A New Collaborative Filtering Algorithm Using K-means Clustering and Neighbors' Voting

Gilda Moradi Dakhel
Faculty of Electrical, Computer & IT Engineering
Azad University, Qazvin Branch
Qazvin, Iran
gilda.moradi@gmail.com

Mehregan Mahdavi
Department of Computer Science and Engineering
University of Guilan
Rasht, Iran
mehregan.mahdavi@gmail.com

***Abstract*— The Collaborative Filtering is the most successful algorithm in the recommender systems' field. A recommender system is an intelligent system can help users to come across interesting items. It uses data mining and information filtering techniques. The collaborative filtering creates suggestions for users based on their neighbors' preferences. But it suffers from its poor accuracy and scalability. This paper considers the users are m (m is the number of users) points in n dimensional space (n is the number of items) and represents an approach based on user clustering to produce a recommendation for active user by a new method. It uses k-means clustering algorithm to categorize users based on their interests. Then it uses a new method called voting algorithm to develop a recommendation. We evaluate the traditional collaborative filtering and the new one to compare them. Our results show the proposed algorithm is more accurate than the traditional one, besides it is less time consuming than it.**

Keywords- collaborative filtering; recommender systems; minkowski distance; clustering; k-means; neighbors' voting

I. INTRODUCTION

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 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. Recommender systems enable people to share their preferences, and then they offer people the most interesting items. They are an intelligent technique to deal with the problem of information overload. They can create useful suggestion for customers and make them come across their interesting items. Collaborative Filtering (CF) is the most successful method that matches people with similar tastes and then provides personalized recommendations on this basis. There are two basics entities in this algorithm: User and Item. Users use rating to show their own opinion on items [1, 2, 3]. Recommender systems usually use collaborative filtering algorithms or a combination of the collaborative filtering and the other filtering algorithms. The aim is to find users who have similar tastes and suggest items that were mostly selected by these related users [4, 5, 6].

TABLE I. A SAMPLE OF USER-ITEM DATABASE

	$Item_1$	**$Item_2$**	**$Item_3$**	**…**	**$Item_n$**
$User_1$	0	4	3	…	0
$User_2$	5	0	2	…	4
$User_3$	4	5	1	…	5
…	…	…	…	…	…
$User_m$	0	4	5	…	3

In this context a user rates items. The items are typically movies, books, articles, music and all kinds of existent digital libraries. 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 database such as Tab .1. The ratings are between *1* and *5*. Each user rates multiple items between *1* and *5*. *1* for the lowest interest means user doesn't like the item, *5* 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 [7, 8, 9]. These datasets are employed to predict some items that the active user has not seen them 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 [10]. A recommender system uses the collaborative filtering algorithm to predict unknown rates for "no vote" items and recommends 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 show our experimental results, and at the end we have a summary and conclusion of our paper in section VI.

II. COLLABORATIVE FILTERING

The collaborative algorithm works based on similar users. The theory of this algorithm is each person has similar tastes as his/her friends and relatives do. It produces recommendations based on a subset of users that are called neighbors. These neighbors are the most similar users to the active user. The algorithm has two main steps. At the first

978-1-4577-2152-6/11/$26.00 ©2011 IEEE

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 [11].

This algorithm has two main types, Memory-based and Model-based. Model-based algorithms use machine learning techniques like Bayesian networks, Neural networks and etc. You can find more information about the model-based algorithms in [10, 12]. In this paper we just focus on the memory-based algorithms.

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 the active user, a prediction on no rated items for him/her can be generated. On 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 [10].

A. Measuring the Similarity of Users and Items

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 kinds of memory-based algorithm. If the algorithm computes the similarity between two items *i* and *j* ($sim_{i,j}$) it is called item-based collaborative filtering and if it computes the similarity between two users *u* and *v* ($sim_{u,v}$) it is called user-based collaborative filtering algorithm [10].

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

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

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

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 a set of co-rated users includes some users who rated both items *i* and *j*. $\overline{r_i}$ 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 a set of co-rated items which means items that both users *i* and *j* have rated them. $\overline{r_i}$ is the average rating of *i*th user on co-rated items ($i \cap j$) [8, 10, 12, 13, 14].

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 [10 ,15].

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 the threshold-based selection, we choose 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 [7].

C. Prediction Computation

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

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

where *Neighbors* are selected in previous section (see section II.B). $\overline{r_u}$ 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 II.A).

For the item-based collaborative filtering we have another way to produce a prediction (5).

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

$sim_{u,n}$ is the similarity between the target item *i* and the neighbor item *n* [7, 10].

III. CHALLENGES OF THE COLLABORATIVE FILTERING

Even though the collaborative filtering has been successfully used in recommender systems, there still remain some problems that commonly identified as Scalability problem, Sparsity problem and Cold-start problem [14].

A. The Scalability Problem

The collaborative filtering needs a lot of heavy computations. They grow nonlinearly with increasing the number of users and items. On the other hand, the collaborative filtering fails with the growth of number of users and items [11]. Calculating similarity takes most of the computational time. If we have *U* numbers of users and *I* numbers of items the time complexity becomes *O(UxI)*. This

problem is called as the scalability problem [15]. 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 [10].

B. The Sparsity Problem

The recommendation systems' dataset is very large (an *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 negative impact on the result of the 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 [11].

C. The Cold-Start Problem

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 [10]. 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 [16].

In the paper [17] they have proposed a novel efficiently association clusters filtering (ACF) algorithm. They used clustering and also filtering. ACF algorithm determines clusters models based on the ratings database. They say users in the same cluster will have the same interests. They use opinion of a cluster to guess unknown ratings. They said this approach increases the prediction scope and improves the accuracy.

Also this research [18] has represented 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.

There is a lot of work to solve the cold-start problem like [17, 18, 19, 20, 21].

IV. NEW COLLABORATIVE FILTERING ALGORITHM BASED ON USER CLUSTERING AND VOTING

In this section we represent our proposed algorithm based on user clustering and neighbors' voting. 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 [20]. In the rest of this section we will explain our algorithm that uses k-means clustering.

Traditional collaborative filtering uses similarity measuring. But in this paper, we use Minkowski Distance of order γ instead of similarity (6) [21]. In this method we suppose all users correspond to points in the *n*-dimensional space. The nearest neighbors of a user that are in a cluster will be defined by Minkowski distance.

$User1 = (r1_1, r1_2, \ldots, r1_n)$

$User2 = (r2_1, r2_2, \ldots, r2_n)$

...

$Userm = (rm_1, rm_2, \ldots, rm_n)$

where ri_k denotes the normalized rating of user i on k^{th} item.

$$d\left(user_i, user_j\right) = \sqrt[\gamma]{\sum_{k=1}^{n}\left(ri_k - rj_k\right)^{\gamma}} \quad (6)$$

This is Minkowski distance. It is introduced as Euclidean distance When γ=2, it is City block distance when γ=1 [23, 24, 25].

A. User Clustering

We use k-means clustering algorithm to cluster our dataset. Also we employ Minkowski distance (6) 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 Minkowski distance 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. Algorithm K-means

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 [7, 22].

B. 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.

$$f_{u,i} = \arg\max_{r \in R} \sum_{n \in Neighbors} \delta(r, f_{n,i}) . \quad (7)$$

$$\delta(r, f_{n,i}) = \begin{cases} 1 & if\left(r = f_{n,i}\right) \\ 0 & otehrwise \end{cases} . \quad (8)$$

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 [24]. We have illustrated the operation of the algorithm with an example. The rating range is between A and E. Fig. 2, Fig. 3 and Fig. 4 show a summary of our algorithm.

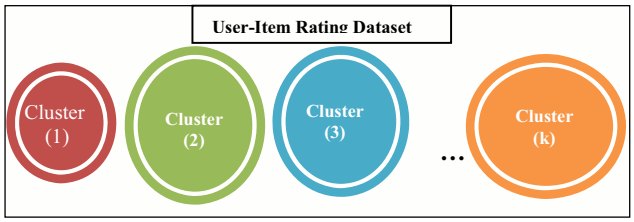

Figure 2. User clustering

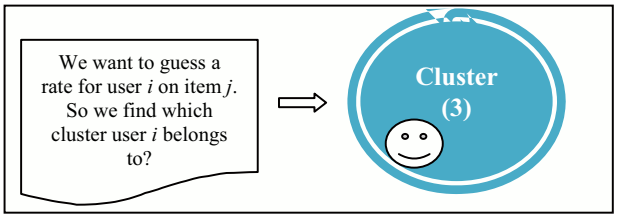

Figure 4. Finding neighbors of user i

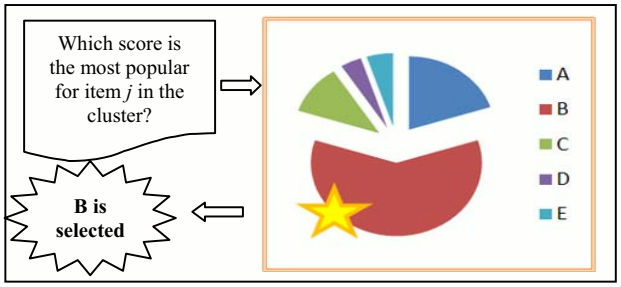

Figure 5. Producing prediction

In this example we want to predict a score for user i on item j. At first we cluster our dataset to k clusters as Fig. 2. Then we find the cluster that includes user i as Fig.3. Finally we use voting to find prediction. Fig. 4 shows rate B is our prediction.

C. 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:

$$f_{u,i} = \arg\max_{r \in R} \sum_{n \in Neighbors} \lambda_n \delta(r, f_{n,i}) . \quad (9)$$

where

$$\lambda_n = \frac{1}{d(user_u, user_n)^2} . \quad (10)$$

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 [24].

V. EXPERIMENT RESULTS

We have implemented two types of traditional collaborative filtering algorithm and the new proposed algorithm. In the rest of this section we compare their accuracy and running time. We have tested the algorithms with different sizes of neighborhood and K, here we show the average of our results.

All our experiments were performed by *MATLAB*. We ran the program on a Windows based NoteBook with *IntelCore2Duo* processor having a speed of *2.66GHz* and *4GB* of RAM.

A. Dataset

We used MovieLens collaborative filtering data set: *(http://www.grouplens.org/)* to evaluate our proposed algorithm. This dataset has *943* users that 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 used *5*-fold cross validation to be fair to compare.

B. Metrics

There are several methods to measure the accuracy of a recommender algorithm [10]. We used the Normalized Mean Absolute Error (NMAE) metric to measure the accuracy of our proposed approach (11). Normalized Mean Absolute Error (NMAE) is a metric that normalizes MAE to show all errors as percentages of full scale. On the other hand NMAE is a normalized form of MAE. The most widely used metric in recommender system researches is Mean Absolute Error (MAE), which calculates the average of the absolute difference between the our predictions and real ratings:

$$NMAE = \frac{MAE}{r_{\max} - r_{\min}} \qquad (11)$$

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

where n is the total number of ratings-prediction pairs in the test set, $prediction_{u,i}$ is the predicted rating for user u on item i, and $real_{u,i}$ is the actual rating in the real dataset. r_{max} is the upper bound and r_{min} is the lower bound of ratings. The lower NMAE is better and denotes lower errors and more accuracy [7, 10, 14]

C. Compare

In this section we represent our result and compare the proposed collaborative filtering algorithm with the traditional one. These results are divided into two main parts: accuracy results and running time results. K-means Clustering was implemented in both form of the Euclidean distance and the City Block distance. User-based and item-based collaborative filtering algorithms were implemented in various sizes of neighborhood, and then we used an average of results. As the same, k-means clustering was experimented by different values of k, and the result that is shown is the average of experiments. We used pearson correlation (1) as similarity measure for traditional collaborative filtering.

D. Accuracy

We compare the accuracy of our proposed algorithm with the basic collaborative filtering. The results in Tab. 2 and Fig. 5 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. The NMAE for user-based and item-based collaborative filtering is around 85%, but it is around 22% in our experiments. And it is a great deal of change.

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 Euclidean distance is about one nintieth and with city block distance is about one tenth of the traditional user-based running time. We performed this experiment with various sizes of neighborhood and k (see Fig. 6).

VI. CONCLUSION

A recommender system is an intelligence web-based application and is a subset of information filtering systems. It is very useful for e-commerce personalized web sites. It helps customers to make a better decision by recommending interesting items. The most successful algorithm of it commonly 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 recommendation approach based on user clustering and Minkowski distance and a new method to guess "no vote" ratings. Our new method to develop a recommendation is based on neighbors' voting. This research considers the users are m points in n-dimensional space. The Minkowski distance calculates the distance between two users to cluster dataset. This method combines clustering and neighbors' voting to generate predicts. 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.

TABLE II. THE RESULTS COMPARISON

	NMAE	Time(sec)
Proposed Algorithm-Euclidean distance	0.21	161
Proposed Algorithm-Cityblock distance	0.22	1186
User-based CF	0.84	15030
Item-based CF	0.89	386

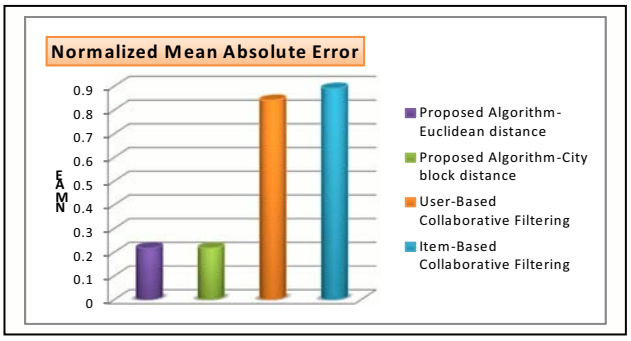

Figure 6. Accuracy Comparison

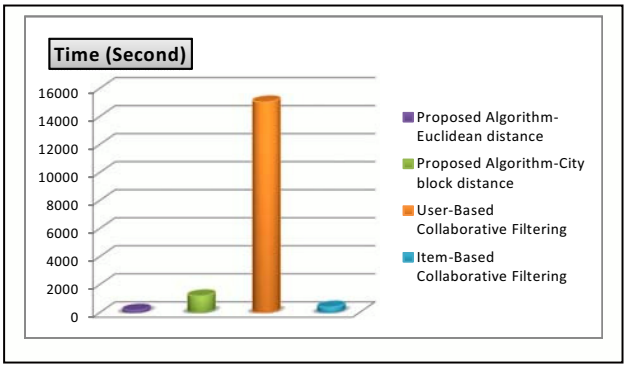

Figure 7. Time Comparison

REFERENCES

[1] P. Symeonidis, A. Nanopoulos and Y. Manolopoulos, "Providing justifications in recommender systems", IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, Vol. 38, Nov. 2008, pp. 1262-1272.

[2] E. Vozalis and K. G. Margaritis, "Analysis of recommender Systems' algorithms", Proc. 6th Hellenic European Conference on Computer Mathematics & its Applications (HERCMA'03), Athens, Greece, 2003.

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

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

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

[6] S. S. Manvi, N. Nalini and L. B. Bhajantri, "Recommender system in ubiquitous commerce", Proc. 3rd International Conference on Electronics Computer Technology (ICECT), Apr. 2011, pp. 434-438.

[7] P. Bedi and S. K. Agarwal, "Managing Security in Aspect Oriented Recommender System", Proc. International Conference on Communication Systems and Network Technologies (CSNT), Jun. 2011, pp. 709-713.

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

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

[10] X. Su and M. Khoshgoftar, "A survey of collaborative filtering techniques", Advances in Artificial Intelligence, Article ID: 421425, Jan. 2009.

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

[12] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions", IEEE Transactions on Knowledge and Data Engineering, Vol. 17, Jun. 2005, pp. 734-749.

[13] M. Balabanovic and Y. Shoham, "Fab: content-based, collaborative recommendation", Communications of the ACM, Vol. 40, Mar. 1997, pp. 66-72.

[14] B. M. Sarwar, G. Karypis, J. Konstan and J. Riedl, " Item-based collaborative filtering recommendation algorithms", Proc. 10th international conference on World Wide Web (WWW' 01), May. 2001.

[15] M. K. Kavitha Devi and P. Venkatesh, "An improved collaborative recommender system", Proc. 1st International Conference on Networks & Communications (NETCOM'09), Dec. 2009.

[16] Y. Gao, H. Qi, J. Liu and D. Liu, "A recommendation algorithm combining user grade-based collaborative filtering and probabilistic relational models", Proc. 4th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'07), Aug. 2007, pp. 67-71.

[17] CH. Huang and J. Yin, "Effective association clusters filtering to cold start recommendations", Proc. 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'10), Yantai, Shandong, Aug. 2010, pp. 2461-2464,

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

[19] J. Liu and G. Deng, "A new-user cold-starting recommendation algorithm based on normalization of preference", Proc. 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08), Oct. 2008, pp. 1-4.

[20] A. I. Schein, A. Popescul, L. H. Ungar and D. M. Pennock, "Methods and metrics for cold-start recommendations", Proce. 25th annual international ACM SIGIR conference on Research and development in information retrieval,Tampere, Finland, Aug. 2002, pp. 253-260.

[21] L. T. Weng, Y. Xu, Y. Li and R. Nayak, "Exploiting item taxonomy for cold-start problem in recommendation making", Proc. 20th IEEE International Conference on Tools with Artificial Intelligence, Dayton, OH, Nov. 2008, pp. 113-120.

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

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

[24] T. M. Mitchell, "Machine learning", Chapter 08, Instance-based learning, Publisher: McGraw-Hill Science/Engineering/Math, March. 1997, pp. 230-247, ISBN: 0070428077.

[25] S. H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions", Mathematical Models and Methods in applied sciences, Vol. 1, issue 4, 2007, pp. 302-307.