Physica A 396 (2014) 66–76

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

Collaborative filtering recommendation algorithm based on user preference derived from item domain features

Jing Zhang, Qinke Peng *, Shiquan Sun, Che Liu

Systems Engineering Institute, School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi Province, China

HIGHLIGHTS

- We use item domain features to construct user preference models.
- We combine user preference models with CF for personalized recommendation.
- We use the multi-attribute decision making method to calculate the user preference.
- Our method integrates domain characteristics into a personalized recommendation.
- Our method aids to detecting the implicit relationships (missed by CF) among users.

ARTICLE INFO

Article history:
Received 14 December 2012
Received in revised form 1 October 2013
Available online 21 November 2013

Keywords:
Personalized recommendation
Collaborative filtering
User preference model
Item domain feature

ABSTRACT

Personalized recommendation is an effective method for fighting "information overload". However, its performance is often limited by several factors, such as sparsity and cold-start. Some researchers utilize user-created tags of social tagging system to depict user preferences for personalized recommendation, but it is difficult to identify users with similar interests due to the differences between users' descriptive habits and the diversity of language expression. In order to find a better way to depict user preferences to make it more suitable for personalized recommendation, we introduce a framework that utilizes item domain features to construct user preference models and combines these models with collaborative filtering (CF). The framework not only integrates domain characteristics into a personalized recommendation, but also aids to detecting the implicit relationships among users, which are missed by the conventional CF method. The experimental results show our method achieves the better result, and prove the user preference model is more effective for recommendation.

1. Introduction

The rapid development of information technology and the current growth and popularity of the Internet have facilitated an explosion of information that has exacerbated the information overload problem [1]. As one of the most useful methods, personalized recommendation, which was first proposed in the 1990s [2,3], adopts knowledge discovery techniques such as data mining and machine learning to discover user interests according to user behavior and then to make recommendations [4–6]. A typical application of personalized recommendation is in electronic commerce, such as book recommendations in Amazon.com [7], movie recommendations in Netflix.com [8], video recommendations in TiVo.com [9], and so on. An efficient recommendation system not only provides appropriate recommendations for users, but also helps the service provider gain substantial profits.

Mainstream recommendation algorithms can be divided into four categories [10]: content-based (CB), collaborative filtering (CF), network-based (NB), and hybrid recommendation (HR). The CB method recommends objects that are similar

* Corresponding author. Tel.: +86 29 82667964.
E-mail addresses: zjhappy@stu.xjtu.edu.cn (J. Zhang), qkpeng@mail.xjtu.edu.cn (Q. Peng).

0378-4371/$ – see front matter
http://dx.doi.org/10.1016/j.physa.2013.11.013

to those previously preferred by the target user. However, this method cannot filter audio, image, or video information [10]. CF has been the most successful recommendation system technology [11]. In CF, we make recommendation according to the assumption that users who have the similar performances would like to choose the similar items. However, the performance of CF is significantly limited by the sparsity of data [10]. NB recommendation utilizes relationships between users and items or relationships among users to construct a network, and then analyzes the network to determine recommendations for users indirectly. However, the "cold-start" problem could not be solved [10]. Finally, HR is currently the most popular approach and it combines at least two recommendation algorithms to determine a recommendation [10].

Many scholars have recently integrated various kinds of information into the recommendation system to improve performance. Such information includes tags, time, trust relationships, browse records, social networks, and so on. For instance, Zheng and Li investigated the importance and usefulness of tags and time information when predicting user preferences and consequently examined how such information could be exploited to build an effective resource–recommendation model [12]. Yin et al. considered the latent value of trust relationships to construct a trust preference network to make recommendations [13]. Kardan et al. introduced an innovative architecture for a recommendation system that took advantage of collaborative tagging and concept maps [14]. Zhang et al. proposed a recommendation approach that combined content and relation analyses in a single model to estimate the relations among users, tags, and resources for tag, item, and user recommendations [15]. Adding information to the recommendation system not only improves the performance, but also enhances the understanding of which factors influence recommendations.

The social tagging system is currently popular with scholars who utilize user-created tags to depict user preferences for personalized recommendation. Kim et al. proposed a CF method to provide enhanced recommendation quality with user-created tags, which were employed to identify and filter user preferences for certain items [16]. Shang et al. studied a personalized recommendation model that used the ternary relations among users, objects, and tags to propose a similarity based on preferences and tagging information [17]. Zhang et al. proposed a recommendation algorithm based on an integrated diffusion on user–item–tag tripartite graphs [18]. Schenkel et al. proposed an incremental threshold algorithm that considered both social ties among users and semantic relations among different tags [19]. Nakamoto et al. created a tag-based contextual CF model, where tag information was taken as user profiles [20]. Tso-Sutter et al. proposed a generic method that enabled tags to be incorporated to the standard CF by reducing ternary correlations into three binary correlations and then applying a fusion method to re-associate these correlations [21]. However, the user-created tag data is very sparse because of human descriptive habits and the diversity of language expression. For instance, a number of users prefer "happy" to express their delight, whereas others prefer "pleasure". Likewise, some users are accustomed to using "awful" to show their dissatisfaction, whereas others prefer "terrible". This characteristic hinders the identification of users who have similar interests through a social tagging system.

Additionally, different domain items often exhibit different characteristics. For instance, performance and quality are important for electronics; genres, directors, and actors compose the main information for movies; style of music and the scope of service are other primary factors that people concern. These characteristics are called domain characteristics in this paper. Given the diversity of domain characteristics, traditional personalized recommendations do not adapt well to all domains. Thus, methods to combine domain characteristics and personalized recommendation are required.

Recently, scholars focus on domain recommendation system. For example, Chen et al. presented a diabetes medication recommendation system based on domain ontology about drug attributes and patient symptoms [22]. García-Crespo et al. presented a semantic expert hotel recommendation system based on consumer experience and hotel characteristics [23]. Xin et al. presented a financial information recommendation algorithm that combined fuzzy clustering and CF [24]. Carrer-Neto et al. presented a social-based content recommender system in the movie domain, which used Semantic Web principles to help users find content that was relevant to their preferences, and likewise proposed social networks for building a novel CF [25]. Wei et al. introduced a news recommender system in which each user was considered as a node of the network. Users could post and recommend news to others, and they also could receive news from others at the same time [26]. However, these recommendation systems only adapt to one special domain very well. Thus, developing the general method that can adapt to every domain is more meaningful. Besides, although domain characteristics are always used to enhance item feature descriptions, there are few studies that have focused on expanding user preferences based on domain characteristics of items.

In this paper, in order to find a better way to depict user preferences and integrate domain characteristics into recommendation system, we propose a framework in which we use item domain features to construct user preference models at first, and then combine these models with CF for domain recommendations. Moreover, this framework could make recommendations to users who have not selected any common items with others. Finally, the framework is applied to the movie domain and exhibits good performance.

2. Methods

2.1. Comparison of the capacity in finding neighbors by using tags and ratings

Given the differences in users describing habits and the diversity of language expression, different users use different words or phrases to describe the same feelings. This characteristic hinders the identification of users who have similar

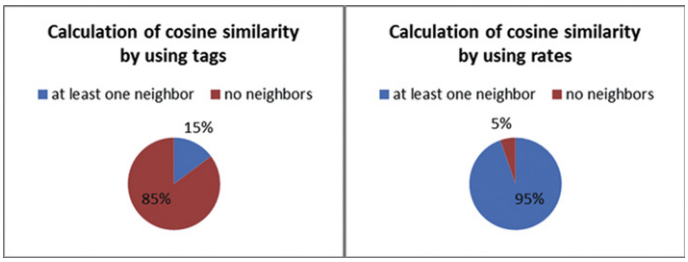

Fig. 1. Comparison of the capacity in finding users with similar interests by using tags and ratings.

interests by user-created tags. To illustrate this, we make a statistics on a benchmark dataset, MovieLens [27], whose details are described in Section 3.1, and the result is shown in Fig. 1.

Fig. 1 shows that only approximately 15% of users in MovieLens have neighbors by using user-created tags. We also make statistics on other datasets and get the similar results.

Compared with tags, the item features are given based on uniform standards and the feature space is smaller than tag space. So, we want to use item features instead of user-created tags to construct user preference model.

Additionally, item always contains features, such as genre, director, actor, and country information in the movie domain and performance and quality in the electronics product domain. These features not only help user to get a better understanding but also might reflect user preferences to some extent. Taking a simple case for example, if a user selects numerous horror films but only a few about love, he or she could be regarded as someone who prefers horror films. In fact, user preference is not as simple as the example, which is always a combination of various factors. Thus, we employ item features to construct a model that represents user preference, and integrates domain characteristics into a personalized recommendation system.

2.2. Modeling user preferences matrix based on item domain features

For all of the above reasons discussed in the last paragraph of Section 2.1, we assume that a user preference could typically be implied from his/her rated items.

We use $V = \{1, 2, \ldots, N\}$ to denote the set of items, and the set of users is $U = \{1, 2, \ldots, M\}$. For convenience, i and j are used to represent an item and a user respectively. The set of item ratings rated by user j is defined as

$$T_j = \left\{t_{ji} | j \in U, i \in V_j, V_j \subseteq V\right\}, \tag{1}$$

where t_{ji} represents the rating of item i rated by user j, V_j is the set of items rated by user j. Obviously, $V_j = \left\{i_{j1}, i_{j2} \ldots, i_{jh}\right\}$ is a subset of V.

The set of domain features attached to item i is denoted as A_i. Taking movie domain for example, the domain features of it include several categories of information, such as genres, directors, actors, countries and so on.

So the set of features that depicts the user preferences is formally defined as follows:

$$S_j = \bigcup_{i \in V_j} A_i = \left\{s_{j1}, s_{j2} \ldots, s_{jr}\right\}, \tag{2}$$

where s_{jr} is one of the features that preferred by user j, S_j is the union of features of items selected by user j. S_j contains domain characteristics.

Thus, the preferences matrix of user j is determined by T_j and S_j

$$E_j = \left(e^j_{nm}\right)_{h \times r} = \begin{array}{c} \\ i_{j1} \\ \vdots \\ i_{jh} \end{array} \begin{array}{c} \begin{array}{ccc} s_{j1} & \cdots & s_{jr} \end{array} \\ \begin{pmatrix} e^j_{11} & \ldots & e^j_{1r} \\ \vdots & \ddots & \vdots \\ e^j_{h1} & \cdots & e^j_{hr} \end{pmatrix} \end{array}, \tag{3}$$

where E_j is an $h \times r$ matrix, of which columns denote the features preferred by user j and rows denote the items rated by user j. e^j_{nm} is determined by the following function:

$$e^j_{nm} = \begin{cases} t_{jn}, & \text{if } s_{jm} \in A_n \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

2.3. Modeling user preferences vector

We use $\vec{W_j} = \left[w_{j1}, w_{j2}, \ldots, w_{jm}, \ldots, w_{jr}\right]$ to denote the preferences vector of user j. w_{jm} is the weight of s_{jm}, and satisfies $0 \leq w_{jm} \leq 1$ and $\sum_{m=1}^{r} w_{jm} = 1$. An infinite number of feasible solutions satisfy these conditions. So the key problem is to find out the optimal solution $\vec{W}^*_j = \left[\left(w_{j1}\right)^*, \left(w_{j2}\right)^*, \ldots, \left(w_{jm}\right)^*, \ldots, \left(w_{jr}\right)^*\right]$ from user preferences matrix E_j.

In Section 2.2, E_j is preferences matrix of user j, of which column $S_j = \{s_{j1}, s_{j2} \ldots s_{jr}\}$ is a finite set of attributes (item features) and row $V_j = \{i_{j1}, i_{j2}, \ldots, i_{jh}\}$ is a discrete set of alternatives (items). In multi-attribute decision making (MADM), E_j is called decision matrix. $\overrightarrow{W_j} = [w_{j1}, w_{j2}, \ldots, w_{jm}, \ldots, w_{jr}]$ is the weight vector of attributes. If $\overrightarrow{W_j}$ is given, we could apply MADM methods to rank the alternatives. TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution) is one of the MADM methods. It is based on the concept that chose alternative who is similar to the ideal solution [28]. Thus we could apply the concept of TOPSIS to build the optimization model to find out the optimal solution $\overrightarrow{W_j}^*$.

The main steps of TOPSIS are summarized as follows [29]: first normalize E_j as $B_j = \left(b^j_{nm}\right)_{h\times r}$ by the function $b^j_{nm} = \frac{e^j_{nm}}{\sqrt{\sum_{n=1}^{h}\left(e^j_{nm}\right)^2}}$; second construct the weighted normalized decision matrix as $Y_j = (y_{nm})_{h\times r} = \left[\left(w_{jm}\right)^* b^j_{nm}\right]_{h\times r}$; third determine the positive ideal solution A^+ and negative ideal solution A^-: for the positive impact attribute $A^+ = \left\{\max_{1\le n\le h}(y_{nm}) \,|m = 1, 2\cdots r\right\}, A^- = \left\{\min_{1\le n\le h}(y_{nm}) \,|m = 1, 2\cdots r\right\}$ and for the negative impact attribute $A^+ = \left\{\min_{1\le n\le h}(y_{nm}) \,|m = 1, 2\cdots r\right\}, A^- = \left\{\max_{1\le n\le h}(y_{nm}) \,|m = 1, 2\cdots r\right\}$; and then calculate the Euclid distances between the alternative and ideal solutions (positive and negative), and denote them as d^+ and d^-; finally, calculate the similarity of alternatives to the ideal solution using the function $C_n = \frac{d^-}{d^+ + d^-}$ and rank the alternatives.

Here, item feature is neither positive impact attribute nor negative impact attribute. So the ideal solution in TOPSIS cannot suit this situation. Usually, people are conditioned to use the average value of feature ratings to closely express the ideal feature rating. Thus the ideal point could be approximately represented as $Y^*_j = (y^*_{nm})_{h\times r} = \left[\left(w_{jm}\right)^* \left(b^j_m\right)^*\right]$, where $\left(b^j_m\right)^* = \frac{1}{h}\sum_{n=1}^{h} b^j_{nm}$. So we build the optimization model based on the concept of TOPSIS to minimize the sum of the squares of the distances between the alternatives and ideal point, and the parameters of each weight are regarded as the decision variables of the model. The functions were shown as follows.

$$\begin{cases} \min \sum_{n=1}^{h}\left[d^j\right]^2 = \sum_{n=1}^{h}\sum_{m=1}^{r}\left[y_{nm} - (y_m)^*\right]^2 = \sum_{n=1}^{h}\sum_{m=1}^{r}\left[b_{nm} - \left(b^j_m\right)^*\right]^2\left[\left(w_{jm}\right)^*\right]^2 \\ \sum_{m=1}^{r}\left(w_{jm}\right)^* = 1 \\ 0 \le \left(w_{jm}\right)^* \le 1, \quad m = 1, 2, \ldots, r \end{cases} \tag{5}$$

and the solution is

$$\left(w_{jm}\right)^* = \left\{\sum_{m=1}^{r}\frac{1}{\sum_{n=1}^{h}\left[b^j_{nm} - \left(b^j_m\right)^*\right]^2}\cdot\sum_{n=1}^{h}\left[b^j_{nm} - \left(b^j_m\right)^*\right]^2\right\}^{-1}. \tag{6}$$

We sort the weights in descending order and reserve the features whose weights are in the top NF in order to reduce the dimensionality and improve the speed of calculation.

2.4. CF based on the user preference model

To integrate user preference model with CF, we assume two users with similar vector would like the same items; hence, we use the following cosine function to calculate their similarity.

$$sim(j, k) = \cos(\overrightarrow{W_j}, \overrightarrow{W_k}) = \frac{\overrightarrow{W_j}\cdot\overrightarrow{W_k}}{\left\|\overrightarrow{W_j}\right\| \times \left\|\overrightarrow{W_k}\right\|}, \tag{7}$$

where $j, k \in U$. We sort the similarities in descending order and select users whose similarities are in the top PL as neighbors of the target user. The set of neighbors of target user j is marked as PL_j.

More specifically, the prediction score p_{ji} employed by user j to rate item i is computed by the following formula:

$$p_{ji} = \frac{\sum_{k\in PL_j} sim\,(j, k)\, t_{ki}}{\sum_{k\in PL_j} sim\,(j, k)}, \tag{8}$$

where $sim\,(j, k)$ is the similarity between users j and k, and t_{ki} is the rating of item i rated by neighbor k of user j. PL_j denotes the set of neighbors of the target user j. We sort p_{ji} in descending order and choose item i of which p_{ji} is in the top L to recommend to the target user. The set of items which recommend to user j is marked as L_j.

2.5. Algorithm

We propose a framework that aims to model user preferences based on item domain features, and combine them with CF for personalized recommendation. This framework consists of three algorithms: the UPM-B-IDF (modeling user preferences matrix based on item domain features), UPV (modeling user preferences vector) and CF-B-UCM (CF based on the user preference model). UPM-B-IDF algorithm aims to use item domain features to model user preferences matrix; UPV algorithm aims to derivate the user preferences vector from user preferences matrix; and CF-B-UCM algorithm aims to combine user preference models with collaborative filtering to provide personalized recommendations. These algorithms are expressed as follows:

UPM-B-IDF algorithm:

Input : $T_j = \left\{t_{ji} | j \in U, i \in V_j, V_j \subseteq V\right\}$ *and* A_i

Output : $E_j = \left(e^j_{nm}\right)_{h\times r}$

for : $n \in V_j (|V_j| = h, |\cdot|$ denotes cardinality)

$S_j = \bigcup_{n\in V_j} A_n = \left\{s_{j1}, s_{j2} \cdots s_{jr}\right\}$

for : $n \in V_j$

for : m from 1 to $|S_j|$ $(|S_j| = r)$

if $(s_{jm} \in A_n)$

$e^j_{nm} = t_{jn}$

then, output $E_j = \left(e^j_{nm}\right)_{h\times r} = \begin{matrix} \\ i_{j1} \\ \vdots \\ i_{jh} \end{matrix} \begin{matrix} \begin{matrix} s_{j1} & \cdots & s_{jr} \end{matrix} \\ \begin{pmatrix} e^j_{11} & \cdots & e^j_{1r} \\ \vdots & \ddots & \vdots \\ e^j_{h1} & \cdots & e^j_{hr} \end{pmatrix} \end{matrix}$

UPV algorithm:

Input : $E_j = \left(e^j_{nm}\right)_{h\times r}$

Output : $\vec{W_j} = \left[w_{j1}, w_{j2}, \ldots, w_{jm}, \ldots, w_{jr}\right], s_{jm} \in S_j, m = 1, 2, \ldots, |S_j|$

for : n from 1 to h

for : m from 1 to r

$$b^j_{nm} = \frac{e^j_{nm}}{\sqrt{\sum_{n=1}^{h}\left(e^j_{nm}\right)^2}}$$

for : m from 1 to r

$$\left(b^j_m\right)^* = \frac{1}{h}\sum_{n=1}^{h} b^j_{nm}$$

for : *m from 1 to r*

$$w_{jm} = \left\{\sum_{m=1}^{r} \frac{1}{\sum_{n=1}^{h}\left[b^j_{nm} - \left(b^j_m\right)^*\right]^2} \cdot \sum_{n=1}^{h}\left[b^j_{nm} - \left(b^j_m\right)^*\right]^2\right\}^{-1}$$

then, output $\vec{W_j} = \left[w_{j1}, w_{j2}, \ldots, w_{jm}, \ldots, w_{jr}\right], s_{jm} \in S_j, m = 1, 2, \ldots, |S_j|$

CF-B-UCM algorithm:

Input : $\vec{W_j} = \left[w_{j1}, w_{j2}, \ldots, w_{jm}, \ldots, w_{jr}\right], s_{jm} \in S_j, m = 1, 2, \ldots, |S_j|$

Output : $L_j = \{l_1 l_2, \ldots, l_L\}$

for : j from 1 to M(M denotes the number of users in the training set)

for : k from 1 to M

if $(k \neq j)$

$$sim(j, k) = \cos(\vec{W_j}, \vec{W_k}) = \frac{\vec{W_j} \cdot \vec{W_k}}{\left\|\vec{W_j}\right\| \times \left\|\vec{W_k}\right\|}$$

sort $(sim(j, 0), sim(j, 1), \ldots, sim(j, m-1))$ *in descending order and choose the* top *PL users, and this set is marked as* PL_j

for : $i \in \left(V - V_j\right)$

using the function $p_{ji} = \frac{\sum_{k\in PL_j} sim(j,k)t_{ki}}{\sum_{k\in PL_j} sim(j,k)}$ *to predict the rating of* i

sort $(p_{j1}, p_{j2}, \ldots, p_{j(|V-V_j|)})$ *in descending order and choose the top L items and this set is marked as* $L_j = \{l_1, l_2, \ldots, l_L\}$

The Computational Complexities of UPM-B-IDF algorithm and UPV algorithm are both $O(h \times r)$, and the CF-B-UCM algorithm is $O(M^2 \log M)$.

3. Experiments and results

3.1. Data sources

We use a benchmark dataset, MovieLens, to evaluate the effectiveness of our recommendation algorithm. MovieLens, which was released in the HetRec2011 framework [30], contains 2113 users, 10 197 movies, 20 movie genres, 4060 directors, 72 countries, 13 222 tags, and 855 598 ratings. We use 10-fold cross-validation to evaluate the performance of this algorithm, in which the dataset is randomly partitioned into ten sets. In each of ten runs, one of the sets serves as the testing set, and the remaining nine are combined into the training set. After ten runs, each set had served in testing set one time and training set nine times. We calculate the average of ten results as the final result.

3.2. Performance evaluation metrics

We employ six metrics: NMAE (normalized mean absolute error), Recall, Precision, F-measure, Inter-diversity, and Popularity [13,31,32], to investigate the performance of the proposed algorithm. The first three are accuracy metrics, whereas the fourth is the comprehensiveness of Precision and Recall. Inter-diversity (De) measures the personalization of recommendations for different users with different habits and tastes. The last metric measures the capability to recommend dark (less popular) items.

Before introducing the concrete definitions of the above metrics, we should interpret symbols L_j, Te_j and $|\cdot|$. L_j represents the recommendation list for user j. Te_j is the set of items rated by user j in the testing set. $|\cdot|$ denotes cardinality.

NMAE is the most common method for evaluating the accuracy of a recommender algorithm by comparing the numerical prediction values against user raw ratings.

$$\mathrm{NMAE} = \frac{1}{M(r_{\max} - r_{\min})} \sum_{j=1}^{M} \frac{1}{|Te_j|} \sum_{l \in Te_j} |r_{jl} - p_{jl}| \tag{9}$$

where M denotes the number of users, $r_{\max}$ and $r_{\min}$ represent, respectively, the maximum and minimum values in the rating range, r_{jl} is raw rating of item l by user j, p_{il} denotes the predicted rating of item l for user j.

Precision is defined as the ratio of the number of items appearing in both L_j and Te_j to the total number of items in L_j. Therefore, the Precision of the whole system is given by

$$\mathrm{Pr}\, e = \frac{1}{n} \sum_{j} \frac{|L_j \bigcap Te_j|}{L_j}, \tag{10}$$

where Pr e represents Precision, n is the number of users. Precision is also called Hitting rate [33]. A larger Precision corresponds to better performance.

Recall is defined as the ratio of the number of items appearing in both L_j and Te_j to the total number of items in Te_j. Therefore, the Recall of the whole system is given by

$$Re = \frac{1}{n} \sum_{j} \frac{|L_j \bigcap Te_j|}{Te_j}, \tag{11}$$

where Re represents Recall. A larger Recall corresponds to better performance.

Recall refers to the ratio of the hit set size to the test set size, whereas Precision denotes the ratio of the hit set size to the top L set size. However, these two metrics are often contradictory. For instance, increasing the number L tends to increase Recall but decreases Precision. Thus, the F-measure metric is the comprehensiveness of Precision and Recall.

$$\text{F-measure} = \frac{(\beta^2 + 1) * \mathrm{Pr}\, e * Re}{\beta * (\mathrm{Pr}\, e + Re)}, \tag{12}$$

where β is used to regulate the importance of both Precision and Recall. Usually, $\beta = 1$, and the F-measure is labeled as F_1.

Given that De basically measures the diversity of items between two recommendation lists, D_{ab} can be quantified by the Hamming distance between L_a and L_b, that is,

$$D_{ab} = 1 - \frac{|L_a \bigcap L_b|}{len}, \tag{13}$$

where L_a and L_b are the recommendation lists of user a and user b, len is the length of the recommendation list.

Therefore, the Inter-diversity of the whole system *De* is given by

$$De = \frac{2}{n(n-1)} \sum_{a \neq b} D_{ab}. \tag{14}$$

A greater *De* generally yields more personalized recommendations for different users.

The popularity of L_j is defined as the average degree of popularity of the items in L_j. Therefore, the popularity of the whole system is given by

$$\mathrm{Pop} = \frac{1}{n} \sum_{j} \frac{1}{|L_j|} \sum_{i \in L_j} k_i, \tag{15}$$

where i is an element in L_j, k_i is the degree of i, which represents the number of users who have watched i. A smaller popularity corresponds to a stronger capability to recommend dark items. Thus, a smaller value is preferred.

3.3. Experiments and results

We make comparisons between UPM-based CF and three other widely applied recommendation algorithms: tags-based CF, ratings-based CF and items-based CF.

UPM-based: In the UPM-based method, we calculate cosine similarity by user preference model to find out neighbors, and then combine with CF for personal recommendation.

Tags-based: In the tags-based method, we calculate cosine similarity by tags which users used to find out neighbors, and then combine with CF for personal recommendation.

Ratings-based: In the ratings-based method, we calculate cosine similarity by ratings which users rated to find out neighbors, and then combine with CF for personal recommendation.

Items-based: In the items-based method, we calculate cosine similarity by items which users chose to find out neighbors, and then combine with CF for personal recommendation.

Considering that the length of the recommendation list would affect the recommendation performance, we evaluate the quality of recommendation lists with different lengths ranging from 10 to 100 at intervals of 10. To ensure fairness, we optimize the parameters of the other compared methods to achieve their best performance. In UPM-based, we test all possible combinations of five categories of features in MovieLens, and find out the combination containing countries, genres, actors, and directors gets the best result. The results are shown from Figs. 2 to 7.

Fig. 2 shows the accuracy of recommender algorithms. The tags-based method gets the best NMAE; the UPM-based is the second-best and the NMAE of the ratings-based is close to the items-based. However, there are few movies that can be recommended to target users by the tags-based method, which means many of target users' recommendation lists cannot reach the desired number. It is because fully utilizing personalized characteristics makes it difficult to find neighbors of target users.

Figs. 3–5 demonstrate that the UPM-based method exhibits the best performance. Compared with the second-best approach, Precision and Recall are both improved, and F1 are improved nearly 3.5% at $L = 40$. The tags-based method is the worst one because target users have identified few neighbors by tags.

Fig. 6 shows the *De* results of the four experiments. The tags-based method exhibits the best performance, which indicates that this approach gives the most personalized recommendations for different users. This result is in accordance with

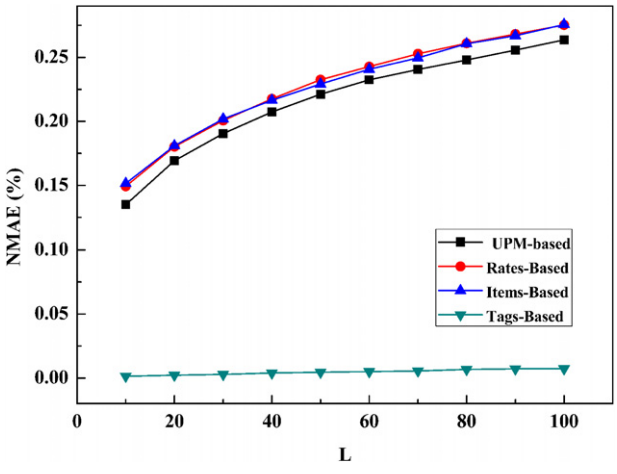

Fig. 2. NMAE of the four experiments.

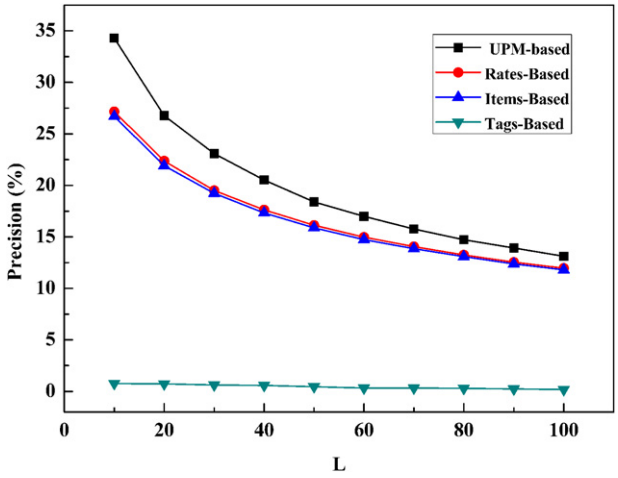

Fig. 3. Precision of the four experiments.

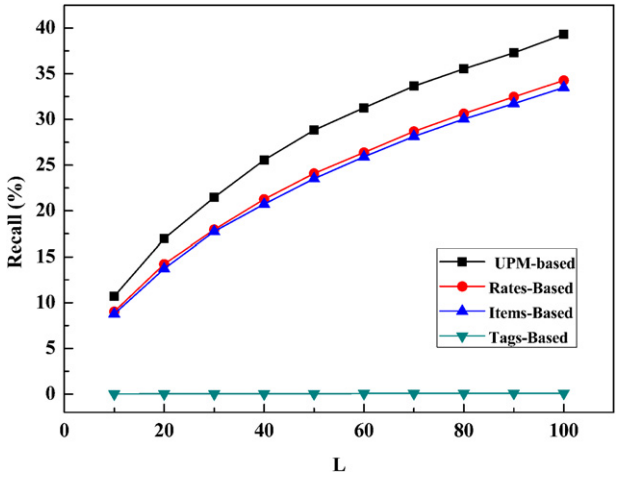

Fig. 4. Recall of the four experiments.

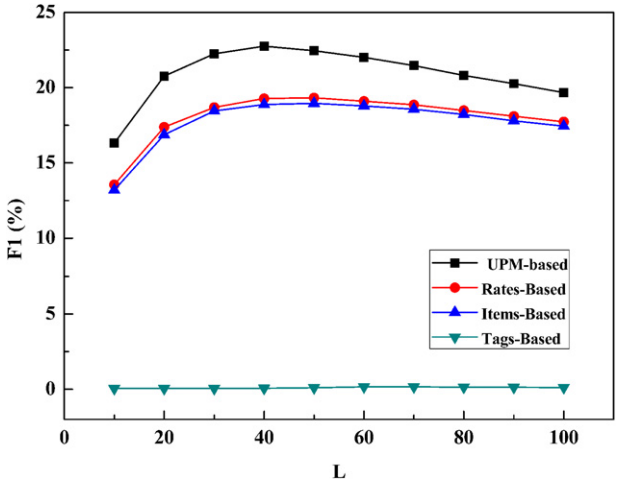

Fig. 5. F1 of the four experiments.

actual conditions. This is because the tags-based method makes the best of personalized characteristic at the cost of Precision and Recall.

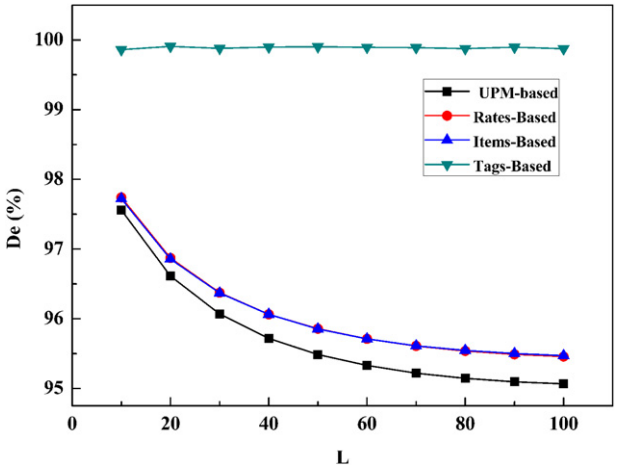

Fig. 6. *De* of the four experiments.

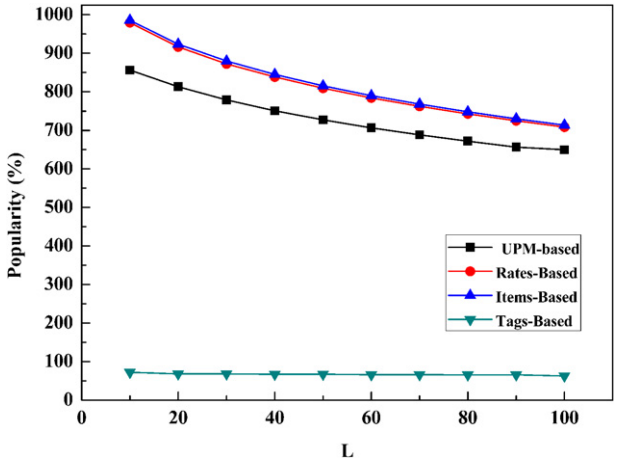

Fig. 7. Popularity of the four experiments.

Table 1
The actual length of the recommendation lists.

The length of recommendation list	Method			
	UPM-based	Ratings-based	Items-based	Tags-based
$L = 10$	21 130	21 130	21 130	704
$L = 20$	42 260	42 260	42 260	735
$L = 30$	63 390	63 390	63 390	752
$L = 40$	84 520	84 520	84 520	768
$L = 50$	105 650	105 650	105 650	780
$L = 60$	126 780	126 780	126 780	791
$L = 70$	147 910	147 902	147 879	802
$L = 80$	169 040	168 982	168 918	809
$L = 90$	190 170	190 002	189 886	813
$L = 100$	211 300	210 927	210 730	815

Fig. 7 shows the capability to recommend dark items. The tags-based method is evidently superior. Compared with the rating-based and user-based methods, the UPM-based method exhibits minimal improvement. This is because that the tag-based method makes the best of personalized characteristic to improve diversity while decrease accuracy.

However, the tags-based method has a serious defect that only a few of movies can be recommended to target users. To illustrate this problem, we count the actual length of the recommendation lists by different approaches and the results are shown in Table 1.

In the conventional CF approach, the recommendation system identifies users who share the same preferences with the target user by selection relation, consequently recommending items that the target user has not yet chosen. Thus, if a user

Table 2a
Features of movies.

ID	Movie	Features of movie
1	The usual suspects	Crime/thriller
2	Road trip	Comedy
3	The godfather	Crime/action
4	Avatar	Fantasy/adventure/action

Table 2b
Features of users' preference.

User	Features of preference model
A	Crime/thriller/comedy
B	Crime/thriller/action
C	Fantasy/adventure/action

Table 3
The number of special users.

Length	$PL = 10$	$PL = 20$	$PL = 30$	$PL = 40$	$PL = 50$	$PL = 60$	$PL = 70$	$PL = 80$	$PL = 90$	$PL = 100$
Number	165	276	357	422	501	572	645	711	790	856

has not selected any common items with others, he or she would be isolated though this user may have similar interests with others. However, the user preference model can alleviate this problem. Even if a user has not chosen or rated any items that are common with others, we would employ the user preference model to identify users who have similar interests. For example, there are four movies in Table 2a. User A has rated the first two movies, user B has rated the first and third movies, and user C has rated only the last movie. So, user C cannot be recommended by the conventional CF approach. However, user preference model is derived from movies domain features. Preference model of user C has features that are common with user B's, which can be seen in Table 2b. Thus user C can find neighbors by calculating cosine similarity of preference models to get recommendation list.

Table 3 shows the number of users who have no item in common with one another but could be neighbors based on the user preference model, with the length of the neighbor list ranging from 10 to 100 at intervals of 10.

In Table 3, the number of neighbors, who have not been identified except by the user preference models, increases along with the growth of neighbor list. This makes the performance of the UPM-based method superior to compared methods.

4. Conclusion

In this paper, we have proposed a framework, in which the user preference model has been constructed based on item domain features and then has combined with CF for personalized recommendation. Compared with the existing methods, the UPM-based method integrates domain characteristics with a personalized recommendation system. Moreover, the framework enables us to make recommendations to users who have not selected any common items with others. Our method gets better results and suits domain recommendation more.

We apply this framework to movie domain in this paper, but the framework can be applied to other domains. However, not all features are good for recommendation. Thus, designing an algorithm for automatically selecting features could be the further research.

Acknowledgments

This work was jointly supported by the National Natural Science Foundation of China (grant numbers 61173111, 60774086) and the Ph. D. Programs Foundation of Ministry of Education of China (grant number 20090201110027). Thank HetRec2011 for providing us with the data set MovieLens.

References

[1] D. Rosenberg, Early modern information overload, J. Hist. Ideas 64 (2003) 1–9.
[2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, GroupLens: an open architecture for collaborative filtering of netnews, in: Processing 1994 Computer Supported Cooperative Work Conference, 1994, pp. 175–186.
[3] W. Hill, L. Stead, M. Rosenstein, G. Furnas, Recommending and evaluating choices in a virual community of use, in: Processing Conference Human Factors in Computing Systems, 1995, pp. 194–201.
[4] P. Resnick, H.R. Varian, Recommender systems, Commun. ACM 40 (1997) 56–58.
[5] N.J. Belkin, Helping people find what they don't know, Commun. ACM 43 (2000) 58–61.
[6] J.-G. Liu, K. Shi, Q. Guo, Solving the accuracy–diversity dilemma via directed random walks, Phys. Rev. E 85 (2012) 016118.
[7] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item, collaborative filtering, IEEE Internet Comput. 7 (2003) 76–80.

[8] J. Bennett, S. Lanning, The Netflix prize, in: Proceedings of KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007.
[9] K. Ali, W. van Stam, TiVo: making show recommendations using a distributed collaborative filtering architecture, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 394–401.
[10] J.-G. Liu, T. Zhou, B.-H. Wang, Research progress in personalized recommendation system, Prog. Nat. Sci. 19 (2009) 1–15.
[11] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. 22 (2004) 5–53.
[12] N. Zheng, Q.-D. Li, A recommender system based on tag and time information for social tagging systems, Expert Syst. Appl. 38 (2011) 4575–4587.
[13] C.-X. Yin, Q.-K. Peng, T. Chu, Personal artist recommendation via a listening and trust preference network, Physica A 391 (2012) 1991–1999.
[14] A.A. Kardan, S. Abbaspour, F. Hendijanifard, A hybrid recommender system for e-learning environments based on concept maps and collaborative tagging, in: The Proceedings of the 4th International Conference on Virtual Learning, ICVL 2009, 2009 pp. 300–307.
[15] Y. Zhang, B. Zhang, K.-N. Gao, P.-W. Guo, D.-M. Sun, Combining content and relation analysis for recommendation in social tagging systems, Physica A 391 (2012) 5759–5768.
[16] H.N. Kim, A.T. Ji, I. Ha, G.S. Jo, Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation, Electron. Commerce Res. Appl. 9 (2010) 73–83.
[17] M.-S. Shang, Z.-K. Zhang, T. Zhou, Y.-C. Zhang, Collaborative filtering with diffusion-based similarity on tripartite graphs, Physica A 389 (2010) 1259–1264.
[18] Z.-K. Zhang, T. Zhou, Y.-C. Zhang, Personalized recommendation via integrated diffusion on user-item-tag tripartite graphs, Physica A 389 (2010) 179–186.
[19] R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J.X. Parreira, G. Weikum, Efficient top-k querying over social-tagging networks, in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2008, p. 523.
[20] R.Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, H. Kato, Y. Inagaki, Reasonable tag-based collaborative filtering for social tagging systems, in: Proceedings of the 2nd ACM Workshop on Information Credibility on the Web, 2008, pp. 11–18.
[21] K.H.L. Tso-Sutter, L.B. Marinho, L. Schmidt-Thieme, Tag-aware recommender systems by fusion of collaborative filtering algorithms, in: Proceedings of the 2008 ACM Symposium on Applied Computing, 2008, pp. 1995–1999.
[22] R.-C. Chen, Y.-H. Huang, C.-T. Ba, S.-M. Chen, A recommendation system based on domain ontology and SWRL for anti-diabetic drugs selection, Expert Syst. Appl. 39 (2012) 3995–4006.
[23] Ángel Garcí-Crespo, José Luis López-Cuadrado, Ricardo Colomo-Palacios, Israel González-Carrasco, Belén Ruiz-Mezcua, Sem-Fit: a semantic based expert system to provide recommendations in the tourism domain, Expert Syst. Appl. 38 (2011) 13310–13319.
[24] Z.-Y. Xin, Z.-F. Ma, M. Gu, Collaborative-filtering recommendation algorithm based on Fuzzy clustering for domain information service, Comput. Sci. 34 (2007) 128–130.
[25] Walter Carrer-Neto, Marí Luisa Hernández-Alcaraz, Rafael Valencia-Garcí, Francisco Garcí-Sánchez, Social knowledge-based recommender system, application to the movies domain, Expert Syst. Appl. 39 (2012) 10990–11000.
[26] D. Wei, T. Zhou, G. Cimini, P. Wu, W.-P. Liu, Y.-C. Zhang, Effective mechanism for social recommendation of news, Physica A 390 (2011) 2117–2126.
[27] MovieLens is a benchmark dataset. Available at http://www.movielens.org/login.
[28] C.-L. Hwang, K. Yoon, Multiple Attribute Decision Making, Springer-Verlag, 1981.
[29] C.-L. Hwang, Y.-J. Lai, T.-Y. Liu, A new approach for multiple objective decision making, Comput. Oper. Res. 20 (1993) 889–899.
[30] HetRec 2011 is the abbreviation of the 2nd international workshop on information heterogeneity and fusion in recommender systems. Available at http://ir.ii.uam.es/hetrec2011.
[31] J. Shao, L. Yao, R.-Y. Cai, Y. Zhang, Lasso-based tag expansion and tag-boosted collaborative filtering, advances in multimedia information processing 2010 part II, Lecture Notes in Comput. Sci. 6298 (2011) 559–570.
[32] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Eigentaste: a constant time collaborative filtering algorithm, Inf. Retr. J. 4 (2001) 133–151.
[33] T. Zhou, J. Ren, M. Medo, Y.-C. Zhang, Bipartite network projection and personal recommendation, Phys. Rev. E 76 (2007).