t_2 to t_1, and finally look through news event of "South Florida All-Star Charity Game" from t_1 to t_0. In terms of how to improve the accuracy of recommendations, it would be more reasonable to explicitly consider news readers' interest drift, or say, interest evolution, when modeling the users' reading profiles.

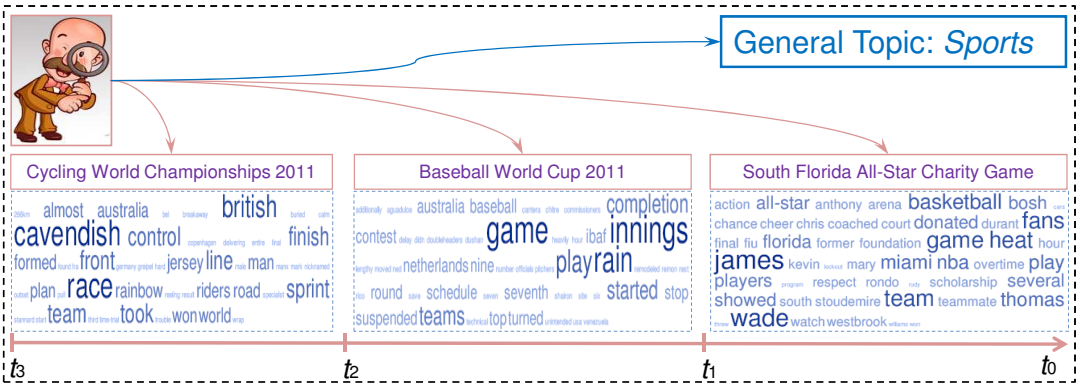


Figure 5.1: An example of interest drift.

Another problem is the stationary profiling. Recall that in content-based recommender systems, the recommendation is often achieved by calculating the affinity between a given user's profile and items and selecting top ranked ones. This strategy favors the similar items and gives lower ranking to the items with topics different from a user's profile. Therefore, if a user uses such a system for a long time, his/her profile tends to become too monotonous, without diverse topics. A typical approach to alleviating this issue involves incorporating diversity into the recommendation result, i.e., to recommend news articles with diverse topics in a single recommendation session [ZH08]. In our previous work [LWL+11], we model the news recommendation problem as a budgeted maximum coverage problem, in which we explicitly consider the diversity among news items in the recommended list. However, most methods in this direction only take into account the similarity between items, without considering the opinions of other users.

In [LZL11], we study the problem of how to model news readers' temporal reading interests and meanwhile broaden their preference. We initially provide an experimental study on the evolution of user interests, and show that most users' reading preferences indeed change over time, whereas their long-term interests vary slightly. Then, we extend our solution of LOGO, in which the long-term and short-term read-

ing preferences of users are seamlessly integrated when recommending news items to individual online users. We construct the long-term profile of a given user based on a *time sensitive weighting* scheme [DL05], and the short-term profile by analyzing the latest reading history of the user. For recommendation, we build a user-item affinity graph based on both long-term and short-term profiles [LZYL14], and then perform absorbing random walk model [ZGVGA07] to select news articles with diverse topics. In this way, we can not only provide relevant news articles with respect to users' interests, but also broaden users' preferences by introducing diverse topics.

5.2.2 Prior Approaches to Temporal Dynamics

In the following, we highlight the existing literatures that are relevant to this work, mainly on two aspects: temporal dynamics in recommendation and random walk models in recommendation.

Temporal Dynamics in Recommendation

A lot of research work related to temporal dynamics in recommender systems have been proposed in recent years. Most of the research efforts along this line focus on adding temporal aspects into the collaborative filtering model. For example, [Kor09] studied the problem of multi-characteristics shift in recommender systems. They proposed to track the time changing behavior throughout the life span of the data via collaborative filtering approaches. [XYZ+10] proposed a session-based temporal graph with incorporates temporal information to model long-term and short-term preferences simultaneously. [LHCA10] investigated the temporal characteristics of the recommender's top-N recommendations, in particular, the temporal diversity of the recommended items. They examined how differently user rating patterns would affect the temporal diversity. A detailed survey can be referred to [Sot11]. Our work

is orthogonal to the existing ones in a sense that we consider the temporal interest drift for a single user.

Another direction related to the temporal dynamics of users involves context-aware recommendation approaches, which consider different user-oriented contexts (e.g., time, location, etc.) when modeling a user's preference over news items [AT11, CBC08, SKP+13]. For example, [MGÁRLGMM13] presents a context-aware recommendation system for journalists to enable the identification of similar topics across different sources. News contextual features, e.g., time, current user interests, location or existing trends, are taken into account for user profiling. Context-aware recommendation approaches can be easily integrated into our proposed framework, i.e., they can serve as the alternatives of the identification of short-term user profiles.

Random Walk in Recommendation

The random walk model is generally applied to collaborative filtering in solving personalized recommendation problems. Examples involve movie recommendations [LY08, WB08, YK08]. In these systems, the transition probabilities are calculated by normalizing the similarity matrix from the movie ratings, and the top-N movies with the highest score are recommended to users. Although positive results are shown in the domain of movie recommendations, it is not clear that directly applying the random walk model in personalized news recommendation can also achieve promising performance. The reason lying in behind is that the data characteristics as well the the problem setting of news recommendation is significantly different from those of movie recommendation:

- The news data in news reading communities change significantly. A gigantic amount of newly-published news articles will be added to the repository everyday, while in movie recommenders, the movie repository is relatively stable.

- The news preference is often binary, i.e., a user may choose to click (1) or not click (0) an article, whereas in movie recommenders, the ratings are explicitly given by users, in a range of $[1, 5]$ or $[1, 100]$. The approaches based on explicit ratings might not be suitable to the scenario of news recommendation.
- Some methods in movie recommendation use arbitrary similarity measures to construct the similarity matrix. However, in news reading community, it is possible that the similarity between two objects depends on the involvement of other objects. Therefore, it would be more reasonable to calculate the transition probabilities using the graph representing the data.

One of the research work that is related to ours is [OTF09], in which a recommendation problem is envisioned as a node selection problem on a graph, giving high scores to nodes that are well connected to the older choices, and at the same time well connected to unrelated choices. This work claims that by using the relatedness between nodes, it is possible to make reasonable yet surprising recommendations. The idea is similar to ours in a sense that we also consider the relatedness between nodes; however, our proposed random walk based method utilizes historical data to construct user profiles, and new data to construct user-item affinity graph. The distinction between historical data and new data serves to reduce the scale of the targets being analyzed, i.e., we do not have to incorporate users' reading histories into the affinity graph. In addition, we reason on the diversity among the recommended news list, which can help broaden users' reading interest to some extent.

5.2.3 An Overview on Temporal Dynamics Model

Figure 5.2 depicts an overview of our proposed framework. We maintain a reading preference model for each online user. This model is essentially a content-based filtering model, involving the long-term and short-term interest profiles. The former is

built based on a time sensitive weighting scheme [DL05], and the latter is represented by the latest reading interest of the user. Our approach assumes that the user's reading preference would evolve over time, and in a long run, the general preference would be relatively stable.

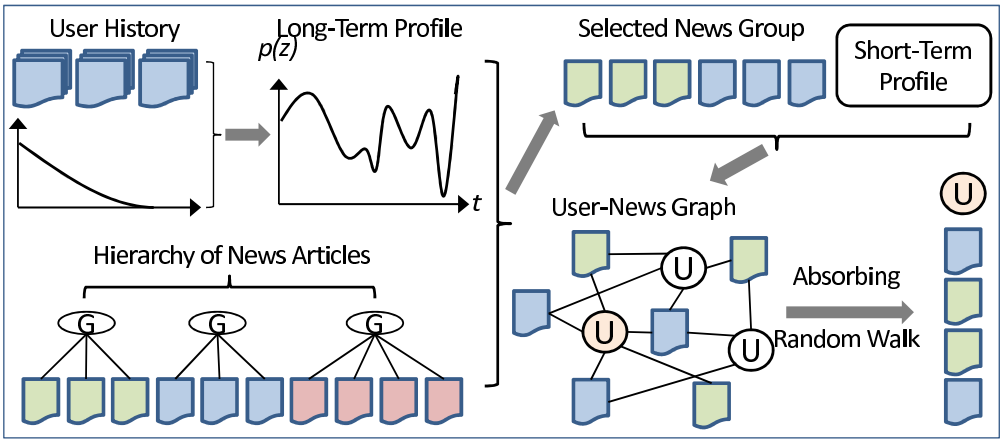


Figure 5.2: System overview of LOGO.

Given a collection of newly-published news articles, we first construct a news hierarchy purely based on the content of news articles using hierarchical agglomerative clustering algorithm, and then use Dunn's Validity Index [Dun73] to decide the best layer of the dendrogram. In this way, we can avoid to decide the number of clusters in news dataset. Using the long-term user profile weighted by time decay factor, we can easily filter the news groups that are similar to the long-term user profile. After that, the short-term user profile is utilized to construct a user-article affinity graph. By performing absorbing random walk starting from the target user [ZGVGA07], in which the escape probabilities of nodes will decrease once they are ranked, we can obtain a list of ranked news articles and recommend them to the target user. The recommended news list contains relevant news articles with respect to the target user's reading interest (i.e., the graph is built upon the user's short-term profile), and further the articles within it have great diversities (i.e., based on the elegant property of the absorbing random walk). Hence, our proposed recommendation framework

can not only satisfy news readers' reading preferences, but also broaden their reading interests in a long run.

To summarize, our proposed recommendation framework consists of three interleaved modules:

1. Long-term and short-term user profiling: In this module, we first segment a user's reading history into multiple time frames, and build long-term and short-term user profile using a time-sensitive weighting scheme. The long-term user profile serves to model a user's general topic interests, whereas the short-term one is able to capture the recent/current user preferences over fine-grained news topics.

2. News group selection: In this module, the newly-published news articles are initially organized as a two-level news hierarchy, where the first level contains news groups with respect to general news topics, and the second level is composed of specific news articles within each news group. When it comes to the recommendation, our proposed framework initially selects a list of news groups that are similar to the long-term profile, i.e., these news groups are close to the general topic preference of users.

3. News article recommendation: Our goal is to select news articles relevant to a user's short-term profile, and at the meantime, to introduce more diversity in order to expand the user's reading interest. To this end, we first build a user-news affinity graph based on the similarities between users' short-term profiles and news articles within selected news groups, and then perform absorbing random walking on the graph to finalize the recommendation.

In the following, the detailed algorithmic information for each module will be introduced.

5.2.4 Temporal Dynamics in User Profiling

In this section, we introduce the profiling paradigm that employs a time-weighting scheme to model user profiles. Specifically, we first segment a user's reading history into multiple time frames, and build long-term and short-term user profile using a time-sensitive weighting scheme. The long-term user profile serves to model a user's general topic interests, whereas the short-term one is able to capture the recent/current user preferences over fine-grained news topics. Such a profiling paradigm enables us to capture both long-term and short-term user preferences, which differentiates our work from existing solutions to time-sensitive user profiling.

Long-Term Profile

In our proposed method, we employ Latent Dirichlet Allocation (LDA) [BNJ03] as the language model to detect latent topics, and represent the topic distribution of the news collection as a topic vector, each entry of which denotes the weight of the representative words in each topic. The long-term user profile is constructed using a time sensitive weighting scheme. Formally, given the reading history $\mathcal{H}$ of a specific user u, $\mathcal{H}$ is initially divided into multiple segments based on a uniform time period T[1], $\mathcal{H} = \{\mathcal{H}_{t_0}, \mathcal{H}_{t_1}, \mathcal{H}_{t_2}, \cdots, \mathcal{H}_{t_n}\}$, where t_0 means the current time period. For each time period t_i, $i = 0, 1, 2, \cdots, n$, we summarize the corresponding reading history $\mathcal{H}_{t_i}$ using LDA and generate a short-term profile $\mathcal{P}_{t_i}$. Note that we assume the user's reading preference evolves over time. We then define a time function $f(t)$ to find appropriate time weights for each $\mathcal{P}_{t_i}$ in the order that the recent reading histories are able to contribute more to the long-term profile. The user's long-term profile $\mathcal{P}_u$ can be represented as

$$\mathcal{P}_u = f(t_0) \cdot \mathcal{P}_{t_0} + f(t_1) \cdot \mathcal{P}_{t_1} + \cdots + f(t_n) \cdot \mathcal{P}_{t_n}, \tag{5.1}$$

[1]In the experiment, T is empirically chosen as 3 days, as shown in Section 5.2.6.

where $\mathcal{P}_{t_i}$ and $\mathcal{P}_u$ are all topic vectors. Intuitively, we are concerned with a user's recent reading preference, as the recent one represents the user's current reading interest. More recent reading histories should have higher weights. Therefore, $f(t)$ is a monotonic decreasing function, which reduces uniformly with time t and the value of the time weight lies in the range $[0, 1]$. Motivated by [DL05], we choose an exponential form for the time function, which is able to describe the gradual decay of the importance of past reading histories as time goes [AHWY04]. The time function is as follows:

$$f(t) = e^{-\lambda \cdot t}, \tag{5.2}$$

where λ represents the profile decay rate. In the experiment, we choose λ as the inverse of the time period T that we are using to divide the entire reading history of the user.

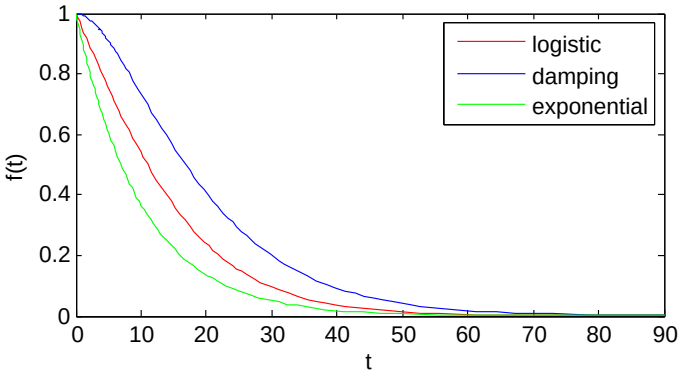


Figure 5.3: Comparison of different time functions.

The exponential function can satisfy our need well. However, there are some other time functions which might also be useful, such as logistic function ($f(t) = \frac{2}{1+e^{\lambda \cdot t}}$), and damping function ($f(t) = (1 + \lambda \cdot t)e^{\lambda \cdot t}$). From Figure 5.3, we can observe the difference between these three time functions, where λ is set to be 1/10. It is straightforward that for all these three functions, the gradient of the curve at the data points close to zero is steeper than that of the data points far away from zero,

which perfectly fits the actual scenario of the user's reading history – more recent histories should carry higher weight than the older ones, and also the weight value should decrease slower as time is far from the current time. However, the exponential function shows the greatest decay speed, compared with the other two functions. In the experiment, we provide empirical comparison among the performance using these three functions respectively, and detailed analysis on how to select λ.

Short-Term Profile

Once we obtain the long-term profile for the given user, we can easily deduce the short-term profile about a user's recent preference. For simplicity, we choose the latest short-term profile $\mathcal{P}_{t_0}$ to achieve this goal. The reason behind this lies in the fact that $\mathcal{P}_{t_0}$ can represent the user's current reading preference, and therefore the most recent reading history would be more beneficial when filtering news items to the end user.

Group Selection using Long-Term Profile

Algorithm 5.1 describes the algorithmic procedure of news selection. When dealing with the newly-published news articles, we first divide the article set into distinct news groups using hierarchical clustering algorithm based on the cosine similarity of news content, where the inner cluster similarity is evaluated by average-link metric. To generate groups of news articles, we use Dunn's Validity Index [Dun73] as the metric to "cut" the dendrogram at certain level of the news hierarchy. This validity measure is based on the idea that high-quality clustering produces well-separated compact clusters. The measure is able to identify sets of compact clusters with a small variance between members of the cluster, and well separated where the means of different clusters are sufficiently far apart, as compared to the within cluster variance.

In general, the larger Dunn's Index, the better the clustering. In this way, we do not have to specify the number of clusters when performing clustering on news articles.

Algorithm 5.1 Group and news selection procedure.

Input: A set of newly-published articles $\mathcal{N}$, user u's profile $\mathcal{P}_u$ and $\mathcal{P}_{t_0}$
Output: A set of recommended news articles

`GroupSelection` :

1. Perform hierarchical clustering on $\mathcal{N}$;
2. Generate a two-level news hierarchy $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_k\}$, where each $\mathcal{C}_i$ is linked a list of news articles, and k is optimized by Dunn' index;
3. Compare $\mathcal{P}_u$ with each $\mathcal{C}_i$, and select top ranked news groups.

`NewsSelection` :

1. Select similar articles from each news group based on $\mathcal{P}_{t_0}$, as a candidate news set V;
2. Construct a user-article affinity graph G based on user set U and news set V;
3. Start from the target user u, perform absorbing random walk on G;
4. Output a list of selected top ranked articles.

Once the collection of news items is well-separated, we summarize each news group using LDA, and present each group as a topic vector to facilitate the group filtering with the long-term user profile. Formally, provided that the summarized news clustering result $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_k\}$, where $\mathcal{C}_i$, $i = 1, 2, \cdots, k$ is a topic vector and k is optimized by Dunn's index, our recommender system automatically calculates the cosine similarity between the long-term user profile $\mathcal{P}_u$ and $\mathcal{C}_i$, and selects the top ranked news groups for further processing. The system sorts the similarity scores in descending order and then selects groups with the similarity higher than the median, which serves as a dynamic threshold to select similar news groups. Up to this point, we can obtain a list of news groups that the user might prefer in a long-run perspective. It is straightforward that the selected news groups would probably cover the user's general interest based on the topic distribution of the long-term user profile.

News Selection using Short-Term Profile

When filtering news item in each news group, $\mathcal{P}_{t_0}$ (the latest short-term profile) is used to compare with news articles within each group. Note that the latest short-term profile represents the most recent reading preference of the user, and therefore it is quite reasonable to compare it with news items in each news group. A naïve way of selecting news articles for recommendation is to first process news articles using LDA, and then compare them with the topic vector of $\mathcal{P}_{t_0}$. Given the processed news articles (often represented as a topic vector), we can adopt a greedy algorithm to sequentially pick up the news article with the largest similarity. To integrate recommended news items from different news groups into the final recommendation list, top ranked items within each group can be selected, and the number of items selected in each group can be proportional to the interest weight of the user's long-term profile on the corresponding news group. However, if we select recommended articles in this way, the algorithm cannot easily adapt to the change of the user's reading preference. For example, a user may be interested in news articles related to NBA playoff; once NBA playoff ends, he/she may shift reading interest to other sports-related topics. Nevertheless, the system following the aforementioned strategy may not be able to detect such an interest shift, and may keep recommending newsfeeds of NBA playoff. We therefore need to consider how to adapt user interest shifts, or say, how to broaden users' reading preference in daily recommendation activities.

5.2.5 Personalized Absorbing Random Walk

In our work, motivated by [ZGVGA07], we propose to use personalized absorbing random walk (ARW) on user-item affinity graph to support temporal news recommendation. Different from general random walks that provide relevance ranking of items, absorbing random walk turns articles that are ranked in the previous process

into absorbing states, and hence lowers the probability that such articles are chosen again. The recommended news articles are relevant to a user's short-term profile, and also belong to the coarse-grained topic group preferred by the user. In this way, our recommender system can guarantee that a user could read "somewhat novel" news articles without sacrificing too much recommendation accuracy.

Intuition: Using Random Walk

A natural solution to the problem discussed in Section 5.2.4 is to use collaborative filtering for news selection. To avoid the changeless profiles of users, we can consult on other users' opinions or interests, and hence "borrow" their preference for the target user. We therefore propose to first construct a user-article affinity graph based on the similarities of newly-published news articles and all the users' short-term profiles, and then employ a random walk model to select articles for recommendation.

Personalized recommendation via random walk on graphs has been extensively studied in recent years [GP06, JE09, LDJ+09, LY08]. Typical recommendation approaches in this direction perform random walk on user×item graphs extracted from historical accessing data. However, in a news reading community, news repository suffers from frequent updating, i.e., a huge amount of newly-published news articles are often added to the repository. In such a case, these new items have very limited, or even no accessing history that can be used for graph construction. Therefore, it is infeasible to directly use random walk on these data.

In our work, we initially construct user-news affinity graph based on the similarities between users' short-term profiles and newly-published news articles by predefining a similarity threshold for edges. In general, news articles in a specific topic group describe different fine-grained topics or events, while belonging to a general topic. A user usually prefers a subset of fine-grained topics, but not all of them. By using per-

sonalized absorbing random walk model, we can help expand a user's reading interest via other users' opinions over the newly-published articles.

Absorbing Random Walk

A user-news affinity graph has two types of nodes: users and articles. The edges in the graph represent the similarity between a user's profile and an article. Formally, let $U = \{u_1, \cdots, u_{|U|}\}$ denote the set of users, $V = \{v_1, \cdots, v_{|V|}\}$ denote the article set, and E be the edge set. Then $G = \{U, V, E\}$.

A PageRank [Hav02] model ranks nodes in a graph using

$$\vec{\mathrm{S}} = \alpha \cdot M \cdot \vec{\mathrm{S}} + (1 - \alpha) \cdot \vec{d}, \tag{5.3}$$

where $\vec{\mathrm{S}}$ is the score vector, α is the damping factor, M is a transition matrix that defines the transition probabilities and $\vec{d}$ is a personalized user-specific vector, indicating which node the random walker will jump to after a restart:

$$d(v) = \begin{cases} 1, & v = v_u \\ 0, & otherwise \end{cases} \tag{5.4}$$

where v_u denotes a user node. $\vec{d}$ serves to avoid the reducibility of the transition matrix.

For the transition matrix M, in our problem setting, it involves three different transition probabilities, i.e., $v \to u$, $u \to v$ and $v \to v$. $p(v|u)$ and $p(u|v)$ can be easily calculated using the similarities between a user's profile and an article, and for simplicity, we restrict them to be equal. The reason we consider $p(v|v)$ is that in news recommendation, it is typical that a user has a reading sequence over the recommended articles. However, in our model, we did not consider $p(u|u)$ since the user-news affinity graph is built based on the similarities but not the actual accessing histories; therefore, it is not reasonable to calculate $p(u|u)$ based on those similarities.

In the graph G, the conditional probability of v_j given v_i can be interpreted as the transition probability $p(v_j|v_i)$ for a random surfer to jump once from the news node v_i to news node v_j via all the connected user nodes u_k, which can be expressed as

$$p(v_j|v_i) = \sum_{k=1}^{|U|} p(v_j|u_k)p(u_k|v_i), \tag{5.5}$$

where $p(u_k|v_i)$ is the probability that a random surfer jumps from news node v_i to user node u_k, $p(v_j|u_k)$ is the probability that this surfer then jumps from u_k to news node v_j, and $p(v_j|v_i)$ is the marginal probability distribution over all the users in U whose profiles are similar to p_i. Inspired by [LDJ+09], $p(v_j|v_i)$ can be treated as a first-order similarity between article i and j, since the calculation is made by considering the news readers whose profiles are similar to both articles.

In absorbing random walk [ZGVGA07], the key step is to turn articles that are ranked in the previous process into absorbing states. To this end, we first absorb the node that represents the target user, and perform random walk on the user-item affinity graph. Intuitively, nodes that are strongly connected to the target user node will have fewer visits, as the walker tends to become absorbed soon after visiting them; comparatively, nodes that are far away from the target user node will have more visits. Therefore, the expected number of visits in an absorbing Markov chain can be calculated in order to decide whether it reaches the absorbing state or not.

Assume that K is the selected article list. Each node $k \in K$ has an absorbing state, which corresponds to the status that $S_{kk} = 1$ and $S_{ki} = 0, \forall i \neq k$. We can easily arrange entries of S by

$$\tilde{S} = \begin{bmatrix} \mathbf{I}_K & \mathbf{0} \\ R & Q \end{bmatrix}, \tag{5.6}$$

where $\mathbf{I}_K$ is the identity matrix on S, and R and Q denote the submatrices with respect to rows of unranked articles. According to [DS84, ZGVGA07], the expected number of visits in the absorbing random walk can be calculated by $N = (\mathbf{I} - Q)^{-1}$,

and can then be normalized by $\mathbf{v} = \frac{N^T \mathbf{1}}{n-|K|}$, where $|K|$ is the size of K. The absorbing state can then be decided by selecting the one with the largest expected number of visits, i.e., $k_{|K|+1} = \arg\max_{|K|+1}^{n} v_i$. After the absorbing states (corresponding to the nodes on the graph) are obtained, the recommendation can be made by selecting these nodes, excluding the ones that represent users.

In each iteration of absorbing random walk, we need to calculate the fundamental matrix $N = (\mathbf{I} - Q)^{-1}$, which involves inverting an $(n - |K|) \times (n - |K|)$ matrix (K is the size of K matrix). Such computation is quite expensive due to the inversion of a matrix. However, the Q matrix will be reduced by one row and one column, as one item is ranked and is turned to absorbed state in each iteration, which enables us to apply the matrix inversion lemma (Sherman-Morrison-Woodbury formula) [PTVF92]. We then only need to invert the Q matrix once in the first iteration, but not in subsequent iterations. Such an improvement can significantly reduce the running time of absorbing random walk.

5.2.6 Empirical Evaluation

Real World Dataset

For experiments, we gather news articles along with users' access history[2] from popular news websites from Jan 15th, 2011 to Apr 16th, 2011 [LWL+11]. We preprocess the data by removing news articles that are rarely accessed (i.e., the accessed frequency is less than 10 times per day) and by storing users with frequent online reading behaviors (i.e., users who read news articles every day and read more than 10 pieces of news each day). After preprocessing, a total of 103,540 news items are stored, along with 3,430 users, each day in average with 1,125 news articles published. We also

[2]The data is collected from a commercial party, providing us access to their back-end logs.

keep the timestamp data that a user clicks an article in order to model the temporal changes of user interest.

User Interest Evolution over Time

Recall that our proposed recommendation framework is based on the assumption that the user's reading preference would be relatively stable in a long run, while the content read by the user might change significantly. To validate this assumption, we empirically segment the user's reading history into 3-day time slots and 15-day time slots, respectively, and then examine the profile in 3-day time slot based on word distribution related to general topics (represented by word frequency), and the profile in 15-day time slot based on topic distribution (detected by LDA).

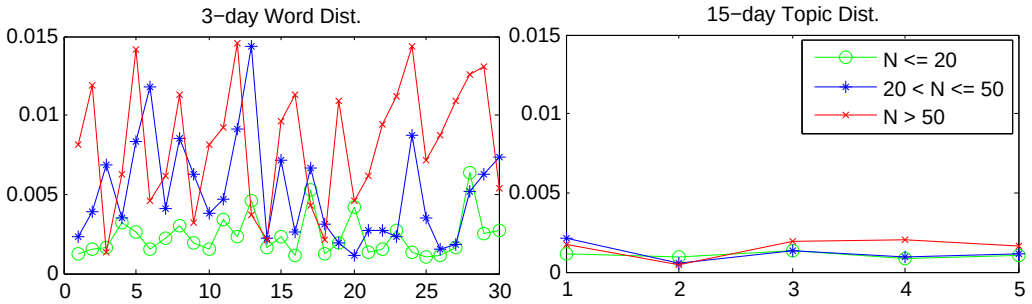


Figure 5.4: KL-Divergence of short-term word distributions (topic-related) and long-term topic distributions over different user groups.

Specifically, we investigate the possibility of interest evolution on different user groups, since users with different news access patterns, such as different reading frequency every day, may have distinct news topic preferences, and therefore the dynamic interest on news articles may vary a lot. To do that, we divide the user pool into three groups based on their reading habits. Suppose a user reads N news articles per day, then the three groups are: (i) $N \leq 20$ (25%); (ii) $20 < N \leq 50$ (38%); (iii) $N > 50$ (37%). To verify whether the user's interest evolves over time, we calculate the KL-Divergence [KL51] between two profiles of adjacent time slots for each

user, and then average the value of different time slots over the three user groups. Figure 5.4 shows the result, where X-axis represents the time slot pair, and Y-axis denotes KL-Divergence value. As is evident in Figure 5.4, all the three user groups exhibit the interest evolution: the general topics (topic distribution in the right figure) are relatively stable in a long run while the specific content (word distribution in the left figure) of those topics that a user might be interested in changes significantly in a short run. Particularly, users with higher access frequency on news articles shift their reading preference more dramatically than the other two groups of users.

Time Function Comparison

In Section 5.2.4, we have analyzed the effects of different time functions. To better understand how the time function influences the final recommendation result, we evaluate the performance of different time weighting schemes (logistic, damping and exponential).

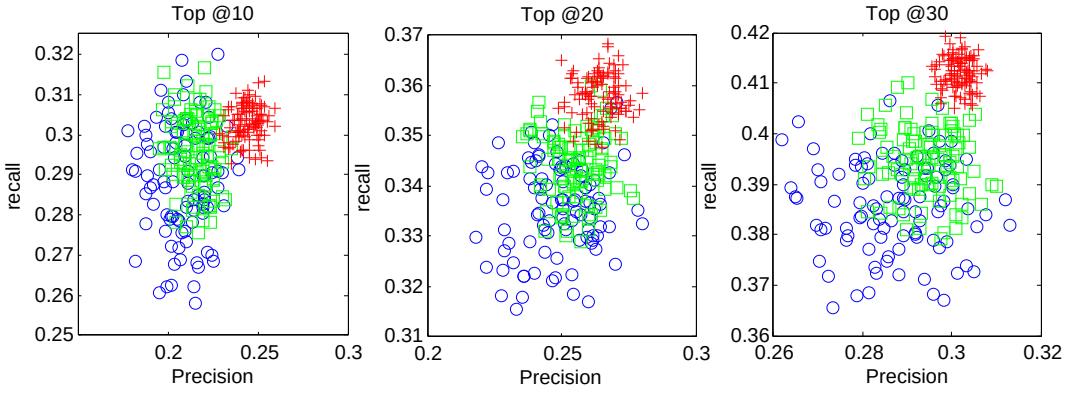


Figure 5.5: Precision-recall plot for different time weighting schemes. Remark: ◯ represents the performance using damping function; □ denotes the performance using logistic function; and + represents the one using exponential function.

For evaluation, we choose 5 time ranges for each user, and use the reading history before the time range to construct the user's profile. The reading history within the time range serves as the ground truth. For each approach, we recommend news

items (top @10, top @20 and top @30) to 100 randomly selected users, and plot the averaged precision and recall pair of recommendation results over 5 time ranges, where each time range contains 3 days. We construct the user's profile based on the reading history earlier than each time range. The recommendation is achieved by applying absorbing random walk model on each approach (i.e., the comparison is performed in the same environment). Note that the decay factor λ is empirically set to 1/3. In next section, λ will be tuned based on the recommendation result. As is depicted in Figure 5.5, besides the higher ratio of precision and recall, the performance distribution of exponential function is more compact than the others, which demonstrates the efficacy and stability of our proposed time weighting scheme.

Parameter Tuning

The value of the decay factor λ denotes the decay rate of the user's reading preference over time. The higher the value of λ, the faster old reading histories decay and the lower the importance of the historical information compared to more recent profiles. To better capture how the recommendation result is influenced by λ, we use different time periods T to segment the user's reading history (the corresponding decay factor $\lambda = 1/T$), and adopt the same experimental setup with the above procedure. Instead of using precision and recall, we calculate the F1-score for each selected user, and plot the averaged F1-score for each λ. The tuning result is shown in Figure 5.6. As is depicted, for top @10, @20, @30 recommended news articles, the performance achieves the best when $\lambda = 1/3$, meaning that the time slot of each history segment is 3 days. Therefore, we set λ as 1/3 for the following experimentation.

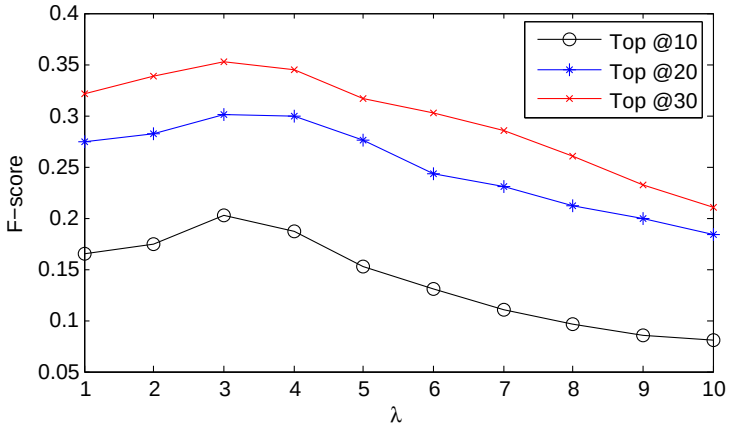


Figure 5.6: λ tuning curve.

Evaluation on Random Walk Models

In our work, we propose to employ absorbing random walk model for news article selection. To this end, the initial step is to construct the user-item affinity graph, which involves two types of edges, i.e., user-article and article-article edges. By including the edges of article-article, we argue that it can be regarded as the transition probability for a random surfer to jump once between two article nodes via all the connected user nodes, which can naturally interpreted as the first-order similarity between articles. Then absorbing random walk is performed on the affinity graph. By introducing the absorbing state into the model, it is able to obtain a static ranking with diverse items selected. In the following, we first evaluate how the recommendation performance can be affected by using different types of edges to construct the affinity graph, and then investigate how random walk models will influence the results.

Comparison of Different Graph Constructions

In this section, we plan to evaluate different constructions of a user-item affinity graph in order to understand the importance of different types of edges for recommendation. The constructions of a user-item affinity graph include: (1) only considering

user-article edges (UA); (2) considering both user-article and article-article edges (UA-AA); (3) considering both user-article and user-user edges (UA-UU); and (4) considering user-article, user-user and article-article edges (UA-UU-AA). The experiment setup is similar to the one introduced in Section 5.2.6, and the threshold of affinity values for all the methods is empirically set to be 0.1. We perform 10-fold cross validation, recommend news articles (top @10, top @20 and top @30) to each set of test users, and calculate the averaged F1-score to compare the performance of different methods. The result is depicted in Figure 5.7.

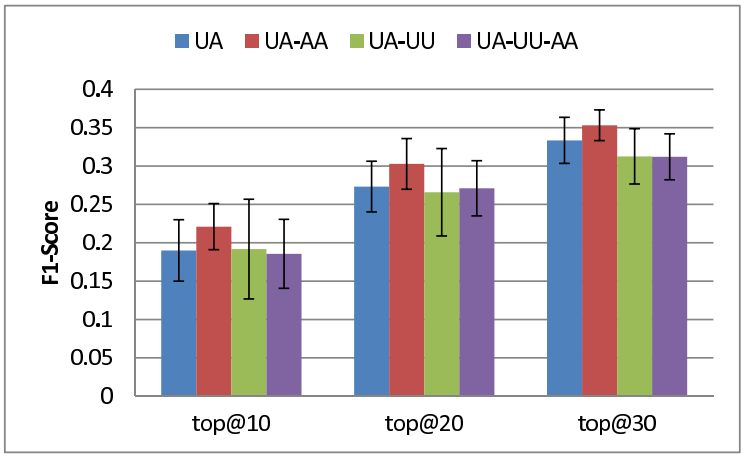


Figure 5.7: Comparison of graph constructions.

As shown in Fig. 5.7, the graph construction based on user-article and article-article edges (UA-AA) outperforms other construction methods in terms of F1-score, with smaller deviation. The result is statistically significant at the 1 percent level. The way of constructing the graph using user-article and user-user edges performs the worst. The reason behind it is that when we perform absorbing random walk model on the graph, the absorbing states on user nodes contribute less to news selection, compared with the ones on article nodes. Therefore, it is not beneficial for news selection to add user-user edges into the graph, and may result in larger deviation of F1-scores, as shown in Figure 5.7.

Comparison of Random Walk Models

The recommended news list provided by our framework shows a great diversity on topic aspects. Such diversity is oriented from the sparkle of absorbing random walk model. In this section, we investigate the item diversity introduced by absorbing random walk model. To this end, we compare the recommendation result using (1) greedy selection based on the similarities of users' short-term profiles and news articles (Greedy); (2) general random walk (GRW); and (3) absorbing random walk (ARW). The experiment setup is similar to the one introduced in Section 5.2.6. We employ set diversity described in [ZH08] as the diversity evaluation metric. The article set diversity is defined as the *average dissimilarity* of all pairs of news articles in the recommendation list, as introduced in Section 2.2.

Figure 5.8: Comparison of news selections.

We user similar experimental setting as introduced in the previous section, and calculate the averaged set diversity for comparison. The result is shown in Figure 5.8. The result is statistically significant at the 1 percent level. As depicted, ARW achieves the best in terms of the set diversity, i.e., the recommended article list contains diverse news items. The set diversity of the recommended list by ARW is relatively stable when the recommended list size increases. The reason is straightforward: by

performing absorbing random walk, the absorbing state of a node may influence the other nodes that directly connect to it, and therefore the selection strategy will avoid selecting nodes with close distance. Comparatively, the results by greedy selection and general random walk model are relatively worse, since in essence, they select and rank news articles based on the similarity between the profile of the target user and the articles.

Comparison with Other Methods

Our proposed recommendation framework takes into account the long-term and short-term interest profiles of the users in an integrated way, where the long-term profile is designed to filter preferable news groups, and the short-term one is used to build user-article affinity graph. In order to verify the effectiveness of the two-stage news selection strategy in LOGO, we use the long-term (LT) and short-term (ST) profiles, respectively, to perform the recommendation task, as two baseline methods. In each baseline, news articles are selected greedily based on a single profile, either the long-term one (LT-Greedy) or the short-term one (ST-Greedy). We also utilize absorbing random walk into these two baselines (LT-ARW and ST-ARW). In addition, we implement two existing approaches, [DDGR07] (Goo) and [LDP10] (ClickB) for comparison. The former is a collaborative filtering based method, whereas the latter is a content-based method. The experimental setting is the same as the previous procedure. We then perform 10-fold cross validation and recommend news items (top @10, top @20 and top @30) to each set of test users. For comparison, we compute the averaged precision, recall and F1-score of recommendation results for these users over the 5 time ranges. Table 5.1 shows the comparison results.

From the result, we observe that: (1) Simply using one single profile (long-term or short-term) cannot guarantee the double-effect of the general topic interest and

Method	Top @10			Top @20			Top @30		
	P	R	F	P	R	F	P	R	F
LT-Greedy	0.1531	0.2095	0.1773	0.2314	0.3115	0.2517	0.2614	0.3469	0.3012
ST-Greedy	0.1614	0.2184	0.1901	0.2385	0.2936	0.2610	0.2660	0.3366	0.2806
LT-ARW	0.1598	0.2217	0.1826	0.2367	0.3202	0.2648	0.2758	0.3541	0.3069
ST-ARW	0.1767	0.2365	0.1944	0.2479	0.3278	0.2763	0.2800	0.3520	0.3144
Goo	0.1901	0.2374	0.2101	0.2516	0.3306	0.2913	0.2813	0.3606	0.3160
ClickB	0.1873	0.2408	0.2048	0.2502	0.3219	0.2835	0.2856	0.3554	0.2987
LOGO-Greedy	0.2186	0.2496	0.2165	0.2698	0.3542	0.3103	0.3198	0.4011	0.3400
LOGO-ARW	0.2253	0.2586	**0.223**	0.2788	0.3643	**0.3231**	0.3285	0.4210	**0.3583**

Table 5.1: Comparison among different recommenders.

the recent reading preference; (2) Treating the user's reading interest over different topic categories independently, like ClickB, cannot effectively capture the exact reading interest of users; (3) The quality of the recommendation result can be improved using absorbing random walk, compared with the one that greedily selects news articles with respect to the target user's profile; and (4) Our proposed recommender LOGO outperforms other methods by providing a seamless integration of long-term and short-term user profiles, and an elegant news selection strategy introduced by absorbing random walk.

5.2.7 Summary of LOGO

The contribution of this work is three-fold:

- We emphasize the effect of user interest evolution when modeling user profiles, and represent the user's reading preference with seamless integration of long-term and short-term user profiles;
- We construct a two-stage news selection strategy, where the long-term profile is firstly utilized to differentiate news groups with specified preference, and then the short-term profile is applied to filter specific news articles to individual users;

- We build a user-item affinity graph based on the filtered news articles and the user's short-term profile, and then perform absorbing random walk on such a graph to select news articles with diverse topics.

Our work presents a novel way to capture users' reading interest drift and at the mean time, to broaden users' preference.

5.3 Dynamics Modeled by Item Taxonomy

In E-commerce recommender systems, there is a special class of recommendation problems, in which a user's behavior might evolve over time, i.e., within different stages the user will prefer distinct items, as shown in *Scenario 1*.

Scenario 1: A customer with a baby needs to purchase some products of baby care. Within different growth stages of his/her baby, the desired products are significantly different.

In this scenario, the customer's purchasing interest evolves over time. It is not reasonable to utilize item-based method for personalized recommendation, as the items purchased in different time periods might significantly different. Further, it is a non-trivial task to effectively capture the evolution of customer's purchasing interest. In general, an item taxonomy is often associated with a recommender system, by which customers can easily navigate different categories of products. In this preliminary work, we propose to employ such taxonomic knowledge to formalize users' long-term preference, and in the meantime, to capture the evolution of users' interest. However, only long-term preference cannot provide enough evidence against the user's current desire, as illustrated in *Scenario 2*.

Scenario 2: A customer with a 6-month baby needs to change the type of formula for his/her baby.

In this scenario, the customer needs to purchase some new items. In general, long-term preferences model users' purchasing interest in a long run, e.g., what brands that the user prefers. However, for customers who step into a new purchasing stage, he/she would desire new types of items that have never been purchased before by this user. Therefore, only considering the customer's long-term preference will not capture his/her desire. To handle such issue, we propose to explore the user's short-term interest by analyzing similar users' behaviors. Here "similar" indicates that two users might have similar purchasing behaviors within the identical purchasing stage. By this way, we can easily resolve the situation that the customer desires new types of products within new stages.

5.3.1 Problem Formation

In our work, we employ item taxonomy to facilitate the recommendation procedure. In general, item taxonomies are often designed or applied in E-commerce websites, e.g., *Amazon*. In particular, we have the following elements within an E-commerce community:

- **User Repository**: $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$. $\mathcal{U}$ contains all users or customers within the E-commerce community.
- **Product Repository**: $\mathcal{P} = \{p_1, p_2, \cdots, p_m\}$. $\mathcal{P}$ is composed of all the products provided by the E-commerce website.
- **User Rating**: $\mathcal{R} = \{R_1, R_2, \cdots, R_n\}$. $\mathcal{R}$ includes all the rating information of users over products.
- **Item Taxonomy**: $\mathcal{T}$ over item category set $\mathcal{C} = \{c_1, c_2, \cdots, c_l\}$. The taxonomy $\mathcal{T}$ captures the hierarchical structure of $\mathcal{C}$ in the E-commerce. For simplicity,

we require that one product can only fall into one specific category, i.e., $\{p \rightarrow c_i | \neg c_j, c_j \neq c_i, p \rightarrow c_j\}$.

Problem (Recommendation with Stage): *Within an E-commerce community, given $\mathcal{U}$, $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{T}$, recommend items to a target user, guaranteeing that the user's current purchasing desire is maximally satisfied.*

To resolve the above problem, we can decompose it into two different subproblems based on the characteristics of the problem itself.

Subproblem 1 (Stage Identification): *Given an item taxonomy and a target user's history, identify the user's current preference on item categories.*

To identify the recommendation stage for the target user, we need to consider the entire purchasing history of this user. Simply modeling the user's behavior based on transactional purchasing data cannot capture the evolution of the user's exact interest. Fortunately, the item taxonomy can provide us a meaningful and elegant explanation of the user's evolved behavior.

Subproblem 2 (Item Recommendation): *Given a target user's current stage, recommend a list of items to the user such that his/her preference will be maximally satisfied.*

Once we obtain the recommendation stage of the target user, a natural way to recommend items is to refer other users' behaviors within the same stage. In our problem setup, we have additional resources to use, e.g., the item taxonomy, to enrich the correlations among users, and therefore we are able to provide more reasonable recommendation list.

5.3.2 Stage Identification

The item taxonomy semantically encapsules the correlations among items, which is suitable to capture users' preferences over different item categories. Specifically in our

work, we first separate a user's purchasing history into different stages, and then model the user's long-term preference by employing the item taxonomy. Given a target user, we construct a multi-modal graph based on the users' purchasing behaviors similar to the target user, and then perform inference on this graph to finalize the recommendation.

The long-term user profile consists of multiple stage profiles. Formally, given the purchasing history $\mathcal{H}$ of the target user u, $\mathcal{H}$ can be initially divided into multiple segments based on a specific time scheme. Here the time scheme used to segment $\mathcal{H}$ can be learned according to the detailed content of $\mathcal{H}$ using statistical or machine learning techniques. Taking into account the time complexity, hereby, we simply define a uniform time frame T as a segment unit, and then separate $\mathcal{H}$ based on T, i.e., $\mathcal{H} = \{H_{t_0}, H_{t_1}, \cdots, H_{t_n}\}$, where t_0 means the current stage, i.e., the latest purchasing period. For each time period t_i, $i = 0, 1, \cdots, n$, we model the user's behavior H_{t_i} using the item taxonomy.

5.3.3 User Profile Generation

As illustrated in Fig 5.9, to model the stage profile $\mathcal{H}_{t_0}$ of user u_1, we first extract the transactional data of u_1, $R_1 = \{p_{41}, p_{51}, p_{21}\}$. For each transaction or item in R_1, e.g., p_{41}, we locate the fine-grained category that the item directly belongs to, e.g., c_4, and then traverse the taxonomy from bottom to top, to obtain the category path starting from this category, e.g., $c_4 \rightarrow c_1$. Following this example, we then transfer the rating score on p_{41} to the category path, i.e., assigning "1" to each category in the path. Subsequently, we aggregate the rating score for each category and formalize the user's stage profile as a weighted category vector.

Again for the previous example, assume the basic category vector is in the form of $\langle c_1, c_2, c_3, c_4, c_5 \rangle$, then the weighted category vector $R^{t_0}_{u_1} = \langle 1, 2, 0, 1, 1 \rangle$, where each en-

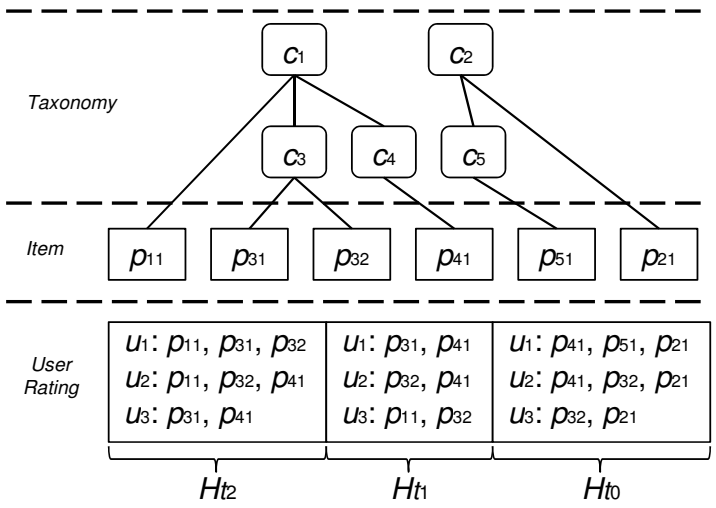


Figure 5.9: An example of item taxonomy.

try represents the user's implicit rating over the corresponding category. In such representation, we assign more weight on the categories of the upper level in the item taxonomy, by which we can avoid being trapped into specific topics and losing the overall recognition of the user's preference. The weighted category vector is l_2-normalized. Finally, we can denote user u_1's long-term profile as $\mathbb{R}_{u_1} = \{R_{u_1}^{t_0}, R_{u_1}^{t_1}, R_{u_1}^{t_2}, \cdots\}$, and also we can identify the current recommendation stage of u_1 as H_{t_0}.

The identification of the user's current recommendation stage can help determine what information should be used in the item recommendation, i.e., to consider other users' stage profiles. For example, there are three users u_1, u_2 and u_3, whose long-term profiles are $\{R_{u_1}^{t_0}, R_{u_1}^{t_1}, R_{u_1}^{t_2}\}$, $\{R_{u_2}^{t_0}, R_{u_2}^{t_1}, R_{u_2}^{t_2}, R_{u_2}^{t_3}, R_{u_2}^{t_4}\}$, and $\{R_{u_3}^{t_0}, R_{u_3}^{t_1}, R_{u_3}^{t_2}, R_{u_3}^{t_3}\}$. Given the target user u_1 with the recommendation stage H_{t_0}, we need to analyze the stage profiles of users u_2 and u_3, that is, $R_{u_2}^{t_2}$ and $R_{u_3}^{t_1}$, since these stage profiles indicate that their stages are identical to u_1's current stage. In this sense, the long-term and short-term profiles of a user are dependent in a way that the short-term profile is derived from the long-term profile and also the long-term profiles of different users can contribute to the success of collaborative filtering. The subsequent procedures are all based on this scheme. Note that the evolution of a user's purchasing interest

exists within the user's long-term profile, whereas the interest would be relatively stable within the user's short-term profile.

5.3.4 Model Refinement

As proposed in the previous section, the item taxonomy serves to connect users and items in a semantic way, in which users' rating behaviors are amplified via the category path. In such a model, we can easily capture the correlations between different elements, i.e., users, items and categories. In the following, we further formalize the taxonomy-based profiling model by specifying three different similarities.

- *User-User Similarity* ($\mathcal{S}_U$): The user-user similarity originates from two different components: user-item similarity S_{UI} and user-category similarity S_{UC}. Given two users u_1 and u_2, S_{UI} can be computed by considering the *Jaccard similarity* between the item sets purchased by these two users, I_{u_1}, I_{u_2}, whereas S_{UC} can be derived from the *Cosine similarity* between the identical stage profile vectors of these two users R_{u_1}, R_{u_2}. Specifically,

$$S_{UI}(u_1, u_2) = \frac{I_{u_1} \cap I_{u_2}}{I_{u_1} \cup I_{u_2}}, S_{UC}(u_1, u_2) = \frac{R_{u_1} \cdot R_{u_2}}{||R_{u_1}||||R_{u_2}||}.$$

  Finally, the similarity between two users, $S_U(u_1, u_2)$ can be calculated as:

$$S_U(u_1, u_2) = \lambda \cdot S_{UI}(u_1, u_2) + (1 - \lambda) \cdot S_{UC}(u_1, u_2). \tag{5.7}$$

  In this way, we can easily capture the correlations between two users from both the taxonomic level and real purchasing behavior level, simultaneously. This similarity can help obtain user links.

- *Item-Item Similarity* ($\mathcal{S}_I$): Similarity between items can be computed using item-to-item collaborative filtering. Given a product p, our goal is to find products similar to p; Here by "similar" we mean that the user sets, U_{p_1}, U_{p_2}, pur-

chasing the two items have substantial overlap. To this end, we use the *Jaccard similarity* to capture $S_I(U_{p_1}, U_{p_2})$, as

$$S_I(U_{p_1}, U_{p_2}) = \frac{U_{p_1} \cap U_{p_2}}{U_{p_1} \cup U_{p_2}}. \tag{5.8}$$

- *Category-Category Similarity* ($\mathcal{S}_C$): This similarity can help quantify the semantic correlation between two different categories in the item taxonomy. In our work, we employ *Information Content* (IC) [Res95] to compute $\mathcal{S}_C$, which measures the amount of information of a given category c from its probability of occurrence, in our case, the probability that items under c are purchased. The larger the IC, the more popular the category. IC can be computed as the negation of the logarithm of the probability of occurrence, i.e., $IC(c) = -\log p(c)$. To derive the similarity between two categories, we use the similarity measure proposed in [Lin98], which evaluates the correlation between a pair of concepts as the IC of the *Least Common Subsumer* (LCS) in a give taxonomy, i.e., an indication of the amount of information that two categories share in common. Given two categories c_1 and c_2, the similarity $S_C(c_1, c_2)$ can be computed as

$$S_C(c_1, c_2) = \frac{2 \times IC(LCS(c_1, c_2))}{IC(c_1) + IC(c_2)}. \tag{5.9}$$

5.3.5 Item Recommendation

Once we locate the recommendation stage of the customer, a simple solution to the recommendation is to employ user-to-user collaborative filtering to retrieve items that the customer might prefer. However, such paradigm might result in the overemphasis on popular items, and thus make popular items more popular, while the items in the long tail have little chance to be recommended to the user. To resolve this issue, we investigate the effect of using the item taxonomy to increase the possibility that items in the long tail are being recommended. To this end, we first construct a multi-modal

graph, and then given a target customer u, we perform random walk on this graph starting from u for recommendation.

In detail, the multi-modal graph consists of multiple resources, e.g., users, products and categories, associated with multiple types of correlations, e.g., user-user, item-item, category-category relations and etc. These resources are all originated from the user's purchasing behaviors within the same stage. We encapsule the graph as an adjacency matrix. Formally, let U, P, C denote customers, products and categories respectively, and let U_p, U_c, P_c denote user-product, user-category and product-category relations, then we can build a block-wise adjacency matrix $\mathbb{W} = \begin{pmatrix} U_u & U_p & U_c \\ U_p & P_p & P_c \\ U_c & P_c & C_c \end{pmatrix}$, where U_u, P_p, C_c denote user-user, product-product and category-category relations. Here the entries of U_u is calculated by S_U defined by Eq.(5.7), P_p by Eq.(5.8) and C_c by Eq.(5.9). The entries of other blocks in $\mathbb{W}$ can be derived by purchasing behaviors or the item taxonomy. For example, for entries in U_p, each one is a binary value ("1" or "0"), representing whether the corresponding user has been purchasing the product or not.

Given a target user u, we perform random walk on the multi-modal graph starting from u. Here we employ *random walk with restart* (RWR) [PYFD04] to retrieve items for recommendation, such that the recommended items are not deviated much from the user's purchasing interest. Specifically, we construct a matrix $\mathbb{A}$ using normalized graph Laplacian, i.e., $\mathbb{A} = D^{-1/2}\mathbb{W}D^{-1/2}$, in which D is a diagonal matrix with its (i, i)-element equal to the sum of the i-th row of $\mathbb{W}$. The algorithmic detail is described in Algorithm 5.2.

In Algorithm 5.2, we first build a multi-modal graph whose nodes include users, products and categories, and the weighted edges between nodes are quantified by the adjacency matrix $\mathbb{W}$. Here we normalize $\mathbb{W}$ using graph Laplacian (line 2) to make

Algorithm 5.2 RECOMMENDATION USING RWR

input: An adjacency matrix $\mathbb{W}$, and a target user u_q.
output: A recommended item list l.

1. Let $\overrightarrow{\mathbf{v_q}} = 0$ for all its entries, except a '1' for the q-th entry;
2. Normalize $\mathbb{W}$ using $\mathbb{A} = D^{-1/2}\mathbb{W}D^{-1/2}$;
3. Initialize $\overrightarrow{\mathbf{u_q}} = \overrightarrow{\mathbf{v_q}}$;
4. **while** $\overrightarrow{\mathbf{u_q}}$ not converged **do**
5. $\quad \overrightarrow{\mathbf{u_q}} = (1-\delta)\mathbb{A}\overrightarrow{\mathbf{u_q}} + \delta\overrightarrow{\mathbf{v_q}}$;
6. **end while**
7. Select top ranked subset $\mathbf{e}$ from $\overrightarrow{\mathbf{u_q}}$ as the recommendation base;
8. **if** e_i is a user **then**
9. $\quad$ Select items that e_i has purchased recently and put them into l;
10. **end if**
11. **if** e_i is a product **then**
12. $\quad$ Put e_i into l;
13. **end if**
14. **if** e_i is a category **then**
15. $\quad$ Select items that contribute more to IC of e_i and put them into l;
16. **end if**
17. **return** l;

it suitable for the random walk computation. Next, our method executes RWR (line 3-6) to retrieve elements highly related to a given user q. The procedure will end once the input vector $\overrightarrow{\mathbf{u_q}}$ converges. Here the parameter δ controls the steps of each random walk; in our case, we expect that the random walk be not so deep on the graph so that the selected items will conform to the user's purchasing interest.

After $\overrightarrow{\mathbf{u_q}}$ is finalized, our method choose a subset of elements that are top ranked in $\overrightarrow{\mathbf{u_q}}$, which might include users, products or categories. Therefore, to retrieve the recommended items, we need to consider three distinct cases (Line 8-16). In this way, the resulted recommendation list is composed of products that are originated from different recommendation disciplines.

5.3.6 Empirical Evaluation

Real-World Data

The dataset used in our experiment is collected from *Amazon.com*. It consists of customers' order information related to *Baby Care*, ranging from Jan 21st, 2005 to Mar 5th, 2009. This dataset contains 133,039 orders of 1,000 anonymous customers on 2,187 products. The ratings in this dataset are implicit rating, i.e., the binary rating originated from the purchasing behavior of customers. The time span of the customer's order history varies significantly, ranging from several months to 4 years. The average number of orders for customers is 133. The item taxonomy covered by the dataset consists of 37 classes and it is a three-layer hierarchy: the root class, "Baby Care" and subclasses, e.g., "Baby Diaper", "Baby Formula", "Feeding Accessories", and etc. For each category, the number of products contained varies from tens to hundreds. Hereby, the taxonomy is simple and items ratings in the dataset are relatively dense. Due to the space limit, we are unable to list all the classes contained in the taxonomy.

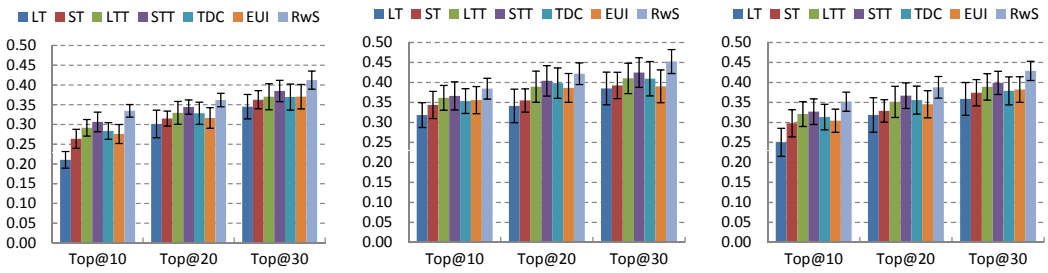
Profile Generation and Taxonomy

Long-term and short-term profiles both contribute to the success of recommendation. To verify this, we use the long-term and short-term profiles, respectively, to perform the recommendation task. Also in our framework, we employ the item taxonomy to enrich the representation of users' profiles. Therefore, we setup four baseline methods as follows: (1) Long-term without taxonomy (LT): use each customer's entire purchasing history to construct profiles and perform collaborative filtering; (2) Short-term without taxonomy (ST): use each customer's recent purchasing history[3] to build

[3]Here we empirically set the "recent" history as the last 90 days behavior, and when performing collaborative filtering, we only consider all customers' recent histories.

profiles and perform collaborative filtering; (3) Long-term with taxonomy (LTT): use item taxonomy to enrich each customer's long-term profile and perform collaborative filtering; (4) Short-term with taxonomy (STT): use item taxonomy to enrich each customer's short-term profile and perform collaborative filtering. In addition, we implement two existing approaches, [ZLST04] (TDC) and [NFT$^+$10] (EUI) for comparison, where the former considers both user proximity and item proximity, and the latter uses taxonomy to recommend novel items to users.

For evaluation of our method (RwS for short), we empirically set the time frame T to be 90 days, and use the purchasing history within the latest stage as the testing data, whereas the purchasing history before the latest stage is regarded as the source of profiling. Note that in our method, user similarities are calculated based on the stage closest to the target user's current stage. The parameters λ and δ will be tuned in the following subsection. For each approach, we recommend products (top@10, top@20 and top@30) to all the users in the dataset, and then compute the averaged precision, recall and F1-score of the recommended list. Fig 5.10 shows the comparison results.



(a) Precision comparison. (b) Recall comparison (c) F1-score comparison.

Figure 5.10: Performance comparison for different recommendation algorithms.

From the results, we observe that (i) The performance of algorithms with item taxonomy involved is superior to the ones without taxonomy. It is evident that by using item taxonomy, users' preference can be better captured, since the categories or classes have more distinguishable power than simple items on identifying a user's

purchasing interest. (ii) By considering the short-term profiles, the accuracy of the recommended item list can be slightly improved, which is straightforward because we are more concerned with users' recent activities. (iii) Our proposed method significantly outperforms other candidates, which demonstrates the efficacy of our algorithm in handling the problem of RwS.

Model Validation

In Section 5.3.2 we introduced three distinct similarity measurements, i.e., *user-user*, *item-item* and *concept-concept*. The encapsulation of these three measures enables the method to capture the correlations among different resources (users, items and categories), and therefore renders the recommendation more reasonable. Also, the item taxonomy can help better interpret the customers' exact purchasing interest. To verify this claim, we evaluate the performance of using different measurements. In the experiment, we choose to use user-user and item-item similarities respectively, as two baselines, and then recommend items (top@10, top@20 and top@30) to a set of randomly selected users (100 users). Note that the parameters λ and δ are set to be 0.2 and 0.7, respectively. Detailed parameter tuning process can be found in next section. We plot the average precision and recall pair of recommendation results. As depicted in Fig 5.11, besides the higher ratio of precision and recall, the performance distribution of the unified multi-modal graph is more compact than the others, which demonstrates the stability of the strategy for multi-modal graph construction.

5.3.7 Summary of RwS

In summary, the contribution of our paper is three-fold:

- We define a new class of recommendation problem, named *Recommendation with Stage* (RwS), in which a user's preference evolves over time.

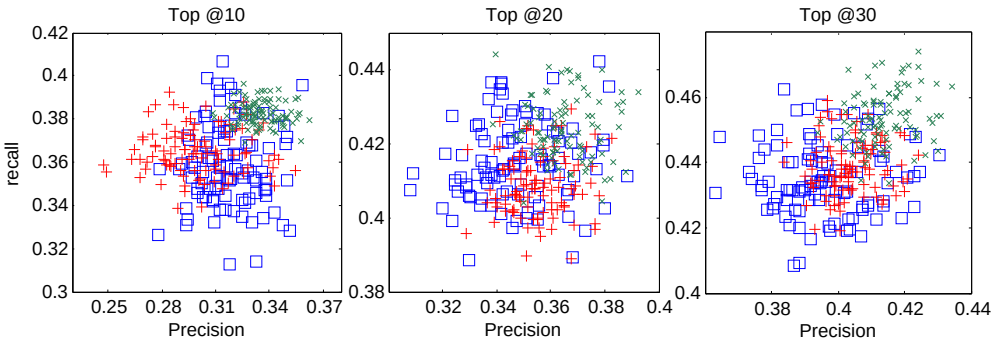


Figure 5.11: Precision-recall plot for different model constructions. Remark: □ represents the performance using user-user similarity; + denotes the performance using item-item similarity; and x represents the one using the integration of user-user, item-item and category-category similarities.

- We introduce a taxonomy-oriented approach to model a user's preference. We propose to model a user's preference not only on item level, but also on the semantic correlations among the categories that the items belong to.
- We propose a novel graph-based model for recommendation, in which a multi-modal weighted graph, including users, items and concepts, is constructed, and then random walk is performed on this graph for inference.

5.4 Concluding Remarks

This chapter studies the problem of interest dynamics in a recommender system, and introduces two previous work [LZL11, LHL12] along this direction.

For the work of LOGO, we initially provide an experimental study on the evolution of user interests in real-world news recommender systems. To better capture the interest evolution issue, our proposed recommender seamlessly integrates the long-term and short-term reading preferences of users when recommending news items. The time sensitive weighting scheme on the long-term profile, along with the two-stage news recommendation framework, shows promising performance compared with other

existing methods. In addition, we employ personalized absorbing random walk on user-article affinity graph (constructed based on the similarities of newly-published articles and users' short-term profiles) to adaptively select recommended news articles. When averaged with a relatively static ranking, one can balance relevance and diversity in the recommended list. In this way, we can not only satisfy users' reading interest, but also broaden their preferences according to the temporal dynamics.

As future work, one possible extension is to allow users to given feedbacks about the recommendation results, and then transform these feedbacks to constraints when performing absorbing random walk. In practice, the feedbacks could be the ratings of users, or the recommended rankings of items. By considering such feedbacks for recommendation, the user's reading preference can be well captured, and consequently the recommended result can be more reasonable and meaningful. Another direction along this research could be integrating users' long-term and short-term profiles when performing absorbing random walk. In this way, we do not have to decouple the process of selecting news groups and select news articles, and therefore can have a unified solution to personalized news recommendation with temporal dynamics.

For the work of RwS, we introduced a problem – Recommendation with Stage in E-commerce, and then proposed a novel method that utilizes item taxonomy to resolve this problem. Our approach is capable to effectively capture a customer's purchasing interest via a seamless integration of the customer's long-term profile and short-term profile, where the long-term profile is derived from the entire purchasing history, and the short-term profile is originated from the identification of the current recommendation stage of the customer. For recommendation purpose, we focus on the current recommendation stage and proposed a multi-modal graph ranking method to obtain the recommended item list. Evaluation on a real-world dataset demonstrates the effectiveness of our proposed method.

Notice that the performance of our proposed method depends on the quality of the item taxonomy. In our future work, we plan to investigate how the taxonomy would influence the performance of the proposed method. In addition, the segmentation of a user's purchasing history is based on a uniform time frame in our method, which is not quite sufficient since the durations of purchasing stages of different users might vary a lot. Therefore, we will also take into account different segmentation methods in our future work.

CHAPTER 6

APPLICATIONS

The previous introduced issues of recommender systems, i.e., understanding behaviors, relations and dynamics, reflect a user's profile from different perspectives. User behaviors represent the usage pattern of a user in a long run, interest dynamics show the temporal change of a user's preference in a session-based interval, and user relations denote the correlation between the target user and other elements within a recommender system. By integrating these three research objectives, we are able to comprehensively understand the exact preferences of users, and construct high-quality user profiles. Built on top of it, in this research we propose a recommendation framework that is capable of capturing diverse information needs of users, and consequently providing reasonable and meaningful recommendations according to the well-constructed user profiles. Further, the proposed framework for personalized recommendation is a generalized solution to various recommendation domains. Despite of different characteristics of recommender systems, users who are using these systems exhibit similar behaviors (e.g., click patterns, complex relations and dynamical interest drifts). Hence, the framework can handle recommendation cases of different systems.

6.1 Objective

Based on the solutions to the aforementioned three issues, a systematic framework is presented to give the guidance on building an effective recommender system. An overview of the recommendation framework is depicted in Figure 6.1. As shown in the figure, the framework includes three interleaved modules:

1. *Profiling*: The profile of a user in the system is constructed based on multiple aspects, including the sharing interests with other users, the long-term preference and the short-term taste. The profiling is able to handle the changing patterns of coarse-grained and fine-grained perspectives in different time granularities.

2. *Recommendation*: The recommendation result can be obtained by carefully analyzing the relations of user profiles and the item repository. In particular, the items are initially organized into a two-level item hierarchy. The long-term profile of users is adopted to select general item categories from the top level, whereas the short-term profile is employed to filter and rank items in the bottom level.

3. *Evaluation* The evaluation module takes into account various recommendation evaluation criteria, including *accuracy*, *diversity*, *reciprocity*, *vitality*, etc. These criteria provide helpful guidance in refining the profiling and recommendation models from different perspectives.

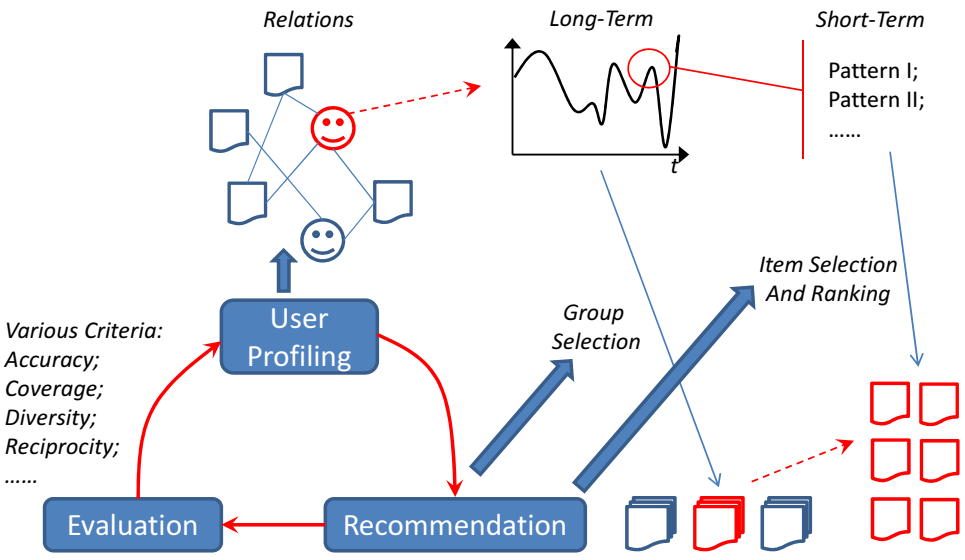


Figure 6.1: The recommendation framework.

The three interleaved modules operate in an iterative way in real-world applications. The *Profiling* module constructs user profiles based on different types of information; the *Recommendation* module takes user profiles as input, and generates recommended results to users; and the *Evaluation* module analyzes the recommendation logs and performs experimental evaluation on the running algorithms to provide more robust and workable parameters for the models.

6.2 A Representative Application

In my previous work [HLLP13], an online recruiting system is presented, called iHR. iHR is a collaborative solution designed for job markets, on which job seekers and recruiters are able to cooperatively build an effective job matching platform. It provides users functionalities to organize their career development, to improve their corporation culture, and to construct extensive relation network.

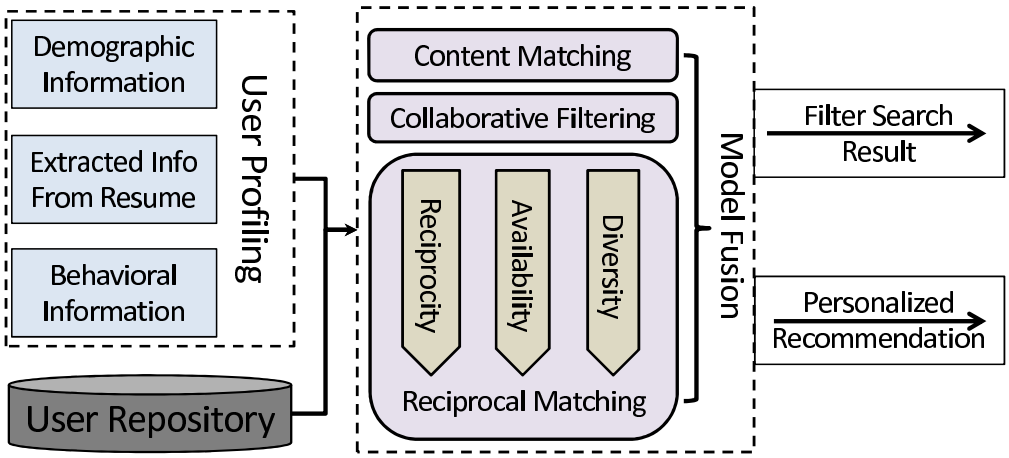


Figure 6.2: The system overview of iHR.

In iHR, registered users are categorized into two groups: job seekers and recruiters. For job seekers, iHR enables them to input their basic information, to upload and update their resumes, and to receive instant recommendation for job positions; For

recruiters, similar functionalities are provided, except for the recommendation of job applicants. Figure 6.2 shows the system overview of iHR.

The information processing and representation functionalities are integrated into three critical modules: *User Profiling*, *Advanced Search* and *Recommendation*. The seamless integration of these modules makes the system more user-friendly and customer-centric. In the following, we describe functional details of these modules.

User Profiling: To construct user profiles for job seekers and recruiters, iHR considers multiple information resources, including users' basic information, extracted text from uploaded files or links and uses behaviors. The fusion of these types of information enables us to comprehensively understand a user's exact interest.

Advanced Search: In iHR, advanced search functionality for users is provided in order to quickly obtain the information they are interested in. Although search is commonly used in most job matching systems, iHR uses some query expansion techniques [MRS08] to expand the query keywords for more comprehensive results. In this preliminary work, we do not focus on introducing the search service as it has been extensively studied by many existing systems[AP02, SBR02].

Recommendation: iHR provides flexible interfaces for users to obtain recommendation results related to their preferences. We implement three distinct recommendation strategies, including content filtering, collaborative filtering, and reciprocal recommendation, and also provide integrated recommendation results to users. For reciprocal recommendation, we extensively investigate the bilateral correlation between job seekers and recruiters, i.e., the reciprocity, and therefore help achieve the win-win situation among them.

Besides the major modules mentioned above, in iHR we also provide an interactive channel for job seekers and recruiters. Figure 6.3 shows an example of a job seeker's comment management.

Figure 6.3: The comment management of job seekers.

6.2.1 User Profiling

A natural way to enhance the search and recommendation experience is to maintain and utilize a user's profile. Users' profiles provide us extensive evidence to understand the major preference of both job seekers and recruiters, and great facilities to adjust the search and recommendation output for individuals. In iHR, a user's profile is composed of three components, i.e., the *basic* information, the *extracted* information and the *behavioral* information. In the following, we describe the detailed profiling for each component.

Basic Information

A user's basic information includes demographic data and explicit preference specified by the user. Such information is common in most job matching systems. Table 6.1 shows sample features of the basic information we collect from users.

Table 6.1: Bilateral features in a job matching system. Features with the prefix need are all the preference features, whereas the others are the self-description features.

Job Seekers				Recruiters			
scalar	categorical		free text	scalar	categorical		free text
age	language	marriage	education	salary	job_type	need_degree	company_desc
work_length	sex	degree	self_desc	need_amount	trade_type	need_sex	job_desc
need_salary	work_status	need_job_type	ability	need_work_length	property_type	need_language	need_ability

In general, the user-specified features can be scalar, categorical or free text, as shown in Table 6.1. For scalar features, we first transform them into different ranges, e.g., we can transform the length of working experience as 0-3 years, 3-5 years, 5-10 years and 10+years. We then encode these ranges as binary features, e.g., if a user's working experience falls into the range of 0-3 years, then this feature (0-3 years) would be set to true (1), and all other ranges would be set to false (0). With categorical attributes, we use the same strategy of dealing with working experiences ranges. For free text features, we transform the text into an l_2-normalized tf.idf-based term vector, and then combine this vector with the vector obtained based on other types of attributes. Note that the demographic information and the preference are separately processed. We denote the demographic vector as $\boldsymbol{p}^s$ and the preference vector as $\boldsymbol{p}^p$.

Extracted Information

Another way to collect a user's information is to analyze his/her external information, e.g., resumes and home pages. In general, such information might be represented as .doc, .pdf, or .html files, and the text information can be extracted from these files. Take the resumes as an example. The text in resumes contains multiple types of information, e.g., personal information (demographics), educational details (graduation school, degree, major), experiences (activities, research, skills), etc. Automatically extracting structured information from resumes of different styles and formats would

be challenging. [YGZ05] analyzes the hierarchy of resume information, and then proposes a cascaded two-pass information extraction framework to automatically extract useful features. However, such a paradigm requires substantial effort to estimate parameters in the model, and therefore cannot handle large-scale resume extraction.

In our system, we simplify the extraction procedure by treating each resume as a text document, and then extracting important features from the text. Specifically, we sample 20% of resumes from the resume repository and transform them to plain text. To ensure the coverage of sampling, we sample resumes from different domains, e.g., *Internet Technology*, *Chemical Engineering* and *Business Management*, etc. We then select a pool of features (words) from each domain, according to the weight information of each feature. Here the weight is represented as the tf.idf value of the word. For each domain, we empirically choose 1,000 features and then finalize the extracted profile as a feature vector. We concatenate such vector to $\boldsymbol{p}^s$. Note that each user's $\boldsymbol{p}^s$ has an additional label, i.e., the domain name, for further comparison.

Behavioral Information

Besides the static information provided by users, the system has various ways to interact with users, and consequently collects users' behavioral activities. For instance, we provide functionalities such as searching and recommendation: users can feed some input, e.g., keywords, into the system, and then click on some preferred profiles from the search result; users can also receive recommendations from the system, and then choose some of them to view. Such behavioral information would be helpful to construct the preference of users and improve the quality of user profiles.

In our system, a user's activities, including searching and clicking, are automatically recorded and maintained in the format of log files. Table 6.2 shows an interpreted log file example of recruiters searching job applicants. Note that in *Search Criteria*,

Table 6.2: Sample log file.

User Identification	*Search Time*	*Search Criteria*	*Duration*	*Clicks*
ce20180c04c41580	09/Jan/2012 13:53:50	Project Manager, IT, 10	217	John, Michael
38f04d74e6511375	09/Jan/2012 13:54:32	Programmer, IT, 3	501	Chris, Ben, Shanny,$\cdots$
acef953fe42596a0	09/Jan/2012 13:55:02	Programmer & Java, IT, 2	432	Chris, Shanny, Murphy,$\cdots$
3110dbe2758556bf	09/Jan/2012 13:55:28	Data Analyst, Business, 8	283	Mary, Denver, Sam
72bf2bde4b64e457	09/Jan/2012 13:56:31	Consultant, Finance, 4	621	Jack, Dory, Devon,$\cdots$
437e052155f01a80	09/Jan/2012 13:57:09	Data Analyst, Business, 2	421	Mary, Sam, Denver, Nara,$\cdots$
21947eb595423a15	09/Jan/2012 13:58:50	Sales, Automobile, 3	142	Claydon, Chark, Edda, $\cdots$
3d5f469d3d7097d3	09/Jan/2012 13:59:29	Sales, Automobile, 5	239	Edda, Jamie

the three values correspond to *Keywords*, *Domain* and *Working years*, and the *Duration* means that the time that the user spends on reading candidates profiles. The log file in our system is parsed into two components: the *search interest* and the *click interest*. *Search interest* is obtained from the *Search Criteria*, presented by an interest vector, where each entry denotes a keyword associated with the domain, and the weight of the entry is the normalized duration. This vector is concatenated to the user's preference vector. *Click interest* is obtained from the *Clicks*. Such information is useful when we perform collaborative filtering based recommendation to individuals.

6.2.2 Recommendation

The recommendation module is designed for users who do not have definite preferences on either job positions or job applicants. For example, for job seekers who only have some *general career interest with a broad range of preferences*, the results generated purely based on the job search module might not be able to satisfy such users' appetite, i.e., the results need to be further refined to help the job seeker figure out his/her preferred jobs.

The recommendation module includes three different submodules, categorized by the recommendation techniques, i.e., *content-filtering* module, *collaborative filtering*

module, and *reciprocal recommendation* module. In the following, we will discuss the algorithmic details within each submodule.

Content Filtering

The principle of content filtering methods is to sequentially find items from the search result similar to the target user's preference in terms of "content". In iHR, the "content" refers to user profiles that are generated, including job seekers' and recruiters' profiles. Specific similarity measurements can be adopted to evaluate the relatedness between the target user and items in the search result, e.g., the relatedness between a job seeker and a series of job posts.

In this module, we focus on evaluating how relevant the users in the search result are to the target user. Formally, given a target user u's profile p_u and a set of search result $\mathcal{Q}_u$, our goal is to select a subset $\hat{\mathcal{Q}}_u \subset \mathcal{Q}_u$ such that $\forall v \in \hat{\mathcal{Q}}_u$, v's profile p_v is relevant to p_u in terms of a predefined relevance measurement. Under the scenario of job matching, we only consider the features that indicate the preference of the target user when calculating the relevance. For example, assume a job seeker u has his/her preference profile p_u^p, e.g., what types of jobs and which salary range that u prefers. After u feeds some keywords into the system, the system returns a set of job posts $\mathcal{Q}_u$. Within $\mathcal{Q}_u$, we rank all the candidates v based on the similarity

$$sim(p_u^p, p_v^s) = \frac{p_u^p \cdot p_v^s}{\|p_u^p\| \times \|p_v^s\|}, \tag{6.1}$$

and then choose the top ranked ones as the final recommendation result. Note that in our system p_u^p and p_v^s are processed to have the same cardinality.

Collaborative Filtering

Collaborative filtering methods are designed based on user's historical accessing behaviors, e.g., what kinds of job posts have been clicked by a job seeker before. It

considers "similar" users' accessing history, and then recommends to the target user a list of items that have been accessed by these "similar" users. Therefore, the key step is to find "similar" users in terms of the accessing history.

In our system, we expand the concept of collaborative filtering to a broader case, i.e., to consider some content information when calculating the similarity between users. Specifically, we take into account a user u's preference profile p_u^p and the search history profile p_u^h. The similarity between two users u and v (both are job seekers or recruiters), $sim(u, v)$, can be calculated as

$$sim(u, v) = \frac{p_u^p \cdot p_v^p}{\|p_u^p\| \cdot \|p_v^p\|} + \frac{p_u^h \cdot p_v^h}{\|p_u^h\| \cdot \|p_v^h\|} \tag{6.2}$$

After obtaining a list of users similar to the target user u, we sequentially check the search result to see if users in the list have accessed them or not. We then rank the search result based on the access count and recommend top ranked ones to the target user.

Reciprocal Recommendation

The aforementioned two modules do not take into account the properties of job matching systems. In this module, we analyze the special characteristics within the job matching domain, and then propose a reciprocal strategy. Recently, a special class of recommender systems, called reciprocal recommender, has emerged. Reciprocal recommender systems refer to systems from which users can obtain recommendations of other individuals by satisfying preferences of both users being involved. Examples of reciprocal recommenders include online dating services, mentor-mentee matching, consumer-to-consumer marketplaces, and etc. The job matching system is also a type of reciprocal recommender systems.

Properties in Reciprocal Recommenders

Reciprocity: In traditional user×item recommender systems, only unilateral preferences are considered, i.e., the users' preferences on items. However, in the domain of job matching, both users being involved in the recommendation have their preferences against with each other; in such a situation, only considering unilateral preference might not be reasonable. In other words, the success of a match depends on the bilateral preference, but not solely on the user who receives the recommendation. This is the key feature of a reciprocal recommender that differentiates it from the traditional user×item recommendation paradigm.

In our system, job seekers and recruiters have their self-descriptive information, and also the preferences on either job positions or applicants. Given a target user u and a search result list $\mathcal{Q}_u$, we are interested in finding a relevant user $v \in \mathcal{Q}_u$, such that (u,v) is a successful match. By *relevance*, we mean that the self-description of user v, p_v^s, matches the preference profile of user u, p_u^p, and at the meantime, the self-description of user u, p_u^s, matches the preference of user v, p_v^p. Therefore, the *relevance* includes two components, $rel(u \sim v)$ and $rel(v \sim u)$. Here the relevance is calculated using the cosine similarity between the vectors. We then formalize the relevance between u and v as $rel(u, v) = rel(u \sim v) \cdot rel(v \sim u)$.

Availability: In traditional recommenders, an item can be preferred by a great amount of users, e.g., a music album by the musician Michael Jackson. However, in a job matching system, people have limited availability towards other people, e.g., a job seeker cannot have 100 interviews with different companies simultaneously. Therefore, when designing the recommendation strategy for job matching systems, we need to consider the availability of users so that all the users can obtain reasonable recommendation results. In iHR, we intentionally record the number of times that a user has been recommended to other users. If a user has been recommended to

other users so many times in a time range, e.g. a week, we regard the availability of this user as low; similarly, if a user is a new registered user, then this user has higher availability. Particularly, for job seekers and recruiters, we set different thresholds to indicate the extent of their availability. For example, the threshold for job seekers is set to be 20, i.e., if the number of times that a job seeker is recommended to other users exceeds 20 in a time range, we will not recommend this job seeker any more. Similarly, the threshold for recruiters is set to be 50.

Diversity: In general, the search result contains a lot of records, some of which might be similar in terms of specific features. For example, two job candidates may have the same GPA and similar education background. If we recommend both of them to a recruiter, the recruiter may spend time and other resources to distinguish which candidate is better. In our system, we try to go beyond such established paradigm. Instead, we provide users diverse recommendation results to help them efficiently classify the candidates. Here "diverse" means that the candidates in the recommended result might exhibit different personal strengths.

Recommendation Methodology

In job matching system, when a user, e.g., a job seeker, searches job positions, he/she might have more preference on the top ranked results. When the user scrolls down the browser and clicks on the job posts, he/she might lose patience to view the details of the result. In other words, the interestingness of job posts with respect to a job seeker could be regressive (the situation also holds when a recruiter searches job candidates), which is known as the "submodularity". Hence in this recommendation module, based on the "submodularity", we model the recommendation problem as a budgeted maximum coverage problem [KMN99], and incorporate the special properties into the solution.

Introduction to Submodularity: Let E be a finite set and f be a real valued nondecreasing function defined on the subsets of E that satisfies

$$f(T \cup \{\varsigma\}) - f(T) \leqslant f(S \cup \{\varsigma\}) - f(S), \tag{6.3}$$

where $S \subseteq T$, S and T are two subsets of E, and $\varsigma \in E \setminus T$. Then f is called a **submodular** function [NWF78]. By adding one element to a larger set T, the value increment of f can never be larger than that by adding one element to a smaller set S. Submodularity modeling has been employed into multiple research areas, e.g., document summarization [LLL11, LB10], news recommendation [LWL+11], graph mining [THW+11], etc.

The budgeted maximum coverage problem is then described as: given a set of elements E where each element is associated with an influence and a cost defined over a domain of these elements and a budget B, the goal is to find out a subset of E which has the largest possible influence while the total cost does not exceed B. This problem is NP-hard as indicated in [KMN99]. However, [KMN99] proposed a greedy algorithm which picks up the element that increases the largest possible influence within the cost limit each time and guarantees the influence of the result subset is $(1 - 1/e)$-approximation. Submodularity resides in each "pick up" step. A key property is that submodular functions are closed under nonnegative linear combinations [LKG+07], which is useful to define a submodular function over several strategies.

Submodularity Model: In our recommendation model, we consider the properties including *Reciprocity*, *Availability* and *Diversity*. Given a target user u and a search result $\mathcal{Q}$, we try to sequentially select items from $\mathcal{Q}$ and then put them into a new set $\mathcal{S}$. The selection strategy can be described as follows (note that ς is the item being selected). After selection ς, we expect that (1) $\mathcal{S}$ should provide more relevance to the target user u; and (2) The diversity in $\mathcal{S}$ should not deviate too much. Based

on the above strategies, we define a function f to measure the quality of the current selected set $\mathcal{S}$ against $\mathcal{Q}$. The function f is defined as a linear combination of two submodular functions, described as

$$f(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{v_1 \in \mathcal{S}} \frac{1}{\tau} \cdot rel(u, v_1) + \frac{1}{\binom{|\mathcal{S}|}{2}} \sum_{\substack{v_1, v_2 \in \mathcal{S} \\ v_1 \neq v_2}} -sim(v_1, v_2), \tag{6.4}$$

where v_1 and v_2 denote users in $\mathcal{Q}$, τ indicates the availability of the user v_1, $rel(\cdot, \cdot)$ represents the relevance between two users (e.g., a job seeker and recruiter), and $sim(\cdot, \cdot)$ denotes the similarity between two users (e.g., two job seekers).

In Eq.(6.4), two components are involved corresponding to the user selection strategy listed above. The former aims to evaluate how relevant that the selected user set $\mathcal{S}$ is to the target user u, whereas the latter gives us the evidence that how diverse the selected set $\mathcal{S}$ is. Note that we use τ to indicate the availability of a user, i.e., the number of times that the user has been recommended to other users. We take the reverse of τ to reduce the possibility of the user being selected into $\mathcal{S}$. $f(\mathcal{S})$ balances the contribution of different components, and clearly the two components are naturally submodular functions. Based on the non-negative linear invariability of the submodularity function [LKG+07], $f(\mathcal{S})$ is also a submodular function.

Suppose ς is the candidate user, the quality increase is therefore represented as

$$I(\varsigma) = f(\mathcal{S} \cup \{\varsigma\}) - f(\mathcal{S}). \tag{6.5}$$

The goal is to select a list of users from $\mathcal{Q}$ with the largest possible quality increase under the budget. A greedy algorithm is employed to solve this problem. Note that in iHR, the budget is set to 100, i.e., to provide users at most 100 candidates by refining the search result.

Recommendation Fusion

The aforementioned recommendation methods capture different aspects of the relevant results. In our system, we provide a recommendation fusion strategy to integrate the recommended results. Specifically, different weights are assigned to the ranking scores of results obtained from the three methods. Formally, let r^{cot}, r^{cof} and r^{rep} denote the ranking scores from content filtering, collaborative filtering and reciprocal filtering, our recommendation fusion model towards selecting an item i can be described as

$$r_i = \alpha \times r_i^{cot} + \beta \times r_i^{cof} + \gamma \times r_i^{rep}, \tag{6.6}$$

where α, β and γ represent the weights of the corresponding scores, $s.t.\ \alpha+\beta+\gamma = 1$. Different weighting schemes of α, β and γ characterize different information needs of users:

- If α dominates, it indicates that a user has relatively clear preference on the result, and therefore the recommendation will be primarily based on this user's preference profile p^p;
- If β dominates, it indicates that a user does not have definite requirement on what information should be provided, and therefore the recommendation can be performed based on similar users' preferences;
- If γ dominates, it indicates that a user is concerned with the status of the recommended result, e.g., a job seeker will consider the availability of a position. The recommendation is achieved by considering the "reciprocal" property within a job matching community.

In our current system, the three scores are treated equally. Figure 6.4 shows some recommended results to a job seeker who is looking for "Manager" positions. The left

Figure 6.4: An illustrative example of the recommendation fusion model.

part of the figure describes the scores of the recommended job positions in a scale of 1-5.

6.2.3 Evaluation

Up to now, iHR has been deployed online for practical use, with over 699,600 visits per day. In our system, we propose a reciprocal recommendation method that emphasizes the bilateral correlations between job seekers and recruiters.

Empirical Evaluation

We perform quantitative evaluation on our proposed reciprocal recommendation strategy. The data used for experiments is a sampled data set collected from iHR, including the profiles and activities for users from Jan, 2008 to Oct, 2011. We calculate the user relevance based on the transformed feature space. The data statistics is depicted in Table 6.3.

Table 6.3: Statistics of the dataset.

Basic Statistics		training	testing
Job Seekers	199,999	176,423	23,576
Recruiters	46,629	29,850	6,779
# of p_u^s	860	–	–
# of p_u^p	928	–	–
# of p_v^s	928	–	–
# of p_v^p	860	–	–
Activities	664,943	493,128	171,815

Experiment Setup

For experiments, we split the data set into training and testing sets. Each set includes two sets of users, associated with their interactive activities, as shown in Table 6.3. For each user in the testing set, we recommend top ranked users (top@10, top@20 and top@30) at each week of the testing range using different strategies. Within the testing set, each user has a series of activities, e.g., adding job positions as favorite. Based on these activities, we use different metrics to evaluate the quality of the recommended list, as introduced in Section 2.2.

Set Evaluation: For comparison, we compute the averaged precision and recall based on users' activities. Specifically, the ground truth of a user u's activities, including who have been clicked or contacted by u, is denoted by M, and the recommended user list by algorithms is denoted by N. Then the precision (P) and recall (R) can be computed as

$$P = \frac{M \cap N}{N}, \quad R = \frac{M \cap N}{M}. \tag{6.7}$$

We then compute the F_1-score of the recommendation results, i.e., $F_1 = \frac{2PR}{P+R}$.

Ranking Evaluation: We employ Normalized Discount Cumulative Gain (NDCG) to evaluate the ranking quality of the recommended list based on a user's actual activity sequence. NDCG at position n is defined as

$$NDCG@n = N(n) \times \sum_{i=1}^{n} \frac{2^{r_i} - 1}{\log_2(i+1)}, \tag{6.8}$$

where $N(n)$ is the NDCG at n of the ideal ranking list, and r_i is the relevance rating of item at rank i. In our scenario, $r_i = 1$ if the user has clicked on or contacted with the recommended users and 0 otherwise.

Comparison with Other Methods

Here we only compare the reciprocal strategy in iHR with other existing methods. We choose two recently published collaborative filtering methods [HKV08, LHZC10] as our baselines. [HKV08] (CFIF for short) proposed treating the data as indication of positive and negative preference associated with vastly varying confidence levels, which is a pure collaborative filtering approach. [LHZC10] (OCCF for short) exploited the rich user information available in community-based interactive information systems, and incorporated user information into modeling the recommendation. For this method, we use the neighborhood model as the baseline. We also implement GBDTs [DMAY10], RECON [PRC+10] and CCR [AKY+11] for comparison. We use F_1-score and NDCG to compare these algorithms with iHR. The feature set used in all the methods are identical to the one in our proposed method, and also the parameters are optimally tuned.

Table 6.4: Comparison with existing methods. (The bold font indicates the best performance. * indicates the statistical significance at $p < 0.01$.)

Methods	top@10		top@20		top@30	
	F_1	NDCG	F_1	NDCG	F_1	NDCG
CFIF	0.2301	0.3174	0.3121	0.3813	0.3481	0.4036
OCCF	0.2485	0.3320	0.3219	0.3929	0.3569	0.4127
GBDTs	0.2567	0.3592	0.3304	0.4131	0.3718	0.4432
RECON	0.2604	0.3608	0.3247	0.4025	0.3839	0.4507
CCR	0.2431	0.3745	**0.3573**	0.3987	0.3912	0.4729
iHR	**0.2718***	**0.3720**	0.3501	**0.4316***	**0.4098***	**0.4875***

The results are shown in Table 6.4. It is evident that iHR significantly outperforms the baselines on both F_1-score and NDCG. The two collaborative filtering based

methods cannot effectively handle the reciprocal task. We investigated the recommendation results of both methods and found that users in most recommended matches are relevant. However, a significant reason that both users in a match have few or even no interactions is that the recommended user has been recommended to multiple users, and therefore he/she has limited availability. The three reciprocal methods being compared can slightly improve the recommendation performance; however, they only focus on different aspects of the reciprocal community. Instead, iHR with reciprocal recommendation provides a comprehensive overview of the reciprocal network, and therefore achieves the best.

A User Study

In order to evaluate the efficacy of iHR, we present a survey to each valid user to collect user experiences. The purpose of the survey is to evaluate how users feel about the results generated by different algorithms. These three methods are parallel in the system, i.e., users have the choice to check different recommendation results from different algorithms. The survey covers several aspects for evaluation, including *relevance*, *interpretability*, *diversity* and *ordering*. Sample questionnaire statements [PCH11] are listed in Table 6.5.

Based on these aspects, we define the corresponding indices to measure the satisfaction of online users (i.e., job seekers and recruiters). Each experience index is rated by users in a range of 1 to 5, where 1 – "Execrable", 2 – "Below Average", 3 – "Average", 4 – "Above Average", and 5 – "Exceptional". We collect users' feedbacks on these experience indices from October 2011 to January 2012. At the end of the evaluation period, we have obtained over 500,000 valid feedbacks from users. To analyze the experience result, we calculate the percentages of users with different ratings on the indices, and then plot them in Figure 6.5.

Table 6.5: Sample questionnaire statements used in our survey. (Remark: The scale is 1-5. 1 – worst, 5 – best. Reverse scale: 1 – best, 5 – worst.)

Aspects	*Statements*
Relevance	• The items in the list matched my interests. • The recommender gave me good suggestions. • I am not interested in the items recommended to me (reverse scale).
Interpret-ability	• The recommender explains why the candidates (recruiters) are recommended to me. • The recommender shows me details to help me digest the recommended results.
Diversity	• The items recommended to me are diverse. • The items recommended to me are similar to each other (reverse scale).
Ordering	• The recommended results maintain a relatively reasonable ordering. • The candidates (recruiters) that perfectly match my preference are listed at the top of the result.

From the result, we observe that the reciprocal recommendation method outperforms the other two methods in terms of user experience. Particularly, for the "Relevance" index, over 50% of users vote the reciprocal method as above average, thanks to the paradigm that considers the mutual relevance between job seekers and recruiters. For the "Interpretability" index, over 65% of users regard the recommendation result more interpretable, since we explicitly present the bilateral relations between job seekers and recruiters, which renders the result more explainable. For the "Diversity" and "Ordering" indices, over 51% and 54% of users vote the reciprocal method as above average. The underlying reason is that we elaborately design the selection strategy by considering the properties of job matching systems.

6.3 Concluding Remarks

This chapter presents an integrated recommendation framework for related applications. The framework contains three interleaved modules: *Profiling*, *Recommendation*

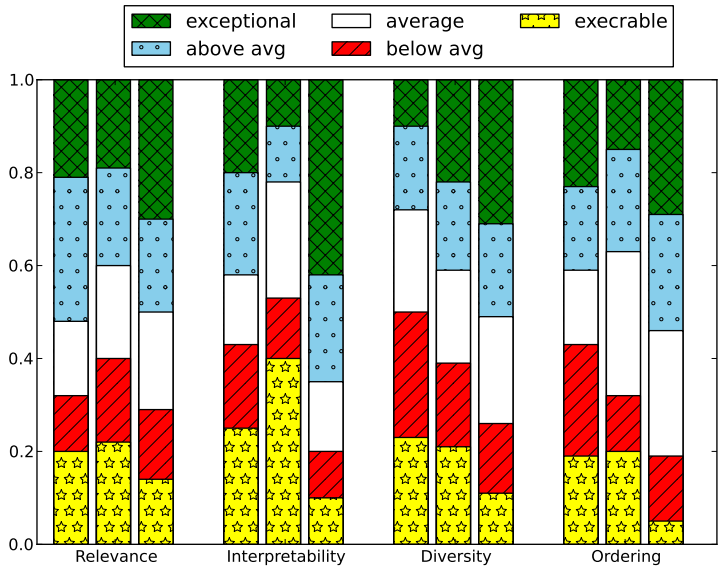


Figure 6.5: User experience results on different experience indices. For each index, Bar1 represents the results of content filtering, Bar2 shows the results of collaborative filtering, and Bar3 indicates the results of reciprocal recommendation.

and *Evaluation*, which operated in an iterative way. An representative application is described under the setting of this recommendation framework, which illustrates the efficacy of the framework.

CHAPTER 7

SUMMARY AND FUTURE WORK

Recommender systems aim to provide personalized recommendation of products or services to users and alleviate the problem of *information overload* on the web [AT05]. Based on customers' preferences or information needs, personalization can be achieved in these systems by comparing user profiles with items in a large item repository. The recommendation technologies enable customers to obtain interesting information, as well as service providers to acquire remarkable economic benefit. Hence, recommender systems play different roles in satisfying the requirements of both customers and service providers.

In this dissertation, we explored multiple user-oriented issues in the current generation of personalized recommender systems, including understanding user behaviors, complex user relations and users' changing preferences. By delving into these three related yet representative issues, we designed and developed corresponding solutions to these issues, and presented an integrated framework that synthesizes the solutions together. The recommendation framework is able to assist service providers in building an effective recommendation application.

The ultimate goal of this research, by tapping into three different yet interleaved issues that have not been well studied in personalized recommender systems, is to comprehensively understand user preferences for high-quality user profiling and reasonable recommendation. Different from the existing work that only tackles issues of single aspects of recommendation, the proposed work has the following merits with respect to personalized recommender systems:

1. Previous research efforts focus on modeling user clicks on a set of items [LDP10], whereas the click sequence is often being ignored. In this research, users' habits of using recommender systems can be effectively captured by understanding

accessing patterns of users. Based on the learned habits, system designers can have a clear sense of how to personalize each user's accessing experience. The understanding of user habits is essential for building user-friendly interactive systems [AW98].

2. Collaborative filtering in prior art mainly models the relations between users and items, e.g., using neighborhood-based methods [BK07a, DK11] or latent factors [BK07b, KBV09, Pat07]. However, modern recommender systems often involve complex correlations among high-dimensional user profiles and item characteristics. In this research, complex relations within recommender systems can be comprehensively understood in a user-oriented manner, and important relations with users involved can be distinguished for high-quality profile generation.

3. Prior recommendation approaches often formalize the temporal variability of user interests into a temporal factor [Kor10, XCH+10]. However in general, a user's preference over items would be relatively stable or vary slightly in a long-term period, whereas the content accessed by the user may change frequently in short terms. Simply using temporal factors to model interest dynamics may lose an overview of user preference. In this research, user inclination over items can be captured by integrating both long-term and short-term profiles.

By integrating these three research objectives, we are able to comprehensively understand the exact preferences of users, and construct high-quality user profiles. Built on top of it, the proposed recommendation framework is capable of capturing diverse information needs of users, and consequently providing reasonable and meaningful recommendations according to the well-constructed user profiles. Further, the proposed framework for personalized recommendation is a generalized solution to various recommendation domains. Despite of different characteristics of recommender

systems, users who are using these systems exhibit similar behaviors (e.g., click patterns, complex relations and dynamical interest drifts). Hence, the framework can handle recommendation cases of different systems.

The issues studied in this dissertation are all user-oriented issues. One interesting direction is to consider the contextual information of users when recommending items to them. Here the contextual information involves the contexts that the target user may have. For example, in a movie recommender system, a user may prefer to watch drama in the evening with his girlfriend, and action movies in the weekend. Hence, the preference of users may change with different contexts; modeling the contextual information into user profiles enable the recommendation engine to provide more reasonable recommendation results for users.

Another interesting direction to extend my work is to scale up the processes involved in the recommendation framework. Currently, the proposed solutions discussed in this dissertation operate on medium scale data sets. It is possible that when the scale of the data goes tens of millions or even billions, the performance of algorithms will be deteriorated. In [LWL+11], we have explored a scalable mechanism that utilizes the technique of *Locality Sensitive Hashing* to roughly partition the data into different clusters. This may work in applications that have the requirement to group items or users. However, in general scenarios, the recommendation framework depends heavily on machine learning algorithms, and therefore the key factor to improve the efficiency is originated from the infrastructure perspective, e.g., by virtue of advanced computing frameworks (such as Map-Reduce and Bulk Synchronous Parallel) combined with powerful computing clusters.

BIBLIOGRAPHY

[ABB06] S. Agarwal, K. Branson, and S. Belongie. Higher order learning with graphs. In *Proc. of ICML*, pages 17–24. ACM, 2006.

[AC10] Deepak Agarwal and Bee-Chung Chen. flda: matrix factorization through latent dirichlet allocation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 91–100. ACM, 2010.

[ACW12] Deepak Agarwal, Bee-Chung Chen, and Xuanhui Wang. Multi-faceted ranking of news articles using post-read actions. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 694–703. ACM, 2012.

[AHCSZ06] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM'06: Workshop on Link Analysis, Counter-terrorism and Security*, 2006.

[AHWY04] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu. A framework for projected clustering of high dimensional data streams. In *Proc. of VLDB*, pages 852–863, 2004.

[AKY+11] J. Akehurst, I. Koprinska, K. Yacef, L. Pizzato, J. Kay, and T. Rej. Ccr-a content-collaborative reciprocal recommender for online dating. In *Proc. of IJCAI*, pages 2199–2204, 2011.

[AP02] J.T. Addison and P. Portugal. Job search methods and outcomes. *Oxford Economic Papers*, 54(3):505, 2002.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[AT11] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. 2011.

[AW98] Ahmad M Ahmad Wasfi. Collecting user access patterns for building user profiles and collaborative filtering. In *Proceedings of the 4th international conference on Intelligent user interfaces*, pages 57–64. ACM, 1998.

[BDN11] Kliček Božidar, Oreški Dijana, and Begičević Nina. Temporal recommender systems. In *Proceedings of the 10th WSEAS international conference on Applied computer and applied computational science*, pages 248–253. World Scientific and Engineering Academy and Society (WSEAS), 2011.

[BHK98] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[BHS06] Erik Brynjolfsson, Yu Jeffrey Hu, and Michael D Smith. From niches to riches: The anatomy of the long tail. 2006.

[BK07a] Robert M Bell and Yehuda Koren. Improved neighborhood-based collaborative filtering. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. sn, 2007.

[BK07b] Robert M Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 43–52. IEEE, 2007.

[BKV07] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104. ACM, 2007.

[BNJ03] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[BP99] D. Billsus and M.J. Pazzani. A hybrid user model for news story classification. *Internatinal Center for Mechanical Sciences*, pages 99–108, 1999.

[BPC00] Daniel Billsus, Michael J Pazzani, and James Chen. A learning agent for wireless news access. In *Proceedings of the 5th international conference on Intelligent user interfaces*, pages 33–36. ACM, 2000.

[BS97] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[BTC+10] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. Music recommendation by unified hypergraph: combining social media information and music content. In *Proceedings of the international conference on Multimedia*, pages 391–400. ACM, 2010.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[CBC08] Iván Cantador, Alejandro Bellogín, and Pablo Castells. Ontology-based personalised and context-aware recommendations of news items. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 1, pages 562–565. IEEE, 2008.

[CCL+09] W.Y. Chen, J.C. Chu, J. Luan, H. Bai, Y. Wang, and E.Y. Chang. Collaborative filtering for orkut communities: discovery of user latent behavior. In *Proc. of WWW*, pages 681–690. ACM, 2009.

[CCYX09] Huanhuan Cao, Enhong Chen, Jie Yang, and Hui Xiong. Enhancing recommender systems under volatile userinterest drifts. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1257–1266. ACM, 2009.

[CMBT02] D.H. Cunningham, D.D. Maynard, D.K. Bontcheva, and M.V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of ACL*, 2002.

[CP09] W. Chu and S.T. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th International Conference on World Wide Web*, pages 691–700. ACM, 2009.

[CZC08] W.Y. Chen, D. Zhang, and E.Y. Chang. Combinational collaborative filtering for personalized community recommendation. In *Proc. of SIGKDD*, pages 115–123. ACM, 2008.

[CZL+11] Tianqi Chen, Zhao Zheng, Qiuxia Lu, Weinan Zhang, and Yong Yu. Feature-based matrix factorization. *arXiv preprint arXiv:1109.2271*, 2011.

[DDGR07] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

[DGM08] Souvik Debnath, Niloy Ganguly, and Pabitra Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th international conference on World Wide Web*, pages 1041–1042. ACM, 2008.

[DI99] Joaquin Delgado and Naohiro Ishii. Memory-based weighted majority prediction. In *SIGIR Workshop Recomm. Syst. Citeseer*, 1999.

[DK04] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[DK11] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[DL05] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492. ACM, 2005.

[DLS07] Marco Degemmis, Pasquale Lops, and Giovanni Semeraro. A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. *User Modeling and User-Adapted Interaction*, 17(3):217–255, 2007.

[DMAY10] F. Diaz, D. Metzler, and S. Amer-Yahia. Relevance and ranking in online dating systems. In *Proc. of SIGIR*, pages 66–73, 2010.

[DS84] Peter G Doyle and J Laurie Snell. *Random walks and electric networks*, volume 22. Mathematical association of America, 1984.

[Dun73] J.C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Cybernetics and Systems*, 3(3):32–57, 1973.

[DY08] L. Ding and A. Yilmaz. Image segmentation as learning on hypergraphs. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*, pages 247–252. IEEE, 2008.

[For10] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[GDH04] Evgeniy Gabrilovich, Susan Dumais, and Eric Horvitz. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *Proceedings of the 13th international conference on World Wide Web*, pages 482–490. ACM, 2004.

[GFMG06] Miha Grčar, Blaž Fortuna, Dunja Mladenič, and Marko Grobelnik. knn versus svm in the collaborative filtering framework. In *Data Science and Classification*, pages 251–260. Springer, 2006.

[GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, 1999.

[GP06] Marco Gori and Augusto Pucci. Research paper recommender systems: A random-walk based approach. In *Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on*, pages 778–781. IEEE, 2006.

[GP07] Marco Gori and Augusto Pucci. Itemrank: A random-walk based scoring algorithm for recommender engines. In *IJCAI*, volume 7, pages 2766–2771, 2007.

[GS04] T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[GV00] K. George and K. Vipin. Multilevel k-way hypergraph partitioning. *VLSI Design*, 11(3):285–300, 2000.

[Hav02] Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM, 2002.

[HKBR99] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[HKTR04] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[HLL12] Wenxing Hong, Lei Li, and Tao Li. Product recommendation with temporal dynamics. *Expert Systems with Applications*, 39(16):12398–12406, 2012.

[HLLP13] Wenxing Hong, Lei Li, Tao Li, and Wenfu Pan. ihr: an online recruiting system for xiamen talent service center. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1177–1185. ACM, 2013.

[HLZM] Y. Huang, Q. Liu, S. Zhang, and D.N. Metaxas. Image retrieval via probabilistic hypergraph ranking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3376–3383. IEEE.

[Hof99] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 50–57. ACM, 1999.

[Hof03] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual*

international ACM SIGIR conference on Research and development in informaion retrieval, pages 259–266. ACM, 2003.

[Hof04] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

[HSS01] Uri Hanani, Bracha Shapira, and Peretz Shoval. Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction*, 11(3):203–259, 2001.

[HTKK08] T.H. Hwang, Z. Tian, R. Kuangy, and J.P. Kocher. Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction. In *2008 Eighth IEEE International Conference on Data Mining*, pages 293–302. IEEE, 2008.

[ISY06] Tomoharu Iwata, Kazumi Saito, and Takeshi Yamada. Recommendation method for extending subscription periods. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 574–579. ACM, 2006.

[ISY07] Tomoharu Iwata, Kazumi Saito, and Takeshi Yamada. Modeling user behavior in recommender systems based on maximum entropy. In *Proceedings of the 16th international conference on World Wide Web*, pages 1281–1282. ACM, 2007.

[JE09] Mohsen Jamali and Martin Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 397–406. ACM, 2009.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[JNT10] J. Jancsary, F. Neubarth, and H. Trost. Towards context-aware personalization and a broad perspective on the semantics of news articles. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 289–292. ACM, 2010.

[Joa98] Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features.* Springer, 1998.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[KC03] Hyoung R Kim and Philip K Chan. Learning implicit user interest hierarchy for context in personalization. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 101–108. ACM, 2003.

[KL51] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[KM03] Jeremy Z Kolter and Marcus A Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 123–130. IEEE, 2003.

[KMM+97] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[KMN99] S. Khuller, A. Moss, and J.S. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[Kor08] 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. ACM, 2008.

[Kor09] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456. ACM, 2009.

[Kor10] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[KS00] Ivan Koychev and Ingo Schwab. Adaptation to drifting user's interests. In *Proceedings of ECML2000 Workshop: Machine Learning in New Information Age*, pages 39–46, 2000.

[KSKÇ13] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V Çatalyürek. Diversified recommendation on graphs: pitfalls, measures, and algorithms. In *Proceedings of the 22nd international conference on World Wide Web*, pages 715–726, 2013.

[LB10] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Proc. of NAACL-HLT*, pages 912–920. Association for Computational Linguistics, 2010.

[LC12] Xin Li and Hsinchun Chen. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems*, 2012.

[LCGF10] Hangzai Luo, Peng Cai, Wei Gong, and Jianping Fan. Semantic entity-relationship model for large-scale multimedia news exploration and recommendation. In *Advances in Multimedia Modeling*, pages 522–532. 2010.

[LCLS10] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[LdGS11] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. 2011.

[LDJ+09] Ming Li, Benjamin M Dias, Ian Jarman, Wael El-Deredy, and Paulo JG Lisboa. Grocery shopping recommendations based on basket-sensitive random walk. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1215–1224. ACM, 2009.

[LDP10] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.

[LHC09] Neal Lathia, Stephen Hailes, and Licia Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 796–797. ACM, 2009.

[LHCA10] 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. ACM, 2010.

[LHL12] Lei Li, Wenxing Hong, and Tao Li. Taxonomy-oriented recommendation towards recommendation with stage. In *Web Technologies and Applications*, pages 219–230. Springer, 2012.

[LHZC10] Y. Li, J. Hu, C.X. Zhai, and Y. Chen. Improving one-class collaborative filtering by incorporating rich user information. In *Proc. of CIKM*, pages 959–968. ACM, 2010.

[Lin98] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th international conference on machine learning*, volume 1, pages 296–304, 1998.

[LKG+07] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429. ACM, 2007.

[LL11] D. Li and S. Li. Hypergraph-based inductive learning for generating implicit key phrases. In *Proceedings of the 20th international conference companion on World wide web*, pages 77–78. ACM, 2011.

[LL12] Lei Li and Tao Li. Meet: a generalized framework for reciprocal recommender systems. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 35–44. ACM, 2012.

[LL13] Lei Li and Tao Li. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 305–314. ACM, 2013.

[LLL11] J. Li, L. Li, and T. Li. Mssf: a multi-document summarization framework based on submodularity. In *Proc. of SIGIR*, pages 1247–1248. ACM, 2011.

[LNK07] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[LPK+13] Lei Li, Wei Peng, Saurabh Kataria, Tong Sun, and Tao Li. Frec: A novel framework of recommending users and communities in social media. In *Proceedings of the 22st ACM international conference on Information and knowledge management*. ACM, 2013.

[LPP07] Tong Queue Lee, Young Park, and Yong-Tae Park. A similarity measure for collaborative filtering with implicit feedback. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, pages 385–397. Springer, 2007.

[LPP08] Tong Queue Lee, Young Park, and Yong-Tae Park. A time-based approach to effective recommender systems using implicit feedback. *Expert systems with applications*, 34(4):3055–3062, 2008.

[LSMW98] Page Lawrence, Brin Sergey, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[LSY03] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[LWL+11] Lei Li, Dingding Wang, Tao Li, Daniel Knox, and Balaji Padmanabhan. Scene: a scalable two-stage personalized news recommendation system. In *SIGIR*, pages 125–134, 2011.

[LY08] Nathan N Liu and Qiang Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90. ACM, 2008.

[LZL11] Lei Li, Li Zheng, and Tao Li. Logo: a long-short user interest integration in personalized news recommendation. In *Proceedings*

of the fifth ACM conference on Recommender systems, pages 317–320. ACM, 2011.

[LZYL14] Lei Li, Li Zheng, Fan Yang, and Tao Li. Modeling and broadening temporal user interest in personalized news recommendation. *Expert Systems with Applications*, 41(7):3168–3177, 2014.

[MGÁRLGMM13] Alejandro Montes-García, Jose María Álvarez-Rodríguez, Jose Emilio Labra-Gayo, and Marcos Martínez-Merino. Towards a journalist-based news recommendation system: The wesomender approach. *Expert Systems with Applications*, 40(17):6735–6741, 2013.

[MKR03] Batul J Mirza, Benjamin J Keller, and Naren Ramakrishnan. Studying recommendation algorithms by graph analysis. *Journal of Intelligent Information Systems*, 20(2):131–160, 2003.

[MLDO07] Shanle Ma, Xue Li, Yi Ding, and Maria E Orlowska. A recommender system with interest-drifting. In *Web Information Systems Engineering–WISE 2007*, pages 633–642. 2007.

[MMN02] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI*, pages 187–192, 2002.

[MMO08] Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. Metrics for evaluating the serendipity of recommendation lists. In *New frontiers in artificial intelligence*, pages 40–46. Springer, 2008.

[MRS08] C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*, volume 1. Cambridge University Press, 2008.

[MSDR04] Stuart E Middleton, Nigel R Shadbolt, and David C De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, 2004.

[MZ04] Benjamin Marlin and Richard S Zemel. The multiple multiplicative factor model for collaborative filtering. In *Proceedings of the twenty-first international conference on Machine learning*, page 73. ACM, 2004.

[NA98] Atsuyoshi Nakamura and Naoki Abe. Collaborative filtering using weighted majority prediction algorithms. In *ICML*, volume 98, pages 395–403, 1998.

[NFT+10] M. Nakatsuji, Y. Fujiwara, A. Tanaka, T. Uchiyama, K. Fujimura, and T. Ishida. Classical music for rock fans?: novel recommendations for expanding user interests. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 949–958. ACM, 2010.

[NWF78] GL Nemhauser, LA Wolsey, and ML Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[OKMA07] Chihiro Ono, Mori Kurokawa, Yoichi Motomura, and Hideki Asoh. A context-aware movie preference model using a bayesian network for recommendation and promotion. In *User Modeling 2007*, pages 247–257. 2007.

[OTF09] Kensuke Onuma, Hanghang Tong, and Christos Faloutsos. Tangent: a novel,'surprise me', recommendation algorithm. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 657–666. ACM, 2009.

[Pat07] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[PB97] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

[PB07] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. 2007.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[PCH11] P. Pu, L. Chen, and R. Hu. A user-centric evaluation framework for recommender systems. In *Proc. of RecSys*, pages 157–164. ACM, 2011.

[PG11] M. Pennacchiotti and S. Gurumurthy. Investigating topic models for social media user recommendation. In *Proc. of WWW*, pages 101–102. ACM, 2011.

[POM09] M.A. Porter, J.P. Onnela, and P.J. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.

[PPL01] 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 Seventeenth conference on Uncertainty in artificial intelligence*, pages 437–444, 2001.

[PRC+10] L. Pizzato, T. Rej, T. Chung, I. Koprinska, and J. Kay. Recon: a reciprocal recommender for online dating. In *Proc. of Recommender Systems*, pages 207–214. ACM, 2010.

[PTVF92] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. Numerical recipes in c: The art of scientific computing (; cambridge, 1992.

[PU03] A. Popescul and L.H. Ungar. Statistical relational learning for link prediction. In *IJCAI workshop on learning statistical models from relational data*, volume 2003, 2003.

[PYFD04] J.Y. Pan, H.J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 653–658. ACM, 2004.

[Res95] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. *Arxiv preprint cmp-lg/9511007*, 1995.

[RIS+94] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[RN07] Francesco Ricci and Quang Nhat Nguyen. Acquiring and revising preferences in a critique-based mobile recommender system. *Intelligent Systems, IEEE*, 22(3):22–29, 2007.

[RRS11] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[RV97] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[RZCG+10] M. Rosen-Zvi, C. Chemudugunta, T. Griffiths, P. Smyth, and M. Steyvers. Learning author-topic models from text corpora. *ACM Transactions on Information Systems*, 28(1):4, 2010.

[RZGSS04] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proc. of UAI*, pages 487–494, 2004.

[SBR02] B. Smyth, K. Bradley, and R. Rafter. Personalization techniques for online recruitment services. *Communications of the ACM*, 45(5):39–40, 2002.

[SCFS12] M. Sachan, D. Contractor, T.A. Faruquie, and L.V. Subramaniam. Using content and interactions for discovering communities in social networks. In *Proc. of WWW*, pages 331–340. ACM, 2012.

[SFPY07] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696. ACM, 2007.

[SG11] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[SJY08] L. Sun, S. Ji, and J. Ye. Hypergraph spectral learning for multi-label classification. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 668–676. ACM, 2008.

[SKKR00] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[SKKR01] 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. ACM, 2001.

[SKP+13] Jeong-Woo Son, A Kim, Seong-Bae Park, et al. A location-based news article recommendation with explicit localized semantic analysis. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 293–302. ACM, 2013.

[SKR99] J Ben Schafer, Joseph Konstan, and John Riedi. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[Sot11] Pedro G. Campos Soto. Temporal models in recommender systems: An exploratory study on different evaluation dimensions. Master's thesis, Universidad Autónoma de Madrid, 2011.

[SPUP02] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[STF06] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383. ACM, 2006.

[SZF07] Vincent Schickel-Zuber and Boi Faltings. Using hierarchical clustering for learning theontologies used in recommendation systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 599–608. ACM, 2007.

[THK09] Z. Tian, T.H. Hwang, and R. Kuang. A hypergraph-based learning algorithm for classifying gene expression and arraycgh data with prior knowledge. *Bioinformatics*, 25(21):2831, 2009.

[THW+11] Hanghang Tong, Jingrui He, Zhen Wen, Ravi Konuru, and Ching-Yung Lin. Diversified ranking on large graphs: an optimization

viewpoint. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1028–1036. ACM, 2011.

[TPNT09] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656, 2009.

[WB08] Derry Tanti Wijaya and Stéphane Bressan. A random walk on the red carpet: rating movies with user reviews and pagerank. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 951–960. ACM, 2008.

[WB11] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.

[WHZ10] F. Wu, Y.H. Han, and Y.T. Zhuang. Multiple hypergraph clustering of web images by miningword2image correlations. *Journal of Computer Science and Technology*, 25(4):750–760, 2010.

[WK07] G. Wachman and R. Khardon. Learning from interpretations: a rooted kernel for ordered hypergraphs. In *Proceedings of the 24th international conference on Machine learning*, pages 943–950. ACM, 2007.

[WMYL06] Fei Wang, Sheng Ma, Liuzhong Yang, and Tao Li. Recommendation on item graphs. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 1119–1123. IEEE, 2006.

[WQF07] Bing Wu, Luo Qi, and Xiong Feng. Personalized recommendation algorithm based on svm. In *Communications, Circuits and Systems, 2007. ICCCAS 2007. International Conference on*, pages 951–953. IEEE, 2007.

[WTZ10] Ziqi Wang, Yuwei Tan, and Ming Zhang. Graph-based recommendation on social networks. In *Web Conference (APWEB), 2010 12th International Asia-Pacific*, pages 116–122. IEEE, 2010.

[WXLN07] Li-Tung Weng, Yue Xu, Yuefeng Li, and Richi Nayak. Improving recommendation novelty based on topic taxonomy. In *Web Intelligence and Intelligent Agent Technology Workshops, 2007 IEEE/WIC/ACM International Conferences on*, pages 115–118. IEEE, 2007.

[XCH+10] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222, 2010.

[XYZ+10] 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. ACM, 2010.

[XZWY12] Zhiheng Xu, Yang Zhang, Yao Wu, and Qing Yang. Modeling user posting behavior on social media. In *Proc. of SIGIR*, pages 545–554. ACM, 2012.

[Yaj06] Yasutoshi Yajima. One-class support vector machines for recommendation tasks. In *Advances in Knowledge Discovery and Data Mining*, pages 230–239. 2006.

[YGZ05] K. Yu, G. Guan, and M. Zhou. Resume information extraction with cascaded hybrid model. In *Proc. of ACL*, pages 499–506. ACL, 2005.

[YJCZ09] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *Proc. of SIGKDD*, pages 927–936. ACM, 2009.

[YK08] Hilmi Yildirim and Mukkai S Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 131–138. ACM, 2008.

[YYT06] K. Yu, S. Yu, and V. Tresp. Soft clustering on graphs. *Advances in Neural Information Processing Systems*, 18:1553–1560, 2006.

[ZBL+04] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16:321–328, 2004.

[ZGVGA07] Xiaojin Zhu, Andrew B Goldberg, Jurgen Van Gael, and David Andrzejewski. Improving diversity in ranking using absorbing random walks. In *HLT-NAACL*, pages 97–104, 2007.

[ZH08] Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 123–130. ACM, 2008.

[ZHS05] D. Zhou, J. Huang, and B. Scholkopf. Beyond pairwise classification and clustering using hypergraphs. Technical report, 2005.

[ZHS07] D. Zhou, J. Huang, and B. Scholkopf. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in Neural Information Processing Systems*, 19:1601, 2007.

[ZK12] Lawrence Zitnick and Takeo Kanade. Maximum entropy for collaborative filtering. *arXiv preprint arXiv:1207.4152*, 2012.

[ZLHL13] Li Zheng, Lei Li, Wenxing Hong, and Tao Li. Penetrate: Personalized news recommendation using ensemble hierarchical clustering. *Expert Systems with Applications*, 40:2127–2136, 2013.

[ZLST04] C.N. Ziegler, G. Lausen, and L. Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 406–415. ACM, 2004.

[ZML+06] D. Zhou, E. Manavoglu, J. Li, C.L. Giles, and H. Zha. Probabilistic models for discovering e-communities. In *Proc. of WWW*, pages 173–182, 2006.

[ZRMZ07] Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4):046115, 2007.

[ZSH05] D. Zhou, B. Schölkopf, and T. Hofmann. Semi-supervised learning on directed graphs. *Advances in neural information processing systems*, pages 1633–1640, 2005.

VITA

LEI LI

Aug 2009 – Now	Ph.D., Computer Science Florida International University Miami, Florida
Sep 2005 – Jan 2008	M.E., Software Engineering Beihang University Beijing, China
Sep 2000 – Jul 2004	B.S., Electrical Engineering Beihang University Beijing, China

PUBLICATIONS

Tao Li and Lei Li. Music Data Mining: An Introduction. Music Data Mining, Pages 3-42. CRC Press, 2011, ISBN 9781439835524.

Lei Li, Li Zheng, Fan Yang and Tao Li. Modeling and Broadening Temporal User Interest in Personalized News Recommendation. Expert Systems with Applications 41(7), Pages 3168-3177, 2014.

Lei Li and Tao Li. An Empirical Study of Ontology-Based Multi-Document Summarization in Disaster Management. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2014. In Press.

Li Zheng, Lei Li and Tao Li. PENETRATE: Personalized News Recommendation Using Ensemble Hierarchical Clustering. Expert Systems with Applications, Pages 2127-2136, 2013.

Jingxuan Li, Lei Li and Tao Li. Multi-document summarization via submodularity. Applied Intelligence, Pages 1-11, 2012.

Wenxing Hong, Lei Li and Tao Li. Product recommendation with temporal dynamics. Expert Systems with Applications, Pages 12398-12406, 2012.

Lei Li, Dingding Wang, Shunzhi Zhu and Tao Li. Personalized News Recommendation: A Review and An Experimental Investigation. Journal of Computer Science and Technology, Pages 754-766, 2011.

Lei Li, Wei Peng, Saurabh Kataria, Tong Sun and Tao Li. FRec: A Novel Framework of Recommending Users and Communities in Social Media. In Proceedings of the 22st ACM International Conference on Information and Knowledge Management, Pages 1765-1770, 2013.

Wenxing Hong, Lei Li, Tao Li and Wenfu Pan. iHR: an online recruiting system for Xiamen Talent Service Center. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, Pages 1177-1185, 2013.

Lei Li and Tao Li. News Recommendation via Hypergraph Learning: Encapsulation of User Behavior and News Content. In Proceedings of the 6th ACM International Conference on Web Search and Data Mining, Pages 305-314, 2013.

Lei Li and Tao Li. MEET: A Generalized Framework for Reciprocal Recommender Systems. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Pages 35-44, 2012.

Chen Lin, Runquan Xie, Lei Li, Zhenhua Huang and Tao Li. PRemiSE: Personalized News Recommendation via Implicit Social Experts. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Pages 1607-1611, 2012.

Lei Li, Wenxing Hong and Tao Li. Taxonomy-oriented recommendation towards recommendation with stage. In Proceedings of the 14th Asia-Pacific Web Conference, Pages 219-230, 2012.

Lei Li, Li Zheng and Tao Li. LOGO: A Long-Short User Interest Integration in Personalized News Recommendation. In Proceedings of the 5th ACM Conference on Recommender Systems, Pages 317-320.

Lei Li, Dingding Wang, Tao Li, Daniel Knox and Balaji Padmanabhan. SCENE : A Scalable Two-Stage Personalized News Recommendation System. In Proceedings of the 34th International ACM SIGIR conference on Research and Development in Information Retrieval, Pages 125-134, 2011.

Lei Li, Dingding Wang, Chao Shen and Tao Li. Ontology-Enriched Multi-document Summarization in Disaster Management. In Proceedings of the 33rd International ACM SIGIR conference on Research and Development in Information Retrieval, Pages 819-820, 2010.