

# Typicality-Based Collaborative Filtering Recommendation

Yi Cai, Ho-fung Leung, Qing Li, *Senior Member, IEEE*, Huaqing Min, Jie Tang, and Juanzi Li

**Abstract**—Collaborative filtering (CF) is an important and popular technology for recommender systems. However, current CF methods suffer from such problems as data sparsity, recommendation inaccuracy, and big-error in predictions. In this paper, we borrow ideas of object typicality from cognitive psychology and propose a novel typicality-based collaborative filtering recommendation method named *TyCo*. A distinct feature of typicality-based CF is that it finds “neighbors” of users based on user typicality degrees in user groups (instead of the corated items of users, or common users of items, as in traditional CF). To the best of our knowledge, there has been no prior work on investigating CF recommendation by combining object typicality. *TyCo* outperforms many CF recommendation methods on recommendation accuracy (in terms of MAE) with an improvement of at least 6.35 percent in Movielens data set, especially with sparse training data (9.89 percent improvement on MAE) and has lower time cost than other CF methods. Further, it can obtain more accurate predictions with less number of big-error predictions.

**Index Terms**—Recommendation, typicality, collaborative filtering

## 1 INTRODUCTION

COLLABORATIVE filtering (CF) is an important and popular technology for recommender systems. There has been a lot of work done both in industry and academia. These methods are classified into user-based CF and item-based CF. The basic idea of user-based CF approach is to find out a set of users who have similar favor patterns to a given user (i.e., “neighbors” of the user) and recommend to the user those items that other users in the same set like, while the item-based CF approach aims to provide a user with the recommendation on an item based on the other items with high correlations (i.e., “neighbors” of the item). In all collaborative filtering methods, it is a significant step to find users’ (or items’) neighbors, that is, a set of similar users (or items). Currently, almost all CF methods measure users’ similarity (or items’ similarity) based on corated items of users (or common users of items). Although these recommendation methods are widely used in E-Commerce, a number of inadequacies have been identified, including:

- *Data Sparsity*. The data sparsity problem is the problem of having too few ratings, and hence, it is difficult to find out correlations between users and items [1]. It occurs when the available data are insufficient for identifying similar users or items. It

is a major issue that limits the quality of CF recommendations [2].

- *Recommendation accuracy*. People require recommender systems to predict users’ preferences or ratings as accurately as possible. However, some predictions provided by current systems may be very different from the actual preferences or ratings given by users [2]. These inaccurate predictions, especially the big-error predictions, may reduce the trust of users on the recommender system.

With the above-mentioned issues, it is clear that a good mechanism to find “neighbors” of users is very important. A better way to select “neighbors” of users or items for collaborative filtering can facilitate better handling of the challenges.

We note that using rated items to represent a user, as in conventional collaborative filtering, only captures the user’s preference at a low level (i.e., item level). Measuring users’ similarity based on such a low-level representation of users (i.e., corated items of users) can lead to inaccurate results in some cases. For example, suppose Bob has only rated five typical war movies with the highest ratings while Tom has rated other five typical war movies with his highest ratings. If we use traditional CF methods to measure the similarity between Bob and Tom, they will not be similar at all, for the reason that there is no corated items between Bob and Tom. However, such a result is intuitively not true. Even though Bob and Tom do not have any corated items, both of them are fans of war movies and they share very similar preference on war movies. Thus, we should consider them to be similar to a high degree. Besides, the more sparse the user rating data is, the more seriously traditional CF methods suffer from such a problem.

In reality, people may like to group items into categories, and for each category there is a corresponding group of people who like items in the category [3]. Cognitive psychologists find that objects (items) have different

• Y. Cai and H. Min are with the School of Software Engineering, South China University of Technology, Panyu District, Guangzhou 510006, Guangdong, China.

• H.-f. Leung is with the Department of Computer Science, The Chinese University of Hong Kong, Shatin, Hong Kong, China.

• Q. Li is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China.

• J. Tang and J. Li are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.

Manuscript received 1 June 2011; revised 24 Feb. 2012; accepted 26 Dec. 2012; published online 4 Jan. 2013.

Recommended for acceptance by L. Khan.

For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-06-0309. Digital Object Identifier no. 10.1109/TKDE.2013.7.

typicality degrees in categories in real life [4], [5], [6]. For instance, people may consider that a sparrow is more typical than a penguin in the concept of “bird,” and “Titanic” is a very typical romance movie, and so on. Similarly, different people may have different degrees of typicality in different user groups (i.e., sets of persons who like items of particular item groups). For instance, Raymond is a very typical member of the concept “users who like war movies” while not so typical in the concept “users who like romance movies.” The typicality of users in different user groups can indicate the user’s favor or preference on different kinds of items. The typicality degree of a user in a particular user group can reflect the user’s preference at a higher abstraction level than the rated items by the user.

Thus, in this paper, we borrow the idea of object typicality from cognitive psychology and propose a typicality-based CF recommendation approach named *TyCo*. The mechanism of typicality-based CF recommendation is as follows: First, we cluster all items into several item groups. For example, we can cluster all movies into “war movies,” “romance movies,” and so on. Second, we form a user group corresponding to each item group (i.e., a set of users who like items of a particular item group), with all users having different typicality degrees in each of the user groups. Third, we build a user-typicality matrix and measure users’ similarities based on users’ typicality degrees in all user groups so as to select a set of “neighbors” of each user. Then, we predict the unknown rating of a user on an item based on the ratings of the “neighbors” of at user on the item.

A distinct feature of the typicality-based CF recommendation is that it selects the “neighbors” of users by measuring users’ similarity based on user typicality degrees in user groups, which differentiates it from previous methods. To the best of our knowledge, there has been no prior work on using typicality with CF recommendation. *TyCo* provides a new perspective to investigate CF recommendations. We conduct experiments to validate the *TyCo* and compare it with previous methods. Experiments show that typicality-based CF method has the following several advantages:

- It generally improves the accuracy of predictions when compared with previous recommendation methods.
- It works well even with sparse training data sets, especially in data sets with sparse ratings for each item.
- It can reduce the number of big-error predictions.
- It is more efficient than the compared methods.

This paper is organized as follows: Section 2 introduces the background and related work. The typicality-based CF method is introduced in details in Section 3. In Section 4, we evaluate the proposed typicality-based CF method using the MovieLens<sup>1</sup> data set as well as Netflix data set,<sup>2</sup> and compare it with previous methods. The differences among *TyCo* and existing recommendation methods are discussed in Section 5. We conclude the paper in Section 6.

## 2 BACKGROUND AND RELATED WORK

### 2.1 Prototype View and Typicality

In cognitive psychology, object typicality is considered as a measure of the goodness degree of objects as exemplars in concepts [4]. It depends on salient properties shared by most of the objects of that concept, which generally include salient properties that are not necessary for defining the concept [3], as well as those that are.

In the prototype view of concepts [7], a concept is represented by a best prototype abstracted by the property list that consists of the salient properties of the objects that are classified into this concept. The salient properties defining the prototype include both necessary and unnecessary properties. It has been found that typicality of an instance can be determined by the number of its properties which it shares with the concept prototype. For example, the property “can-fly” will probably appear in the prototype of the concept “bird” because most birds can fly. So birds that can fly will be judged as more typical than those that cannot. A prototype of a concept is considered as the best example of the concept, and is abstracted to be a feature list. Although the prototype view can explain many different aspects of how concepts and properties are represented in human’s mind, there are also situations in which it fails to give a thorough explanation. For example, there is virtually no prototype to represent the concept “animal.” It cannot explain the co-occurring relations among properties of an instance, either.

An object is considered as more typical in a concept if it is more similar to the prototype of the concept. Vanpaemel et al. [8] propose a model that extends the prototype view. They consider that a concept is represented by some abstractions (prototypes) deduced from exemplars of the concept. An object is considered to be an instantiation of an abstraction that is most similar to it. Typicality of an object is determined by matching its properties with those of the abstraction that is most similar to it. Vanpaemel et al. show that both the prototype model and exemplar model are special cases of the model they propose, and such a combined model is better than the prototype model and exemplar model.

Barsalou [9] measures two factors named *central tendency* and *frequency of instantiation*, which affect the object typicality in a concept. Central tendency is the degree of an object’s “family resemblance.” The more an object is similar to other members of the same concept (and the less it is similar to the members of the other concepts), the more typical it is in the concept. Frequency of instantiation of a cluster of similar objects in a concept is an estimate of how often people experience, or consider, objects in the cluster as members of a particular concept. Objects of a cluster with higher frequency of instantiation in a concept are more familiar to people, and thus are considered as more typical in the concept.

There are some works on measuring object typicality in computer science. Rifqi [10] proposes a method to calculate object typicality in large databases, which is later extended by Lesot et al. [11]. In their works, the typicality of an object for a category depends on its resemblance to other members of the category, as well as its dissimilarity to members of

1. <http://www.grouplens.org/>.

2. <http://www.netflixprize.com/>.

other categories. Au Yeung and Leung [12] have formalized object typicality in a model of ontologies, in which the typicality of an object in a concept is the degree of similarity matching between the object property vector and the prototype vector of the concept. All these works focus on developing methods to calculate object typicality in concepts. There has been no work on integrating typicality in collaborative filtering recommendation.

## 2.2 Recommender Systems

There have been many works on recommender systems and most of these works focus on developing new methods of recommending items to users (e.g., works in [13], [14]). The objective of recommender systems is to assist users to find out items which they would be interested in. Items can be of any type, such as movies, jokes, restaurants, books, news articles, and so on. Currently, recommendation methods are mainly classified into collaborative filtering (CF), content based (CB), and hybrid methods [2]. For the reason that we are focusing on proposing a new CF method, we will introduce the related works about CF methods in more details.

### 2.2.1 Content-Based Recommender Systems

The inspiration of this kind of recommendation methods comes from the fact that people have their subjective evaluations on some items in the past and will have the similar evaluations on other similar items in the future.

The descriptions of items are analyzed to identify interesting items for users in CB recommender systems. Based on the items a user has rated, a CB recommender learns a profile of user's interests or preferences. According to a user's interest profile, the items which are similar to the ones that the user has preferred or rated highly in the past will be recommended to the user. For CB recommender systems, it is important to learn users' profiles. Various learning approaches have been applied to construct profiles of users. For example, Mooney and Roy [15] adopt text categorization methods in LIBRA system to recommend books. A detailed discussion about CB recommender systems is given by Pazzani and Billsus [16].

### 2.2.2 Collaborative Filtering

CF recommendation methods predict the preferences of active users on items based on the preferences of other similar users or items. For the reason that CF methods do not require well-structured item descriptions, they are more often implemented than CB methods [2], and many collaborative systems are developed in academia and industry. There are two kinds of CF methods, namely user-based CF approach and item-based CF approach [2].

The basic idea of user-based CF approach is to provide recommendation of an item for a user based on the opinions of other like-minded users on that item. The user-based CF approach first finds out a set of nearest "neighbors" (similar users) for each user, who share similar favorites or interests. Then, the rating of a user on an unrated item is predicted based on the ratings given by the user's "neighbors" on the item.

The basic idea of item-based CF approach is to provide a user with the recommendation of an item based on the

other items with high correlations. Unlike the user-based CF, the item-based CF approach first finds out a set of nearest "neighbors" (similar items) for each item. The item-based CF recommender systems try to predict a user's rating on an item based on the ratings given by the user on the neighbors of the target item. For example, Sarwar et al. [17] discuss different techniques for measuring item similarity and obtaining recommendations for item-based CF; Deshpande and Karypis [18] present and evaluate a class of model based top- $N$  recommendation algorithms that use item-to-item or item set-to-item similarities for recommendation.

For both user-based CF and item-based CF, the measurement of similarity between users or items is a significant step. Pearson correlation coefficient, cosine-based similarity, vector space similarity, and so on are widely used in similarity measurement in CF methods [2].

There are some hybrid methods such as [13]. Besides, Huang et al. [1] try to apply associative retrieval techniques to alleviate the sparsity problem. Hu et al. [19] explore algorithms suitable for processing implicit feedbacks. Umyarov and Tuzhilin [20] propose an approach for incorporating externally specified aggregate ratings information into CF methods.

Recently, latent factor model has become popular. A typical latent factor model associates each user  $u$  with a user-factor vector  $p_u$ , and each item  $i$  with an item-factor vector  $q_i$ . The prediction is done by taking an inner product:  $\hat{r}_{ui} = b_{ui} + p_u^T q_i$ . The more involved part is parameter estimation. Some recent works such as [21] have suggested modelling directly only the observed ratings to avoid overfitting through an adequate regularized model. Zhang et al. [22] explore matrix factorization method by flexible regression priors. However, latent factor models such as SVD face real difficulties to explain predictions. Backstrom et al. [23] adopt supervised random walks for predicting and recommending links in social networks. Lee et al. [24] rank entities on Graph by random walks for multidimensional recommendation. Zhou et al. [25] propose a method named functional matrix factorization to handle the cold-start problem. Ma et al. [26] use a collective probabilistic factor model for website recommendation. Leung et al. [27] propose a collaborative location recommendation framework based on coclustering.

### 2.2.3 Hybrid Recommender Systems

Several recommender systems (e.g., [28] and [29]) use a hybrid approach by combining collaborative and content-based methods, so as to help avoid some limitations of content-based and collaborative systems. A naive hybrid approach is to implement collaborative and CB methods separately, and then combine their predictions by a combining function, such as a linear combination of ratings or a voting scheme or other metrics. Melville et al. [28] use a CB method to augment the rating matrix and then use a CF method for recommendation.

Some hybrid recommender systems combine item-based CF and user-based CF. For example, Ma et al. [13] propose an effective missing data prediction (EMDP) by combining item-based CF and user-based CF.

### 3 TYPICALITY-BASED COLLABORATIVE FILTERING

In this section, we propose a typicality-based collaborative filtering approach named *TyCo*, in which the “neighbors” of users are found based on user typicality in user groups instead of co-rated items of users. We first introduce some formal definitions of concepts in *TyCo* in Section 3.1. The mechanism of *TyCo* is then described in Section 3.2. We introduce its technical details in Sections 3.3, 3.4, 3.5, and 3.6.

#### 3.1 Preliminaries

Assume that in a CF recommender system, there are a set  $U$  of users, and a set  $O$  of items. Items can be clustered into several item groups and an *item group* is intuitively a set of similar items. For example, movies can be clustered into action movies, war movies, and so on. Each movie belongs to different movie groups to different degrees. The choice of clustering method is application domain dependent, and is out of the scope of this paper. For instance, based on the keyword descriptions of movies, we can use Topic Model-based clustering [30], [31] for the task of obtaining movie groups and the degrees of movies belonging to movie groups. In other application domains, other clustering approaches (such as [32], [33]) can also be used. In this paper, we will not discuss clustering methods further.

The formal definition of an item group is given in the following.

**Definition 3.1.** An item group denoted by  $k_i$  is a fuzzy set of objects, as following:

$$k_i = \{O_1^{w_{i,1}}, O_2^{w_{i,2}}, \dots, O_h^{w_{i,h}}\},$$

where  $h$  is the number of items in  $k_i$ ,  $O_x$  is an item, and  $w_{i,x}$  is the grade of membership of item  $O_x$  in  $k_i$ .

Users who share similar interests on an item group could form a community, and we name such a community as a *user group*. Users have different typicality degrees in different user groups. In other words, for each item group  $k_i$ , we define a corresponding user group (i.e., a fuzzy set of users who like objects in  $k_i$ ) to some degrees. For instance, Bob and Raymond are very interested in war movies but not so interested in romance movies, while Amy and Alice like romance movies very much but do not like war movies. Thus, Bob and Raymond are typical users in the user group of users who like war movies, but not typical users in the user group corresponding to romance movies; while Amy and Alice are typical users in the user group of users who like romance movies but not typical in that of war movies.

We consider a user group  $g_i$  corresponding to an item group  $k_i$  as a fuzzy concept “users who like the items in  $k_i$ .” Note that users may have different typicality degrees in different  $g_i$ . The following is the formal definition of a user group.

**Definition 3.2.** A user group  $g_i$  is a fuzzy set of users, as follows:

$$g_i = \{U_1^{v_{i,1}}, U_2^{v_{i,2}}, \dots, U_m^{v_{i,m}}\},$$

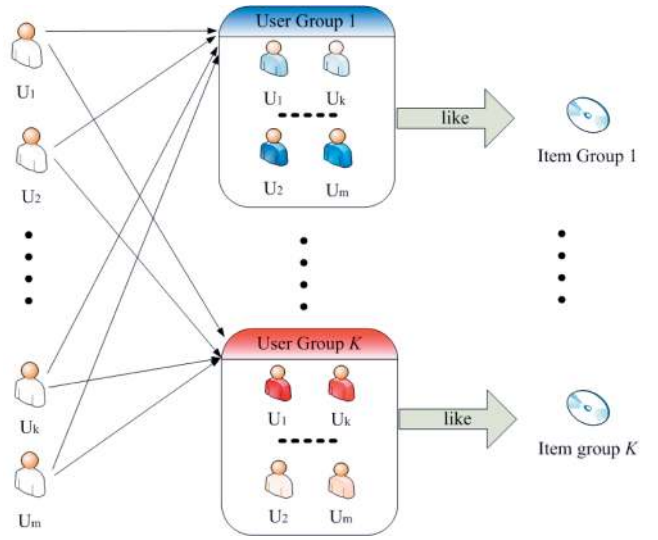


Fig. 1. The relations among users, user groups, and item groups.

where  $m$  is the number of users in the group  $g_i$ ,  $U_x$  is a user, and  $v_{i,x}$  is the typicality degree of user  $U_x$  in user group  $g_i$ .

The relations among users, user groups, and item groups are as shown in Fig. 1. Users possess different typical degrees in different user groups: the darker a user is in Fig. 1, the more typical it is in that user groups. For examples,  $U_1$  and  $U_k$  are typical in user group  $g_k$  but not typical in  $g_1$ , while  $U_2$  and  $U_m$  are typical in  $g_1$  but not typical in  $g_k$ .

For the reason that users have different typicality degrees in different user groups, we represent a user by a *user typicality vector* defined below:

**Definition 3.3.** A user typicality vector  $\vec{U}_j$  of a user  $U_j$  is a vector of real numbers in the closed interval  $[0, 1]$ , defined by

$$\vec{U}_j = (v_{1,j}, v_{2,j}, \dots, v_{n,j}),$$

where  $n$  is the number of user groups and  $v_{i,j}$  is the typicality degree of user  $U_j$  in the user group  $g_i$ .

Thus, for all users, we can obtain a *user-typicality matrix* as follows.

**Definition 3.4.** A user-typicality matrix, denoted by  $M_\tau$ , is a matrix with the  $i$ th row being user typicality vector of user  $U_i$ :

$$M_\tau = \begin{Bmatrix} \vec{U}_1 \\ \vdots \\ \vec{U}_m \end{Bmatrix} = \begin{Bmatrix} v_{1,1}, v_{2,1}, \dots, v_{n,1} \\ \vdots \\ v_{1,m}, v_{2,m}, \dots, v_{n,m} \end{Bmatrix},$$

where  $m$  is the number of users,  $n$  is the number of user groups and  $\vec{U}_i$  is the user typicality vector of user  $U_i$ .

Fig. 2 shows an example user-typicality matrix.

#### 3.2 Mechanism of TyCo

The mechanism of *TyCo* is as follows: given a set  $O = \{O_1, O_2, \dots, O_h\}$  of items and a set  $U = \{U_1, U_2, \dots, U_m\}$  of users, a set  $K = \{k_1, k_2, \dots, k_n\}$  of item groups is formed. For each item group  $k_i$ , there is a corresponding user group  $g_i$ . Users have different typicality degrees in each  $g_i$ . Then, a user typicality vector  $\vec{U}_i$  is built for each user, from which

	$g_1$	$g_2$	$g_3$	$g_4$	$g_5$	$g_6$
$U_1$	0.87	0.75	0.92	0.12	0.32	0.28
$U_2$	0.34	0.21	0.38	0.89	0.85	0.94
$\vdots$	...	...	...	...	...	...
$U_k$	0.81	0.79	0.89	0.15	0.29	0.31
$\vdots$	...	...	...	...	...	...
$U_m$	0.41	0.22	0.35	0.90	0.88	0.92

Fig. 2. An example of user-typicality matrix in TyCo.

user-typicality matrix  $M_\tau$  is obtained. After obtaining users' similarity based on their typicality degrees in user groups, a set  $\bar{N}_i$  of "neighbors" is obtained for each user. Then, we predict the rating of an active user on an item based on the ratings by "neighbors" of that user on the same item.

Selecting "neighbors" of users by measuring users' similarity based on their typicality degrees is a distinct feature, which differentiates our approach from previous CF approaches. In TyCo, we measure the similarity of two users  $U_a$  and  $U_b$  based on the comparison of their typicality vectors in  $M_\tau$ . For example, according to Fig. 2,  $U_1$  and  $U_k$  are similar users. The reason is that  $U_1$  and  $U_k$  possess similar typicality degrees in all user groups  $g_1$  to  $g_6$ . Similarly,  $U_2$  is similar to  $U_m$ . In previous CF methods, similarity of two users depends on the comparison of corated items of them (i.e., the more similar ratings of two users on corated items are, the more similar the two users are). Fig. 3 shows an example user-item rating matrix in the traditional CF approach.

In TyCo, as a user is described by a user typicality vector, each element in the vector can be considered as a feature of a user. Such a representation can indicate the user's preference on items at a higher abstraction level than representing a user by a set of rated items. Furthermore, measuring users' similarity based on users' typicality can overcome the limitation of traditional CF mentioned in Section 1. Let us reconsider the example of Bob and Tom in Section 1. In that example, Bob has only rated five typical war movies with the highest ratings while Tom has only rated other five typical war movies with the highest ratings. Using TyCo we can find that Bob and Tom are very typical users in the user group "users who like war movies" and not typical users in other user groups (because they have not rated any other kinds of movies in their histories). Then, we can find that Bob and Tom are very similar by comparing their typicality vectors. Such a result is intuitively more reasonable than that of traditional CF.

We present details of measuring user typicality, selecting "neighbors" of users, and predicting ratings in the following sections.

### 3.3 Item Typicality Measurement

As introduced above, the typicality of an object in a concept depends on the central tendency of the object for the prototype of the concept. In other words, if an object is more similar to the prototype of a concept, it has a higher typicality degree in the concept. Generally, an item is

	$i_1$	$i_2$	...	$i_k$	...	$i_n$
$U_1$	5	?	...	3	...	4
$U_2$	?	?	...	4	...	5
$\vdots$	...	...	...	...	...	...
$U_k$	2	5	...	?	...	3
$\vdots$	...	...	...	...	...	...
$U_m$	5	4	...	2	...	?

Fig. 3. An example of user rating matrix in traditional CF.

represented by a set of properties, which, following our previous work [34], we shall call *item property vector*. For example, keywords, actors, directors, and producers are properties of a movie and these properties can form an item property vector to represent a movie. For each item group  $k_j$ , we can extract a prototype to represent the item group. The prototype of  $k_j$  is represented by a set of properties denoted by the *prototype property vector* of  $k_j$   $\vec{t}_{k_j}$ , as follows:

$$\vec{t}_{k_j} = (p_{k_j,1} : r_{k_j,1}, p_{k_j,2} : r_{k_j,2}, \dots, p_{k_j,m} : r_{k_j,m}),$$

where  $m$  is the number of the properties of the prototype of concept (item group)  $k_j$ , and  $r_{k_j,i}$  is a real number (between 0 and 1), which indicates the degree the prototype of concept  $k_j$  possesses the property  $p_{k_j,i}$ .<sup>3</sup> The typicality of an item  $O_y$  in an item group  $k_j$ , denoted by  $w_{j,y}$ , depends on the similarity between the item  $O_y$  and the prototype of  $k_j$ , i.e.,

$$w_{j,y} = Sim(\vec{t}_{k_j}, \vec{p}_{O_y}),$$

where  $\vec{t}_{k_j}$  is the prototype property vector of item group  $k_j$ ,  $\vec{p}_{O_y}$  is the item property vector of  $O_y$ , and  $Sim$  is a similarity function. We shall present some possible similarity functions in Section 3.5.

We regard an item group as a fuzzy set and there is only one prototype to represent an item group (i.e., a cluster of similar items). Thus, the frequency of instantiation of the unique prototype for the item group is 1, and the typicality degree of an item in an item group only depends on the central tendency. Based on the works in [36] and [9], the central tendency of an object to a concept is affected by the degrees of the internal similarity and external dissimilarity. Internal similarity is the similarity of the object property vector of the item and the prototype property vector of the item group. External dissimilarity is the similarity of the object property vector of the item and prototype property vectors of other item groups.

In TyCo, the similarity between a prototype of a concept  $c$  and an object  $a$  is calculated by the following function:

$$Sim : P \times T \rightarrow [0, 1],$$

where  $T$  is the set of all prototype vectors, and  $P$  is the set of all object property vectors. For the dissimilarity between the unique prototype of a concept  $c$  and an object  $a$  in

3. Following [4] and [35], we consider the prototype as the mean of a cluster.

our method, we define it as the complement of similarity, as follows:

$$Dissimilar(\vec{p}_a, \vec{t}_c) = 1 - Sim(\vec{p}_a, \vec{t}_c).$$

The output of the  $Sim$  function is a real number in the range of  $[0, 1]$ . The object  $a$  is identical to the prototype of concept  $c$  if  $Sim(\vec{t}_c, \vec{p}_a) = 1$ . If  $Sim(\vec{t}_c, \vec{p}_a) = 0$ , then the object  $a$  is not similar to the prototype of concept  $c$  at all.

The internal similarity of object  $a$  for concept  $c$ , denoted by  $\beta(\vec{p}_a, \vec{t}_c)$ , is determined as follows:

$$\beta(\vec{p}_a, \vec{t}_c) = Sim(\vec{p}_a, \vec{t}_c).$$

In our method, the external dissimilarity  $\delta(\vec{p}_a, \vec{t}_c)$  is considered as the average of dissimilarities of the object and other concepts excluding  $c$ , and is defined as follows:

$$\delta(\vec{p}_a, \vec{t}_c) = \frac{\sum_{x \in C/\{c\}} Dissimilar(\vec{p}_a, \vec{t}_x)}{N_c - 1},$$

where  $C$  is the set of concepts in the domain, and  $N_c$  is the number of concepts.

The central tendency of an object  $a$  to concept  $c$ , denoted by  $\alpha(\vec{p}_a, \vec{t}_c)$ , is defined as an aggregation of internal similarity and external dissimilarity of  $a$  for  $c$ :

$$\alpha(\vec{p}_a, \vec{t}_c) = Aggr(\beta(\vec{p}_a, \vec{t}_c), \delta(\vec{p}_a, \vec{t}_c)),$$

where  $Aggr$  is an application-dependent aggregation function [11] used to combine the effects of internal similarity and external dissimilarity, as discussed in [36]. The following is a possible function as an example to aggregate the internal similarity and external dissimilarity:

$$\alpha(\vec{p}_a, \vec{t}_c) = \beta(\vec{p}_a, \vec{t}_c) \cdot \delta(\vec{p}_a, \vec{t}_c). \quad (1)$$

### 3.4 User Typicality Measurement

The prototype of a user group is computed from the properties describing users [4]. Most data sets of existing recommender systems have little information related to users' interests, and the ratings of users on items are the main related information for describing users' interests. For the reason that a user group  $g_x$  is a fuzzy concept "users who like the items in the corresponding item group  $k_x$ ," we regard a prototype of a user group  $g_x$  as consisting of two properties. One property is "having rated items in the corresponding item group  $k_x$  to the highest degrees," denoted by  $p_{g_x,r}$ , and another property is "having frequently rated items in the corresponding item group  $k_x$ ," denoted by  $p_{g_x,f}$ . Thus, we abstract the prototype of a user group  $g_x$  to be represented by a *prototype property vector* of  $g_x$ , denoted by  $\vec{t}_{g_x}$ , as follows:

$$\vec{t}_{g_x} = (p_{g_x,r} : 1, p_{g_x,f} : 1).$$

The value 1 means that the prototype possesses both the properties  $p_{g_x,r}$  and  $p_{g_x,f}$  to the degree of 1.

To measure the typicality of a user  $U_i$  in a user group  $g_x$ , we need to build a user property vector for  $U_i$  and compare it with the prototype of the user group  $g_x$  to obtain its central tendency. According to the prototype property vector  $\vec{t}_{g_x}$ , the central tendency degree of a user  $U_i$  for a user group  $g_x$  depends on the ratings of the user  $U_i$  on items

in the corresponding item group  $k_x$  and the number of items that user  $U_i$  has rated in  $k_x$ . For users in a user group  $g_x$ , the higher the ratings of a user on items in the corresponding item group are, and the more frequently the user rates items in  $g_x$ , the more typical the user is as a member in  $g_x$ .

We also need to build for each user group a *user property vector* to represent a user, so that we can compare it with the prototype property vector of the user group. The user property vector of a user  $U_i$  for a user group  $g_x$  is denoted by  $\vec{p}_{i,g_x}$  and given by

$$\vec{p}_{i,g_x} = (p_{g_x,r} : s_{g_x,r}^i, p_{g_x,f} : s_{g_x,f}^i),$$

where  $s_{g_x,r}^i$  and  $s_{g_x,f}^i$  are the degrees to which the user  $U_i$  possesses the property  $p_{g_x,r}$  and  $p_{g_x,f}$ , respectively.

We consider that  $s_{g_x,r}^i$  is the weighted sum average aggregation of all ratings of  $U_i$  on items in  $k_x$  which corresponds to  $g_x$ , and is given by

$$s_{g_x,r}^i = \frac{\sum_{y=1}^n w_{x,y} \cdot R_{i,y}}{n \cdot R_{max}},$$

where  $n$  is the number of items user  $U_i$  has rated in the item group  $k_x$ ,  $R_{i,y}$  is the rating of  $U_i$  on item  $O_y$ ,  $w_{x,y}$  is the degree of  $O_y$  belonging to item group  $k_x$ , and  $R_{max}$  is the maximal rating value. Besides,  $s_{g_x,f}^i$  is the degree of oftenness of the user's rating items in item group  $k_x$ , which is calculated as follows:

$$s_{g_x,f}^i = \frac{N_{x,i}}{N_i} = \frac{N_{x,i}}{\sum_{y=1}^n N_{y,i}},$$

where  $n$  is the number of item groups (i.e., the number of user groups),  $N_{x,i}$  is the number of items having been rated by user  $U_i$  in the item group  $k_x$ , and  $N_i$  is the number of items having been rated by  $U_i$  in all item groups.

The typicality of a user  $U_i$  in a user group  $g_x$  depends on the comparison of user property vector  $\vec{p}_{i,g_x}$  and prototype property vector  $\vec{t}_{g_x}$ . Since all values of properties in  $\vec{t}_{g_x}$  are 1, the typicality of user  $U_i$  in  $g_x$ , denoted by  $v_{i,g_x}$ , is calculated by a combination function  $\tau_{g_x}(U_i)$  of  $s_{g_x,r}^i$  and  $s_{g_x,f}^i$ . Here, we present a possible combination function<sup>4</sup> as follows:

$$\tau_{g_x}(U_i) = \frac{s_{g_x,r}^i + s_{g_x,f}^i}{2}.$$

Higher values of  $s_{g_x,r}^i$  and  $s_{g_x,f}^i$  indicate higher similarity between the user property vector and the prototype property vector, and thus, the  $U_i$  is more typical in  $g_x$ .

### 3.5 Neighbors Selection

We select a fuzzy set of "neighbors" of user  $U_j$ , denoted by  $\vec{N}_j$ , by choosing users who are sufficiently similar to  $U_j$ , i.e.,

$$\vec{N}_j = \{U_i \mid Sim(U_i, U_j) \geq \gamma\},$$

where  $Sim(U_i, U_j)$  is the similarity of  $U_i$  and  $U_j$  and  $\gamma$  is a threshold to select users who are qualified as "neighbors" of user  $U_j$ .

4. There are other possible functions to combine  $s_{g_x,r}^i$  and  $s_{g_x,f}^i$ , but here we only present a simple one which is also used in our experiment. Please refer to [37] for more details.

The neighbor selection is a very important step before prediction because the prediction ratings of an active user on items will be inaccurate if the selected neighbors are not sufficiently similar to the active user. Instead of selecting the top- $k$  neighbors, we set a threshold  $\gamma$  for selecting neighbors. If the similarity of a candidate user  $U_i$  and the active user  $U_j$  is greater than or equal to the threshold  $\gamma$ ,  $U_i$  will be selected into  $\overrightarrow{N}_j$  (i.e., the set of “neighbors” of  $U_j$ ).

The choice of similarity functions depends on applications. There have been a number of methods proposed to calculate the similarity between two objects, as discussed in [38]. Here, we consider three methods, namely, distance-based similarity, cosine-based similarity, and correlation-based similarity.

*Distance-based similarity.* According to [38], similarity between two objects is derived from the distance between them through decreasing functions. The distance between two users depends on matching of their corresponding properties. The similarity between  $U_i$  and  $U_j$  is measured as follows:

$$Sim(U_i, U_j) = \exp\left(-\sqrt{\sum_{y=1}^n |v_{i,y} - v_{j,y}|^2}\right),$$

where  $n$  is the number of user groups, and

$$\sqrt{\sum_{y=1}^n |v_{i,y} - v_{j,y}|^2}$$

is the euclidean distance between  $\vec{U}_i$  and  $\vec{U}_j$ .

*Cosine-based similarity.* In TyCo, a user is represented by a user typicality vector. In this case, the similarity between users  $U_i$  and  $U_j$  is calculated by computing the cosine of the angle between these two vectors, i.e.,

$$Sim(U_i, U_j) = \frac{\sum_{x=0}^n v_{x,i} v_{x,j}}{\sqrt{\sum_{x=0}^n v_{x,i}^2} \cdot \sqrt{\sum_{x=0}^n v_{x,j}^2}},$$

where “.” is the dot-product operator of two vectors, and  $v_{x,i}$  is the typicality degree of user  $U_i$  in the user group  $g_x$ .

*Correlation-based similarity.* Pearson Correlation is very popular for measuring similarity of users or items in existing CF methods. The similarity between users  $U_i$  and  $U_j$  is calculated as follows:

$$Sim(U_i, U_j) = \frac{\sum_{x=0}^n (v_{x,i} - \bar{v}_i) \cdot (v_{x,j} - \bar{v}_j)}{\sqrt{\sum_{x=0}^n (v_{x,i} - \bar{v}_i)^2} \cdot \sqrt{\sum_{x=0}^n (v_{x,j} - \bar{v}_j)^2}},$$

where  $v_{x,i}$  is the typicality degree of user  $U_i$  in the user group  $g_x$ , and  $\bar{v}_i$  is the average typicality of users  $U_i$  in all user groups.

We evaluate experimentally in Section 4 the different effects of these three similarity functions on recommendation quality.

### 3.6 Prediction

Having obtained the set of “neighbors” of each user, we can predict the rating of an active user  $U_i$  on an item  $O_j$ , denoted by  $R(U_i, O_j)$ , based on the ratings of all “neighbors” of  $U_i$  on  $O_j$ , as follows:

$$R(U_i, O_j) = \frac{\sum_{U_x \in \overrightarrow{N}_i} R(U_x, O_j) \cdot Sim(U_x, U_i)}{\sum_{U_x \in \overrightarrow{N}_i} Sim(U_x, U_i)},$$

where  $U_x$  is a user in the set of “neighbors” of  $U_i$ ,  $R(U_x, O_j)$  is the rating of user  $U_x$  on item  $O_j$ , and  $Sim(U_x, U_i)$  is the similarity between  $U_x$  and  $U_i$ . This function calculates a weighted sum of all ratings given by the “neighbors” of  $U_i$  on  $O_j$ .

## 4 EXPERIMENTS

We report the results obtained from experiments conducted to compare the typicality-based CF (TyCo) with other current CF methods. We want to address the following questions:

1. How does TyCo compare with other current CF methods?
2. Does TyCo have a good performance with sparse training data?
3. Can TyCo obtain a good result with less big-error predictions?
4. Is TyCo more efficient than other existing methods?
5. How does the number of user groups affect the recommendation quality?
6. How do the similarity function and threshold  $\gamma$  affect the recommendation results?

### 4.1 Experimental Setting

#### 4.1.1 Data Set Description

To evaluate our recommendation method, we use the MovieLens data set in the experiments, as this data set has been widely used in previous papers such as [17], [39]. From the MovieLens data set, we obtain 100,000 ratings, assigned by 943 users on 1,682 movies. Each user has rated at least 20 movies, and the ratings follow the 1 (bad) to 5 (excellent) numerical scale. The sparsity level of the data set is  $1 - \frac{100,000}{943 \times 1,682}$ , which is 0.9369. Another data set we use is the Netflix data set, which contains 100,480,507 ratings that 480,189 users have given to 17,770 movies. The sparsity level of Netflix data set is  $1 - \frac{100,480,507}{480,189 \times 17,770}$ , which is 0.9882. We extract keywords of movies from the Internet Movie Database (IMDB),<sup>5</sup> and regard such keywords as the descriptions of movies.

#### 4.1.2 Metrics

To measure statistical accuracy, we use the *mean absolute error* (MAE) metric, which is defined as the average absolute difference between predicted ratings and actual ratings [2]. MAE is a measure of deviation of recommendations from real user-rated ratings, and it is most commonly used and very easy to interpret. It is computed by averaging the all sums of the absolute errors of the  $n$  corresponding ratings-prediction pairs, and can be formally defined as follows:

$$MAE = \frac{\sum_{i=1}^n |f_i - h_i|}{n},$$

where  $n$  is the number of rating-prediction pairs,  $f_i$  is an actual user-specified rating on an item, and  $h_i$  is the

5. <http://www.imdb.com/interfaces>.

TABLE 1  
Sensitivity of  $n$  on MAE with Different Train/Test Ratios

	$\gamma$	$\chi=0.1$	$\chi=0.3$	$\chi=0.5$	$\chi=0.7$	$\chi=0.9$
$n=5$	0.8	0.8106	0.771	0.7478	0.7436	0.7361
$n=10$	0.7	0.8115	0.7771	0.7546	0.7451	0.7394
$n=15$	0.6	0.8117	0.7774	0.7563	0.7502	0.7393
$n=20$	0.6	0.8125	0.7757	0.7568	0.7481	0.735
$n=25$	0.5	0.8136	0.777	0.7576	0.7515	0.739
$n=30$	0.5	0.8129	0.7726	0.7536	0.7438	0.7349
AVG		0.8121	0.7751	0.7544	0.747	0.7373

prediction for a user on an item given by the recommender system. A lower MAE value means that the recommendation method can predict users' ratings more accurately. Thus, for MAE values of a recommendation method, the smaller the better.

Another metric for evaluation is *Coverage*. Coverage measures the percentage of items for which a recommender system is capable of making predictions [2]. For example, if a recommender system can predict 8,500 out of 10,000 ratings on items to be predicted, the coverage is 0.85. So a larger Coverage means that the recommendation method can predict more ratings for users on unrated items. Thus, for Coverage values of a recommendation method, the larger the better.

#### 4.1.3 Experiment Process

For TyCo, we first use the Topic model-based clustering [30] to cluster the movies described by keywords. Based on the clustering results for item groups, we then form a corresponding user group for each item group and build a prototype for each user group. To evaluate TyCo thoroughly. Similar to previous works such as [40], [14] and [13], we conduct two experiments.

The objective of the first experiment is to explore the impact of group number, recommendation quality, and the performance with sparse data, by comparing TyCo with several classic baseline methods. We adopt several classic recommendation methods for the comparison, which include a content-based method with cosine similarity function, a user-based CF with Pearson Correlation Coefficient (UBCF), an item-based CF with Pearson Correlation Coefficient (IBCF), a naive hybrid method, and a CF method with effective missing data prediction in [13]. We divide the data set into two parts, the training set and the test set. We obtain the recommendation predictions based on the training set and use test set to evaluate the accuracy of TyCo. We randomly choose user-movie-rating tuples to form the training and test sets. Besides, we try to test the sensitivity of different scales of the training set and the test set on recommendation results. We adopt a variable named *train/test ratio* denoted by  $\chi$ , as introduced in [17], to denote the percentage of data used as the training and test sets. A value of  $\chi = 0.9$  means that 90 percent of data are used as the training set and the other 10 percent of data are used as the test set. The smaller the value of  $\chi$  is, the more sparse the training data are. We conduct a fivefold cross validation and take the average MAE.

In the second experiment, we compare TyCo with some state-of-the-art methods to further demonstrate the

TABLE 2  
Sensitivity of  $n$  on Coverage with Different Train/Test Ratios

	$\gamma$	$\chi=0.1$	$\chi=0.3$	$\chi=0.5$	$\chi=0.7$	$\chi=0.9$
$n=5$	0.8	0.9401	0.965	0.9711	0.9699	0.9773
$n=10$	0.7	0.9637	0.9794	0.9795	0.9792	0.9847
$n=15$	0.6	0.9764	0.9874	0.9891	0.9895	0.9896
$n=20$	0.6	0.9774	0.9877	0.9889	0.9862	0.986
$n=25$	0.5	0.9798	0.9909	0.9918	0.9923	0.9934
$n=30$	0.5	0.9739	0.9882	0.9902	0.986	0.9882
AVG		0.9685	0.9831	0.9851	0.9838	0.9865

advantages of TyCo on improving the recommendation quality. The compared state-of-the-art methods include a cluster-based Pearson Correlation Coefficient method (SCBPCC) [40], weighted low-rank approximation (WLR) [41], a transfer learning-based collaborative filtering (CBT) [42], and SVD++ [43]. In this experiment, similar to previous works [40], [42], we extract a subset of 500 users from the data set, and select the first 100, 200, and 300 users to form the training sets, named as ML100, ML200, and ML300, respectively (the remaining last 200 users are used as test users). As to the active (test) users, we vary the number of rated items provided by the test users in training from 5, 10, to 20, and give the name Given5, Given10, and Given20, respectively. To evaluate TyCo furthermore, we also compare TyCo with SVD++ using Netflix data.

The experiment is conducted using a PC with a Pentium 4 3.2-GHz CPU, 2-GB memory, Windows XP Professional operating system, and Java J2SE platform.

## 4.2 Experimental Results

### 4.2.1 Impact of Number of User Groups

In TyCo, the number of user groups is the same as the number of item groups. To test the sensitivity of different number of user groups (i.e.,  $n$ ), we run experiments for various  $n$  from 5 to 30, and the best results (with the most suitable parameter  $\gamma$ ) on MAE and coverage are shown in Tables 1 and 2, respectively. According to Tables 1 and 2, we find that the number of user groups has little effect on the recommendation results. Although the MAE values for some  $n$  (e.g.,  $n = 25$ ) are a little bigger than those for other  $n$  (e.g.,  $n = 20$ ), their coverage values are still bigger. Thus, we regard recommendation quality under different  $n$  as stable by setting an appropriate  $\gamma$ .

### 4.2.2 Comparison on Recommendation Quality

As mentioned in Section 4.1.3, we adopt three existing recommendation methods as baselines, which are user-based collaborative filtering with Pearson Correlation Coefficient, item-based collaborative filtering with Pearson Correlation Coefficient, and the EMDP method [13], to compare with the novel typicality-based CF method. Figs. 4 and 5 show the comparison results of TyCo (we set the number of user groups  $n = 20$  and  $\gamma = 0.6$ ) with the baseline methods with different train/test ratios on MAE and coverage, respectively.

From Fig. 4, we can find that TyCo outperforms all other three baseline methods in all train/test ratios on MAE. For example, for train/test ratio  $\chi = 0.9$ , the MAE of TyCo is 0.735 while that of EMDP (the second best result when



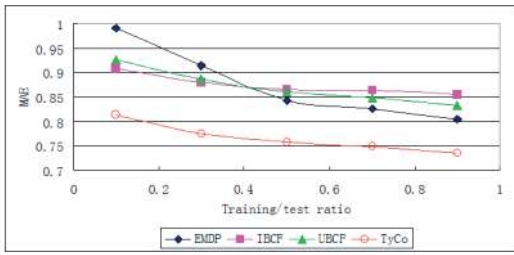


Fig. 4. Comparison on MAE using ML data set.

$\chi = 0.9$ ) is 0.804; for  $\chi = 0.3$ , the MAE of TyCo is 0.7757 while that of IBCF (the second best results when  $\chi = 0.3$ ) is 0.8803 only. It shows clearly that typicality-based CF method has higher recommendation accuracy than all compared methods.

As mentioned in Section 4.1.2, coverage measures the percentage of items for which a recommender system is capable of making predictions. According to Fig. 5, EMDP can obtain the highest coverage with all train/test ratios among all the methods. For UBCF and IBCF, they only achieve a coverage around 0.4 with  $\chi = 0.1$  and around 0.8 with  $\chi = 0.3$ . For TyCo, it can obtain stable coverage around 0.98 with all train/test ratios. TyCo, thus, outperforms UBCF and IBCF, and it is comparable to EMDP in obtaining the high coverage. In other words, TyCo and EMDP can predict more ratings for users on unrated items than UBCF and IBCF.

#### 4.2.3 Performance on Sparse Data

Another interesting finding is that using TyCo can obtain good MAE and coverage values even with low train/test ratio  $\chi$ . For instance, according to Figs. 4 and 5, the MAE value is 0.8125 and coverage value is 0.9685 using TyCo, even with a very low train/test ratio  $\chi = 0.1$ . Such results are comparable to those results obtained by EMDP method with a high train/test ratio  $\chi = 0.7$ . For the baseline methods in our evaluation, the best MAE is 0.9073 obtained by IBCF with  $\chi = 0.1$ , while the best coverage is 1 obtained by EMDP. From Fig. 4, it is clear that TyCo can produce more accurate recommendations than the compared methods with sparse data sets. According to Fig. 5, TyCo and EMDP can predict much more ratings for users on unrated items than other compared methods.

Table 3 shows the improvement of TyCo over the compared methods with sparse training data on MAE and coverage, respectively. TyCo outperforms the compared methods by at least 9.89 percent on MAE (with  $\chi = 0.1$  for IBCF) and as much as 17.56 percent (with  $\chi = 0.1$  for

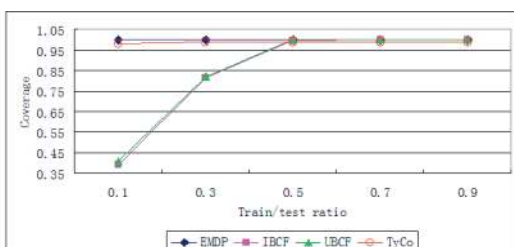


Fig. 5. Comparison on Coverage using ML data set.

TABLE 3  
Improvement of TyCo for Other Methods with Sparse Training Data on MAE and Coverage

	method	$\chi = 0.1$	$\chi = 0.2$	$\chi = 0.3$
MAE	IBCF	9.89%	12.65%	11.69%
	UBCF	11.77%	13.56%	11.55%
	EMDP	17.56%	16.17%	14.92%
	Random Guess	94.72%	97.93%	101.01%
Coverage	IBCF	59.78%	44.98%	17.11%
	UBCF	58.78%	44.29%	16.75%
	EMDP	-2.31%	-1.444%	-1.24%
	Random Guess	-2.31%	-1.444%	-1.24%

EMDP). Besides, TyCo obtains a coverage value close to 1 for all train/test ratios. It outperforms IBCF and UBCF on coverage with improvement of up to 59.78 percent and at least 16.75 percent. Yet TyCo is comparable to EMDP on coverage: the coverage value of EMDP is 1 and that of TyCo is around 0.987 for all  $\chi$ .

For the cases with small train/test ratios, it is difficult for traditional CF methods to find out enough qualified “neighbors” of a user or an item. The reason is that there may be very few corated items of two users (or few common users for two items) in sparse data sets. Thus, the recommendation accuracy is low for traditional collaborative filtering when training data are sparse. However, for our typicality-based CF, the user-typicality matrix built is a dense matrix even when the training data are sparse because all users have different typicality degrees in different user groups. The “neighbors” are selected by measuring users’ similarity based on their typicality degrees in each user group and the number of user groups is usually not large (for example, the number of user groups is 20 in our experiment). In other words, the 1,682 (or 943) dimensions for the traditional user-based CF (item-based CF) to be compared are reduced to only 20 dimensions. Thus, we can obtain a small MAE and large coverage values even with small train/test ratios.

#### 4.2.4 Comparison on Prediction Errors (PEs)

Predictions with big error may reduce the users’ trust and loyalty of recommender systems. Therefore, in our experiments, we also evaluate the prediction errors of the TyCo recommendation method. Fig. 6 compares TyCo with distance-based similarity (we set  $n = 20$  and  $\gamma = 0.6$ ) and other current methods on prediction error, with the train/test ratio  $\chi = 0.9$ . For other train/test ratios (i.e.,  $\chi < 0.9$ ), the comparison results are similar and are not shown here.

For the MovieLens data set, the user ratings are from 1 to 5. In Fig. 6,  $PE = 0$  means that there is no error and the

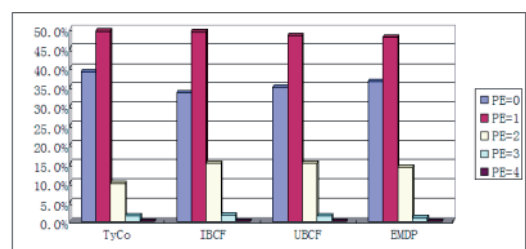


Fig. 6. Comparison on prediction errors.

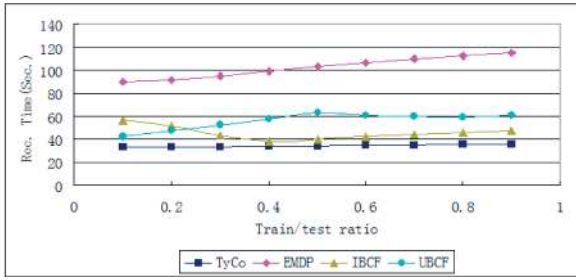


Fig. 7. Comparison of different methods on time to generate recommendations.

prediction rating is the same as the ratings given by users;  $PE = 1$  means that the difference between prediction rating and user's rating is 1, so on and so forth. The biggest prediction error is 4.

From Fig. 6, it can be seen that TyCo has the most number of small-error prediction ratings among all the methods (i.e., 38.96 percent of the predictions with  $PE = 0$  and 49.46 percent with  $PE = 1$ ). Besides, there are 9.93 percent of the prediction ratings with  $PE = 2$ , 1.58 percent with  $PE = 3$  and 0.07 percent with  $PE = 4$  using the TyCo method. For the cases of  $PE \geq 2$ , there are 11.58 percent for predictions by typicality-based method TyCo, 16 percent for EMDP, and about 17 percent for IBCF and UBCF.

It is clear that TyCo can obtain good results with less number of big-error predictions than compared methods.

#### 4.2.5 Comparison on Efficiency

We also compare the time cost of TyCo with that of other methods. Fig. 7 shows the comparison of the time used to generate recommendations by different methods with different train/test ratios.

From Fig. 7, we can see that TyCo is faster than the other methods. UBCF and IBCF need to find out a set of "nearest neighbors" of items or users based on user-item-rating matrix. As the Movielens data set has 943 users and 1,682 movies, we need to build a  $943 \times 1,682$  matrix. It means that we need to compare values in 1,682 dimensions for UBCF, and in 943 dimensions for IBCF. For TyCo, we only need to compare values in  $n$  dimensions ( $n = 20$  in our experiments). EMDP is the slowest because it predicts missing data first and then combines the ratings obtained by a user-based method and an item-based method.

With the increase of train/test ratio, the time cost of TyCo increases slightly. The reason is that most time cost of TyCo is attributed to the computation of user typicality degrees in user groups. Once the user-typicality matrix is built, TyCo can predict the unknown ratings quickly.

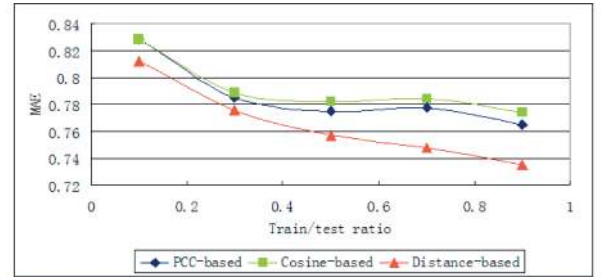


Fig. 8. Impact of different similarity functions on MAE.

#### 4.2.6 Impact of Similarity Functions and $\gamma$

In Section 3.5, we presented three similarity functions that are distance-based similarity, cosine-based similarity function, and Pearson correlation similarity function. Besides, we set a similarity threshold  $\gamma$  in our "neighbors" selection. We conduct experiments here to evaluate the effect of different similarity functions and that of different thresholds on recommendation quality.

Fig. 8 shows the best MAE results (with the best setting of  $\gamma$ ) of different similarity functions with different train/test ratios. According to Fig. 8, the distance-based similarity can achieve smallest MAE values in all train/test ratio. Thus, we consider that the distance-based similarity is more appropriate for typicality-based CF and adopt it in the comparison experiments.

To evaluate the effect of similarity threshold  $\gamma$ , we conduct experiments by setting  $\gamma$  from 0.1 to 0.9 on different  $n$ . Fig. 9 shows the comparison of different  $\gamma$  for  $n = 10$ ,  $n = 20$ , and  $n = 30$ . For each  $n$ , there is a best similarity threshold  $\gamma$ . With the increase of  $n$ , the best  $\gamma$  decreases for obtaining the best result. For example, the best value for  $\gamma$  is 0.7 when  $n = 10$  and it decreases to 0.5 when  $n = 30$ . We think that the reason is as follows: for smaller  $n$  (e.g.,  $n = 5$ ), users belong to user groups with higher degrees because the prototype of each cluster is not so specific; while for larger  $n$  (e.g.,  $n = 35$ ), users belong to user groups with a lower degrees because the prototype of each cluster is more specific.

When we increase  $\gamma$  to a value that is much greater than the best value, MAE increases quickly. The reason is that the requirement to be "neighbors" of a user is more strict as  $\gamma$  increases and there is not enough qualified "neighbors" for each user with large  $\gamma$ . A small set of "neighbors" is not good enough to assist predicting unknown ratings for users and may cause big-error predictions.

#### 4.2.7 Comparison with State-of-the-Art Methods

We also compare our method with some state-of-the-art methods on MAE. Table 4 shows the comparison of TyCo

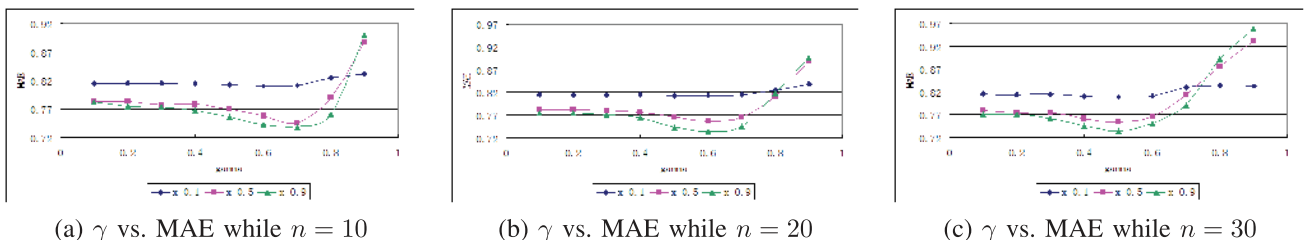


Fig. 9. Impact of different  $\gamma$  on MAE.

TABLE 4  
Comparison with State-of-the-Art Methods on MAE

Training Set	Methods	Given5	Given10	Given 15
ML100	SCBPCC	0.874	0.845	0.839
	WLR	0.915	0.875	0.890
	CBT	0.840	0.802	0.786
	SVD++	0.925	0.911	0.916
	<b>TyCo</b>	<b>0.830</b>	<b>0.799</b>	<b>0.777</b>
ML200	SCBPCC	0.871	0.833	0.828
	WLR	0.941	0.903	0.883
	CBT	0.839	0.800	0.784
	SVD++	0.881	0.815	0.812
	<b>TyCo</b>	<b>0.830</b>	<b>0.775</b>	<b>0.775</b>
ML300	SCBPCC	0.870	0.834	0.819
	WLR	1.018	0.962	0.938
	CBT	0.840	0.801	0.785
	SVD++	0.885	0.815	0.802
	<b>TyCo</b>	<b>0.814</b>	<b>0.762</b>	<b>0.760</b>

with state-of-the-art methods.<sup>6</sup> According to Table 4, TyCo outperforms the other methods in all configurations. Even with sparse data (e.g., ML100 and Given15), TyCo can obtain MAE values that are lower than those by the other methods with less sparse data (e.g., ML300 and Given15). CBT method is the best baseline method among all four baseline methods, which is only a little bit worse than TyCo on the Movielens data set.

To evaluate TyCo further, we also compare TyCo with SVD++ because SVD++ method has a good performance on Netflix data set [21]. We adopt the whole Netflix data set, which is much larger than Movielens, and the experiment setting on Netflix data set are the same as that for the Movielens data set. The comparison results are shown in Fig. 10. The  $x$ -axis shows the train/test ratios and the  $y$ -axis shows the MAE (RMSE) values. From Fig. 10a, we can see that TyCo outperforms SVD++ on Movielens data with all train/test ratios. For small train/test ratios, for example, from 0.1 to 0.3, TyCo has lower MAE values and obviously outperforms SVD++. For train/test ratios from 0.4 to 0.8, the MAE values of TyCo are still lower than SVD++, but they are very close and the improvement is not so significant. According to Fig. 10b, TyCo outperforms SVD++ on MAE using Netflix data set as a whole. For small train/test ratios (from 0.1 to 0.4), the MAE values of TyCo and SVD++ are very close. However, when train/test ratio is greater than or equal to 0.5, as the train/test ratio increases, the improvement of TyCo over SVD++ becomes more and more significant. Furthermore, we can get similar directionality using RMSE by comparing TyCo and SVD++. According to Fig. 10c, the RMSE values of TyCo and SVD++ are very close from  $\chi = 0.1$  to  $\chi = 0.6$ . As the train/test ratio increases ( $\chi \geq 0.7$ ), it is clear that TyCo outperforms SVD++.<sup>7</sup> By the comparison of TyCo and SVD++ on both Movielens and Netflix data set, both TyCo and SVD++ have good performance on recommendation, and TyCo outperforms SVD++ to some extent.

6. The results of other methods have been reported in [43].

7. We obtain a higher value of RMSE using SVD++ than that claimed in [21] because the train/test ratios are different. According to Netflix Prize specification, the train/test ratio in [21] is fixed as 0.9859 while we varied it from 0.1 to 0.9 to show the performance of TyCo in different sparsity.

## 5 DISCUSSIONS

In this section, we analyze the intuition of our method and discuss the differences among TyCo and current methods.

### 5.1 Differences between TyCo and Cluster-Based CF Methods

There are several cluster-based CF methods (such as [44]). The first difference between TyCo and cluster-based CF methods is the way of discovering neighbors of users. Cluster-based methods cluster users into several clusters and such clustering is based on the ratings of users, instead of user interests. In these methods, all users are clustered into several clusters and each user belongs to exactly one cluster only, then the users similar to a specific user  $a$  are those in the same cluster with  $a$ .

There are some cluster-based smoothing CF methods. For example, SCBPCC [40] uses cluster to refine the prediction of users' ratings on items. It predicts and refills the sparse matrix based on the ratings on all items by all users in the same cluster, thereby obtaining a nonsparse matrix. Top- $k$  similar users are selected based on Pearson correlation for conducting CF prediction. Fig. 11 shows the mechanism of discovering similar users in SCBPCC method.

In our method, similar users are not discovered by comparing their item ratings. Fig. 12 shows the mechanism of discovering similar users in TyCo. Different from cluster-based CF methods (e.g., coclustering and cluster-based smoothing CF methods), we conduct fuzzy clustering on users and form several fuzzy clusters instead of crisp clusters, then construct a user-typicality matrix. Similar users to a specific user  $a$  are discovered by similarity match based on the user typicality in all groups in the user-typicality matrix.

Another difference between TyCo and existing cluster-based CF methods are in user representation. In existing cluster-based CF methods, a user is represented by a vector of ratings by the user on items, and such a vector is often sparse because there are many unrated items. Obviously, clustering users based on their ratings on all items is not reasonable enough and will suffer the rating sparsity problem. In TyCo, we consider a user's typicality in all user groups. In other words, a user is represented at a higher (categorical) level as opposed to item level, which is more informative and closer to human's point of view and thinking. Representing a user by a user group typicality vector instead of item vector can be regarded as a dimension reduction on user representation.

### 5.2 Differences among TyCo and Other Methods

The difference between TyCo and previous user-based collaborative filtering is that TyCo finds a user's neighbors based on their typicality degrees in all user group, instead of based on users' ratings on items in previous methods. For item-clustering-based CF, they are based on clustering items, while TyCo is based on users' typicality. That is, item-clustering-based CF is item-based recommendation while TyCo is user-based recommendation. Current hybrid methods are based on combining both collaborative filtering and content-based methods, for example, using some aggregation to aggregate the recommendation results of CF method and content-based method, while TyCo is a neighbor-based recommendation. Latent factor methods (e.g., pLSA and SVD++) use latent factors or concepts to

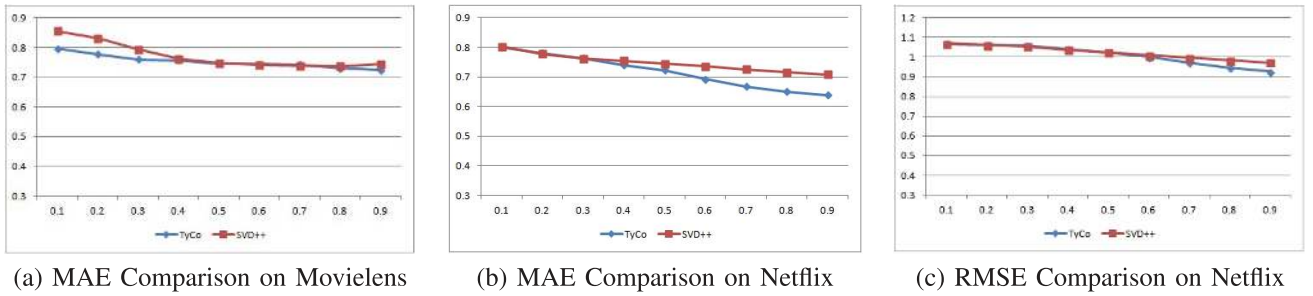


Fig. 10. Comparison of TyCo and SVD++.

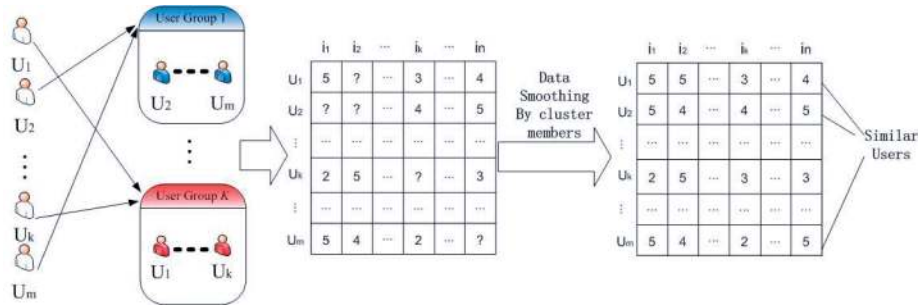


Fig. 11. Mechanism of discovering similar users in SCBPCC.

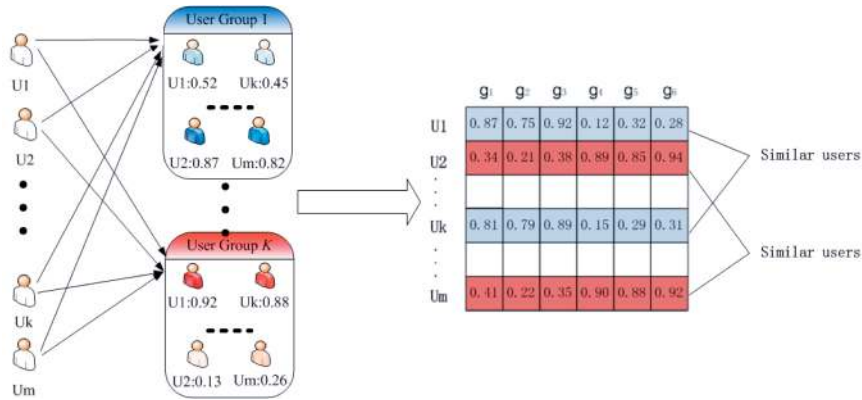


Fig. 12. Mechanism of discovering similar users in TyCo.

find neighbors instead of pure rating. The idea behind such models is to characterize both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. A key to these models is abstracting users by an intermediate layer of user factors. This intermediate layer separates the computed predictions from past user actions and complicates explanations [45]. Different from these latent factor models, TyCo discovers users' neighbors based on typicality and predicts ratings based on neighbors' actual ratings.

## 6 CONCLUSIONS AND FUTURE WORKS

In this paper, we investigate the collaborative filtering recommendation from a new perspective and present a novel typicality-based collaborative filtering recommendation method named TyCo. In TyCo, a user is represented by a user typicality vector that can indicate the user's preference on each kind of items. A distinct feature of TyCo is that it selects "neighbors" of users by measuring users' similarity based on their typicality degrees instead of corated items by users. Such a feature can overcome several

limitations of traditional collaborative filtering methods. It is the first work that applies typicality for collaborative filtering. We conduct experiments to evaluate TyCo and demonstrate the advantages of TyCo. In TyCo, there are some preprocessing procedures, such as constructing user prototype by clustering and measuring user typicality in user groups. The cost of these preprocessing procedures depends on the particular clustering method used. In real-life applications, these procedures can be processed offline. While users' prototypes are constructed, the remained recommendation process which is based on user typicality will be efficient. For large scale applications, we can also first conduct the above preprocessing offline, and then adopt some parallel computing methods (e.g., MapReduce) to speed up the computing.

There are several possible future extensions to our work. In TyCo, we do not specify how to cluster resources so as to find out item groups and the corresponding user groups. One possible future work is to try different clustering methods and see how the recommendation results are affected. How to using parallel computing methods (e.g., MapReduce) to handle the large scale applications is also one of the possible future works.

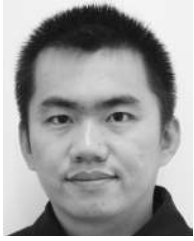
## ACKNOWLEDGMENTS

The authors are grateful to the anonymous reviewers and Mr. Hao Han. The research was supported by National Natural Science Foundation of China (Grant No. 61300137), Guangdong NSFC (S2011040002222 and S2013010013836), the Fundamental Research Funds for the Central Universities, SCUT (2014ZZ0035).

## REFERENCES

- [1] Z. Huang, H. Chen, and D. Zeng, "Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering," *ACM Trans. Information Systems*, vol. 22, no. 1, pp. 116-142, 2004.
- [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. Knowledge and Data Eng.*, vol. 17, no. 6, pp. 734-749, June 2005.
- [3] K.M. Galotti, *Cognitive Psychology In and Out of the Laboratory*, third ed. Wadsworth, 2004.
- [4] G.L. Murphy, *The Big Book of Concepts*. MIT Press, 2002.
- [5] L.W. Barsalou, *Cognitive Psychology: An Overview for Cognitive Scientists*. Lawrence Erlbaum Assoc., 1992.
- [6] S. Schiffer and S. Steele, *Cognition and Representation*. Westview Press, 1988.
- [7] D.L. Medin and E.E. Smith, "Concepts and Concept Formation," *Ann. Rev. of Psychology*, vol. 35, pp. 113-138, 1984.
- [8] W. Vanpaemel, G. Storms, and B. Ons, "A Varying Abstraction Model for Categorization," *Proc. Cognitive Science Conf. (CogSci '05)*, pp. 2277-2282, 2005.
- [9] L.W. Barsalou, "Ideals, Central Tendency, and Frequency of Instantiation as Determinants of Graded Structure in Categories," *J. Experimental Psychology: Learning, Memory, and Cognition*, vol. 11, no. 4, pp. 629-654, Oct. 1985.
- [10] M. Rifqi, "Constructing Prototypes from Large Databases," *Proc. Int'l Conf. Information Processing and Management of Uncertainty (IPMU '96)*, pp. 301-306, 1996.
- [11] M.-J. Lesot, L. Mouillet, and B. Bouchon-Meunier, "Fuzzy Prototypes Based on Typicality Degrees," *Proc. Int'l Conf. Eighth Fuzzy Days '04*, 2005.
- [12] C.M.A. Yeung and H.F. Leung, "Ontology with Likelihood and Typicality of Objects in Concepts," *Proc. 25th Int'l Conf. Conceptual Modeling*, pp. 98-111, 2006.
- [13] H. Ma, I. King, and M.R. Lyu, "Effective Missing Data Prediction for Collaborative Filtering," *Proc. 30th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07)*, pp. 39-46, 2007.
- [14] J. Wang, A.P. de Vries, and M.J.T. Reinders, "Unifying User-Based and Item-Based Collaborative Filtering Approaches by Similarity Fusion," *Proc. 29th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '06)*, pp. 501-508, 2006.
- [15] R.J. Mooney and L. Roy, "Content-Based Book Recommending Using Learning for Text Categorization," *Proc. Fifth ACM Conf. Digital Libraries (DL '00)*, pp. 195-204, 2000.
- [16] M.J. Pazzani and D. Billsus, "Content-Based Recommendation Systems," *The Adaptive Web: Methods and Strategies of Web Personalization*, pp. 325-341, Springer-Verlag, 2007.
- [17] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-Based Collaborative Filtering Recommendation Algorithms," *Proc. 10th Int'l Conf. World Wide Web (WWW '01)*, pp. 285-295, 2001.
- [18] M. Deshpande and G. Karypis, "Item-Based Top-N Recommendation Algorithms," *ACM Trans. Information Systems*, vol. 22, no. 1, pp. 143-177, 2004.
- [19] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets," *Proc. IEEE Eighth Int'l Conf. Data Mining (ICDM '08)*, pp. 263-272, 2008.
- [20] A. Umyarov and A. Tuzhilin, "Improving Collaborative Filtering Recommendations Using External Data," *Proc. IEEE Eighth Int'l Conf. Data Mining (ICDM '08)*, pp. 618-627, 2008.
- [21] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30-37, Aug. 2008.
- [22] L. Zhang, D. Agarwal, and B.-C. Chen, "Generalizing Matrix Factorization through Flexible Regression Priors," *Proc. Fifth ACM Conf. Recommender Systems (RecSys '11)*, pp. 13-20, 2011.
- [23] L. Backstrom and J. Leskovec, "Supervised Random Walks: Predicting and Recommending Links in Social Networks," *Proc. Fourth ACM Int'l Conf. Web Search and Data Mining (WSDM '11)*, pp. 635-644, 2011.
- [24] S. Lee, S.-i. Song, M. Kahng, D. Lee, and S.-G. Lee, "Random Walk Based Entity Ranking on Graph for Multidimensional Recommendation," *Proc. Fifth ACM Conf. Recommender Systems (RecSys '11)*, pp. 93-100, 2011.
- [25] K. Zhou, S.-H. Yang, and H. Zha, "Functional Matrix Factorizations for Cold-Start Recommendation," *Proc. 34th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '11)*, pp. 315-324, 2011.
- [26] H. Ma, C. Liu, I. King, and M.R. Lyu, "Probabilistic Factor Models for Web Site Recommendation," *Proc. 34th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '11)*, pp. 265-274, 2011.
- [27] K.W.-T. Leung, D.L. Lee, and W.-C. Lee, "CLR: A Collaborative Location Recommendation Framework Based on Co-Clustering," *Proc. 34th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '11)*, pp. 305-314, 2011.
- [28] P. Melville, R.J. Mooney, and R. Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations," *Proc. 18th Nat'l Conf. Artificial Intelligence*, pp. 187-192, 2002.
- [29] I.M. Soboroff and C.K. Nicholas, "Combining Content and Collaboration in Text Filtering," *Proc. IJCAI '99 Workshop Machine Learning for Information Filtering*, pp. 86-91, 1999.
- [30] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and Mining of Academic Social Networks," *Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '08)*, pp. 990-998, 2008.
- [31] T. Hofmann, "Probabilistic Latent Semantic Analysis," *Proc. Conf. Uncertainty in Artificial Intelligence (UAI '99)*, pp. 289-296, 1999.
- [32] R. Xu and D. Wunsch, "Survey of Clustering Algorithms," *IEEE Trans. Neural Network*, vol. 16, no. 3, pp. 645-678, May 2005.
- [33] C. Li and G. Biswas, "Unsupervised Learning with Mixed Numeric and Nominal Data," *IEEE Trans. Knowledge and Data Eng.*, vol. 14, no. 4, pp. 673-690, July/Aug. 2002.
- [34] Y. Cai, C. man Au Yeung, and H. fung Leung, *Fuzzy Computational Ontologies in Context*. Higher Education Press and Springer, 2012.
- [35] D.L. Medin and E. Rosch, "Context Theory of Classification Learning," *Psychological Rev.*, vol. 85, pp. 207-238, 1978.
- [36] Y. Cai, H.F. Leung, and A.W.C. Fu, "Multi-Prototype Concept and Object Typicality in Ontology," *Proc. 21st Int'l Florida Artificial Intelligence Research Soc. Conf.*, pp. 470-475, 2008.
- [37] R.R. Yager, "On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-18, no. 1, pp. 183-190, Jan./Feb. 1988.
- [38] S. Santini and R. Jain, "Similarity Matching," *Proc. Asian Conf. Computer Vision (ACCV)*, pp. 571-580, 1995.
- [39] M.G. Vozalis and K.G. Margaritis, "Unison-CF: A Multiple-Component, Adaptive Collaborative Filtering System," *Proc. Third Int'l Conf. Adaptive Hypermedia and Adaptive Web-Based Systems (AH)*, pp. 255-264, 2004.
- [40] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, "Scalable Collaborative Filtering Using Cluster-Based Smoothing," *Proc. 28th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '05)*, 2005.
- [41] N.S. Nati and T. Jaakkola, "Weighted Low-Rank Approximations," *Proc. 20th Int'l Conf. Machine Learning*, pp. 720-727, 2003.
- [42] B. Li, Q. Yang, and X. Xue, "Can Movies and Books Collaborate?: Cross-Domain Collaborative Filtering for Sparsity Reduction," *Proc. 21st Int'l Joint Conf. Artificial Intelligence (IJCAI)*, pp. 2052-2057, 2009.
- [43] Y. Koren, "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model," *Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '08)*, pp. 426-434, 2008.
- [44] A.N.K. Honda and H. Ichihashi, "Collaborative Filtering by User-Item Clustering Based on Structural Balancing Approach," *Int'l J. Computer Science and Network Security*, vol. 8, no. 12, pp. 190-195, 2008.

- [45] Y. Koren, "Factor in the Neighbors: Scalable and Accurate Collaborative Filtering," *ACM Trans. Knowledge Discovery from Data*, vol. 4, pp. 1:1-1:24, Jan. 2010.



**Yi Cai** received the PhD degree in computer science from The Chinese University of Hong Kong. He is currently an associate professor in the School of Software Engineering, South China University of Technology, Guangzhou, China. His research interests include recommendation system, personalized search, Semantic Web, and data mining.



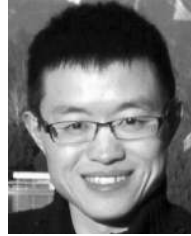
**Ho-Fung Leung** is currently a professor and the chairman in the Department of Computer Science and Engineering, The Chinese University of Hong Kong, China. He has been active in research on intelligent agents, multiagent systems, game theory, and Semantic Web.



**Qing Li** is a professor at the City University of Hong Kong. His research interests include object modeling, multimedia databases, social media, and recommender systems. He is the chairperson of the Hong Kong Web Society, and is a steering committee member of DASFAA, ICWL, and WISE Society. He is a fellow of the IET, a senior member of the IEEE, a member of the ACM SIGMOD and the IEEE Technical Committee on Data Engineering.



**Juanzi Li** received the PhD degree from Tsinghua University in 2000. She is a full professor at Tsinghua University, China. Her primary research interest is to study the semantic technologies by combining natural language processing, Semantic Web, and data mining. She has published about 90 papers in many international journals and conferences such as WWW, the *IEEE Transactions on Knowledge and Data Engineering*, SIGIR, SIGMOD, SIGKDD, ISWC, JoWS, and so on.



*Journal of Web Semantics.*

**Jie Tang** is an associate professor of Tsinghua University, China. His main research interests include data mining algorithms and social network theories. He has published more than 100 research papers in major international journals and conferences including: KDD, IJCAI, WWW, SIGMOD, ACL, the *Machine Learning Journal*, the *ACM Transactions on Knowledge Discovery from Data*, the *Transactions on Knowledge and Data Engineering*, and the



**Huaqing Min** is a professor and the dean of School of Software Engineering, South China University of Technology, China. His research interests include artificial intelligence, machine learning, database, data mining, and robotics.

▷ **For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).**