Enhancing Collaborative Filtering by User Interest Expansion via Personalized Ranking

Qi Liu, Enhong Chen, *Senior Member, IEEE*, Hui Xiong, *Senior Member, IEEE*, Chris H. Q. Ding, *Member, IEEE*, and Jian Chen, *Fellow, IEEE*

***Abstract*—Recommender systems suggest a few items from many possible choices to the users by understanding their past behaviors. In these systems, the user behaviors are influenced by the hidden interests of the users. Learning to leverage the information about user interests is often critical for making better recommendations. However, existing collaborative-filtering-based recommender systems are usually focused on exploiting the information about the user's interaction with the systems; the information about latent user interests is largely underexplored. To that end, inspired by the topic models, in this paper, we propose a novel collaborative-filtering-based recommender system by user interest expansion via personalized ranking, named iExpand. The goal is to build an item-oriented model-based collaborative-filtering framework. The iExpand method introduces a three-layer, user–interests–item, representation scheme, which leads to more accurate ranking recommendation results with less computation cost and helps the understanding of the interactions among users, items, and user interests. Moreover, iExpand strategically deals with many issues that exist in traditional collaborative-filtering approaches, such as the *overspecialization* problem and the cold-start problem. Finally, we evaluate iExpand on three benchmark data sets, and experimental results show that iExpand can lead to better ranking performance than state-of-the-art methods with a significant margin.**

***Index Terms*—Collaborative filtering, latent Dirichlet allocation (LDA), personalized ranking, recommender systems, topic model.**

I. Introduction

THE DEVELOPMENT of recommender systems has been stimulated by the rapid growth of information on the Internet. For information filtering, recommender systems can automatically recommend the few optimal items, which users might like or have interests to buy by learning the user profiles, users' previous transactions, the content of items, etc. [2]. In the recent 20 years, many different types of recommender systems, such as collaborative-filtering-based methods [36], content-based approaches [12], and hybrid approaches [46], have been developed.

A. Collaborative Filtering

Since collaborative-filtering methods only require the information about user interactions and do not rely on the content information of items or user profiles, they have more broad applications [14], [16], [20], and more and more research studies on collaborative filtering have been reported [15], [26], [27]. These methods filter or evaluate items through the opinions of other users [41]. They are usually based on the assumption that the given user will prefer the items which other users with similar preferences liked in the past [2].

In the literature, there are model-based and memory-based methods for collaborative filtering. Model-based approaches learn a model to make recommendation. Algorithms of this category include the matrix factorization [38], the graph-based approaches [14], etc. The common procedure of memory-based approaches is first to select a set of neighbor users for a given user based on the entire collection of previously rated items by the users. Then, the recommendations are made based on the items that neighbor users like. Indeed, these methods are referred to as user-oriented memory-based approaches. In addition, an analogous procedure, which builds item similarity groups using corating history, is known as item-oriented memory-based collaborative filtering [40].

However, existing collaborative-filtering methods often directly exploit the information about the users' interaction with the systems. In other words, they make recommendations by learning a "*user–item*" dualistic relationship. Therefore, existing methods often neglect an important fact that there are many latent user interests which influence user behaviors. To that end, in this paper, we propose a three-layer, user–interests–item, representation scheme. Specifically, we interpret an interest as a requirement from the user to items, while for the corresponding item, the interest can be considered as one of its characteristics. Indeed, it is necessary to leverage this three-layer representation for enhancing collaborative filtering, since this representation leads to better explanation of why recommended items are chosen and helps the understanding of the interactions among users, items, and user interests.

Manuscript received November 8, 2010; revised March 24, 2011 and June 24, 2011; accepted July 12, 2011. Date of current version December 7, 2011. This research was supported in part by the Natural Science Foundation of China under Grants 60775037 and 71028002, by the Key Program of National Natural Science Foundation of China under Grant 60933013, and by the Research Fund for the Doctoral Program of High Education of China under Grant 20093402110017. A preliminary version of this work has been published in the Association for Computing Machinery Conference on Information and Knowledge Management 2010. This paper was recommended by Associate Editor J. Liu.

Q. Liu and E. Chen are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: feiniaol@mail.ustc.edu.cn; cheneh@ustc.edu.cn).

H. Xiong is with the Management Science and Information Systems Department, Rutgers Business School, Rutgers University, Newark, NJ 07102 USA (e-mail: hxiong@rutgers.edu).

C. H. Q. Ding is with the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019 USA (e-mail: chqding@uta.edu).

J. Chen is with the Department of Management Science and Engineering, School of Economics and Management, Tsinghua University, Beijing 100084, China (e-mail: chenj@sem.tsinghua.edu.cn).

Digital Object Identifier 10.1109/TSMCB.2011.2163711

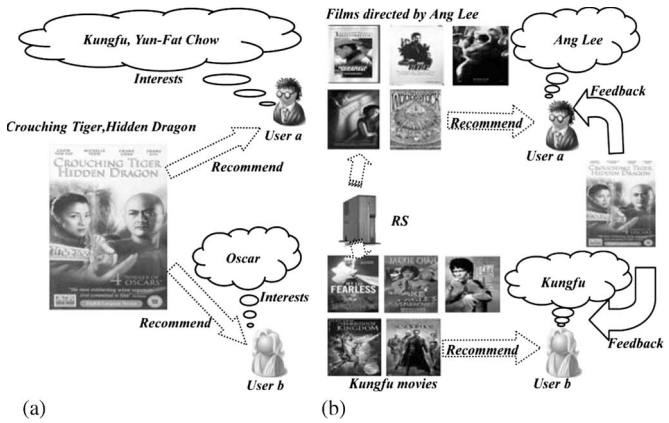


Fig. 1. Simple example of a movie recommender system (where the photos are downloaded from IMDB [http://www.imdb.com/)]. (a) When users decide to watch a movie, there are some latent interests that affect their choices. (b) Users' interests may change after they watch a movie.

B. Motivating Example

Fig. 1(a) shows an example of a movie recommender system. In the figure, user a is interested in *kung fu* movies, while user b likes *Oscar* movies. While both of them have watched the movie *Crouching Tiger, Hidden Dragon*, which was recommended by the system, they have different reasons for watching this movie. Thus, if we can identify user latent interests, we will have a better understanding about the users' requirements, since user interests can better connect users and items. Also, when leveraging the information of user interests for developing recommender systems, we must be aware that user interests can change from time to time under the influence of many internal and external factors. For instance, as shown in Fig. 1(b), after watching the movie *Crouching Tiger, Hidden Dragon*, user interests may be affected by it. For user a, while he is a fan of *kung fu* movies, he may start watching other movies directed by *Ang Lee*. Also, user b may become a fan of *kung fu* movies after her first-time exposure to this *kung fu* movie. If recommender systems cannot capture these changes and only make recommendations according to the user's past interests rather than exploring his/her new preferences, then they are prone to the "*overspecialization*" problem [2].

In addition, in real scenarios, the training data are far less than plentiful and most of the items/users only have a few rating/buying records. At this time, typical measures fail to capture actual similarities between items/users and the system is unable to make meaningful recommendations. This situation is summarized as the cold-start problem [41]. Let us take user b in Fig. 1(a) as an example. If she has only watched one movie *Crouching Tiger, Hidden Dragon* and it has been watched by few people before the rating of user b, for traditional collaborative-filtering systems, it is difficult to find out the similar items or users for both *Crouching Tiger, Hidden Dragon* and user b. However, if we have identified that *Crouching Tiger, Hidden Dragon* belongs to *kung fu* movies and *Oscar* movies, then the system could recommend user b the movies that belong to these two interests or some related interests. Thus, the cold-start problem can be alleviated.

Indeed, the key challenges are how to model latent user interests and the potential correlations and changes between them. It is relatively easy to extract user interests in content-based or hybrid recommender systems by tracking the text information, such as "*keywords*" or "*tags*" [43]. However, for collaborative-filtering systems, it is difficult to identify the user latent interests, since the only information available is the user interaction information with the system.

C. Contributions

To address the aforementioned challenges, in our preliminary work [31], we proposed an item-oriented model-based collaborative-filtering method named iExpand. In iExpand, we assume that each user's rating behavior depends on an underlying set of hidden interests and we use a three-layer, user–interests–item, representation scheme to generate recommendations. Specifically, each user interest is first captured by a latent factor which corresponds to a "*topic*" in topic models. Then, we learn the transition probabilities between different latent interests. Moreover, to deal with the cold-start and "*overspecialization*" problems, we model the possible expansion process of user interests by personalized ranking. In other words, we exploit a personalized ranking strategy on a latent interest correlation graph to predict the next possible interest for each user. At last, iExpand generates the recommendation list by ranking the candidate items according to the expanded user interests. We should note that, compared with previous topic-model-based collaborative-filtering approaches, discovering the correlation between latent interests and using personalized ranking to expand user current interests are the main advantages of iExpand.

In addition, in many previous model-based recommender systems, there are many parameters which are assigned default values. However, the best values for them should be determined in each particular scenario. In iExpand, we develop a model to select parameter values by combining Minka's fixed-point iterations and an evaluation method for topic models.

In this paper, we further explain why topic models can be used to simulate the user latent interests and we demonstrate the way of extracting these interests from the latent Dirichlet allocation (LDA) model by the Gibbs sampling method. In addition, we illustrate how to use iExpand for making online recommendations in the real-world applications. Finally, we provide systematic experiments on three data sets selected from a wide and diverse range of domains, and we use multiple evaluation metrics to evaluate the performance of iExpand. Since iExpand views collaborative filtering as a ranking problem and aims to make recommendations by directly ranking the candidate items, we report the ranking prediction accuracy. As shown in the experimental results, iExpand outperforms four benchmark methods: two graph-based algorithms and two algorithms based on dimension reduction. As many other algorithms formulate collaborative filtering as a regression problem (i.e., rating prediction) [30], we also report the comparison results of the rating predictions. In addition to this, these new experiments provide more insights into the iExpand model, such as the effect of the parameters and the low computational cost.

D. Outline

The rest of this paper is organized as follows. Section II gives the detail of the iExpand method for effective recommendation. In Section III, we show the experimental results and many discussions. In Section IV, we introduce the related work. Finally, Section V concludes this paper.

II. User Interest Expansion

In this section, we first introduce the framework of the iExpand model. Then, we describe each step of the model in detail. In addition, we show how to select parameters. Finally, we address the computational complexity issue.

A. The Framework of the iExpand Model

First of all, the iExpand model assumes that, in recommender systems, a user's rating behavior depends on an underlying set of hidden interests. Inspired by the topic models, in iExpand, each user is represented as a probability distribution over interests and each interest is a probability distribution over items. Fig. 3(a) shows the three-layer representation, *user–interests–item*. What is more is that the iExpand model assumes that the order of items in a user's rating record can be neglected and the users' order in a user set can also be neglected, which means both items and users are exchangeable. In correspondence with the LDA model [8], a topic model that we use in iExpand for extracting user interests, the users are documents, the items are words, and the latent interests are topics, respectively.

We should note that, in terms of items, a latent interest can be viewed as one specific characteristic of the items and the users who have this latent interest will prefer the items with this characteristic. Since an item may have multiple characteristics, it belongs to many latent interests (i.e., polysemy). At the same time, different items may have a similar characteristic; they may at least refer to one same latent interest (i.e., synonymy). Let us take the movies in Fig. 1 for example; the movie *Crouching Tiger, Hidden Dragon* belongs to multiple interests, *Oscar*, *kung fu*, *Yun-Fat Chow*, etc., and there are also many movies that can be denoted by the interest *kung fu*. However, in most cases, it is impossible to understand the item characteristics clearly (e.g., in collaborative-filtering scenario). Fortunately, as a simulation tool, the topic model (e.g., LDA) can be used to learn the meaning and characteristics of items in a data-driven fashion, i.e., from given rating records, possibly without further content or prior knowledge of these items.

Topic models are a type of statistical models, which were firstly proposed in machine learning and natural language processing for discovering the hidden topics (e.g., *Basketball*, *Travel*, and *Cooking*) that occur in a collection of documents. In terms of collaborative filtering, the documents can be viewed as the users, the words are items, and topics become the hidden interests. Based on the hypothesis of topic models, the co-occurrence structure of items in the rating records can be used to recover the latent interest structure and the items that often appear together in one rating record may tend to have the same characteristics. In this way, the latent topics can be used to simulate the real-world interests.

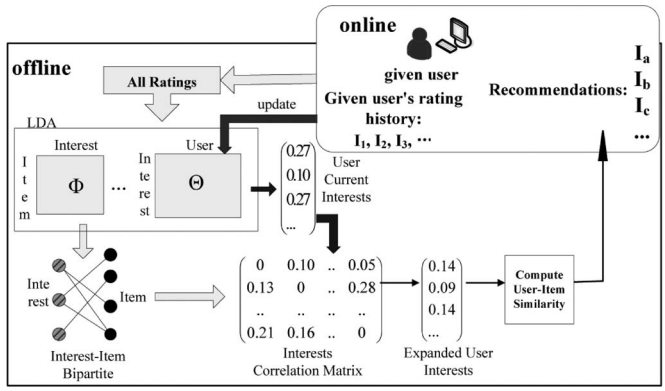


Fig. 2. Framework of the iExpand model. Gray arrows show the general process of the model, while black arrows show the procedure of online recommendations.

TABLE I
Mathematical Notations

Notation	Description
M	number of users
N	number of items
K	number of latent interests
$U = \{U_1, U_2, ..., U_M\}$	the set of users
$I = \{I_1, I_2, ..., I_N\}$	the set of items
$T = \{T_1, T_2, ..., T_K\}$	the set of latent interests
ϕ	a matrix, with ϕ_{ij} equals to $P(I_i\|T_j)$
θ	a matrix, with θ_{ij} equals to $P(T_j\|U_i)$
$\vec{\vartheta}$	a vector, with $\vec{\vartheta}_i$ equals to $P(T_i)$
φ	a matrix, with φ_{ij} equals to $P(T_j\|I_i)$
ψ	a matrix, with ψ_{ij} equals to $P(T_j\|T_i)$
$\theta^{(s)}$	a matrix, and a row is represented as $\vec{\theta}_i^{(s)}$. $\theta_{ij}^{(s)}$ is the probability that a random walk starts from U_i and stops at T_j after s steps

Fig. 2 shows the framework of the iExpand model. From Fig. 2 we can see that, when a user comes, the learning and recommendation process of the iExpand model generally consists of four steps. In the first step, the information about user latent interests is extracted by the inference of the LDA model. In the second step, the correlation graph/matrix of latent interests is established by an item–interest bipartite graph projection. In the third step, for a given user, his/her interest distribution is expanded by letting the current interest vector perform a random walk on the interest correlation graph/matrix. Finally, the candidate items are ranked using expanded user interests and the recommendation list is generated.

Each step of the iExpand model is introduced in the following sections. For better illustration, Table I lists all mathematical notations used in this paper.

B. Extracting User Interests From the LDA Model

In this section, we show how to extract the information about user latent interests from the LDA model. The information about latent interests include the probability distribution of each user over interests, the probability distribution of each interest over items, and the distribution of each interest.

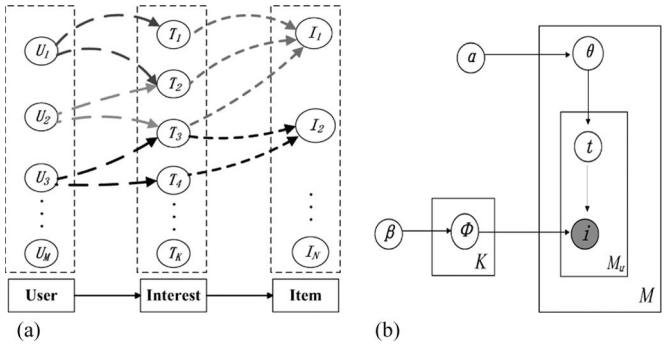


Fig. 3. (a) Three-layer representation scheme. (b) Graphical model representation of LDA.

For collaborative filtering, the LDA model can be represented by a probabilistic graphical model, as shown in Fig. 3(b), where shaded and unshaded variables indicate observed and latent (i.e., unobserved) variables, respectively. In Fig. 3(b), each user in M users is represented as a bag of item tokens M_u, and each token is viewed as an observed variable i. Because LDA can provide an intuitive description of each observed variable i, it is a type of generative probabilistic model. Specifically, this item token is generated from a multinomial distribution over items ϕ_t, specific to an interest t, and interest t is chosen from a multinomial distribution over interests θ_u, specific to this user. Both θ and ϕ are modeled by the Dirichlet distribution, with the hyperparameters α and β, respectively.

Extracting user interests θ from LDA is a latent variable inference process, which is to "invert" the generative model and "generate" latent variables (i.e., the interests' distribution over items ϕ and the users' interest distributions θ) from given observations. After inference, the value of these latent variables should maximize the posterior distribution of the entire user rating records (i.e., given observations). However, learning these latent variables is intractable in general [8]. Thus, many approximations have been proposed, including Gibbs sampling [19], variational inference [8], and so on. The previous research has found that main differences among these approaches could be explained by the different settings of two hyperparameters [5]. In this paper, we choose the Gibbs sampling technique, a form of Markov chain Monte Carlo, which is easy to implement and provides a relatively efficient method for extracting a set of interests from a large rating set.

The Gibbs sampling algorithm begins with the assignment of each item token in users' rating records to a random interest, determining the initial state of the Markov chain. In each of the following iterations of the chain, for each item token, the Gibbs sampling method estimates the conditional distribution of assigning this token to each interest, conditioned on the interest assignments to all other item tokens. An interest is sampled from this conditional distribution and then stored as the new interest assignment for this token. After an enough number of iterations for the Markov chain, the interest assignment for each item token will converge and each token in the rating records is assigned to a "stable" interest. According to the assignment, the distribution of interest T_j over item

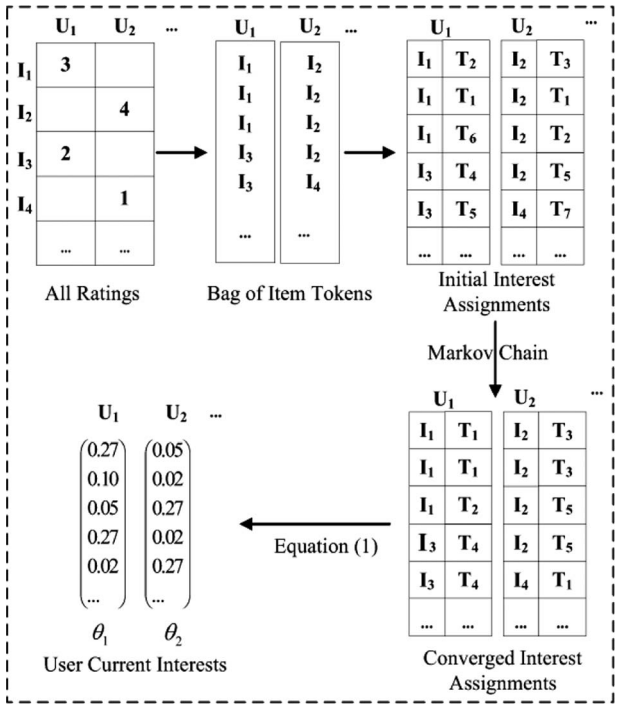


Fig. 4. Example for the LDA inference process based on Gibbs sampling.

I_i (ϕ_{ij}) and user U_i's distribution over interest T_j (θ_{ij}) can be estimated by

$$\phi_{ij} = P(I_i|T_j) = \frac{C_{ij}^{NK} + \beta}{\sum_{n=1}^{N} C_{nj}^{NK} + N\beta}$$
$$\theta_{ij} = P(T_j|U_i) = \frac{C_{ij}^{MK} + \alpha}{\sum_{k=1}^{K} C_{ik}^{MK} + K\alpha} \quad (1)$$

where C^{NK} and C^{MK} are matrices with dimensions $N \times K$ and $M \times K$, respectively. C_{ij}^{NK} denotes the number of times that item I_i is sampled from interest T_j, and C_{ij}^{MK} refers to the number of times that interest T_j is assigned to the items in user U_i's rating record. Fig. 4 shows a simple example for the LDA inference process by extracting two users' interest distributions from Gibbs sampling. From Fig. 4, we can see that users with different preferences will finally get different interest distributions.

In addition, in iExpand, we further extract the probability distribution of each latent interest T_i ($\vec{\vartheta}_i$), and $\vec{\vartheta}_i$ can be estimated by

$$\vec{\vartheta}_i = P(T_i) = \frac{\sum_{m=1}^{M} C_{mi}^{MK} + \alpha}{\sum_{k=1}^{K}\sum_{m=1}^{M} C_{mk}^{MK} + K\alpha}. \quad (2)$$

It is worth distinguishing between our user interests and the latent topics in topic models, like Probabilistic Latent Semantic Analysis (PLSA) or LDA. In iExpand, each user has a distribution on the spectrum of interests, whereas in PLSA/LDA, a topic is a latent variable and the distributions are specified

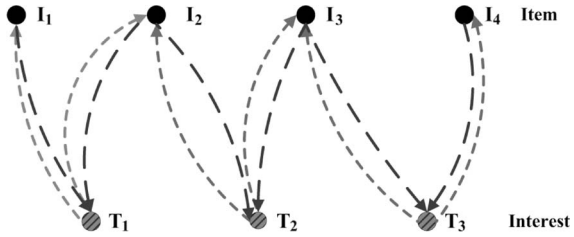


Fig. 5. Example of interest–item bipartite graph. For simplicity, not all the edges between each pair of item and interest are shown.

by the specific topic, i.e., they are class (topic)-conditional distributions. Thus, the model representation of iExpand and PLSA/LDA are significantly different from each other.

C. Correlation Graph of User Interests

In this section, we describe how to compute the transition probabilities between latent interests by the correlation graph. In order to construct the correlation graph of latent interests, we use the items as intermediary entities. φ is created to estimate each item's probability distribution over interests and φ_{ij} can be estimated by

$$\varphi_{ij} = P(T_j|I_i) = \frac{P(T_j, I_i)}{P(I_i)} = \frac{\phi_{ij}\vec{\vartheta_j}}{\sum_{k=1}^{K} \vec{\vartheta_k}\phi_{ik}}. \tag{3}$$

Although the interests may be correlated to each other in reality, in LDA, when α is given, the distributions of interests are independent. Unlike the Correlated Topic Model [7], in iExpand, we model those correlations in the form of probabilities. Specifically, we first use a bipartite graph $G = \langle X, E\rangle$ to represent the relationships between items and interests, with the vertex set $X = I \cup T$, as shown in Fig. 5. In the bipartite graph, the weight of the edge from interest T_j to item I_i is ϕ_{ij} and the weight of the edge from I_i to T_j is φ_{ij}.

Then, by projecting G, we get the relationships between interests, and we use ψ to represent them. Also, ψ_{ij} indicates the recommending strength of interest T_i for T_j, and it can be computed by

$$\psi_{ij} = P(T_j|T_i) = \sum_{n=1}^{N} P(T_j|I_n)P(I_n|T_i) = \sum_{n=1}^{N} \varphi_{nj}\phi_{ni}. \tag{4}$$

At last, the bipartite graph is transformed into a correlation graph which describes the relations between interests, and ψ becomes its correlation matrix. It can be proven easily that each entry in ψ is equal to or greater than zero and ψ is row normalized. In terms of a correlation matrix, ψ_{ij} means the coefficient of correlation between T_i and T_j, from T_i's view. However, in terms of random walk, ψ_{ij} is the probability that current state jumps from T_i to T_j.

D. User Interest Expansion

In this section, we describe the solution for the expansion of user interests. As discussed previously, user interests often change from one to another. If recommender systems just deal with user current interests, the systems will suffer from the *overspecialization* problem and the cold-start problem.

To address this issue, we use **PageRank** [34], a personalized ranking strategy on the user interest correlation graph. We choose this strategy not only because it can create personalized views of interest importance but also because it can predict user interest expansion by exploiting the structure of the interest correlation graph. Given a user interest (a vector), we do repeat PageRank iterations (i.e., guided random walks) until convergence. The final converged PageRank score vector contains the expanded user interests. One can also view this as predicting the next possible interest for each user. Thus, we can make diverse recommendations in a systematic way.

The algorithmic approach here is the personalized ranking [25]. First, for user U_i, we represent his/her current interest model through vector $\vec{\theta_i}^{(0)}$ in which the jth entry $\vec{\theta_i}^{(0)}(j)$ corresponds to a latent interest T_j and is initialized as θ_{ij}. From Section II-C, we know $\vec{\theta_i}^{(0)}$ is normalized and it represents the probability distribution on each latent interest when random walk starts.

Next, let $\vec{\theta_i}$ perform Random Walk with Restart (RWR) [18] (a specific implementation of the personalized ranking) on the correlation graph. Let us consider a random walk that starts from $\vec{\theta_i}^{(0)}$; when arriving at T_j, it randomly chooses T_j's neighbors and keeps walking. For each step, in addition to making such decisions, the random walker goes back to the starting point with a certain probability c, so as to counteract the dependence on far-away parts of the graph.

For example, the process of one step random walk of the user U_i from step s to step $(s+1)$ can be formalized as

$$\vec{\theta_i}^{(s+1)} = (1-c)\vec{\theta_i}^{(s)}\psi + c\vec{\theta_i}^{(0)} \tag{5}$$

while, for all the users, their one-step updates can be formalized as

$$\begin{cases} \theta^{(s)} = \theta, & s = 0 \\ \theta^{(s+1)} = (1-c)\theta^{(s)}\psi + c\theta, & s \geqslant 0 \end{cases} \tag{6}$$

where $\vec{\theta_i}^{(s)}$ serves as the interest vector for U_i after s steps of random walk have completed. All users' interest vectors form a matrix $\theta^{(s)}$ where $\theta_{ij}^{(s)}$ means the steady-state probability that a random walk starts from user U_i and stops at interest T_j after s steps, meanwhile it implies the affinity of T_j with respect to U_i. The bigger $\theta_{ij}^{(s)}$, the closer U_i and T_j.

The personalized ranking is run for all users simultaneously, and it only takes several steps on average before $\theta^{(s)}$ converges. The parameter c indicates the restart probability, and $(1-c)$ is the decay factor used to represent how much relationship is lost in each step.

E. From Expanded User Interests to the Item Recommendation

In this section, we describe the last process of iExpand, the ranking of the items and the generation of recommendation

lists. In iExpand, the items are ranked by their relevance with any given user. The user's possible distribution on latent interests serves as intermediary entities

$$P(I_j|U_i)=\sum_{k=1}^{K} P(I_j|t=k)P_s(t=k|U_i)=\sum_{k=1}^{K}\phi_{jk}\theta_{ik}^{(s)}. \quad (7)$$

It is easy to obtain the top K recommendations by ranking the candidate items. Thus, iExpand directly generates recommendations without the step of predicting rating scores.

In addition, if the user rating has been taken into consideration, iExpand can be used as a rating prediction method, such as the traditional memory-based collaborative-filtering methods. Here, *Pearson Correlation* on expanded user interest vectors can be used to compute user similarities $Sim(U_i, U_h)$. Therefore, the neighborhood $Neighbor(U_i)$ can be formed for user U_i. Then, the rating from user U_i to item I_j can be predicted by

$$\hat{r}_{i,j} = \overline{r}_i + \frac{\sum\limits_{U_h\in \mathrm{Neighbor}(U_i)} \mathrm{Sim}(U_i,U_h) * (r_{h,j}-\overline{r}_h)}{\sum\limits_{U_h\in \mathrm{Neighbor}(U_i)} |\mathrm{Sim}(U_i,U_h)|} \quad (8)$$

where $\overline{r}_i$ and $\overline{r}_h$ are the average rating values for user U_i and U_h, respectively. $r_{h,j}$ refers to the rating value for item I_j from user U_h.

What we discussed earlier is about how to make recommendations in a general iExpand process. However, in real-world applications, we face the challenge of online recommendations. Since users' interest distributions may change quickly from time to time, while the correlation of interests evolves slowly, we can update both the inference process and the correlation graph periodically offline while renewing the user interests whenever he/she rates. For example, when user U_u rates a new item, U_u's interests can be resampled by Gibbs sampling. In each iteration, the interest assignment for every item in U_u's rating record is sampled by

$$P\left(t_i^u = j | t_{\neg i}^u, U_u, \ldots\right) \propto \frac{C_{I_i^u j}^{NK} + \vec{C_{j\neg i}}^{\,u} + \beta}{\sum\limits_{n=1}^{N} C_{nj}^{NK} + \vec{C_j}^{\,u} + N\beta - 1} \frac{\vec{C_{j\neg i}}^{\,u} + \alpha}{\sum\limits_{k=1}^{K} \vec{C_k}^{\,u} + K\alpha - 1} \quad (9)$$

where $t_i^u = j$ means the interest assignment of item I_i^u to interest T_j, $\vec{C}^u$ is a vector, and $\vec{C_j}^{\,u}$ denotes the number of times that interest T_j is assigned to the items in U_u's rating record. Also, $\neg i$ refers to the interest assignments of all other items, not including the current instance. After performing interest resampling, each interest distribution component of U_u can be computed by

$$\theta_{uj} = P(T_j|U_u) = \frac{\vec{C_j^u} + \alpha}{\sum\limits_{k=1}^{K} \vec{C_k^u} + K\alpha}. \quad (10)$$

F. Estimating the Parameters

In this section, we present a method of selecting values for the parameters of iExpand. There are three parameters: the hyperparameters α and β and the number of interests K.

First of all, we select the values for α and β. Previous research works have found that $\alpha = 50/K$ and $\beta = 0.01$ work well with different text collections, and they are often used as the default values [10], [49]. However, Steyvers *et al.* [45] pointed out that good choices for these values depend on the number of interests and the item size. Furthermore, Asuncion *et al.* [5] suggested that hyperparameters play an important role in learning accurate topic models. Therefore, finding the best α, β settings for each scenario is important. There are many ways for learning them [48], among which Minka's fixed-point iteration is widely used. It was proposed by Minka in [33] and was carefully studied by Wallach [48]. In iExpand, each step of fixed-point iteration is formalized as

$$\begin{aligned}
\alpha^* &\longleftarrow \frac{\alpha \sum\limits_{m=1}^{M}\sum\limits_{k=1}^{K}\left[\Psi\left(C_{mk}^{MK}+\alpha\right)-\Psi(\alpha)\right]}{K\sum\limits_{m=1}^{M}\left[\Psi\left(\sum\limits_{k=1}^{K}C_{mk}^{MK}+K\alpha\right)-\Psi(K\alpha)\right]} \\
\beta^* &\longleftarrow \frac{\beta \sum\limits_{k=1}^{K}\sum\limits_{n=1}^{N}\left[\Psi\left(C_{nk}^{NK}+\beta\right)-\Psi(\beta)\right]}{N\sum\limits_{k=1}^{K}\left[\Psi\left(\sum\limits_{n=1}^{N}C_{nk}^{NK}+N\beta\right)-\Psi(N\beta)\right]}.
\end{aligned} \quad (11)$$

Next, in addition to α and β, we choose the right value for the interest number K. In previous works, if categories of the data sets are known, then K will be set equal to that number [9]. However, in most scenarios, the category is unknown and how to set K becomes a problem. In most cases, K is randomly chosen or given a default value [5], [10], [51]. Until now, one possible approach for setting this value is to compute the likelihood of the test data under different K values, then the best one is chosen by a grid search. Exact computation of the posterior probability is intractable, since it requires summing over all possible assignments. However, we can approximate it by an estimator. In this paper, we refer to an approach proposed by Wallach *et al.* [47] named *Chib-style estimation* which was initially proposed as one evaluation method for topic models. The main idea of this approach is first to choose a *special* set of latent topic assignments and then use Bayes' rule to estimate the posterior probability.

Finally, as the posterior probability depends on α, β, and K, we combine these factors together and propose a parameter learning algorithm, as shown in Algorithm 1. In Algorithm 1, inputting the initial values of α and β, we first use Gibbs sampling and Minka's fixed-point iteration to learn optimal values for α and β specific to each number of interest K. Then, Chib-style estimation is used to compute the posterior probability of the test data, under current parameter setting. Lastly, the parameters with the best posterior probability are chosen for the model, and they are used as the default settings for performance comparison in the experimental part.

Algorithm 1: Estimating Parameters (a, b)

input: a is the initial value of α;
b is the initial value of β;
output: the best values for α, β, and K
for *all candidate* K **do**
Initialize $\alpha = a$;
Initialize $\beta = b$;
for $loop \leftarrow 1$ to MAX_LOOP **do**
Gibbs sampling;
Update α, β by (11);
posterior $= \log(Chib - style\ estimation(\alpha, \beta, K))$;
Record the maximum posterior and the corresponding α, β, and K;
Return the best values for α, β, and K

G. Computational Complexity

In this section, we analyze the computational complexity issues for iExpand. Specifically, the time cost for the inference of LDA is $O(M \cdot N \cdot K \cdot l)$, where l is the iteration number of Gibbs sampling. For the bipartite graph projection, most of the time is used to construct the correlation matrix ψ and the time cost in this phase is $O(N \cdot K^2)$. For each user, the cost for random walk is $O(s \cdot K^2)$ on average. Thus, for all the users, it costs $O(s \cdot M \cdot K^2)$. Since $K \ll M$ and $K \ll N$ and the time cost for ranking the items and making recommendations can be neglected, the total computational complexity for the general iExpand process is $O(M \cdot N \cdot K \cdot l)$. As we discussed in Section II-D, in real-world applications, both the inference process and the correlation graph can be updated periodically offline; thus, for online computing, we just need to run less than 30 iterations of Gibbs sampling [37] and one personalized ranking or rating prediction for the current user, both of which can be done efficiently. The online recommendations can be followed by the black arrows shown in Fig. 2.

III. Experimental Results

In this section, we present the experimental results to evaluate the performance of iExpand. Specifically, we demonstrate the following: 1) the results of parameter selection based on Algorithm 1; 2) a performance comparison between iExpand and many other benchmark methods; 3) an analysis of the parameters in personalized ranking; 4) the understanding of interests and interest expansion; and 5) the discussion about the advantages and limitations of the iExpand model.

A. Experimental Setup

All the experiments were performed on three real-world data sets: MovieLens [1], Book-Crossing [55], and Jester [17]. The first one is collected by the GroupLens Research Project and has become a benchmark for evaluating recommender systems. For the last two data sets, we only choose part of them, considering the scalability problem of many of our benchmark methods (i.e., the graph-based algorithms). The detailed information about these three data sets are described in Table II.

TABLE II
Description of Three Data Sets

Data Set	Domain	#Users	#Items	#Records	Sparsity(%)
MovieLens	Movie	943	1,682	100,000	93.70
Book-Crossing	Book	996	1,696	91,084	94.61
Jester	Joke	2,000	100	36,596	81.70

For each user's rating record, we split it into a training set and a test set, by randomly selecting some percentage of the ratings to be part of the training set and the remaining ones to be part of the test set. To observe how each algorithm behaves at different sparsity levels, we construct different sizes of training sets from 10% to 90% of the ratings with the increasing step at 10%. In total, we construct nine pairs of training and test sets, and each split named as $x - (100 - x)$ means x percent ratings are selected to be the training data and the remaining $(100 - x)$ percent ratings for testing.

Benchmark Methods. In order to demonstrate the effectiveness of iExpand, we compare it with many other benchmark methods for both the ranking prediction accuracy and the rating prediction accuracy. For the ranking purpose, we compare it with two graph-based algorithms, ItemRank [18] and L^+ [14], as well as two algorithms based on dimension reduction, LDA and SVD [39], both of which do not take user interests into consideration. Among them, ItemRank is a personalized ranking strategy on the item correlation graph for alleviating the cold-start problem and L^+ is widely used for measuring node similarities in a graph. Similar to iExpand, both LDA and SVD consider the latent factors. However, they are just used for dimension reduction and do not consider the correlation between latent factors. All the aforementioned four methods can be seen as the related methods for iExpand.

For the rating purpose, we also compare it with four existing methods. For the memory-based method, we implemented the user-based collaborative filtering (UCF) [36]. For the model-based method, we chose the RSVD [15] and LDA. In addition to this, we also implemented the graph-based algorithm ItemRank [18]. Both UCF and RSVD are state-of-the-art collaborative-filtering algorithms, and they are widely used for baselines.

Among all these methods, RSVD, UCF, and SVD are originally rating-oriented algorithms and the rest of the methods, including iExpand, are ranking-oriented algorithms. All these methods have been chosen as the baseline methods.

B. Evaluation Metrics

For the purpose of evaluation, we adopted *Degree of Agreement* (DOA) [14], *Top-K* [26], and *Recall* [20], [39] as the evaluation metrics for ranking prediction accuracy. All of them are commonly used for ranking accuracy, and these three metrics try to characterize the recommendation results from different perspectives.

DOA measures the percentage of item pairs ranked in the correct order with respect to all pairs [14], [18]. Let $NW_{U_i} = I - (L_{U_i} \cup E_{U_i})$ denote the set of items that do not occur in

the training and test sets for U_i, where L_{U_i} and E_{U_i} mean the item set that U_i rated in the training and test sets, respectively. Furthermore, we define $check\_order$ as

$$check\_order_{U_i}(I_j, I_k) = \begin{cases} 1, & \text{if } (PR_{I_j} \geq PR_{I_k}) \\ 0, & \text{otherwise} \end{cases}$$

where PR_{I_j} denotes the predicted rank of item I_j in the recommendation list. Then, the individual DOA for user U_i is defined as

$$DOA_{U_i} = \frac{\sum_{j \in E_{U_i}, k \in NW_{U_i}} check\_order_{U_i}(I_j, I_k)}{|E_{U_i}| \times |NW_{U_i}|}.$$

An ideal ranking corresponds to a 100% DOA, and we use DOA to stand for the average of each individual DOA.

Top-K indicates the precision of the selected top K items, and **Recall** measures the ratio of the number of hits to the size of each user's test data [39]. For each user U_i, these two measures are defined as follows:

$$Top - K_{U_i} = \frac{\#hits}{K}, \quad Recall_{U_i} = \frac{\#hits}{|E_{U_i}|}.$$

For the purpose of evaluating the rating effectiveness, we also choose the Mean Absolute Error (**MAE**) and the Root Mean Squared Error (**RMSE**) as the evaluation metrics. Both of them are commonly used in traditional collaborative-filtering systems [2], [15], [20], [27].

C. Parameters in LDA

In this section, we investigate the learning performances of two parameters, namely, hyperparameters and the number of interests, by Algorithm 1. Here, the first 893 users in MovieLens are used as training data and the remaining 50 users form the test set. Similarly, for Book-Crossing, the first 900 users are treated as training samples and the remaining users as test data. Also, for Jester data set, the first 1800 users are treated as training data and the remaining 200 users for testing. For each run of Algorithm 1, we initialize the parameters as $a = 0.5$ and $b = 0.5$ and turn on Minka's updates after 15 iterations, and these settings are similar to the ones in [5].

The estimation of posterior for the test set is computed for K sizes from 50 to 800 for both MovieLens and Book-Crossing and K from 20 to 100 for Jester. The Gibbs sampling algorithm runs 1000 iterations each time. Let us take the MovieLens data set as an example. The results of parameter selection are shown in Fig. 6. The results suggest that the test set are best accounted for by an LDA model incorporating 300 interests and the corresponding best hyperparameter settings are $\alpha = 0.001$ and $\beta = 0.08$. In Fig. 6, we can observe that the best hyperparameters for collaborative filtering are different from those of text applications based on topic models. Finally, the results of parameter selection are summarized in Table III.

D. Performance Comparison

In this section, we present a performance comparison of both effectiveness and efficiency between iExpand and the benchmark approaches: ItemRank [18], L^+ [14], UCF [36], SVD [39],[1] LDA, and RSVD [15]. For the purpose of comparison, we record the best performance of each algorithm by tuning their parameters. The training models of all these algorithms are learned only once, and ratings in the test set have never been used in the training process. Therefore, in order to make a clearer and fairer comparison, we do not take the online recommendation into consideration.

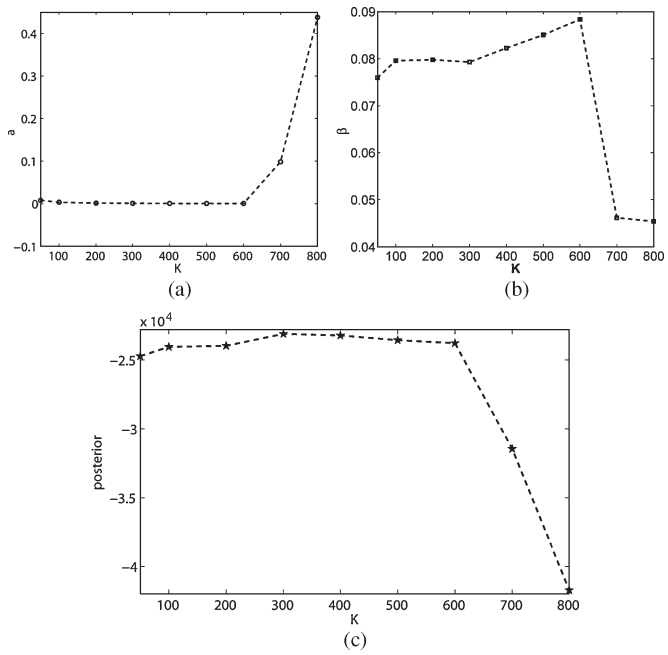


Fig. 6. Results of parameter selection for MovieLens. (a) Best α for different number of interests. (b) Best β for different number of interests. (c) Log-likelihood of posterior for different number of interests.

TABLE III
PARAMETER SETTINGS

Data set	α	β	K	l
MovieLens	0.001	0.080	300	1000
Book-Crossing	0.017	0.237	50	1000
Jester	0.545	0.118	40	1000

First of all, we show a comparison of the effectiveness of all the algorithms. Tables IV and V and Fig. 7 show the performances of their recommendations with respect to different splits and different evaluation metrics. Table IV(a)–(c) illustrates the evaluation results of the DOA/Recall measures. Fig. 7 demonstrates the top K results on the three data sets, and Table V shows the evaluation results of the rating prediction accuracy on the MovieLens data set. Note that we did not report the rating prediction results on the Book-Crossing and Jester data sets because the rating scale is too big for Jester and most of the ratings are 0 in Book-Crossing.

DOA/Recall. In terms of DOA/Recall measures, from Table IV, we can see that iExpand outperforms the other four algorithms in each split. Also, the sparser the data, the more significant improvement can be made. Indeed, both Item-

[1]In our implementation, we rank the items by computing their Pearson correlation with each user. This is slightly different from the implementation in [39]; however, this way can yield better results for our situation.

TABLE IV
PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS BASED ON DOA/RECALL RESULTS. (a) PERFORMANCE COMPARISON ON THE MOVIELENS DATA SET [(LEFT) DOA IN PERCENT. (RIGHT) RECALL IN PERCENT.]. (b) PERFORMANCE COMPARISON ON THE BOOK-CROSSING DATA SET [(LEFT) DOA IN PERCENT. (RIGHT) RECALL IN PERCENT.]. (c) PERFORMANCE COMPARISON ON THE JESTER DATA SET [(LEFT) DOA IN PERCENT. (RIGHT) RECALL IN PERCENT.]

(a)

Split.\Alg	SVD	ItemRank	L^+	LDA	**iExpand**
10 – 90	52.504	76.694	66.853	75.726	**79.043**
20 – 80	67.990	84.402	84.022	83.477	**85.789**
30 – 70	75.334	86.351	86.969	87.280	**87.994**
40 – 60	79.882	87.390	88.774	88.683	**89.060**
50 – 50	83.939	88.217	89.597	89.236	**89.864**
60 – 40	85.567	88.723	90.340	89.992	**90.626**
70 – 30	84.833	88.762	90.604	90.747	**91.202**
80 – 20	85.243	89.002	90.636	90.823	**91.310**
90 – 10	86.911	88.734	90.608	91.110	**91.652**

Split.\Alg	SVD	ItemRank	L^+	LDA	**iExpand**
10 – 90	10.521	24.901	13.467	21.416	**28.196**
20 – 80	13.578	31.940	22.260	33.093	**35.514**
30 – 70	14.996	32.616	25.683	35.146	**36.106**
40 – 60	17.456	31.639	28.055	34.957	**35.240**
50 – 50	17.824	30.464	27.194	33.023	**33.535**
60 – 40	17.530	28.183	25.249	30.463	**31.507**
70 – 30	14.199	24.954	23.329	27.366	**28.484**
80 – 20	11.363	20.761	19.342	23.008	**23.467**
90 – 10	6.941	14.543	12.885	15.692	**16.162**

(b)

Split.\Alg	SVD	ItemRank	L^+	LDA	**iExpand**
10 – 90	46.359	57.734	52.839	57.276	**60.174**
20 – 80	52.711	62.145	60.368	59.891	**62.670**
30 – 70	58.632	64.851	62.846	64.714	**65.682**
40 – 60	62.478	66.998	65.611	67.165	**68.061**
50 – 50	65.665	67.612	67.294	68.295	**69.120**
60 – 40	67.380	68.157	69.152	69.885	**70.437**
70 – 30	69.077	68.692	70.778	70.541	**71.610**
80 – 20	69.881	69.576	71.346	71.927	**72.582**
90 – 10	71.457	69.642	72.707	72.434	**73.372**

Split.\Alg	SVD	ItemRank	L^+	LDA	**iExpand**
10 – 90	5.058	9.210	5.663	7.839	**12.771**
20 – 80	5.708	12.240	6.468	10.130	**13.380**
30 – 70	6.605	13.018	6.547	12.363	**13.895**
40 – 60	6.870	12.509	7.070	12.528	**13.589**
50 – 50	7.079	11.507	7.625	12.088	**12.775**
60 – 40	6.990	9.815	7.854	11.653	**12.172**
70 – 30	5.933	8.297	7.677	10.572	**11.038**
80 – 20	4.547	6.414	6.591	8.184	**8.897**
90 – 10	2.485	4.166	4.625	5.567	**6.274**

(c)

Split.\Alg	SVD	ItemRank	L^+	LDA	**iExpand**
10 – 90	47.668	84.194	56.238	85.987	**86.189**
20 – 80	39.843	87.162	60.933	87.127	**87.329**
30 – 70	36.373	86.752	61.253	86.493	**86.957**
40 – 60	35.112	87.199	68.536	86.782	**87.352**
50 – 50	37.286	87.257	75.567	86.798	**87.310**
60 – 40	35.811	87.707	78.936	87.293	**87.718**
70 – 30	36.607	87.704	80.904	87.096	**87.788**
80 – 20	41.178	87.542	81.599	87.144	**87.650**
90 – 10	48.262	88.467	82.985	88.228	**88.517**

Split.\Alg	SVD	ItemRank	L^+	LDA	**iExpand**
10 – 90	17.728	55.434	20.711	60.250	**60.614**
20 – 80	9.868	58.478	13.580	58.046	**58.967**
30 – 70	8.003	56.709	14.070	56.795	**56.830**
40 – 60	6.489	54.219	14.788	52.978	**54.788**
50 – 50	6.291	50.217	18.637	50.060	**50.335**
60 – 40	4.364	46.425	17.768	45.337	**46.669**
70 – 30	3.310	41.362	14.583	40.845	**41.979**
80 – 20	2.310	35.359	10.022	33.627	**36.293**
90 – 10	1.575	23.825	4.800	23.491	**24.591**

TABLE V
PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS BASED ON RATING RESULTS [(LEFT) MAE. (RIGHT) RMSE]

Split.\Alg	RSVD	ItemRank	CF	LDA	**iExpand**
10 – 90	0.887	0.845	0.919	0.909	**0.844**
20 – 80	0.798	0.796	0.822	0.825	**0.795**
30 – 70	**0.770**	0.775	0.787	0.792	0.777
40 – 60	0.760	**0.749**	0.772	0.778	0.769
50 – 50	**0.751**	0.755	0.756	0.767	0.759
60 – 40	**0.748**	0.754	0.751	0.765	0.759
70 – 30	**0.747**	0.748	0.744	0.758	0.753
80 – 20	**0.740**	0.749	0.741	0.759	0.755
90 – 10	**0.739**	0.757	0.746	0.764	0.763

Split.\Alg	RSVD	ItemRank	CF	LDA	**iExpand**
10 – 90	1.157	1.076	1.172	1.155	**1.075**
20 – 80	1.042	1.009	1.048	1.050	**1.008**
30 – 70	0.995	**0.987**	1.004	1.008	0.988
40 – 60	0.975	**0.949**	0.983	0.992	0.976
50 – 50	**0.957**	0.958	0.961	0.975	0.963
60 – 40	**0.947**	0.958	0.956	0.972	0.962
70 – 30	**0.937**	0.949	0.945	0.961	0.954
80 – 20	**0.933**	0.949	0.942	0.961	0.955
90 – 10	**0.915**	0.957	0.946	0.967	0.965

Rank and iExpand aim at alleviating the sparsity problem and the cold-start problem, and they perform better than L^+, SVD, and LDA (except for Jester) when the training sets are sparse, such as the 10–90 and 20–80 splits. However, iExpand performs much better than ItemRank. For example, in the three 10–90 splits, iExpand achieves nearly two points of improvement on DOA values with respect to ItemRank.

In addition, both LDA and iExpand reduce data dimensions, so they perform better when the data are dense, while SVD, another algorithm based on dimension reduction, does not perform well. This may because of the use of different decomposing techniques. Finally, as the main difference between iExpand and LDA is interest expansion or not and because iExpand can expand user interests and increase the diversity in a properly controlled manner, it performs much better than LDA in all the cases. This means interest expansion can lead to a better performance than only exploiting the current user interests. Another interesting observation is that the smaller and sparser the training set, the more significant improvement is made by iExpand compared with LDA, and when the training set becomes larger and denser, the improvement becomes less obvious. The reason is that, when there are enough interactions between a user and the system, the user has experienced various types of items and his/her preference has been decided. Hence, there will be not much difference

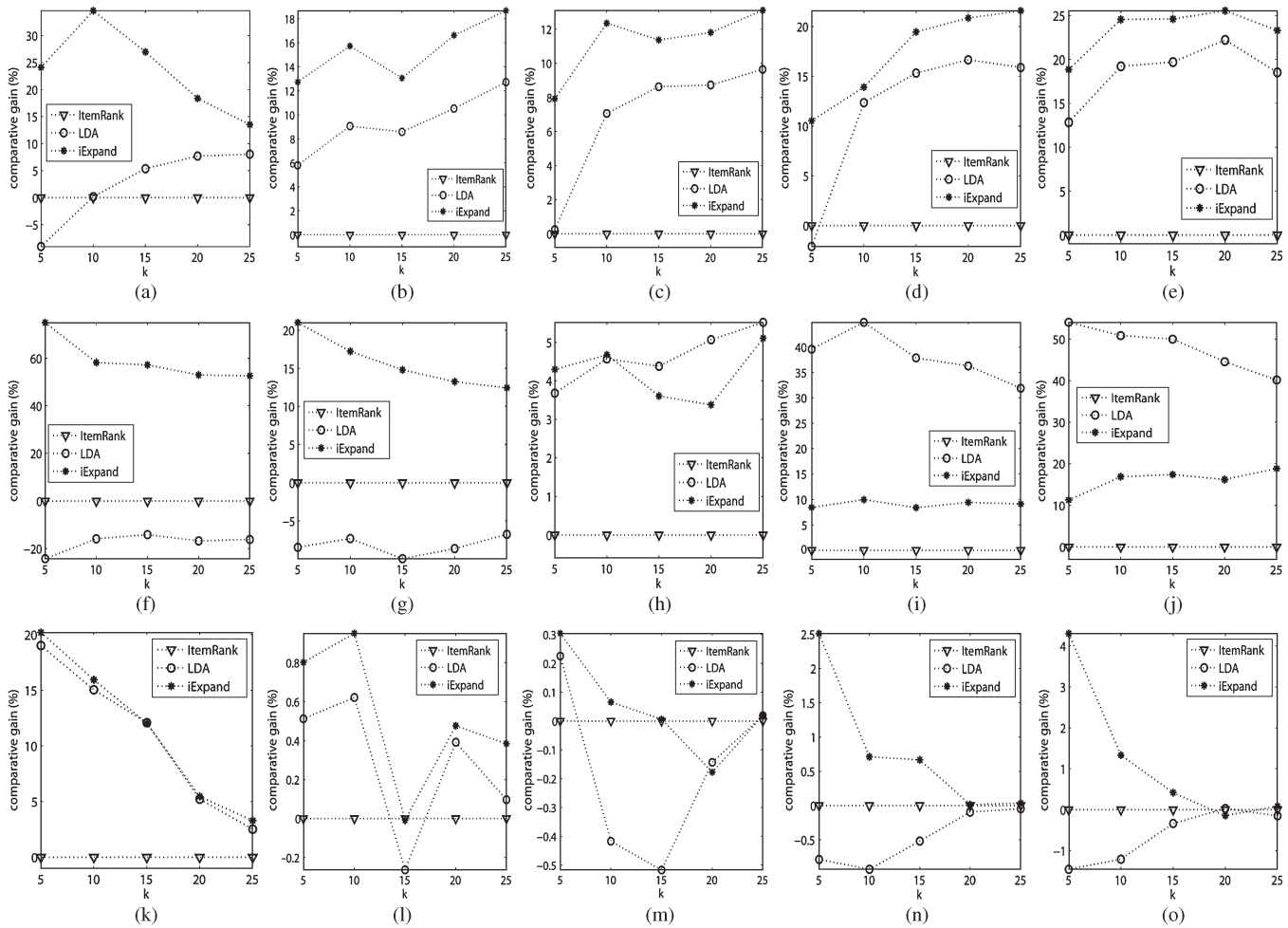


Fig. 7. Performance comparison based on top K results. (a) 10-90, MovieLens. (b) 30-70, MovieLens. (c) 50-50, MovieLens. (d) 70-30, MovieLens. (e) 90-10, MovieLens. (f) 10-90, Book-Crossing. (g) 30-70, Book-Crossing. (h) 50-50, Book-Crossing. (i) 70-30, Book-Crossing. (j) 90-10, Book-Crossing. (k) 10-90, Jester. (l) 30-70, Jester. (m) 50-50, Jester. (n) 70-30, Jester. (o) 90-10, Jester.

from the current interest distribution to the next possible interest distribution.[2]

Top-K. For better illustration, we select five splits from each data set, and we only show the results of the three algorithms with the best top K performances. Fig. 7 shows the comparative results of ItemRank, LDA, and iExpand, where the performance of ItemRank is chosen as the baseline and the comparative results of LDA and iExpand against ItemRank on each k (k ranges from 5 to 25) are demonstrated. In Fig. 7, we can see that iExpand performs better than the baseline on almost every split, while there are more than five splits where LDA performs worse than the baseline. Also, iExpand outperforms LDA, only except for the last two splits of Book-Crossing. In all, in terms of the top K measure, in most cases, iExpand performs better than other methods. Finally, the sparser the data, the more significant improvement can be seen. This is similar to the results of DOA/Recall.

[2]Only one exception is for the Jester data, where LDA performs nearly as well as iExpand on the first two splits. This is because Jester data are a very dense data, which can be seen from the data description in Table II, and this alleviates the advantages of interest expansion.

MAE/RMSE. From Table V, we can see that iExpand performs the best on the two sparsest splits, while in general, RSVD outperforms the other methods in terms of the MAE/RMSE. On the sparse splits, the methods that can discover the indirect correlations and deal with the cold-start problem (i.e., iExpand and ItemRank) get better results than other algorithms (i.e., RSVD, LDA, and UCF). However, on the remaining splits, the rating-oriented methods (i.e., RSVD and UCF) generally perform better than the ranking-oriented methods (i.e.,ItemRank, LDA, and iExpand). Another interesting observation is that these two types of evaluation metrics DOA/Recall/top K and MAE/RMSE lead to inconsistent judgements on the algorithms. The same observation has been reported in many previous works [20], [30].

Note that we chose SVD instead of RSVD for the ranking comparison. The reason is that RSVD led to very bad results which are not comparable with other methods in our ranking experiments. In addition, the question about whether the ranking prediction accuracy or the rating prediction accuracy is more important is beyond the scope of this paper.

Runtime. Next, we compare the computational efficiency of many algorithms. Fig. 8 shows the execution time of these algorithms on each data set. Without a surprise, on both MovieLens

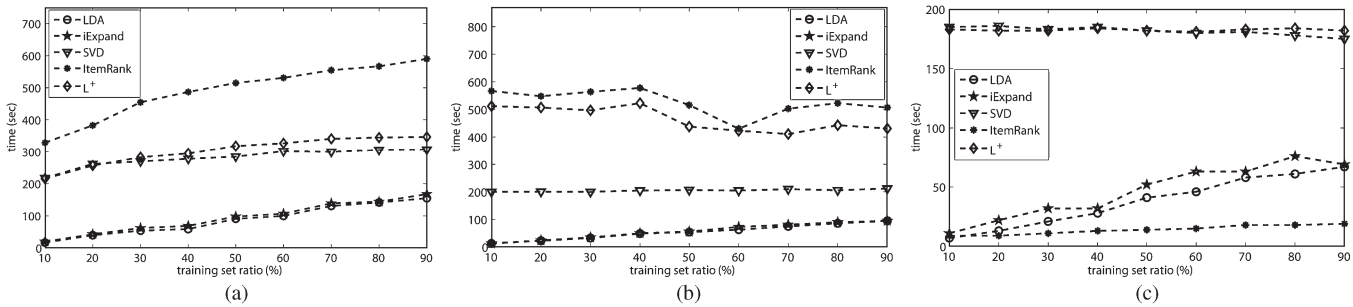


Fig. 8. Comparison of the execution time on each data set. (a) MovieLens data set. (b) Book-Crossing data set. (c) Jester data set.

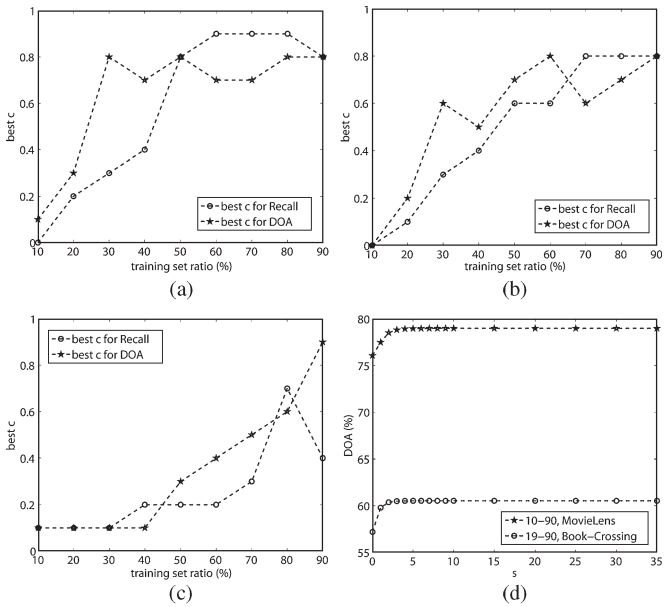


Fig. 9. Effect of parameters in personalized ranking. (a) Best c for MovieLens. (b) Best c for Book-Crossing. (c) Best c for Jester. (d) Steps of random walk for two splits.

and Book-Crossing data sets, among these five model-based collaborative filterings, LDA costs the least time, and with dimension reduction, the execution time of iExpand is almost no longer than that of LDA. Both of them are much faster than three other algorithms with respect to each split. For Jester data set, where there are only 100 items, ItemRank method costs less time than the other approaches. However, in most real-world applications, the number of items are more than thousands, and the time cost for ItemRank will be relatively very high. At the same time, with the increase of the item numbers, its time cost will rise rapidly.

E. Analysis of Parameters in Personalized Ranking

In this section, we provide an analysis of two parameters: the restart probability c and the step of random walk s.

To study the effect of c, we let it vary in the range of [0, 1). When it is 0, random walk will never restart. When c is close to 1, the performance of iExpand will be similar to the LDA algorithm. Fig. 9(a)–(c) shows the relationships between the best value of c with regard to Recall/DOA metrics and the size of training data set for iExpand on three benchmark data sets. In the figure, we can observe that the best c for larger data sets is often bigger than that for smaller data sets. On the one hand, when the training set is large, the correlation graph is dense and there are plenty of direct contacts between vertices. In this scenario, few indirect similarities needs to be considered, and the random walk regresses to one step random walk or there is no need for random walk. On the other hand, when the correlation graph is sparse, random walk does not need to restart frequently for lack of direct contacts. In this scenario, the indirect contacts should be considered, and multistep random walk will perform better than one-step random walk.

As an example, Fig. 9(d) shows the effect of the step of random walk s on the performances of iExpand for two splits. We can see that both curves converge after a few (no more than ten) steps. The results show that random walk does enhance the performance of iExpand and the best performance can be achieved by just a few steps of random walk.

F. Understanding of Interests and Interest Expansion

In this section, we first show the interrelationships between latent interests and explicit interests, and then, we explain the advantages of interest expansion by examples.

In the previous sections, we do not distinguish latent interests and explicit interests deliberately. As we have mentioned, the former is a latent factor extracted by the topic model, while the latter is the one identified in the real world. In this paper, we use latent interests to simulate explicit interests, and research works have shown their one-to-one correspondence [8], [9], [51]. Moreover, Mei *et al.* [32] proposed general approaches for interpreting the meaning of each latent topic. The question is whether every latent interest has a real meaning for use in iExpand. A positive answer is critical for the effectiveness of iExpand for collaborative filtering.

To this end, we consider the first three latent interests extracted from the MovieLens data set. Table VI lists the top five movies for each latent interest identified. As can be seen, all five movies in the first latent interest have the same genres which can be tagged as *Action*, *Adventure*, and *Fantasy*[3] or they can be labeled "Harrison Ford" (and contain one mistake), while movies in the second column all fall into *Comedy* and *Drama*. However, there are several types of movie genres for the third one. After a closer look, we find that all of these movies are generally recognized as *classic* movies and they all have won more than one *Oscar* award. Another observation is that the

[3]This information can be obtained in IMDB. *URL*: http://www.imdb.com/.

TABLE VI
TOP MOVIES IN THE FIRST THREE LATENT USER INTERESTS

Latent interests	Interest 1	Interest 2	Interest 3
movies	Back to the Future	Secrets & Lies	Star Wars
	Return of the Jedi	Il Postino: The Postman	The English Patient
	Raiders of the Lost Ark	My Life as a Dog	The Silence of the Lambs
	Star Wars	Sunset Blvd	Godfather
	The Empire Strike Back	A Room with a View	Pulp Fiction

TABLE VII
EXAMPLE OF USER U_{140} AND THE CORRESPONDING RECOMMENDATION RESULTS

Training set	Test set	Top Recommendations	
		Without Interests Expansion	Interests Expansion
The Devil's Own	The English Patient	Air Force One	Titanic
Contact	Evita	Titanic	**Liar Liar**
Scream	Liar Liar	**Liar Liar**	**The English Patient**
L.A. Confidential	In & Out	**The English Patient**	The Game
Murder at 1600	Ulee's Gold	Conspiracy Theory	The Full Monty
Crash	Fly Away Home	The Full Monty	**In & Out**
Kiss the Girls	Everyone Says I Love You	The Game	Air Force One
U Turn	Mother	Seven Years in Tibet	**Evita**

movie *Star Wars* is given high probability in both latent interests 1 and 3. This verifies that topic models can capture the multiple characteristics of each movie, and each characteristic can be resolved by other movies in the corresponding latent interest.

The aforementioned analysis helps to map each latent interest into explicit interests. This means that, even for collaborative filtering, every latent factor still has a real meaning, although the interpretation may not be as easy and precise as that in text applications. Furthermore, this indicates that, in real applications, if we only get several interest information input by the new user, we can still find out the possible items that a given user may like by the *item–interest* relationship described in iExpand model and thus mitigate the cold-start problem.

In the previous sections, we have showed that interest expansion can lead to a better performance than the method of only exploiting the current user interests. In the following, we will illustrate the difference between these two recommending strategies by a user case. Let us consider the user U_{140} in the MovieLens data set. The ratings of this user can be well divided into two types, *thriller* and *nonthriller*. According to this classification, we select the thriller movies to be the training data and eight of the nonthriller movies to be the test set. Then, we run the two recommending strategies one by one, and we get two types of recommendations in the end. The results are shown in Table VII.

In Table VII, we can see that the top eight recommendations from the algorithm with interest expansion and the method without interest expansion are different from each other. First, the method with interest expansion achieves a better result with more correctly predicted movies. Second, the recommendation results from the method with interest expansion are more diversified.[4] In other words, the interest expansion is more proper to capture the diversified interests and find potential interests for the users. Finally, we would like to point out that this advantage is meaningful to most of the users which can be seen from the results of the performance comparisons shown in Tables IV and V and Fig. 7, while this does not mean it will work for every single user and there may exist users whose interest expansion are different from the majority.

[4]We should note that, in this case, we choose the movie genres as the criterion for diversity, and there may be other appropriate criterions.

G. Discussion

In this section, we analyze the advantages and limitations of the iExpand method. From the experimental results, we can see that there are many key advantages of iExpand. First, iExpand models the implicit relations between users and items through a set of latent user interests. This three-layer representation leads to more accurate ranking recommendation results. Second, iExpand can save the computational cost by reducing the number of *item* dimensions. This dimensionality reduction can also help to alleviate the sparseness problem which is inherent to many traditional collaborative-filtering systems. Third, iExpand enables diverse recommendations by the interest expansion. This can help to avoid the *overspecialization* problem. Finally, iExpand can deal with the cold-start recommendations. This means we only need several items or interests input by the new user, and then, the corresponding items this user may like can be predicted and recommended.

The main limitation of iExpand lies in its "bag of items" assumption, where in each user's rating record, the rating contextual information (e.g., rating time) is totally ignored. However, Ding *et al.* [13] demonstrated that the ratings produced at different times have different impacts on the prediction of future user behaviors. Furthermore, Adomavicius *et al.* [3] presented a systematic discussion on the importance of contextual information when providing recommendations. Thus, it is possible for iExpand to further improve the recommendations by considering the contextual information, such as time stamp and the rating orders.

IV. RELATED WORK

In general, related work can be grouped into four categories. **The first category** has a focus on the graph-based collaborative-filtering methods. Here, the graph-based

collaborative-filtering methods refer to those approaches which use the similarity of graph vertices to make recommendations [14], [18], [44], [50], [52]. In these methods, users and items are treated as vertices of a correlation graph and graph theory is exploited for characterizing the relationship of user–item pairs. The recommendation list is generated by considering how close the candidate items are to a given user. The correlation graph may consist of all users [44], all items [18], [50], [52], or both user and item vertices [14].

While these graph-based collaborative-filtering methods have elegant design ideas, they typically require more memory and have high computational costs due to a large number of vertices. Moreover, most of these methods cannot explain why the items are chosen, and they provide limited understanding of the interactions among users, items, and user interests.

The second category includes the research work related to topic models, which are based upon the idea that documents are mixtures of topics, where a topic is a probability distribution over words. Many kinds of topic models have been proposed, among which PLSA [21] and LDA [8] are most widely used and studied.

Before we describe topic models, we first introduce Latent Semantic Index (LSI), which was first proposed as a method for automatic indexing and retrieval [11]. LSI uses a technique called Singular Value Decomposition (SVD) to find the "latent semantic space" by decomposing the original matrix. LSI/SVD have been used for making recommendations probably since 2000 [28], [39]. Also, many SVD-based rating prediction methods are actually one of the successful competitors for the Netflix prize [15], [27], [28]. These low rank recommenders usually treat collaborative filtering as a regression problem of user ratings. Although they perform well in rating predictions, their effectiveness in generating recommendation lists should be further explored, since the rating prediction accuracy is not always consistent with the ranking accuracy [20], [30].

The following PLSA topic model can be viewed as an enhancement of LSI. PLSA has a sound statistical foundation and has defined a proper generative model of the data [21]. Also, PLSA is based on the observation that user preferences and item characteristics are governed by a few latent semantics. As a statistical model, PLSA is able to capture the complex dependences among different factors using well-defined probabilistic semantics [30]. PLSA has been used both for automatic question recommendation [51] and collaborative filtering [22]. While PLSA has been successfully developed, it suffers from the *overspecialization* problem.

Compared with PLSA, the LDA model possesses fully generative semantics and has also been widely researched [5], [9], [47]. LDA is heavily cited in many text-related tasks, such as finding scientific topics [19] and the information retrieval tasks [49], but its feasibility and effectiveness in collaborative filtering is largely underexplored. Sometimes, topic models were only used to reduce the dimensionality of the data [10], [22], like the function of principal component analysis [17]. In previous topic-model-based collaborative-filtering algorithms, the correlation between latent factors has never been considered; thus, they easily suffer from the *overspecialization* problem and the cold-start problem.

The third category of related work has a focus on solving the *overspecialization* problem in recommender systems. This happens when the user is limited to being recommended the items that are "similar" (with respect to content) to those already rated [2]. In other words, at this time, users' new or latent interests will never be explored. This problem bothers most of the existing recommender systems, particularly for the content-based approaches, where many studies have attempted to find the solutions, for instance, filtering out the items which are too similar to something the user has seen before [6] or introducing some kind of serendipity [24].

Since the *overspecialization* problem can be somewhat alleviated by the use of similar user interests, this problem has been largely ignored by most of the collaborative-filtering works. However, some efforts have been dedicated to the solutions of this issue. Among them, one possible approach is to consider the transitive similarities in item-based collaborative filterings [18], [52]. However, directly computing the transitive similarities between items will increase both the space and time costs. Another approach is to introduce diverse recommendations. For instance, Ziegler *et al.* [55] determined the overall diversity of the collaborative recommendations by introducing the content information. Zhang *et al.* [54] modeled the goals of maximizing the diversity of the recommendations while maintaining adequate similarity to the user query as an optimization problem, and they applied this technique to an item-based recommendation algorithm. Furthermore, a survey about some diversity enhancement algorithms was made in [53]. While the performances of these systems can be improved by introducing diversity, most of them suffer from a tradeoff between diversity and the recommending accuracy. A key reason is that they neglect the fact that diversity should be made by exploiting users' possible interest expansion instead of randomly choosing some explicit interests.

The fourth category of related work is focused on solving the cold-start problem. Cold-start problem will happen when the recommender systems try to give recommendations to the users whose preference are underexplored or try to recommend the new items whose characteristics are also unclear [2]. Thus, it can be further classified as the item-side cold-start problem [42] and the user-side cold-start problem [29].

For the content-based or the hybrid recommender systems, where there are profile descriptions, this problem can be alleviated by understanding items or users with such content information. For instance, to deal with the item-side cold-start problem, Schein *et al.* proposed a probabilistic model that combines item content and the collaborative information for recommendation [42]. To address the user-side cold-start problem, Lam *et al.* proposed a User-Info Aspect Model by using information of users, such as age and gender [29].

However, for collaborative filtering, where there are no content information, the only way to address the cold-start problem is to understand both users and items better from the limited and sparse rating records. For instance, in order to improve the recommendation performance under cold-start conditions, Ahn [4] designed a heuristic similarity measure based on the minute meanings (i.e., proximity, impact, and popularity) of coratings. Aside from exploring information

from the direct relations among items (i.e., coratings), other methods consider the indirect similarities. For instance, Huang *et al.* [23] applied associative retrieval techniques to generate transitive associations in the user–item bipartite graph. In [35], for alleviating the sparsity and the cold-start problems, the authors proposed a method using the trust inferences, which are also transitive associations between users. Meantime, similar to this paper, many random-walk-based similarity methods have been used in [14], [18], and [52]. However, these methods consider the relationship between items or user–item pairs, rather than the correlation between latent interests. Meanwhile, as mentioned previously, with the increase of new items, users, or rating records, both their space and time costs will rise rapidly.

V. Concluding Remarks

In this paper, we exploited user latent interests for developing an item-oriented model-based collaborative framework, named iExpand. Specifically, in iExpand, a topic-model-based method is first used to capture each user's interests. Then, a personalized ranking strategy is developed for predicting a user's possible interest expansion. Moreover, a diverse recommendation list is generated by using user latent interests as an intermediate layer between the user layer and the item layer. There are two key benefits of iExpand. First, the three-layer representation enables a better understanding of the interactions among users, items, and user interests and leads to more accurate ranking recommendation results. Second, since the user interests and the change of the interests have been taken into the consideration, iExpand can keep track of these changes and significantly mitigate the *overspecialization* problem and the cold-start problem.

Finally, an empirical study has been conducted on three benchmark data sets, namely, *MovieLens*, *Book-Crossing*, and *Jester*. The corresponding experimental results demonstrate that iExpand can lead to better ranking performances than state-of-the-art methods including two graph-based collaborative-filtering algorithms and two dimension-reduction-based algorithms. Due to an intellectual use of dimension-reduction techniques, iExpand also has low computational cost and is highly scalable for a large number of users, items, and rating records. In the future, we plan to overcome the limitations of the current model and extend it to go beyond the usual recommendations. In particular, we want to refine the iExpand model so as to deal with the context-aware user–interests mining problem.

Acknowledgment

The authors are would like to thank the anonymous reviewers for their constructive comments. Q. Liu would like to thank the China Scholarship Council for their support.

References

[1] Movielens Datasets, 2007. [Online]. Available: http://www.grouplens.org/node/73#attachments

[2] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[3] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender Systems Handbook*. New York: Springer-Verlag, 2011, pp. 217–253.

[4] H. J. Ahn, "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem," *Inf. Sci.*, vol. 178, no. 1, pp. 37–51, Jan. 2008.

[5] A. Asuncion, M. Welling, P. Smyth, and Y. W. Teh, "On smoothing and inference for topic models," in *Proc. Int. Conf. UAI*, 2009, pp. 27–34.

[6] D. Billsus and M. J. Pazzani, "User modeling for adaptive news access," *User Model. User-Adapted Interaction*, vol. 10, no. 2, pp. 147–180, 2000.

[7] D. M. Blei and J. D. Lafferty, "A correlated topic model of science," *Ann. Appl. Statist.*, vol. 1, no. 1, pp. 17–35, 2007.

[8] D. M. Blei, Y. N. Andrew, and I. J. Michael, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[9] K. R. Canini, L. Shi, and T. L. Griffiths, "Online inference of topics with Latent Dirichlet allocation," in *Proc. 12th Int. Conf. AISTATS*, 2009, vol. 5, pp. 65–72.

[10] W. 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. 18th Int. Conf. WWW*, 2009, pp. 681–690.

[11] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.

[12] S. Debnath, N. Ganguly, and P. Mitra, "Feature weighting in content based recommendation system using social network analysis," in *Proc. 17th Int. Conf. WWW*, 2008, pp. 1041–1042.

[13] Y. Ding and X. Li, "Time weight collaborative filtering," in *Proc. 14th ACM Int. CIKM*, 2005, pp. 485–492.

[14] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 355–369, Mar. 2007.

[15] S. Funk, Netflix Update: Try This at Home, 2006. [Online]. Available: http://sifter.org/~simon/journal/20061211.html

[16] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. J. Pazzani, "An energy-efficient mobile recommender system," in *Proc. 16th ACM SIGKDD Int. Conf. KDD*, 2010, pp. 899–908.

[17] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inf. Retrieval*, vol. 4, no. 2, pp. 133–151, Jul. 2001.

[18] M. Gori and A. Pucci, "A random-walk based scoring algorithm applied to recommender engines," in *Proc. 8th Int. Workshop Knowl. Discov. Web (WebKDD)—Advances in Web Mining and Web Usage Analysis*, 2006, pp. 127–146.

[19] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proc. Nat. Acad. Sci. U.S.A. (PNAS)*, vol. 101, pp. 5228–5235, 2004.

[20] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst. (TOIS)*, vol. 22, no. 1, pp. 5–53, Jan. 2004.

[21] T. Hofmann, "Probabilistic latent semantic analysis," in *Proc. 15th Conf. UAI*, 1999, pp. 289–296.

[22] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst. (TOIS)*, vol. 22, no. 1, pp. 89–115, Jan. 2004.

[23] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM Trans. Inf. Syst. (TOIS)*, vol. 22, no. 1, pp. 116–142, Jan. 2004.

[24] L. Iaquinta, M. de Gemmis, P. Lops, and G. Semeraro, "Introducing serendipity in a content-based recommender system," in *Proc. HIS*, 2008, pp. 168–173.

[25] G. Jeh and J. Widom, "Scaling personalized web search," in *Proc. 12th Int. Conf. WWW*, 2003, pp. 271–279.

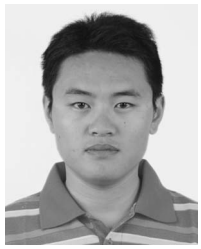
[26] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. KDD*, 2008, pp. 426–434.

[27] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. 15th ACM SIGKDD Int. Conf. KDD*, 2009, pp. 447–456.

[28] M. Kurucz, A. A. Benczur, and K. Csalogany, "Methods for large scale SVD with missing values," in *Proc. KDDCup*, 2007, pp. 31–38.

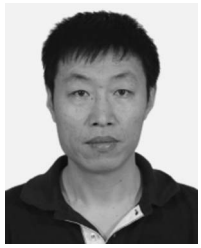
[29] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing cold-start problem in recommendation systems," in *Proc. 2nd Int. Conf. Ubiquitous Inf. Manage. Commun.*, 2008, pp. 208–211.

[30] N. N. Liu, M. Zhao, and Q. Yang, "Probabilistic latent preference analysis for collaborative filtering," in *Proc. 18th ACM CIKM*, 2009, pp. 759–766.
[31] Q. Liu, E. Chen, H. Xiong, and C. H. Q. Ding, "Exploiting user interests for collaborative filtering: Interests expansion via personalized ranking," in *Proc. 19th ACM CIKM*, 2010, pp. 1697–1700.
[32] Q. Mei, X. Shen, and C. Zhai, "Automatic labeling of multinomial topic models," in *Proc. 13th ACM SIGKDD Int. Conf. KDD*, 2007, pp. 490–499.
[33] T. Minka, Estimating a Dirichlet Distribution, 2000. [Online]. Available: http://research.microsoft.com/en-us/um/people/minka/papers/dirichlet/minka-dirichlet.pdf
[34] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Comput. Sci. Dept., Stanford Univ., Stanford, CA, Tech. Rep. 1999-0120, 1998.
[35] M. Papagelis, D. Plexousakis, and T. Kutsuras, "Alleviating the sparsity problem of collaborative filtering using trust inferences," in *Proc. Trust Manage.*, 2005, pp. 224–239.
[36] R. Paul, I. Neophytos, S. Mitesh, B. Peter, and R. John, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proc. ACM Conf. CSCW*, 1994, pp. 175–186.
[37] X. H. Phan, L. M. Nguyen, and S. Horiguchi, "Learning to classify short and sparse text & web with hidden topics from large-scale data collections," in *Proc. 17th Int. Conf. WWW*, 2008, pp. 91–100.
[38] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1257–1264.
[39] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender systems—A case study," in *Proc. ACM WebKDD Workshop*, 2000, pp. 82–90.
[40] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. WWW*, 2001, pp. 285–295.
[41] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *Proc. Adapt. Web*, *Lecture Notes in Computer Science*, 2007, pp. 291–324.
[42] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval (SIGIR)*, 2002, pp. 253–260.
[43] S. Sen, J. Vig, and J. Riedl, "Tagommenders: Connecting users to items through tags," in *Proc. 18th Int. Conf. WWW*, 2009, pp. 671–680.
[44] X. Song, B. L. Tseng, C. Y. Lin, and M. T. Sun, "Personalized recommendation driven by information flow," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval (SIGIR)*, 2006, pp. 509–516.
[45] M. Steyvers and T. Griffiths, "Probabilistic topic models," in *Handbook of Latent Semantic Analysis*, vol. 427. Mahwah, NJ: Lawrence Erlbaum Associates, 2007, pp. 1–15.
[46] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Providing justifications in recommender systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 6, pp. 1262–1272, Nov. 2008.
[47] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. M. Mimno, " Evaluation methods for topic models," in *Proc. 26th Annu. ICML*, 2009, pp. 1105–1112.
[48] H. M. Wallach, "Structured topic models for language," Ph.D. dissertation, Univ. Cambridge, Cambridge, U.K., 2008.
[49] X. Wei and W. B. Croft, "LDA-based document models for ad-hoc retrieval," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval (SIGIR)*, 2006, pp. 178–185.
[50] D. T. Wijaya and S. Bressan, "A random walk on the red carpet: Rating movies with user reviews and pagerank," in *Proc. 17th ACM CIKM*, 2008, pp. 951–960.
[51] H. Wu, Y. Wang, and X. Cheng, "Incremental probabilistic latent semantic analysis for automatic question recommendation," in *Proc. ACM Conf. RecSys*, 2008, pp. 99–106.
[52] H. Yildirim and M. S. Krishnamoorthy, "A random walk method for alleviating the sparsity problem in collaborative filtering," in *Proc. ACM Conf. RecSys*, 2008, pp. 131–138.
[53] M. Zhang, "Enhancing diversity in top-N recommendation," in *Proc. ACM Conf. RecSys*, 2009, pp. 397–400.
[54] M. Zhang and N. Hurley, "Avoiding monotony: Improving the diversity of recommendation lists," in *Proc. ACM Conf. RecSys*, 2008, pp. 123–130.
[55] C. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proc. 14th Int. Conf. WWW*, 2005, pp. 22–32.



Qi Liu received the B.E. degree in computer science from Qufu Normal University, Shandong, China, in 2007. He is currently working toward the Ph.D. degree from the School of Computer and Technology, University of Science and Technology of China, Hefei, China.

He is currently supported by the China Scholarship Council and will stay for a year in Rutgers, The State University of New Jersey, as a Visiting Research Student in the Data Mining Group. His main research interests include intelligent data analysis, recommender systems, and Web data mining. During his Ph.D. study, he has published several papers in refereed conference proceedings and journals.



Enhong Chen (SM'07) received the Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, China.

He is a Professor and the Vice Dean of the School of Computer Science and Technology, USTC. His general areas of research are data mining, personalized recommendation systems, and Web information processing. He has published more than 100 papers in refereed conferences and journals. His research is supported by the National Natural Science Foundation of China, National High Technology Research and Development Program 863 of China, etc. He is the program committee member of more than 20 international conferences and workshops.

Hui Xiong (SM'07) received the B.E. degree from the University of Science and Technology of China, Hefei, China, the M.S. degree from the National University of Singapore, Singapore, and the Ph.D. degree from the University of Minnesota, Minneapolis, MN.

He is currently an Associate Professor and the Vice Department Chair of the Management Science and Information Systems Department, Rutgers University, NJ. His general area of research is data and knowledge engineering, with a focus on developing effective and efficient data analysis techniques for emerging data-intensive applications. He has published over 90 technical papers in peer-reviewed journals and conference proceedings. He is a Coeditor of *Clustering and Information Retrieval* (Kluwer Academic Publishers, 2003) and a Co-Editor-in-Chief of *Encyclopedia of GIS* (Springer, 2008). He is an Associate Editor of the *Knowledge and Information Systems Journal* and has served regularly in the organization and program committees of a number of international conferences and workshops.

Dr. Xiong is a senior member of the Association for Computing Machinery (ACM).

Chris H. Q. Ding (M'09) received the Ph.D. degree from Columbia University, New York, NY.

He is currently a Professor with the Department of Computer Science and Engineering, University of Texas, Arlington (UTA). Prior to joining UTA, he was in the Lawrence Berkeley National Laboratory, University of California, Berkeley, and, prior to that, with the California Institute of Technology, Pasadena. His general research areas are machine learning/data mining and bioinformatics. He also works on information retrieval, Web link analysis, and high-performance computing. His research is supported by National Science Foundation grants and by the University of Texas Regents STARS Award. He has published over 150 research papers in peer-reviewed journals and conference proceedings, and these papers have been cited more than 5000 times. He serves on many program committees of international conferences and gave tutorials on spectral clustering and matrix models. He is an Associate Editor of the journal *Data Mining and Bioinformatics* and is writing a book on spectral clustering to be published by Springer.

Dr. Ding is a member of the IEEE Computer Society since 2000.

Jian Chen (M'95–SM'96–F'08) received the B.Sc. degree in electrical engineering and the M.Sc. and Ph.D. degrees in systems engineering from Tsinghua University, Beijing, China, in 1983, 1986, and 1989, respectively.

He is a Professor and the Chairman of the Management Science Department and the Director of the Research Center for Contemporary Management, Tsinghua University. His main research interests include supply chain management, E-commerce, decision support systems, and modeling and control of complex systems. He has published over 100 papers in refereed journals and has been a principal investigator for over 30 grants or research contracts with the National Science Foundation of China, governmental organizations, and companies. He has presented several plenary lectures.

Dr. Chen is a Ministry of Education Changjiang Scholar and the recipient of the Fudan Management Excellence Award (3rd) and the Science and Technology Progress Award from the Beijing Municipal Government; the Outstanding Contribution Award from the IEEE Systems, Man, and Cybernetics Society; the Science and Technology Progress Award from the State Educational Commission; and the Science and Technology Award for Chinese Youth.