A NOVEL FUZZY-BASED SIMILARITY MEASURE FOR COLLABORATIVE FILTERING TO ALLEVIATE THE SPARSITY PROBLEM

M. SAEED AND E. G. MANSOORI

Abstract. Memory-based collaborative filtering is the most popular approach to build recommender systems. Despite its success in many applications, it still suffers from several major limitations, including data sparsity. Sparse data affect the quality of the user similarity measurement and consequently the quality of the recommender system. In this paper, we propose a novel user similarity measure based on fuzzy set theory along with default voting technique aimed to provide a valid similarity measurement between users wherever the available ratings are relatively rare. The main idea of this research is to model the rating behaviour of each user by a fuzzy set, and use this model to determine the user's degree of interest on items. Experimental results on the MovieLens and Netflix datasets show the effectiveness of the proposed algorithm in handling data sparsity problem. It also outperforms some state-of-the-art collaborative filtering algorithms in terms of prediction quality.

1. Introduction

The explosive growth and variety of information available on the Web in recent years overwhelmed users, leading them to make poor decisions. Recommender systems (RS) have emerged in this context as a solution based on collective intelligence to either predict whether a particular user will like a particular item or identify the collection of items that will be of interest to a certain user [19, 1, 20]. A variety of approaches for recommender systems have been developed that utilize various types of data and analysis tools. Collaborative filtering (CF) is probably the most successful and widely used technique for building RS [36, 35]. The provision of personalized recommendation requires that the CF systems have some information about each user, depending on the type of the employed filtering algorithm [22]. Pure CF approaches take a matrix of given user-item ratings as the only input which express the opinion of users on items.

CF approaches are often classified as being either memory-based or model-based [1]. The traditional CF techniques are said to be memory-based because the original ratings database is used directly for generating the recommendations or making the predictions [29]. On the other hand, model-based approaches use ratings database to learn a predictive model which can be used to predict ratings of users for new items [9]. Memory-based CF methods can be further divided into two groups, namely user-based and item-based algorithms [31]. The user-based algorithms look for users (also called neighbors) similar to the active user, and calculate a predicted rating as a weighted

Received: June 2016; Revised: September 2016; Accepted: January 2017

Key words and phrases: Recommender system, Collaborative filtering, Similarity measure, Data sparsity.

average of the neighbor's ratings on the active item. On the other hand, item-based algorithms look for similar items for an active user [31, 22].

However in practice, systems based on CF algorithms, in spite of their huge success suffer from a range of problems; the most fundamental is the data sparsity problem [32, 30, 26]. Data sparsity refers to the problem of insufficient data, or sparseness, in the rating database. In most recommender systems, many of users only rate a small number of items; this leads to similarity measures having very low accuracy.

In order to improve the prediction accuracy, this paper proposes a heuristic similarity measure based on fuzzy set theory [38]. It aims to provide a valid similarity measure between users with few ratings. The main idea is that it can model the rating behavior of each user by a fuzzy set and use this model to determine the user's degree of interest on items. In order to solve the data sparsity problem, default voting technique [5] is applied. It assigns default values to items that only one of the two users have rated in order to improve the similarity quality of sparse rating databases. The proposed similarity measure not only considers the number of common ratings between users, but also takes the difference of absolute value of ratings between users into account.

The remaining structure of this paper is organized as follows. In Section 2, some related works are reviewed. Section 3 describes the preliminaries. In Section 4, our proposed approach and its features are explained. Section 5 presents the experiments, including details of datasets, evaluation metrics and discussion of results. Finally, a conclusion is presented in Section 6.

2. Related Work

Sparsity is a problem occurring frequently in RS when many users have provided ratings to a limited number of items, or many items have received only a few ratings [8]. The traditional similarity measures such as Pearson correlation coefficient [28] or cosine similarity [5], base the similarity computation step between two users, on the set of common items rated by both users. In the presence of sparseness in the data, many users do not share common items, thus rendering CF useless.

Over the years, a variety of solutions to the data sparsity problem, have been proposed. Luo et al. [24] divide user similarity into two parts: local user similarity and global user similarity. Local similarity is determined based on surprisal-based vector similarity (SVS). Global similarity measures the similarity between two users by further considering the extent to which their neighbors are locally similar (using the local similarity). Therefore, two users become more similar if they can be connected through a series of locally similar neighbors.

Jamali and Ester [17] introduced a similarity measure using the Markov-chain model of a random walk, based on the sigmoid function. This approach can weaken the similarity of small common items among users.

Data smoothing technique is another mostly-used method to improve the recommendation performance in CF. Various sparsity measures [3] were used to enhance accuracy of CF. These sparsity measures were computed based on local and global similarities. Then, an estimating parameter scheme for weighting the various sparsity measures was proposed. The experimental results demonstrated that the proposed estimating parameter outperforms the schemes which kept the parameter constant. Ma et al. [25] proposed a partial missing data prediction algorithm, in which

the information of both users and items was taken into account. In this algorithm, similarity thresholds for users and items were set respectively, and the missing data could be predicted if and only if the intersection of the neighbors of user and the neighbors of item is not empty.

In a work by Cornelis et al. [7], the user/item similarities are modeled as fuzzy relations to resolve data sparsity problem. In this context, Leung et al. [23] presented a collaborative filtering framework based on fuzzy association rules and multi-level similarity. This extends the existing techniques by using fuzzy association rules to address data sparseness and non-transitive associations.

Zhang et al. [39] argued that the traditional user similarity measures require users to rate a minimum number of same items (i.e., a kind of exactly matching). To deal with data sparsity, they introduced a novel user similarity measure based on fuzzy matching of user favorite items. They applied it into Slope One algorithm for rating prediction.

To solve the sparsity problem, Patra et al. [27] addressed this problem by introducing two similarity measures based on Bhattacharyya coefficient (BCFmed and BCFcor) which utilize all rating data in user similarity measurement. The main challenge of the BCF measures is that they ignore differences in two users' opinions on co-rated items. Moreover, these measures are unable to compute user similarity when each of two user's ratings on every rated item have same distances from the item's median rating (in BCFmed) or the item's average rating (in BCFcor).

The transitive associations captured by graph-based methods can be used to recommend items. In graph-based approaches, the data is represented in the form of a graph, where nodes are users, items or both and edges encode the interactions or similarities among the users and items. Fouss et al. [11] suggested applying Euclidean commute time distance as a random walk-based method to compute similarities between nodes.

To reduce the problems from high levels of sparsity in RS databases, certain studies have used dimensionality reduction approaches [4]. The reduction methods are based on matrix factorization which decompose the user-item rating matrix or the sparse similarity matrix [13] into a limited number of latent factors.

3. Background and Preliminaries

3.1. Notations. Suppose that given an RS with a database of m users and n items rated in the range $\{r_{min}, \ldots, r_{max}\}$. We use U to denote the set of users, and I for the set of items. We reserve special indexing letters to distinguish users from items: for users u, v, w and for items i, j, l. The rating values are collected in a $m \times n$ ratings matrix, R, wherein $r_{u,i}$ represents the rating value given by user u to item i. Conventionally, $r_{u,i} = \bullet$ denotes that the rating of user u to item i is unknown. U_i represents the subset of users that have rated item i and I_u denotes the subset of items that have been rated by user u. Also, $I_{u,v}$ shows the items that have been rated by two users u and$v$ and I_{u+v} shows the items that have been rated by at least one of the users u and v. Similarly, $U_{i,j}$ denotes the set of users that have rated both items i and j and U_{i+j} denotes the set of users that have rated at least one of the items i and j. The average of ratings given by user u to the items in I_u is denoted by $\overline{r}_{u,*}$ and $\overline{r}_{*,i}$ corresponds to the average of ratings given to item i by users in U_i. The number of elements in a set S is represented by $|S|$. Finally, the predicted rating for the actual rating $r_{u,i}$ is denoted as $p_{u,i}$.

3.2. Memory-Based Collaborative Filtering. The memory-based CF algorithms use entire user-item rating matrix to generate a prediction for an item. There are two types of approaches for finding prediction: user-based and item-based. In this paper, we focus on the former approach. The user-based CF, also known as k-NN CF, was the first of the automated CF algorithms [14]. The user-based CF algorithms make prediction using the ratings made by other users with similar preferences and tastes. Suppose that $\mathcal{N}_u(i;k)$ are the k users as v with the highest similarity $sim(u,v)$ to u that have rated i. The rating $r_{u,i}$ can be predicted as the weighted average rating given to i by these neighbors:

$$p_{u,i} = \overline{r}_{u,*} + \frac{\sum_{v \in \mathcal{N}_u(i;k)} (r_{v,i} - \overline{r}_{v,*}) \times sim(u,v)}{\sum_{v \in \mathcal{N}_u(i;k)} sim(u,v)} \tag{1}$$

Subtracting the user's mean rating, $r_{v,*}$, compensates for differences in users' use of the rating scale [10]. Therefore, the measurement of the similarity between users plays a fundamental role in the user-based CF algorithms. The most commonly-used measurement techniques for similarities between users are the Pearson correlation coefficient (PCC) [28] and cosine similarity measure (CSM) [5] that are based on the set of common items rated by both users. The PCC measures how two users are linearly correlated to each other. However, it only considers the absolute rating values on co-rated items, while the number of co-rated items is also important for measuring similarity between two users. The PCC is defined as:

$$sim^{PCC}(u,v) = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - \overline{r}_{u,*})(r_{v,i} - \overline{r}_{v,*})}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - \overline{r}_{u,*})^2} \sqrt{\sum_{i \in I_{u,v}} (r_{v,i} - \overline{r}_{v,*})^2}} \tag{2}$$

The CSM assumes that the rating of each user is a point in a vector space, and then evaluates the cosine angle between the rating vectors of two users. The CSM is formulated as:

$$sim^{CSM}(u,v) = \frac{\sum_{i \in I_{u,v}} r_{u,i} \times r_{v,i}}{\sqrt{\sum_{i \in I_u} r_{u,i}^2} \sqrt{\sum_{i \in I_v} r_{v,i}^2}} \tag{3}$$

The constrained Pearson correlation coefficient (CPCC) [34] is a slightly modified version of the PCC that increases the correlation only when both users have rated an item positively or negatively. CPCC is specified as:

$$sim^{CPCC}(u,v) = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - r_{med})(r_{v,i} - r_{med})}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - r_{med})^2} \sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - r_{med})^2}} \tag{4}$$

where r_{med} represents the median value in the rating scale.

Like PCC, the Spearman rank correlation (SRC) [14] computes a measure of correlation between ranks instead of actual preference scores. For SRC, the items, which are rated by a user, are ranked such that the highest-rated item is at first rank. Items with the same rating are assigned the average rank for their position. The computation is then the same as that of the PCC, except that the ranks are used

instead of ratings. The SRC is defined as:

$$sim^{SRC}(u,v) = \frac{\sum_{i \in I_{u,v}} (rk_{u,i} - \overline{rank}_{u,*})(rank_{v,i} - \overline{rank}_{v,*})}{\sqrt{\sum_{i \in I_{u,v}} (rank_{u,i} - \overline{rank}_{u,*})^2} \sqrt{\sum_{i \in I_{u,v}} (rank_{v,i} - \overline{rank}_{v,*})^2}} \quad (5)$$

where $rank_{u,i}$ is the rank of item i for user u and $\overline{rank}_{u,*}$ represents the average of rankings for user u.

Shardanand et al. [34] proposed a measure based on mean square difference (MSD), which evaluates the similarity between two users as the inverse of the average squared difference between the ratings given by those users on the same item. The MSD is formulated as:

$$sim^{MSD}(u,v) = 1 - \frac{\sum_{i \in I_{u,v}} (r_{u,i} - r_{v,i})^2}{|I_{u,v}|} \quad (6)$$

Jaccard (JD) similarity measure [21] is another commonly-used similarity measure. The basic idea is that users are more similar if they have more common ratings. Its drawback is that it does not consider the absolute ratings. It is defined as:

$$sim^{JD}(u,v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \quad (7)$$

Similarity computation is a vital step in the neighborhood-based CF. As these similarity measures have some weaknesses such as data sparsity, cold-start and scalability, many improved similarity measures have been introduced to overcome these drawbacks.

4. The Proposed Similarity Measure

4.1. Basic Idea. One of the most commonly-used measurement technique for similarities between users is JD (equation (7)). It only considers the number of common ratings between two users. Clearly, such a measure has the main disadvantage that it does not take the difference of absolute value of ratings between users into account. In this case, if two users rate the same items but have completely opposite opinions on them, then they are considered to be similar anyway according to JD.

In general, since the scale of ratings is absolute in RS; the system can know which ratings are positive or negative. If r_{med} is the median value in the rating scale, the ratings smaller than r_{med} show negative ratings, those larger than r_{med} represent positive ratings and r_{med} indicates ambivalence. In order to consider the impact of positive, negative and ambivalence ratings, we use a slightly modified version of JD, called like-minded (LM) similarity measure, as:

$$sim(u,v)^{LM} = \frac{|I_u^- \cap I_v^-| + |I_u^0 \cap I_v^0| + |I_u^+ \cap I_v^+|}{|I_u \cup I_v|} \quad (8)$$

where $I_u^- = \{i \in I_u : r_{u,i} < r_{med}\}$, $I_u^0 = \{i \in I_u : r_{u,i} = r_{med}\}$ and $I_u^+ = \{i \in I_u : r_{u,i} > r_{med}\}$ are the set of rated items by user u as negative, ambivalence and positive, respectively. Since it makes use of similarity measures based on similar opinion of users, we have called this similarity measure like-minded. As a drawback, it does not take the differences in the average rating behavior of the users into account. In other words,

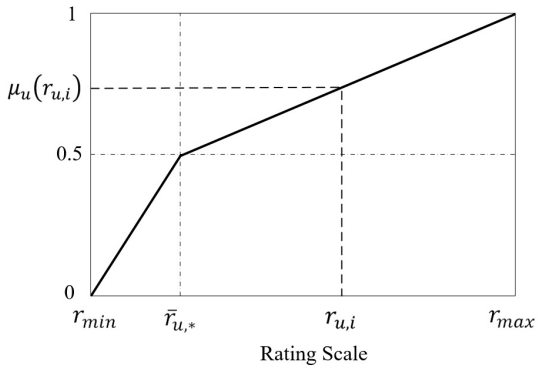

FIGURE 1. The Membership Function $\mu_u(r)$ for Preference Behavior of User u

different users have different preferences of rating. Some people like to rate high, even they do not like the items very much. Likewise, some people tend to rate low, even they like the items very much. Therefore, we can improve the concepts of negative, ambivalence and positive as $I_u^- = \{i \in I_u : r_{u,i} < \overline{r}_{u,*}\}$, $I_u^0 = \{i \in I_u : r_{u,i} = \overline{r}_{u,*}\}$, and $I_u^+ = \{i \in I_u : r_{u,i} > \overline{r}_{u,*}\}$. However, it only considers the number of common items and does not take the user's degree of interest on common items into account.

The main idea of this research is that we can model the rating behavior of each user by a fuzzy set and use this model to determine the user's degree of interest on items. By defining a fuzzy set on universe of rating scale and assigning minimum rating scale as lowest degree of interest 0 and maximum value of the rating scale as degree of interest 1, we obtain the average rating of each user as crossover point for this fuzzy set. The rating values lower than crossover point mean lower preference of user rating to items. Similarly, the higher rating values show more preference of user to items (Figure 1). We define the "degree of interest of user u on items" as membership degree of fuzzy set "user u's ratings to items" as:

$$\mu_u(r) = \begin{cases} \frac{r - r_{min}}{\overline{r}_{u,*} - r_{min}} \times 0.5, & r_{min} \leq r \leq \overline{r}_{u,*} \\ \frac{r - \overline{r}_{u,*}}{r_{max} - \overline{r}_{u,*}} \times 0.5 + 0.5, & \overline{r}_{u,*} \leq r \leq r_{max} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Substituting the set of rated items in equation (8) by their membership degrees in equation (9) and replacing intersection and union operators by fuzzy operators, we get:

$$sim^{FS}(u,v) = \frac{\sum_{i \in I_{u,v}} T(\mu_u(r_{u,i}),\ \mu_v(r_{v,i}))}{\sum_{i \in I_{u+v}} S(\mu_u(r_{u,i}),\ \mu_v(r_{v,i}))} \tag{10}$$

where $T(.,.)$ and $S(.,.)$ are fuzzy T-norm and S-norm operators, respectively. Based on various T-norm and S-norms, a number of qualified methods can be formulated to calculate the similarity between users.

It may happen that there is not any co-rated item between users. The next suggested similarity measure combines equation (10) with default voting [5]. Default voting is a

technique for dealing with sparse rating databases. As mentioned before, the standard similarity measures take into account only items for which both users have submitted ratings. When this rating is too low, coincidental rating commonalities and differences influence the similarity measure too much. To improve the prediction quality of sparse rating databases, the idea is to assign default values to items that are rated by only one of the two users [18]. By extending each user's rating history, our proposed similarity measure can be written as:

$$sim^{DFS}(u,v) = \frac{\sum_{i \in I_{u+v}} T({\mu'}_u(r_{u,i}),\ {\mu'}_v(r_{v,i}))}{\sum_{i \in I_{u+v}} S({\mu'}_u(r_{u,i}),\ {\mu'}_v(r_{v,i}))} \tag{11}$$

where ${\mu'}_u$ is defined as:

$${\mu'}_u(r_{u,i}) = \begin{cases} \mu_u(r_{u,i}), & r_{u,i} \neq \bullet \\ \mu_u(\overline{r}_{*,i}), & r_{u,i} = \bullet \end{cases} \tag{12}$$

It is worth mentioning that when user u has not rated item i, we use the average of ratings on item i as $\overline{r}_{*,i}$. So, those items are selected that their estimated ratings are possibly accurate. Hence, we give more priority to those items that receive more ratings from users and have lower variance. It should be noted that a small variance indicates that the average of ratings of users are more close to actual ones.

Intuitively, if we include all the items in I_{u+v}, it not only influences the performance negatively with respect to the required computational time, but also it affects the quality of similarity by entering additional noise in the computations. Therefore, we define a minimum threshold, L, for the number of items to being taken into account. If the number of co-rated items for computing similarity between users u and v are not sufficient (i.e., $|I_{u,v}| < L$), the default voting values are used to obtain the desired number of items for similarity computation. Finally, by selecting a subset $\mathcal{I}_{u+v}$ of items in I_{u+v} (i.e., $\mathcal{I}_{u+v} \subset I_{u+v}$), equation (2) becomes:

$$sim^{DFS}(u,v) = \frac{\sum_{i \in \mathcal{I}_{u+v}} T({\mu'}_u(r_{u,i}),\ {\mu'}_v(r_{v,i}))}{\sum_{i \in \mathcal{I}_{u+v}} S({\mu'}_u(r_{u,i}),\ {\mu'}_v(r_{v,i}))} \tag{13}$$

4.2. Significance Weighting. It has been shown that the reliability of similarity between users u and v is often affected by the number of common ratings [2]. As a result, the predictions based on the ratings of neighbors which have only a very few co-rated items with active user, lead to poor predictions [14]. In order to take this problem into account, several strategies have been proposed. The principle of all these strategies is to reduce the magnitude of a similarity weight when this weight is computed using only a few co-rated items [25].

In order to achieve the best possible result, we have used the modification described in Ma et al. [25]. It weights the similarity by the number of co-rated items between u and v up to a threshold, γ:

$$sim^{SDFS}(u,v) = \frac{min(|I_{u,v}|,\ \gamma)}{\gamma} \times sim^{DFS}(u,v) \tag{14}$$

This equation means that to each item in $I_{u,v}$, a weighting factor of 1 will be assigned.

Attribute \ Dataset Name	MovieLens 100K	Netflix Tiny
Users	943	4,427
Items	1,682	1,000
Ratings	100,000	56,136
Matrix ratio	0.5606	4.4270
Sparsity rate	93.695%	98.732%
User's Minimum No. of ratings	20	1
User's Maximum No. of ratings	737	314
User's Average No. of ratings	106	12

TABLE 1. Statistics of the MovieLens 100k and Netflix Tiny Datasets

Since the average of item ratings, $\overline{r}_{*,i}$, is used as default voting value, it is reasonable to assign less weights to items based on default voting values in comparison to the actual values. Since the accuracy of average of item ratings depends on the number of ratings received by them, we assign more weights to items which have more ratings. Accordingly, we assign weight 1 to the co-rated items and some significant weights, in equation (16), to items obtained by default voting technique. Therefore, the final proposed equation for computing similarity (denoted SDFS) is as follows:

$$sim^{SDFS}(u,v\ ) = \frac{min\{\sum_{i\in\mathcal{I}_{u+v}} w_i(u,v),\ \gamma\ \}}{\gamma} \times sim^{DFS}(u,v\ ) \tag{15}$$

where w_i is defined as:

$$w_i(u,v) = \begin{cases} 1, & i \in I_{u,v} \\ \frac{min\{|U_i|,\beta\}}{\beta}, & i \notin I_{u,v} \end{cases} \tag{16}$$

where β is a threshold parameter.

4.3. Rating Prediction. After computing the similarity between users by equation (15), we select a set of k most similar users to the active user and generate a predicted value of user u's rating by substituting the similarity measure in equation (1) as:

$$p_{u,i} = \overline{r}_{u,*} + \frac{\sum_{v\in\mathcal{N}_u(i;k)} (r_{v,i} - \overline{r}_{v,*}) \times sim^{SDFS}(u,v)}{\sum_{v\in\mathcal{N}_u(i;k)} sim^{SDFS}(u,v)} \tag{17}$$

5. Experimental Evaluations

5.1. Dataset. Experimental data come from two real-world datasets, MovieLens[1] dataset [14] and Netflix Tiny[2] dataset [37]. The MovieLens dataset has multiple versions of different sizes. Since MovieLens 100K is widely used in many researches, it is used to evaluate the performance of our proposed algorithm. This dataset contains 100,000 ratings collected from 943 users on 1682 movies where each user has rated at least 20 movies. In Netflix Tiny dataset, 4,427 users issued 56,136 ratings for 1,000

[1]http://grouplens.org/datasets/movielens
[2]http://www.prea.gatech.edu/download.html

items where each user has rated at least 1 item. In both datasets, ratings are on an integer scale 1 (bad) to 5 (excellent) where 1 and 2 represent negative ratings, 4 and 5 represent positive ratings, and 3 indicates ambivalence. Each dataset has specific features that may affect the performance of different algorithms. Table 1 summarizes the statistics of these datasets.

5.2. Evaluation Metrics. Our specific task in this paper is to predict scores for items that already have been rated by actual users, and to check how well this prediction helps users in selecting high quality items. Keeping this into account, there are two key dimensions on which the quality of a prediction algorithm can be measured: (i) accuracy and (ii) coverage. We adopt the widely used mean absolute error (MAE) [16] metric for evaluating prediction accuracy, defined as:

$$\text{MAE} = \frac{\sum_{u \in U} \sum_{i \in Test_u} |p_{u,i} - r_{u,i}|}{\sum_{u \in U} |Test_u|} \tag{18}$$

where $Test_u$ is the set of all items in the testing set of user u. MAE measures the average absolute deviation between a recommender system's predicted rating and a true rating assigned by the user.

Another widely used performance measure, the coverage, is also applied in this research to evaluate and analyze the algorithm's behavior with respect to sparsity conditions where there are only a few known ratings. Coverage measures the number of items that a RS can recommend or predict. In this work, the coverage is not considered as the percentage of items that can be predicted from all available ones [12], because an algorithm can increase coverage by making bogus predictions. Hence, coverage and accuracy must be measured simultaneously. Consequently, only those items are selected that have already been rated by users and are in the test set; those items that can be predicted by system [6, 33]. Herlocker et al. have used the term prediction coverage for this metric [15]. It can be defined as:

$$\text{Coverage} = \frac{\sum_{u \in U} \sum_{i \in\ Test_u} \rho_{u,i}}{\sum_{u \in U} |Test_u|} \tag{19}$$

where $\rho_{u,i}$ is an indicator function, defined as:

$$\rho_{u,i} = \begin{cases} 1, & p_{u,i} \neq \bullet \\ 0, & p_{u,i} = \bullet \end{cases} \tag{20}$$

5.3. The Baseline Algorithms. To evaluate the performance of the proposed similarity measure (SDFS), we have implemented user-based CF using PCC, CPCC, LM and JD similarity measures. For the sake of experimental analysis, we have also incorporated per user average (PUA) and per item average (PIA) algorithms. The PUA algorithm returns the average of the ratings that the given user has already entered (given by $p_{u,i} = \overline{r}_{u,*}$). The PIA algorithm gives the average rating for the given item of all users that have rated for that item (given by $p_{u,i} = \overline{r}_{*,i}$).

The proposed similarity measure, SDFS, is implemented using $min(\dots)$ and $max(\dots)$ for T-norm and S-norm operators, respectively. The threshold value $\gamma = 50$ is used in

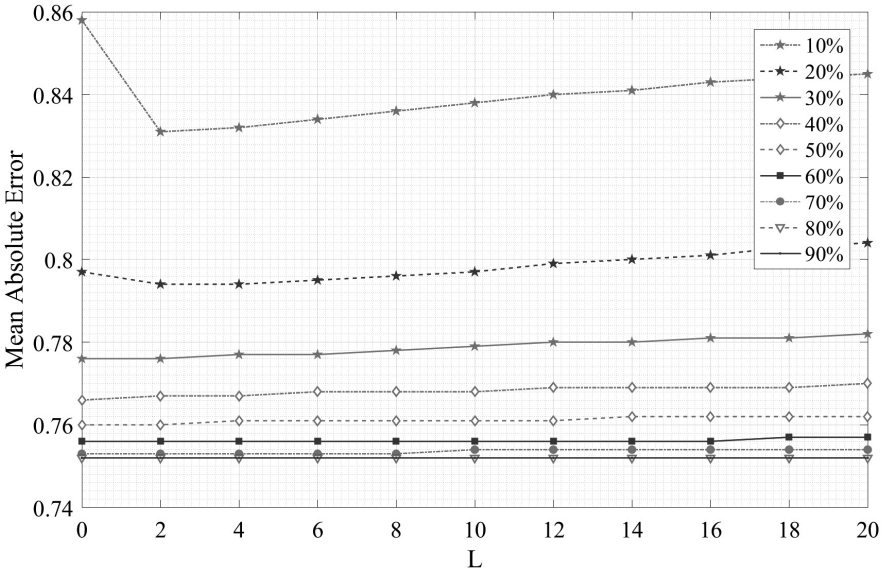

FIGURE 2. Determining the Optimal Value of Parameter L for MovieLens 100K Dataset Under Different Density Conditions

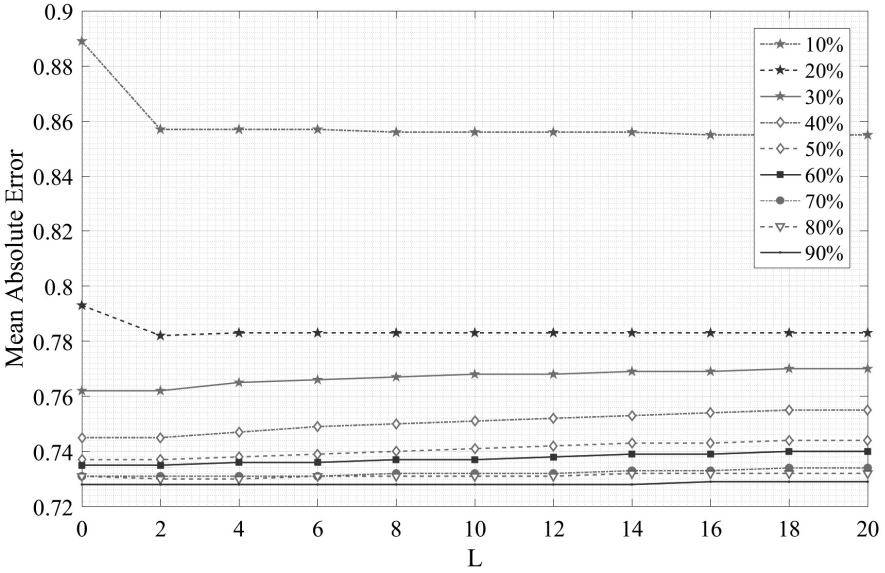

FIGURE 3. Determining the Optimal Value of Parameter L for Netflix Tiny Dataset Under Different Density Conditions

the experiments, as described in Herlocker et al. [14]. For parameter β, the average number of ratings given by users to the items for each dataset is chosen.

We have tuned the user-based CF and item-based CF algorithms to deliver the highest quality prediction. In this regard, they consider every possible neighbor which have positive correlation to form the optimal neighborhoods.

5.4. Optimal Value for Threshold Parameter *L*. The threshold L determines the minimum number of items which are included for similarity computation. If L is set too high, too many items with limited information bring additional noise into the similarity computation. On the other hand, if L is selected too small, the quality of the similarity computation might be negatively affected. To find the optimal value of L, it is varied from 0 to 20 with step 2 while measuring MAE under different density conditions. In this regard, $L = 0$ means that only the co-rated items are considered for computing similarity. So, it is a baseline which can illustrate the effect of parameter L on quality of prediction. The effect of L on MAE under different density conditions for MovieLens 100K and Netflix Tiny datasets are shown in Figure 2 and Figure 3, respectively. Accordingly, the MAE is minimum at $L = 2$ in all densities. Therefore, $L = 2$ is used in the subsequent experiments.

5.5. Performance Evaluation Under the Sparsity Problem. To check the effect of sparsity, we have followed two distinct approaches for constructing the training set. In the first one, 10% of the dataset, chosen randomly, is set aside as test set. The rest of ratings are used for training, wherein we have used 10% to 90% with a step of 10%. In the second approach, the training set is constructed by varying the number of rated items, provided by each user, in 5, 10, 15 and 20 (denoted by Given-5, Given-10, Given-15 and Given-20, respectively). The rest of ratings are used as test set. The above experiments have repeated 5 times and the average of results for each metric is reported.

5.5.1. Performance Evaluation According to Ratings Density. In these experiments, we have evaluated the MAE and coverage of the different algorithms when some distinct percent of ratings are used to build the training sets. In this regard, when the training set contains small percent of ratings, the sparsity condition occurs. In contrast, the higher percentages allow the algorithms to be evaluated under relatively high density conditions. The results for MovieLens 100K and Netflix Tiny datasets are shown in Figure 4 and Figure 5, respectively.

In terms of prediction coverage for MovieLens 100K dataset, the proposed SDFS based CF algorithm outperforms all other baseline algorithms for the ratings percent as shown in Figure 4(a). The prediction coverage of SDFS is equal to PIA. This means that for each item with at least one rating, SDFS can make prediction. Under sparsity conditions, with 10% of ratings, SDFS is still able to make prediction for more than 97% of tests set, while PCC covers 40%, CPCC covers 79%, JD less than 91% and LM reaches 84%. By increasing the percent of ratings in the training set, all algorithms have more information to make prediction and so, their prediction coverage are improved. For the Netflix Tiny dataset, the similar results are observable and in overall, the proposed SDFS outperforms the baseline algorithms for the ratings percent shown in Figure 5(a).

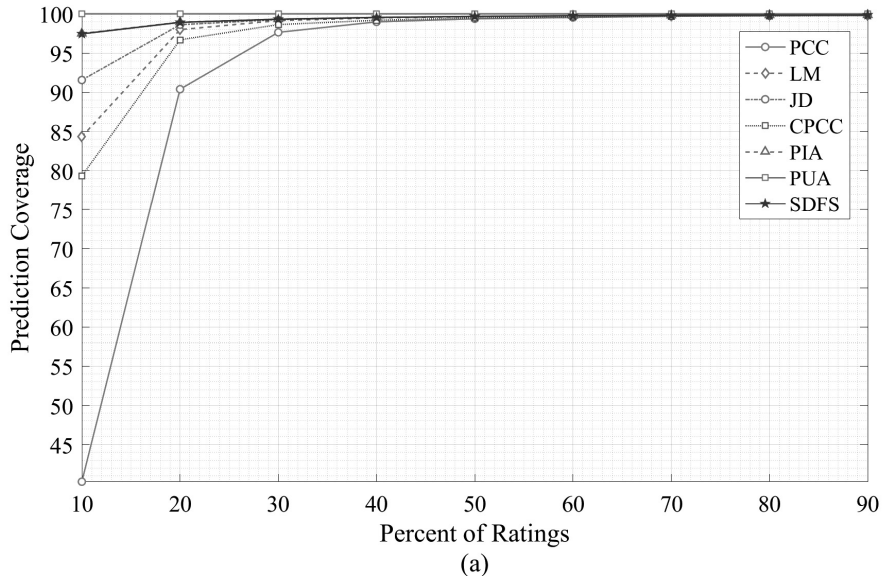

FIGURE 4. Performance of Algorithms for MovieLens 100K Dataset Under Different Ratings Density: (a) Prediction Coverage and (b) MAE

Under sparsity conditions, with 10% of ratings, SDFS is still able to make prediction for more than 97% of tests set while PCC covers 4%, CPCC 53%, JD about 77% and LM reaches 59%.

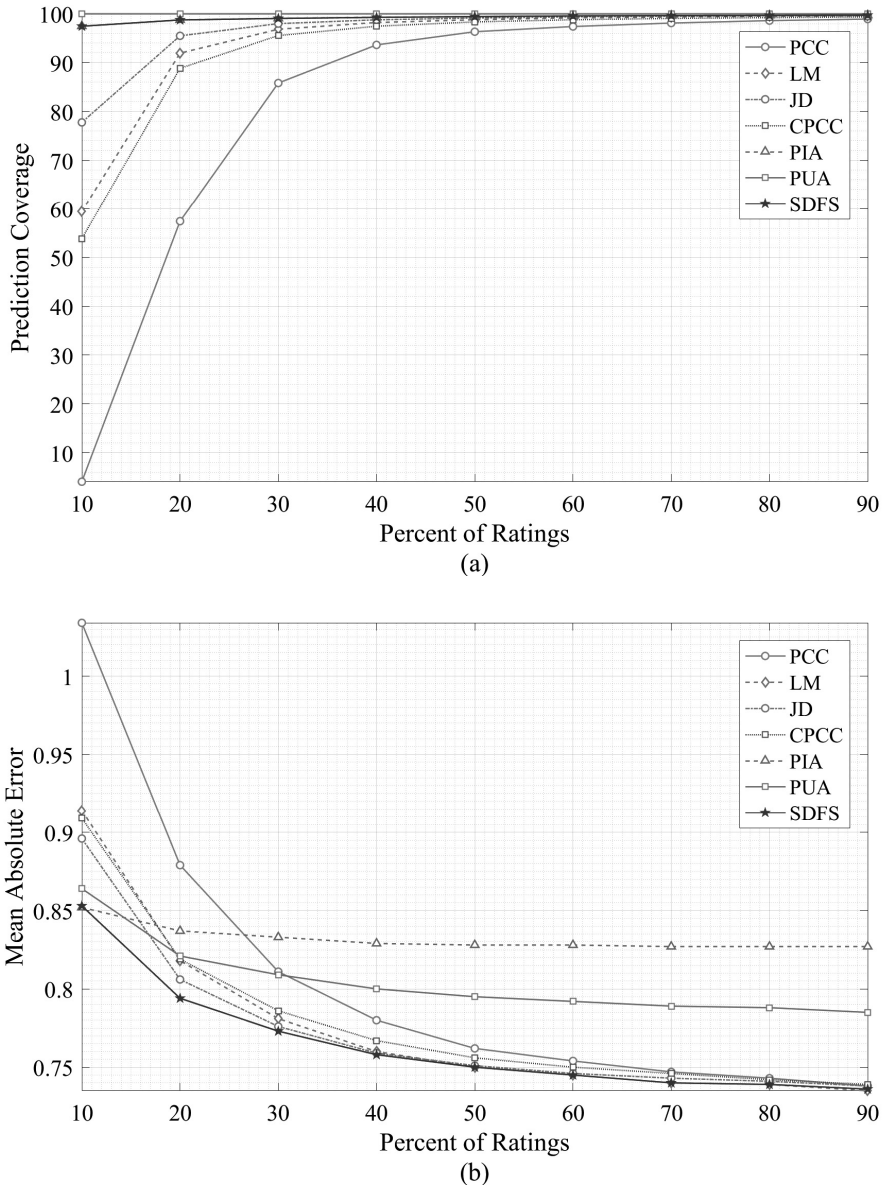

FIGURE 5. Performance of Algorithms for Netflix Tiny Dataset Under Different Ratings Density: (a) Prediction Coverage and (b) MAE

On the other hand, according to MAE plots for MovieLens 100K dataset in Figure 4(b), with less than 40% of the ratings as training set, the proposed SDFS clearly outperforms all other measures. With 10%, high sparsity condition, it improves PCC by 12%, CPCC by 7%, LM by 6% and JD by 4% in terms of MAE.

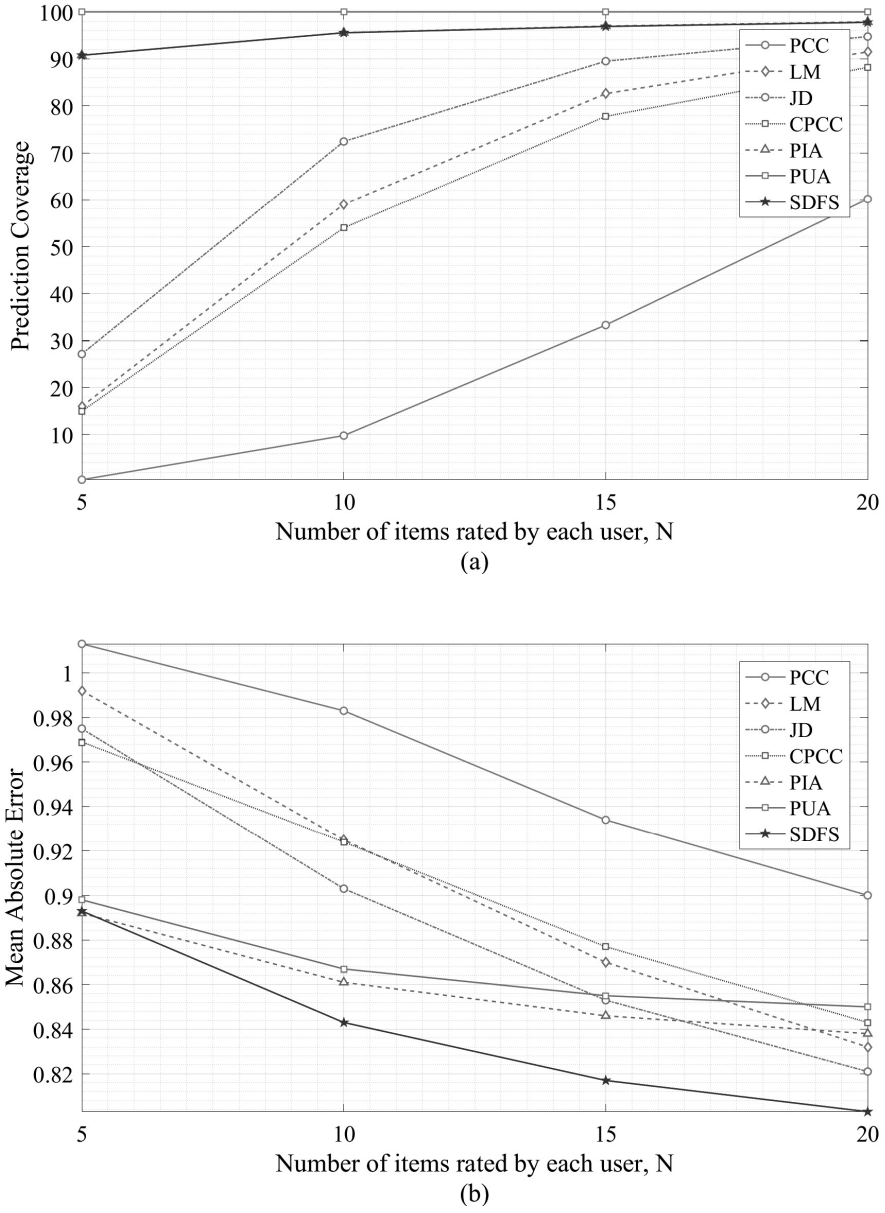

FIGURE 6. Performance of the Different Algorithms for MovieLens 100K Dataset Under Sparsity Conditions, by Given-N Strategy: (a) Prediction Coverage and (b) MAE

Also for coverage, it improves PCC by 12%, CPCC by 7%, LM by 6% and JD by 4%.

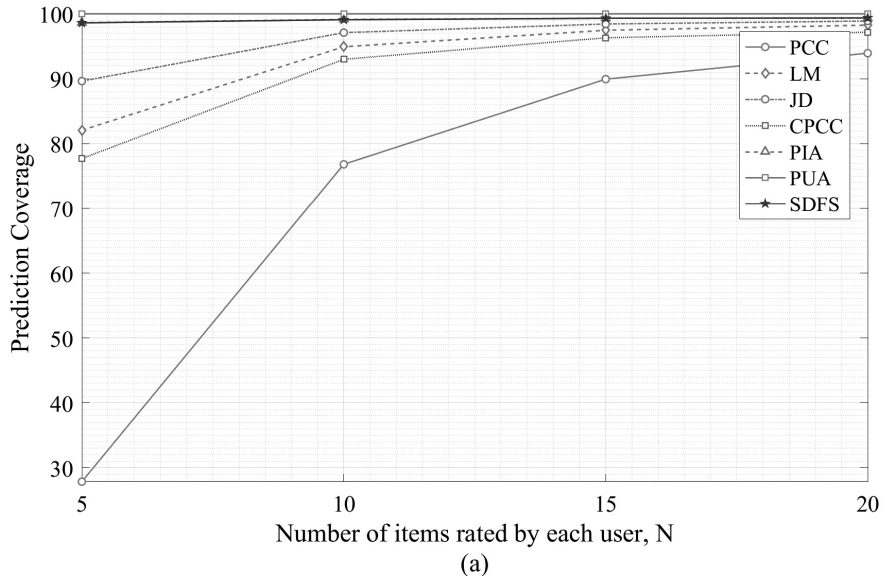

FIGURE 7. Performance of the different algorithms for Netflix Tiny dataset under sparsity conditions, by Given-N strategy: (a) prediction coverage and (b) MAE

As shown in Figure 5(b), the same results can be achieved for Netflix Tiny dataset. With 10% of high sparsity condition, it improves the MAE of PCC by 17%, CPCC by

7%, LM by 7% and JD by 5%. Also, it improves the prediction coverage of PCC by about 17%, CPCC by 6%, LM by 7% and JD by 5%. With a training set of 40% or above, there is no difference in terms of MAE and prediction coverage between proposed SDFS and baseline JD algorithm. In these situations, the items obtained by default voting values are no longer needed and there are enough co-rated items for computing similarity. In terms of MAE, with more than 40%, a negligible improvement (less than 1%) for LM and PCC measures are detected.

In overall, when the users have a lower number of ratings (i.e., the case of 10%), the improvement achieved by SDFS over the baseline measures is relatively larger than the improvements achieved when they have a higher number of ratings (i.e., the case of 40% or more). These results confirm that the proposed SDFS could be particularly beneficial for scenarios under data sparseness.

5.5.2. Performance Evaluation under Sparsity Conditions. In these experiments, the MAE and prediction coverage of the different algorithms under sparsity conditions are evaluated. The tests under sparsity conditions are appropriate indicators of the ability of algorithms to extract more information from the rating matrix. In this regard, the Given-N strategy is used to investigate the action of SDFS under such conditions.

The results for MovieLens 100K dataset is shown in Figure 6, while those for Netflix Tiny in Figure 7. With respect to prediction coverage in Figure 6(a) and Figure 7(a), SDFS outperforms other traditional measures for all number of items rated by each user. For MovieLens 100K dataset, with a training set containing only 5 ratings per user, the proposed SDFS can still make predictions for more than 90% of items, while PCC barely covers 1%, CPCC predicts 15%, LM percentage is less than 17% and JD reaches 28%. For the Netflix Tiny dataset, we observe the similar results as shown in Figure 7(a). With a training set containing only 5 ratings per user, SDFS is able to make predictions for more than 98% of items while PCC barely covers 28%, CPCC predicts 77%, LM percentage is less than 88% and JD reaches 89%.

In terms of MAE, as shown in Figure 6(b) and Figure 7(b), SDFS outperforms other traditional measures for all number of items rated by each user.

6. Conclusion

Memory-based collaborative filtering is the most popular approach in building recommender systems and has been successfully employed in many applications. However, the existing traditional similarity measures for memory-based collaborative filtering cannot provide reliable prediction for data sparsity condition as they cannot utilize full ratings information while finding neighborhood of active users. In order to address this problem, a similarity measurement based on fuzzy set theory along with default voting technique was proposed to provide a valid similarity measurement between users wherever there are relatively few ratings available. Via experiments on MovieLens and Netflix Tiny datasets, we concluded that the proposed similarity measure performs better against traditional similarity measures. Moreover, the experimental results confirmed that applying default voting technique to user similarity computation can provide more accurate results and so, the performance can be more pronounced when the data are sparse.

This work suggests several interesting directions for future work. Besides the problem of predicting a rating for a given item, recommending a list of items to a user is also a natural task for recommender systems. Investigating various T-norm and S-norm fuzzy operators, for similarity computation is also an interesting direction for future work. Moreover, the threshold parameter L can be set adaptively according to ratings density. Finally, this work can be extended by combining with other similarity measures, in order to benefit from their respective advantage points.

References

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

[2] C. C. Aggarwal, Recommender Systems: The Textbook, Springer, 2016.

[3] D. Anand and K. K. Bharadwaj, *Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities*, Expert Systems with Applications, **38(5)** (2011), 5101–5109.

[4] J. Bobadilla, F. Ortega, A. Hernando and A. Gutirrez, *Recommender systems survey*, Knowledge-Based Systems, **46** (2013), 109–132.

[5] J. S. Breese, D. Heckerman and C. Kadie, *Empirical analysis of predictive algorithms for collaborative filtering*, In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, (1998), 43–52.

[6] L. Chen, G. Chen and F. Wang, *Recommender systems based on user reviews: the state of the art*, User Modeling and User-Adapted Interaction, **25(2)** (2015), 99–154.

[7] C. Cornelis, X. Guo, J. Lu and G. Zhang, *A Fuzzy Relational Approach to Event Recommendation*, In Proceedings of the 2nd Indian International Conference on Artificial Intelligence, **5** (2005), 2231–2242.

[8] C. Desrosiers and G. Karypis, *A novel approach to compute similarities and its application to item recommendation*, In Pacific Rim International Conference on Artificial Intelligence, (2010), 39–51.

[9] C. Desrosiers and G. Karypis, *A comprehensive survey of neighborhood-based recommendation methods*, In Recommender Systems Handbook, (2011), 107–144.

[10] M. D. Ekstrand, J. T. Riedl and J. A. Konstan, *Collaborative filtering recommender systems*, Foundations and Trends in Human-Computer Interaction, **4(2)** (2011), 81–173.

[11] F. Fouss, A. Pirotte, J. M. Renders and M. Saerens, *Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation*, IEEE Transactions on Knowledge and Data Engineering, **19(3)** (2007), 355–369.

[12] M. A. Ghazanfar and A. Prugel-Bennett, *Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems*, Expert Systems with Applications, **41(7)** (2014), 3261–3275.

[13] K. Goldberg, T. Roeder, D. Gupta and C. Perkins, *Eigentaste: A constant time collaborative filtering algorithm*, Information Retrieval, **4(2)** (2001), 133–151.

[14] J. L. Herlocker, J. A. Konstan, A. Borchers and J. Riedl, *An algorithmic framework for performing collaborative filtering*, In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (1999), 230–237.

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

[16] R. J. Hyndman and A. B. Koehler, *Another look at measures of forecast accuracy*, International Journal of Forecasting, **22(4)** (2006), 679-688.

[17] M. Jamali and M. Ester, *Trustwalker: a random walk model for combining trust-based and item-based recommendation*, In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (2009), 397–406.

[18] D. Jannach, M. Zanker, A. Felfernig and G. Friedrich, Recommender systems: an introduction, Cambridge University Press, 2010.

[19] G. Karypis, *Evaluation of item-based top-n recommendation algorithms*, In Proceedings of the 10th International Conference on Information and Knowledge Management, (2001), 247–254.

[20] J. A. Konstan and J. Riedl, *Recommender systems: from algorithms to user experience*, User Modeling and User-Adapted Interaction, **22(1-2)** (2012), 101–123.
[21] G. Koutrika, B. Bercovitz and H. Garcia-Molina, *FlexRecs: expressing and combining flexible recommendations*, In Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, (2009), 745–758.
[22] A. S. Lampropoulos and G. A. Tsihrintzis, Machine Learning Paradigms, Springer, 2015.
[23] C. W. Leung, S. C. Chan and F. Chung, *A collaborative filtering framework based on fuzzy association rules and multiple-level similarity*, Knowledge and Information Systems, **10(3)** (2006), 357–381.
[24] H. Luo, C. Niu, R. Shen and C. Ullrich, *A collaborative filtering framework based on both local user similarity and global user similarity*, Machine Learning, **72(3)** (2008), 231–245.
[25] H. Ma, I. King and M. R. Lyu, *Effective missing data prediction for collaborative filtering*, In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (2007), 39–46.
[26] P. Massa and P. Avesani, *Trust metrics in recommender systems*, In Computing with Social Trust, (2009), 259–285.
[27] B. K. Patra, R. Launonen, V. Ollikainen and S. Nandi, *A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data*, Knowledge-Based Systems, **82** (2015), 163–177.
[28] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom and J. Riedl, *Grouplens: an open architecture for collaborative filtering of netnews*, In Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, (1994), 175-186.
[29] F. Ricci, L. Rokach and B. Shapira, *Introduction to recommender systems handbook*, Springer, (2011), 1–35.
[30] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, *Analysis of recommendation algorithms for e-commerce*, In Proceedings of the 2nd ACM Conference on Electronic Commerce, (2000), 158–167.
[31] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, *Item-based collaborative filtering recommendation algorithms*, In Proceedings of the 10th International Conference on World Wide Web, (2001), 285-295.
[32] J. B. Schafer, J. Konstan and J. Riedl, *Recommender systems in e-commerce*, In Proceedings of the 1st ACM Conference on Electronic commerce, (1999), 158–166.
[33] G. Shani and A. Gunawardana, *Evaluating recommendation systems*, In Recommender Systems Handbook, Springer US, (2011), 257–297.
[34] U. Shardanand and P. Maes, *Social information filtering: Algorithms for automating word of mouth*, In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, (1995), 210-217.
[35] Y. Shi, M. Larson and A. Hanjalic, *Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges*, ACM Computing Surveys (CSUR), **47(1)** (2014), 3.
[36] X. Su and T. M. Khoshgoftaar, *A survey of collaborative filtering techniques*, Advances in Artificial Intelligence, (2009), 2–19.
[37] R. Yera, J. Castro and L. Martnez, *A fuzzy model for managing natural noise in recommender systems*, Applied Soft Computing, **40** (2016), 187–198.
[38] L. A. Zadeh, *Fuzzy sets*, Information and Control, **8(3)** (1965), 338–353.
[39] Z. Zhang, X. Tang and D. Chen, *Applying user-favorite-item-based similarity into slope one scheme for collaborative filtering*, Computing and Communication Technologies (WCCCT), 2014 World Congress on, (2014), 5–7.

Masoud Saeed*, School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran
E-mail address: msaeedmz@uk.ac.ir

Eghbal G. Mansoori, School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran
E-mail address: mansoori@shirazu.ac.ir

∗Corresponding Author