International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 5 (2013) pp. 524-531

Providing an Effective Collaborative Filtering Algorithm Based on Distance Measures and Neighbors' Voting

Gilda Moradi Dakhel[1], Mehregan Mahdavi[2]

[1] Faculty of Electrical, Computer & IT Engineering
Azad University, Qazvin Branch, Qazvin, Iran
gilda.moradi@gmail.com

[2] Department of Computer Science and Engineering
University of Guilan, Rasht, Iran
mehregan.mahdavi@gmail.com

***Abstract*: A recommender system produces a list of suggestions for users based on their preferences. It is an intelligent system that can help users to come across their interesting items. It is widely used in personalized web systems, social networks, e-commerce and etc. It uses data mining and information filtering techniques. Mostly, recommender systems employ the collaborative filtering algorithm that is one of the most successful techniques. The collaborative filtering creates suggestions for users based on their neighbors' preferences. This algorithm suffers from its poor accuracy and scalability. This paper represents a new approach to produce a useful recommendation for an active user. It assumes that the users are m (m is the number of users) points in n dimensional space (n is the number of items) and represents a method based on users' distance. We introduce different distance measures instead of traditional similarity measures. In addition to this, we employ clustering algorithms to improve our recommendations. We use k-means clustering algorithm to categorize users based on their interests. Then our proposed algorithm introduces a new method called voting algorithm to develop a recommendation. It is based on neighbors' opinion about unknown products of the active user. This idea is similar to what happens in real life. In the real world, if there are a lot of options for us to choose from, we use other's help and make our choices based on the suggestions of our family and friends who have got the same preferences as us. We evaluate this new idea and the result of our experiments shows that the proposed algorithm is more accurate than the traditional ones; besides it is less time consuming than previous algorithms.**

***Keywords*:** Recommender system, Collaborative filtering, Data normalization, K-means clustering, Neighbors' votes.

I. Introduction

Recommender Systems are techniques to produce suggestions of items for users. The suggestions are various decision-making algorithms, such as what movie to watch, what items to buy, what music to listen to, or what online books to read. "Item" is a general term to denote what the system suggests to users. An RS normally focuses on a specific type of digital libraries (e.g., Libraries of CDs, or movies, books). Recommender Systems typically apply methodologies from Information Retrieval and data mining. Recommender Systems try to predict what the most suitable products or services are, based on the user's interests [1].

People can share their preferences with recommender systems. A recommender system offers people the most interesting items. They are an intelligent technique to deal with the problem of information overload. They create useful suggestion for customers and make them come across their interesting items. This idea comes from the reality. In the real world if there are a lot of options for us to choose among them, we use other's help and make our choices based on the suggestions of our family and friends who have got the same preferences as us. If in the virtual world we have a great deal of option, who or what can help us? The best answer is a recommender system. A recommender system works as a friend to suggest interesting items to users. In this paper we use this idea to introduce our new algorithm. In fact, it suggests a few items from many possible choices by detecting their past behaviors. In these systems, the user interests are influenced by the hidden behaviors of the users. Finding the information about user interests is often critical step of producing recommendations. Collaborative filtering based recommender systems; the information about latent user interests is largely underexplored [2].

Recommender systems usually use collaborative filtering algorithms or a combination of the collaborative filtering and the other filtering algorithms. Collaborative filtering is usually focused on exploiting the information about the user's interaction with the systems. It is the most popular used method that matches people with similar tastes and then provides personalized recommendations on this basis. There are two basic entities in this algorithm: User and Item. Items are the objects that are suggested. Items may be characterized by their complexity and their value or utility which utilizes

users' known preference to generate predictions of the unknown preferences [1]-[3], [12]-[16]. Users are able to rate items. They use rating to show their own opinion on items. The aim is to find users who have similar tastes and suggest items that were mostly selected by these related users [3]-[5], [17], [33]. In this paper the items are typically music, movies, books, articles and any kind of existent digital contents. A rate is a numerical score or a letter grade that users may assign to the items. For example IMDB (www.imdb.com) has a user-item dataset such as Table 1. The ratings are between *1* and *10*. Each user rates multiple items between *1* and *10*. *1* for the lowest interest means user doesn't like the item, *10* for the highest interest means user likes it best and *0* for "no vote" means the user have not seen item so far. "*m*" and "*n*" denote the total number of users and items. These datasets are employed to predict some items that the active user has not seen them so far and might like them. Traditional collaborative filtering has a dataset of *m* users $\{user_1, user_2, \ldots, user_m\}$ and *n* items $\{item_1, item_2, \ldots, item_n\}$. Each $user_i$ has a set of items that he/she has rated them in different scores [3], [4], [6], [16], [29], [30]. A recommender system uses the collaborative filtering algorithm to predict unknown rates for "no vote" items and offer best of them to active user.

In section II we define the traditional collaborative filtering algorithm. Challenges of traditional collaborative filtering are presented in section III. Our proposed algorithm is introduced in section IV. In section V we represent an improvement for our proposed algorithm. Section VI shows our experimental results, and at the end we have a summary and conclusion of our paper in section VII.

Table 1. A Sample of User-Item database

	Item1	Item2	Item3	…	Itemn
User1	*0*	*4*	*3*	…	*0*
User2	*10*	*0*	*2*	…	*4*
User3	*8*	*10*	*1*	…	*10*
…	…	…	…	…	…
Userm	*0*	*4*	*10*	…	*7*

II. Collaborative Filtering

The collaborative filtering algorithm produces recommendations based on a subset of users that are called neighbors. These neighbors are the most similar users to the active user. The theory of this algorithm is each person has the similar tastes as his/her friends and relatives do. It has two main steps. At the first step it computes the similarity between the active user and all other users in database. At the second step it develops recommendations for the active user based on the first step [31]. This algorithm has two primary types, Memory-based and Model-based. Model-based algorithms use machine learning techniques like Bayesian networks, Neural networks and etc. to produce a recommendation. You can find more information about the model-based algorithms in [7], [8], [18]. In this paper we just focus on the memory-based algorithm.

Memory-based collaborative filtering algorithms employ the user-item database to generate a recommendation. Each user has a group of people with similar interests called neighbors. After finding neighbors of an active user, a prediction on no rated items for him/her can be generated. In the other hand, the algorithm computes the similarity between two users or two items *i* and *j* which is named $sim_{i,j}$; Then produces a recommendation for the active user [7], [18]. We conduct a review on its steps:

A. Similarity measuring

The main step of memory-based algorithm is to measure the similarity between two items or users. There are many different ways to calculate the similarity between users or items. In this paper we review these three methods: Pearson Correlation (1), Cosine-based similarity (2), and Adjusted Cosine-based similarity (3). Also there are two kind of memory-based algorithm. If the algorithm computes the similarity between two items *i* and *j* ($sim_{i,j}$), then it is called item-based collaborative filtering and if it computes the similarity between two users *u* and *v* ($sim_{u,v}$) called user-based collaborative filtering algorithm [7].

$$sim_{i,j} = \frac{\sum_{m\in(i\cap j)}\left(r_{i,m}-\bar{r_i}\right)\left(r_{j,m}-\bar{r_j}\right)}{\sqrt{\sum_{m\in(i\cap j)}\left(r_{i,m}-\bar{r_i}\right)^2}\sqrt{\sum_{m\in(i\cap j)}\left(r_{j,m}-\bar{r_j}\right)^2}} \quad (1)$$

$$sim_{i,j} = \frac{\sum_{m\in(i\cap j)} r_{i,m} r_{j,m}}{\sqrt{\sum_{m\in(i\cap j)} r_{i,m}^2 r_{j,m}^2}} \quad (2)$$

$$sim_{i,j} = \frac{\sum_{m\in(i\cap j)}\left(r_{i,m}-\bar{r_i}\right)\left(r_{j,m}-\bar{r_j}\right)}{\sqrt{\sum_{m\in(i\cap j)}\left(r_{i,m}-\bar{r_i}\right)^2 \sum_{m\in(i\cap j)}\left(r_{j,m}-\bar{r_j}\right)^2}} \quad (3)$$

In these formulas $sim_{i,j}$ introduces the correlation or similarity between two users or items *i* and *j*. In the item-based collaborative filtering, *i* and *j* are two items and $i \cap j$ is the set of co-rated users includes some of the users who rated both items *i* and *j*. $\bar{r_i}$ is the average rating of the *i*th item by co-rated users. $r_{i,m}$is the rating of user *i* on item *m*.

For the user-based collaborative filtering, *i* and *j* are two users and $i \cap j$ is the set of co-rated items which means the items that both users *i* and *j* have rated them. $\bar{r_i}$ is the average rating of *i*th user on the co-rated items ($i \cap j$) [3]-[8], [10], [17]-[20], [29]-[33].

B. Selecting Neighbors

The number of neighbors is an experimental value which you have to find. If you select a large number of users as neighbors, you get more accuracy but it takes more time to generate recommendations. So, you have to choose between accuracy and running time [7], [18], [20].

There are two techniques for selecting neighbors. One of them is Threshold-based selection and the other one is the top-*N* selection. According to Threshold-based selection, we choose

those users whose similarity exceeds a certain threshold value as neighbors. But in the top-*N* technique, *N* is an input and it means we select *N*-most similar neighbors [6], [18].

C. Prediction

User-based algorithm develops a prediction for the active user u, on an item i (4):

$$prediction_{u,i} = \frac{\sum_{n \in Neighbors} \left(r_{n,i} - \overline{r_n}\right) sim_{u,n}}{\sum_{n \in Neighbors} \left|sim_{u,n}\right|} + \overline{r_u} \tag{4}$$

Where Neighbors are selected in previous section (see section 2.2.). $\overline{r_u}$ and $\overline{r_n}$ are the average ratings for the user u and user n on all other rated items (all rated items except i). $sim_{u,n}$ is the similarity of the active user u and the neighbor user i. It is already calculated (see section 2.1.). For the item-based collaborative filtering we have another way to produce a prediction (5).

$$prediction_{u,i} = \frac{\sum_{n \in Neighbors} r_{u,n} sim_{i,n}}{\sum_{n \in Neighbors} \left|sim_{i,n}\right|} \tag{5}$$

$sim_{u,n}$ is the similarity between the target item *i* and the neighbor item *n* [6], [7], [18].

III. Challenges of the Collaborative Filtering

The challenges of collaborative filtering are widely discussed in many related papers. So we indicate a brief of their main content. Even though the collaborative filtering has been successfully used in recommender systems, there still remain some problems that commonly identified as the cold-start problem, the scalability problem and the sparsity problem [21], [22].

A. The Cold-Start Problem

Collaborative filtering cannot provide personalized recommendations until a user profile has been created. This is known as the cold-start problem. Several algorithms try to learn and create the new users' profiles as part of the sign up process by asking them to provide feedback [10], [21].

The cold start problem occurs when a new user has just signed up the system. It is difficult to find similar users and items because there is not enough information about his/her rating or purchase history. And as the same for new item, new items cannot be recommended until some users vote them. This problem reduces the accuracy of a recommendation system which relies on comparing users. This problem is also called the new user problem or new item problem [7]. So, collaborative filtering cannot produce recommendation for new users, because there is not any rating of them, and for the same reason, new items cannot be recommended, either [22], [27].

In the paper [23] they propose a novel efficiently association clusters filtering (ACF) algorithm. They use clustering and also filtering. ACF algorithm determines clusters models based on the ratings database. They say the users in the same cluster will have the same interests. They can use the opinion of the cluster to guess the unknown ratings. They said this approach increases the prediction scope and improves the accuracy. Also this research [24] presents a solution to the cold-start problem. They introduced a collaborative filtering recommendation algorithm based on the implicit information of the new users and multi-attribute rating matrix. They combined implicit information of new users with other rating information to produce a User-Item Rating Matrix (UIRM). They said their experiment resulted validate the feasibility of the proposed algorithm.

Paper [10] presents a new method that uses the idea of pairwise comparison between items. It uses a lazy decision tree that compares pairwise at the decision nodes. Based on the user's response to a certain comparison, it selects what pairwise comparison should next be asked. Their results indicate that the pairwise approach provides more accurate recommendations and requires less effort when signing up newcomers. There are a lot of works to solve the cold-start problem like [18], [23]-[27].

B. The Scalability Problem

Collaborative Filtering needs a lot of heavy computations. They grow nonlinearly with increasing the number of users and items. In the other hand, Collaborative filtering fails with the growth of number of users and items [31]. Calculating the similarity takes most of the computational time. If we have *U* numbers of users and *I* numbers of items, then the time complexity becomes *O(UxI)*. This problem is called as scalability problem [20]. A solution for this problem is to run the time-consuming training step offline, and then the online prediction producing will take a much shorter time [7].

C. The Sparsity Problem

The recommendation systems' dataset is very large (a *UxI* matrix). Even users that are very active, they have seen just a few of items available in the database, also plenty of items have been rated by only a few of the total number of users available in the database. This problem is named as the sparsity problem. It has a negative impact on the result of a collaborative filtering algorithm. Because of this problem, it is possible that the similarity between two users cannot be defined. Also when the value of similarity is calculated, because of insufficient information it is not very reliable [31].

IV. New Collaborative Filtering Algorithm

Traditional collaborative filtering uses similarity measuring to find neighbors. But in this paper, first we normalize database, then we use distance measuring instead of similarity measuring. At the end we represent a new method to produce predictions based on neighbors' opinion. The rest of this section includes steps of our proposed algorithm.

A. Data Normalization

At first we should normalize the database. The User-Item dataset is not smooth and the distribution of rating is not uniform; the minority of items has much more rating chances, while the majority of the items have much less rating chances. The ratings should be normalized based on different preferences. It leads to more accuracy according to the experiments. The normalization is introduced as follows (see Formula (6):

$$r_{u,i}^{norm} = \frac{r_{u,i} - mean_i}{var_i} \quad (6)$$

where $r_{u,i}$ is the rating of user u on item i. $mean_i$ denotes the mean of rating values on item i. var_i is the variance of rating values on item i [25]. In this paper $r_{u,i}$ means the real rating user u on item i, and $r_{u,i}^{norm}$ means the normalized rating of user u on item i.

B. Distance measuring

The normalized database is used by this section. Our approach assumes all instances of database correspond to points in the n-dimensional space I^n.

$User1 = (r^{norm}1_1, r^{norm}1_2, \ldots, r^{norm}1_n)$

$User2 = (r^{norm}2_1, r^{norm}2_2, \ldots, r^{norm}2_n)$

...

$Userm = (r^{norm}m_1, r^{norm}m_2, \ldots, r^{norm}m_n)$

and

$Item1 = (r^{norm}1_1, r^{norm}2_1, \ldots, r^{norm}m_1)$

$Item2 = (r^{norm}1_2, r^{norm}2_2, \ldots, r^{norm}m_2)$

...

$Itemn = (r^{norm}1_n, r^{norm}2_n, \ldots, r^{norm}m_n)$

where the $r^{norm}i_k$ in the *Useri and* the $r^{norm}i_k$ in the *Itemk* denotes the normalized rating of user i on k^{th} item. The distance between *Useri* and *Userj*, or between *Itemi* and *Itemj* is given by different distance measures in Table. 2 [18], [9], [11].

Distance is known as a quantitative value of how far apart two points are. It is a scientific and mathematical definition of distance. The opposite of distance is Similarity, or on the other hand the synonym of distance is Dissimilarity. This paper denotes a distance as *d* and a similarity measure as *sim*. Notice a high similarity leads to a low distance. In distance based method we suppose all users correspond to points in the *n*-dimensional space. The nearest neighbors of a user will be defined by a distance measure.

The main distance measures listed in Table 2. Minkowski distance of order γ is shown in it. The Minkowski distance is introduced as Euclidean distance When $\gamma=2$, as City block distance when $\gamma=1$, and as Chebyshev distance when $\gamma=\infty$. Minkowski is a general form of Euclidean, City block and Chebyshev distances. Sørensen, Gower, Canberra and Lorentzian distances are given in Tab. 2 [9].

C. Selecting Neighbors

As we said the traditional collaborative filtering uses similarity measuring. But in this paper, we use distance measuring of instead of similarity (6) [34]. You can select any one of distance measures in Table 2. The defined distance measures compute the distance between *Useri* and *Userj*, that they are listed in Table. 2. These measures are used instead of the users' similarity measures in the section 2.1.

According to the section 2.2 there is a top-N selection method for selecting neighbors. In addition to this selection method, we introduce a clustering method that uses K-means algorithm to cluster dataset and select nearest neighbors. Clustering is a method to make several groups of similar data. It divides data into groups or "clusters" such that similar points are in same cluster and dissimilar points are in different clusters. In this paper we use k-means clustering that is the most important algorithm in data mining [28]. Also we employ a distance measure of Tab. 2 instead of similarity formulas which were used in collaborative filtering. Following pseudo code in Fig. 1 shows the base of this clustering.

Input: Dataset *User-Item Rating Matrix*, number of clusters *k*
Output: Set of cluster centers *C*, cluster membership vector *S*
-Initialize cluster centers *C*
-Do{
// Data Assignment
- Find closet cluster center for each user in User-Item Dataset using a distance measure and update *S* such that s_i is cluster ID of $user_i$.

// Relocation of centers
- Update *C* such that c_i is mean of users in *i*th cluster

While (C doesn't change)

Figure 1. K-means Algorithm

The input of this algorithm is the rating dataset, and the output will be a set of cluster centers and cluster memberships. *k* users from User-Item Dataset randomly are selected and assigned these k users as initial set of cluster centers C [9], [18], [30].

D. Producing Predictions

In this step we have already had *k* clusters such that each cluster includes similar users as neighbors. Now, we want to compute the unknown scores of the active user to the items. At first we introduce discrete-valued target function of the form $f: I^n \rightarrow R$ (7) where R is the finite set of rating range. For example if the rating range is between *1* and *5*, then $R = \{1, 2, 3, 4, 5\}$. Function *f* denotes the rating of each user on items.

(7)

$$f_{u,i} = \arg\max_{r \in R} \sum_{n \in Neighbors} \delta(r, f_{n,i}),$$

$$\delta(r, f_{n,i}) = \begin{cases} 1 & if\left(r = f_{n,i}\right) \\ 0 & otehrwise \end{cases}$$

where *Neighbors* are all user in a cluster that includes *u*, and $f_{n,i}$ is the rating of $user_n$ on $item_i$. The value $f_{u,i}$ is the result of this algorithm and it is the most common value of *f* among the *Neighbors*. If there is more than one maximum value for $f_{u,i}$, then you can use the mean value of maximum values [11].

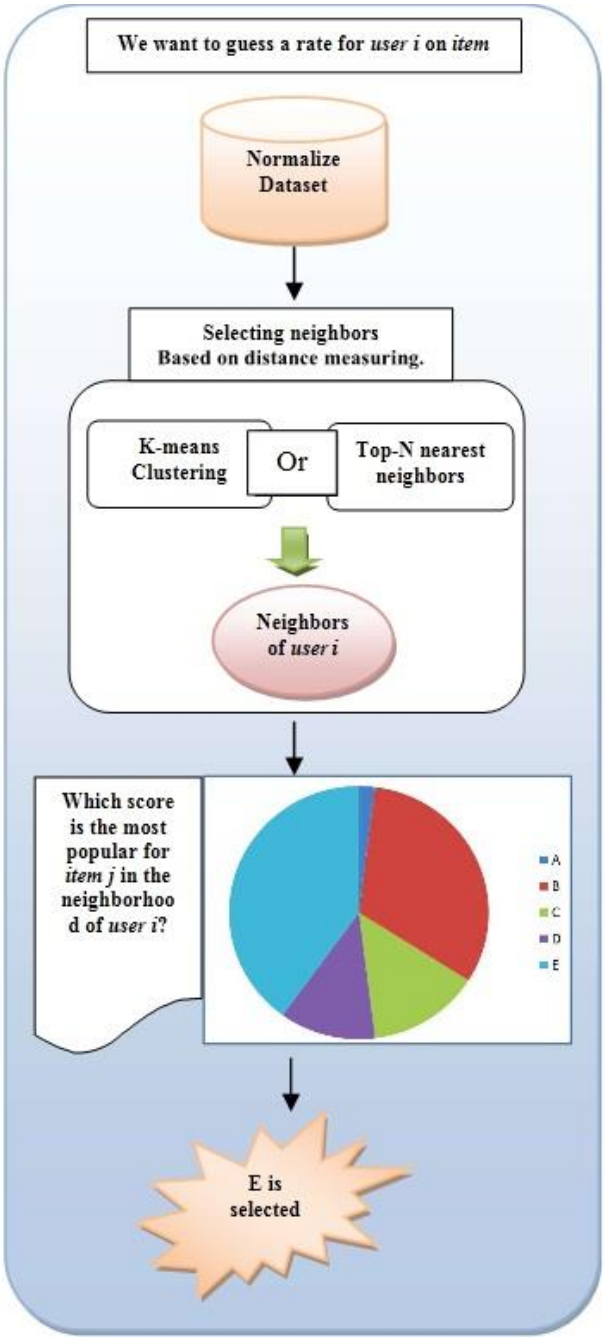

Figure 2. Proposed Algorithm structure

We have illustrated the structure of our proposed algorithm with an example. The rating range is between *A* and *E*. Fig. 2 shows a summary of our algorithm. In this example we want to predict a score for user *i* on item *j*. At first we normalize our dataset. Then we should find neighbors of user *i*. we have two options, we can select top-*N* nearest neighbors or we can cluster our dataset to *k* clusters. Finally we use voting to find prediction. Fig. 2 shows rate E is our prediction.

V. Distance Weighted Algorithm

We introduce a useful refinement to this algorithm. It assigns a weight to each of the neighbors according to their distance to the active user. It gives greater weight to nearest neighbors. This can be developed by replacing $f_{u,i}$ in the first line of last formula (7) by this following formula:

(8)

$$f_{u,i} = \arg\max_{r \in R} \sum_{n \in Neighbors} \lambda_n \delta(r, f_{n,i})$$

where

(9)

$$\lambda_n = \frac{1}{d(user_u, user_n)^2}$$

In this idea, the most popular score of neighbors' to the target item rating in a cluster will be the prediction for the unknown score [11].

Table 2. Definition of Distance Measures

Measures	Formulas
Minkowski	$d_{MK} = \sqrt[p]{\sum_{k=1}^{n} \left(\left| ri_k^{norm} - rj_k^{norm} \right| \right)^p}$
City block	$d_{CB} = \sum_{k=1}^{n} \left| ri_k^{norm} - rj_k^{norm} \right|$
Euclidean	$d_{Euc} = \sqrt{\sum_{k=1}^{n} \left(ri_k^{norm} - rj_k^{norm} \right)^2}$
Chebyshev	$d_{Cheb} = \sqrt[\infty]{\sum_{k=1}^{n} \left(\left| ri_k^{norm} - rj_k^{norm} \right| \right)^\infty}$
Sørensen	$d_{Sor} = \frac{\sum_{k=1}^{n} \left| ri_k^{norm} - rj_k^{norm} \right|}{\sum_{k=1}^{n} \left(ri_k^{norm} + rj_k^{norm} \right)}$
Gower	$d_{Gow} = \frac{1}{n} \sum_{k=1}^{n} \left| ri_k^{norm} - rj_k^{norm} \right|$

Canberra	$d_{Can} = \sum_{k=1}^{n} \frac{\left\| ri_k^{norm} - rj_k^{norm} \right\|}{ri_k^{norm} + rj_k^{norm}}$
Lorentzian	$d_{Lor} = \sum_{k=1}^{n} \ln\left(1 + \left\| ri_k^{norm} - rj_k^{norm} \right\|\right)$

VI. Experimental Result

All our experiments were performed using *MATLAB*. We ran the program on a Windows based NoteBook with *IntelCore2Duo* processor having a speed of *2.66GHz* and *4GB* of RAM.We implemented the traditional collaborative filtering algorithm with the three types of similarity measures. We implemented the new proposed algorithm as well. We compared their accuracy based on the size of neighborhood and the time of process.

A. Dataset

We use MovieLens collaborative filtering data set: *(http://www.grouplens.org/)* to evaluate the performance of our proposed algorithm. *943* users rated on *1682* movies and every user has at least *20* ratings. The users rated the movies between *1* and *5*. We divided User-Item rating dataset into *80%* of the training set and *20%* of the test set. Also we use *5*-fold cross validation for our results.

B. Metrics

The most widely used metric in collaborative filtering researches is Mean Absolute Error (MAE), which calculates the average of the absolute difference between the predictions and real ratings (see Formula 10). Various recommender systems use different numerical rating scales. Normalized Mean Absolute Error (NMAE) normalizes MAE to express errors as percentages of full scale (see Formula 11). NMAE is a metric to measure the accuracy of recommender algorithms, too. We use this metric to compare our proposed approach with other collaborative filtering methods.

(10)

$$MAE = \frac{\sum_{(u,i)\in test} \left| prediction_{u,i} - real_{u,i} \right|}{n_{test}}$$

Where n is the total number of ratings-prediction pairs, $prediction_{u,i}$ is the predicted rating for user u on item i, and $real_{u,i}$ is the actual rating. The lower MAE denotes more accurate and better prediction.

(11)

$$NMAE = \frac{MAE}{r_{max} - r_{min}}$$

Where r_{max} and r_{min} are the upper and lower bounds of the ratings. In the other hand NMAE is normalized form of MAE [7], [9], [28].

C. Compare

In this section we represent our result and compare the proposed collaborative filtering algorithm with the traditional collaborative filtering. These results are mainly divided into two parts: accuracy result and performance result. We compare the algorithms with the sensitivity of size of neighborhood. The size of neighborhood has considerable impact on the recommendation quality. We evaluated our approach with NMAE to compare.

D. Accuracy

We compare the accuracy of our proposed algorithm with the basic collaborative filtering. The results in Tab. 3, Tab. 4 , Fig. 3 and Fig. 4 show our approach is more accurate (approximately 60%) than the basic collaborative filtering. They show our proposed collaborative filtering has a satisfactory quality of recommendation with various sizes of *k* for clustering and *N* for top-*N* nearest neighbors. The NMAE for user-based and item-based collaborative filtering is around 85%, but it is around 25% in our experiments. And it is a great deal of change.

Table 3. Traditional Collaborative Filtering

	NMAE	Time(sec)
User-based CF	***0.84***	***15030***
Item-based CF	***0.89***	***386***

Table 4. Proposed Algorithm using Clustering

	NMAE	Time(sec)
Proposed Algorithm-Euclidean distance	***0.21***	***161***
Proposed Algorithm-Cityblock distance	***0.22***	***1186***
Proposed Algorithm-Chebyshev distance	***0.29***	***385***
Proposed Algorithm-Sorencen distance	***0.26***	***311***
Proposed Algorithm-Gower distance	***0.25***	***421***
Proposed Algorithm-Canberra distance	***0.28***	***904***
Proposed Algorithm-Lorentzian distance	***0.29***	***1000***

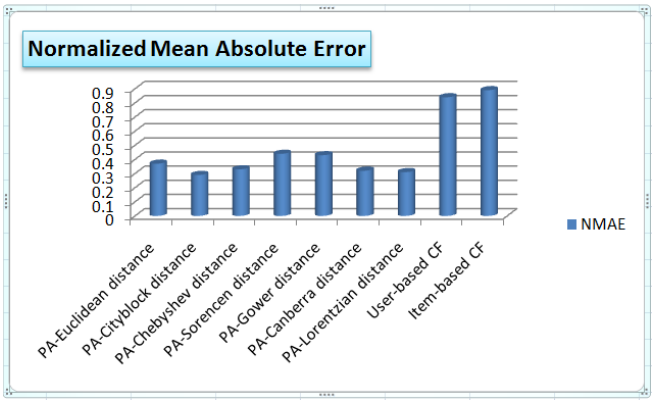

Figure 3. Accuracy comparison-(using top-N nearest neighbors for proposed algorithm)

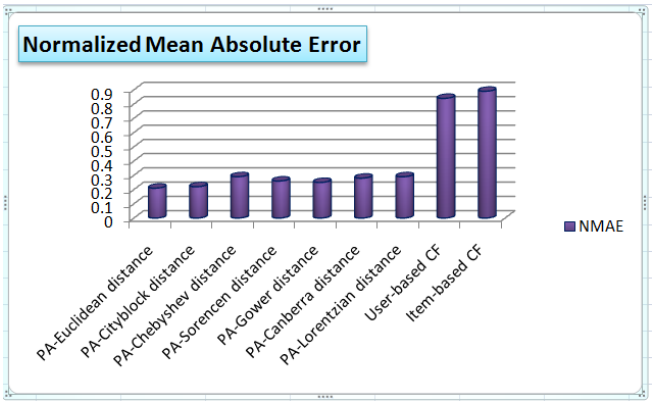

Figure 4. Accuracy comparison-(using clustering for proposed algorithm)

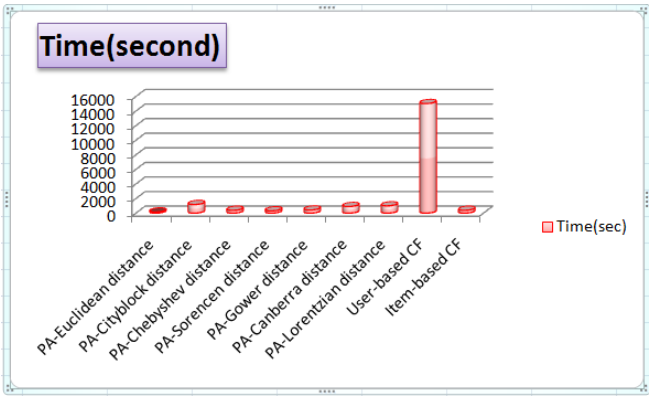

Figure 6. Time comparison-(using clustering for proposed algorithm)

E. Performance

In this part we want to compare the scalability of the algorithms. As we discussed the collaborative filtering algorithm suffers from the scalability problem. With the growth of user-Item dataset, the time of generating recommendations increases especially in user-based algorithm. Our approach is less time consuming than traditional user-based collaborative filtering but it does not have major difference with item-based collaborative filtering. Our proposed algorithm with different distance measures is about one tenth of the traditional user-based running time. We performed this experiment with various sizes of neighborhood and *k* (see Fig. 4 and 6).

Table 5. Proposed algorithm using top-N nearest neighbors

	NMAE	Time(sec)
Proposed Algorithm-Euclidean distance	***0.37***	***560***
Proposed Algorithm-Cityblock distance	***0.29***	***891***
Proposed Algorithm-Chebyshev distance	***0.33***	***702***
Proposed Algorithm-Sorencen distance	***0.44***	***531***
Proposed Algorithm-Gower distance	***0.43***	***654***
Proposed Algorithm-Canberra distance	***0.32***	***1092***
Proposed Algorithm-Lorentzian distance	***0.31***	***1231***

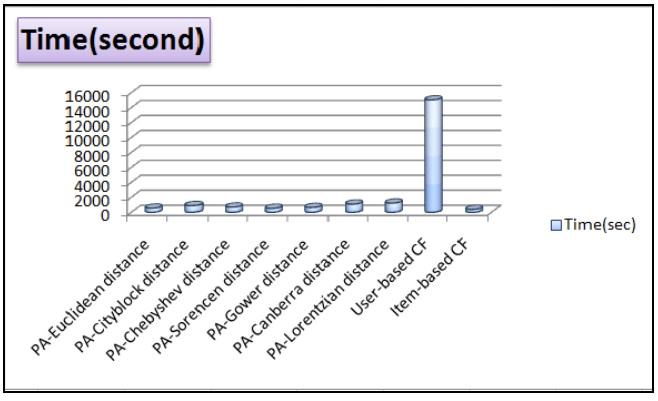

Figure 5. Time comparison-(using top-N nearest neighbors for proposed algorithm)

VII. Conclusion

A recommender system is presented as a Web-based application that is known for its usage on e-commerce personalized Web sites, with the purpose of helping customers in the decision making and product selection process by providing a list of recommended items. Recommender systems employ information filtering algorithms to predict items. The most successful algorithm in this field is Collaborative Filtering. Even though this algorithm is the best, it suffers from poor accuracy and high running time. To solve these problems this paper proposed a personalized recommendation approach based on distance measure. We use k-means clustering algorithm for finding neighbors. At the end we produce recommendations based on users' voting and opinion. This research considers the users are *m* points in *n*-dimensional space. The distance measures calculate the distance between two users. We implemented all algorithms with MovieLens dataset and then we compared the results. The results showed our approach is more accurate and more time-consuming than the traditional algorithm.

References

[1] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor, "Recommender Systems Handbook", Springer, 1st Edition, 2011.

[2] L. Qi, Ch. Enhong, X. Hui, C.H.Q. Ding, CH. Jian, "Enhancing Collaborative Filtering by User Interest Expansion via Personalized Ranking" Journal of Systems, Man, and Cybernetics, Part B: Cybernetics, pp. 218 – 233, 2012.

[3] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, E. Duval, "Context-Aware Recommender Systems for Learning: A Survey and Future Challenges", *IEEE Transaction on Learning Technologies*, pp. 1 - 1, 2012.

[4] P. Symeonidis, A. Nanopoulos, Y. Manolopoulos, "Providing justifications in recommender systems", *IEEE Transaction on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, pp. 1262 – 1272, 2008.

[5] J. Zhan, CH. Hsieh, I. Wang, T. Hsu, CH. Liau, D. Wang, "Privacy-preserving collaborative recommender systems", *IEEE Transaction on* Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 40, pp. 472 – 476, 2010.

[6] S. Gong, "A collaborative filtering recommendation algorithm based on user clustering and item clustering", Journal of Software, pp. 745 – 752, 2010.

[7] X. Su, M. Khoshgoftar, "A survey of collaborative filtering techniques", *Journal of Advances in Artificial Intelligence*, 2009.

[8] G. Adomavicius, A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions", *Journal of Knowledge and Data Engineering*, vol. 17, pp. 734 – 749, 2005.

[9] S. H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions", *Journal of Mathematical Models and Methods in applied sciences*, vol. 1, issue 4, pp. 302 – 307, 2007.

[10] L. Rokach, S. Kisilevich, "Initial Profile Generation in Recommender Systems Using Pairwise Comparison", *IEEE Transaction on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, June. 2012, pp. 1 – 6, 2012.

[11] T. M. Mitchell, "Machine learning", *Chapter 08, Instance-based learning, McGraw-Hill Science/Engineering/Math*, pp. 230 - 247, 1997.

[12] O.H. Embarak, "A method for solving the cold start problem in recommendation systems", *In Proceedings of the International Conference on Innovations in Information Technology*, pp. 238 – 243, 2011.

[13] S. Dongting, L. Zhigang, ZH. Fuhai, "A novel approach for collaborative filtering to alleviate the new item cold-start problem", *In Proceedings of the 11th International Symposium on Communications and Information Technologies*, pp. 402 - 406, 2011.

[14] CH. Huayue, "Personalized Learning Resources Recommendation Model Based on Transfer Learning", *In Proceedings of the International Conference on Computer Science and Electronics Engineering*, vol. 2, pp. 14 – 16, 2012.

[15] D. Bouneffouf, A. Bouzeghoub, A.L. Gancarski, "Following the User's Interests in Mobile Context-Aware Recommender Systems: The Hybrid-e-greedy Algorithm", *In Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops*, pp. 657 - 662, 2012.

[16] A. Aysha, A. KaziMasudul, K. Heung-Nam, S. Abdulmotaleb El, "Social network and user context assisted personalization for recommender systems", *In Proceedings of the International Conference on Innovations in Information Technology*, pp. 95 - 100, 2012.

[17] E. Vozalis, K. G. Margaritis, "Analysis of recommender Systems' algorithms", *In Proceedings of the 6th Hellenic European Conference on Computer Mathematics & its Applications*, Athens, Greece, 2003.

[18] G. Moradi Dakhel, M. Mahdavi, "A New Collaborative Filtering Algorithm Using K-means Clustering and Neighbors' Voting", *In Proceedings of the* 11th International Conference on Hybrid Intelligent Systems, Melacca, vol. 1, pp. 179 – 184, 2011.

[19] B. M. Sarwar, G. Karypis, J. Konstan, J. Riedl, "Item-based collaborative filtering recommendation algorithms", *In Proceedings of the The 10th international conference on World Wide Web*, pp. 285-295, 2001.

[20] M. K. Kavitha Devi, P. Venkatesh, "An improved collaborative recommender system", *In Proceedings of the 1st International Conference on Networks & Communications*, pp. 386-391, 2009.

[21] F. Ullah, G. Sarwar, L. Sung Chang, P. Yun Kyung, M. Kyeong Deok, K. Jin Tae, "Hybrid recommender system with temporal information", *In Proceedings of the International Conference on Information Networking*, pp. 421 – 425, 2012.

[22] Y. Gao, H. Qi, J. Liu, D. Liu, "A recommendation algorithm combining user grade-based collaborative filtering and probabilistic relational models", *In Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery*, Aug. 2007, pp. 67-71, 2007.

[23] CH. Huang, J. Yin, "Effective association clusters filtering to cold start recommendations", *In Proceedings of the 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'10),* , pp. 2461 - 2464, 2010.

[24] Y. Hang, CH. Guiran, W. Xingwei, "A cold-start recommendation algorithm based on new user's implicit information and multi-attribute rating matrix", *In Proceedings of the 9th International Conference on Hybrid Intelligent Systems (HIS'09)*, pp. 353 -358, 2009.

[25] J. Liu, G. Deng, "A new-user cold-starting recommendation algorithm based on normalization of preference", *In Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1 - 4, 2008.

[26] A. I. Schein, A. Popescul, L. H. Ungar and D. M. Pennock, "Methods and metrics for cold-start recommendations*", In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 253 - 260, 2002.

[27] L. T. Weng, Y. Xu, Y. Li, R. Nayak, "Exploiting item taxonomy for cold-start problem in recommendation making", *In Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence*, pp. 113 - 120, 2008.

[28] J. Ghosh, A. Liu, "The to ten algorithms in data mining", *Chapter 02, k-means, by Taylor & Francis Group, LLC*, 2009.

[29] SH. Narayanaswamy, "A concept-based framework and algorithms for recommender systems", *Master Thesis, Document No. ucin1186165016. , Department of Computer Science, University of Cincinnati*, 2007.

[30] Y. Qu, X. Yang, T. Huang, "Survey of recommendation systems and algorithms", *EE 380L: Data Mining*, 2000.

[31] M. Papagelis, "Crawling the algorithmic foundations of recommendation", *Master Thesis, Computer Science Department, School of Sciences and Engineering, University of Crete, Heraklion*, 2005.

[32] M. Balabanovic, Y. Shoham, "Fab: content-based, collaborative recommendation", *Magazine of Communications of the ACM*, vol. 40, pp. 66 - 72, 1997.

[33] L. Terveen, W. Hill, "Beyond recommender systems: helping people help each other", *HCI in the New Millennium*, Jack Carroll, ed., Addison-Wesley, 2001.

[34] K. Teknomo, "Similarity Measurement", http:\\people.revoledu.com\kardi\ tutorial\Similarity\.

Biography

Gilda Moradi Dakhel graduated in "Computer Engineering - Software" from Azad University of Lahijan in 2008. She started his postgraduate study in "Computer Engineering - Software" at Azad University of Qazvin in 2009. Her current research includes recommender systems. She worked at Mechatronic Research Laboratory (MRL) in Azad University of Qazvin from March 2010 to May 2011 as Analyst, Designer and Developer. She has been a member of a RoboCup Rescue Simulation Agent League, which achieved 3rd place in the International Iran Open RoboCup Competition 2011 in Rescue Simulation League. She is currently working as a software engineer.

Mehregan Mahdavi received his PhD from the School of Computer Science and Engineering, University of New South Wales, in 2006. He received his MS degree in Software Engineering from Amirkabir University of Technology in 1997 and his BS degree in Software Engineering from Ferdowsi University of Mashad in 1993. He has been an Assistant Professor in the Department of Computer Science and Engineering, University of Guilan since 2006. He also worked as a Research Academic Level B for Macquarie E-Learning Centre of Excellence in 2005-2006. He has researched a range of topics in the area of Web and Database including Web Caching, Web Portals, Web Services, Grid Computing, and Web Data Integration. His research area also includes Identity and Access Management, Electronic Commerce, Recommender Systems, Electronic Learning and Human Computer Interaction. He has been a member of the IEEE since 2004.