A Collaborative Filtering Recommender System Based on User's Time Pattern Activity

Pooyan Adibi
School of Electrical and Computer Engineering
Shiraz University
Shiraz, Iran
pooyan.adibi@gmail.com

Behrouz Tork Ladani
Dept. of Computer Engineering and Information Technology
University of Isfahan
Isfahan, Iran
ladani@eng.ui.ac.ir

Abstract—**Collaborative filtering has been considerably successful in improving recommender systems both in the literature and commercial applications. Most of the algorithms designed up to now consider users' ratings equally and do not pay attention to the fact that users' interests or requirements might change over the time. In this paper a collaborative filtering based recommender system is designed which tries to find each user's interests to each group of items, thus resulting to a better prediction of ratings a user will give to an item in the near future. This goal is achieved through using the ratings' timestamp, predefined groups of items, and defining a new similarity measure among users. Unlike standard collaborative filtering methods and many new ones in which similarity between users is defined as a single number, in this research we define similarity between users as "group similarity" which is an array of similarity values between items of each group rated by two users. Predefined groups for items e.g. genres for movies, are used as groups for items. Also for calculating similarity, different weights will be dedicated to ratings of each user based on the ratings' timestamp, i.e. a rating with higher timestamp will receive a higher weight. Empirical tests show that our proposed algorithm works better than standard User-based and Item-based collaborative filtering methods in the case of predicting users' interests in the near future with higher precision. Also it is empirically shown that our algorithm works considerably well for cold-start users.**

Keywords-Recommender systems; collaborative filtering; ratings' timestamp; group similarity; cold-start users

I. INTRODUCTION

People use recommendations from others in everyday life. Conversations, News, magazines guides belonging to different products or services are just some examples of the ways people can get to know about others' opinions. Recommender systems use this social fact to help people find valuable information about plenty of books, restaurants, movies, web pages, foods, and any kind of products or services available on internet [1]. Recommender systems are an important part of researches about information overload since the beginning of 1990s [2].

Recommender systems are best known for their use on e-commerce websites. These systems use input about users' interests and try to find a list of favorite items for them [3]. Recommender systems use different methods for preparing recommendations. One of the most successful methods is collaborative filtering [1][4]. In summary, collaborative filtering methods use previous activities of the target user in order to find suitable recommendations for him. Activities include ratings, writing reviews or comments, searches, clicks etc. Standard collaborative filtering methods and many others do not take into account the time each activity is done, and do not pay attention to the fact that users' interests or requirements might change over the time [5][6]. Whatever a user's number of activities among a group of items to be a higher value, the probability to which recommender system would recommend him other items belonging to the same group will be higher. These recommendations can be useful while the target user is yet interested to items of that group. But when his interest changes and differs from what he used to like in the past, recommending items of previous group would no longer help.

For example consider a typical user who is looking for a cell phone in an online store. In a simple situation and regardless of complexity of calculations, collaborative filtering recommender systems will find other users whose activities (ratings) are more similar to the target user's activities. These similar users to target user are called *nearest neighbors*. Then items which target user has not seen (rated) yet and are appreciated by neighbors will be recommended. Whatever the target user's activities to be more concentrated on cell phones, the other users chosen as his nearest neighbors will necessarily be the ones who are more active in that domain. Now consider a situation in which the target user has found a suitable cell phone and is not looking for that anymore. Suppose he is looking for a book and his activities will be done in that domain from now on. A small number of systems detect this change in interest or requirement, thus because they find neighbors for each user based on his activities in the past, still the users who are most active in domain of cell phones will be chosen as his nearest neighbors. This is more significant when the target user's activities are mostly concentrated on a special group of items.

In this paper a collaborative filtering recommender system is designed which tries to find interested groups of items in a given period of time for each user, and predicts items which assumes to be appreciated by the target user in the near future more precisely. This goal is achieved through using the ratings' timestamp, groups of items, and defining a new similarity measure among users. Our new similarity measure tries to reduce influence of some problems with traditional similarity measures which we discuss in the next section. Unlike standard

collaborative filtering methods and many new ones in which similarity between users is defined as a single number, in this research we define similarity between users as "group similarity" which is an array of similarity values between items of each group. Predefined groups for items e.g. genres for movies, are used as groups for items. Also for calculating similarity, different weights will be dedicated to ratings of each user based on the ratings' timestamp, i.e. a rating with higher timestamp will receive a higher weight. As one advantage to the traditional methods of calculating similarity which used to have some problems with cold-start users, our similarity measure works well for cold-start users too. The reason why we have concentrated on cold-start users is that when a user keeps on his activities in a new group of items, he can be treated like a new user to that group and therefore his previous ratings and neighbors cannot be much useful in predicting items of new group. In fact, after analyzing ratings of each user and the groups each rating belongs to, we can say that for each user cold-starting conditions might occur more than once, even after the first time he begins his activities in the system. Cold-starting conditions could occur based on how many times the target user changes his group of interests.

Rest of the paper is organized as follows. In section 2 a typical classification of Recommender systems is introduced and features of the methods belonging to each class are briefly described. At the end of section 2, two cases which play an important role in the proposed method i.e. similarity measure and using timestamp of ratings are discussed. In section 3, our proposed method is described in more details. In section 4, the data set used for evaluation is introduced and evaluation results as well as comparing them with standard collaborative filtering methods are represented. Finally section 5 concludes this research.

II. Related Work

From the mid 1990s which recommender systems were more seriously noticed, different methods are designed in order to predict users' degree of interest to different items. In [1] recommender systems are classified into three classes: collaborative methods, content-based methods, and hybrid methods.

Content-based systems try to recommend items that are more similar to items appreciated by the target user [7]. Similarity between items is defined based on predefined features of items. The features mostly include keywords and textual descriptions for each item. Because of this, only those features which are able to be described as text can be used in calculations. This method would not work well for items like videos and audios.

Collaborative filtering systems use ratings of other users instead of predefined features of items. These systems are divided into memory-based and model-based methods [8]. Memory-based systems use all ratings of users for recommendation. Users can be compared based on ratings they have given to common items. Then for each user a group of other users who have rated those items more similarly can be recognized as his neighbors. The same way, items can be compared based on common users who have rated them, and for each item a group of other items which have the most similarity can be recognized as its neighbors. These methods are called *User-based* and *Item-based* methods respectively.

On contrary with Memory-based methods, there are Model-based methods which use a subset of ratings in order to learn a model. Derived model will be used in predictions [1]. There are various ways for deriving a model from ratings which include k-means clustering, Bayesian model, linear regression, probabilistic relational model etc.

Hybrid methods try to uses advantages of both content based and collaborative filtering methods. Hybrid methods use both features of items and ratings from other users. Other sources of information like demographic and social information about users can be used as well.

A. Similarity Measure

In collaborative filtering methods the most important part of actions is finding similarity. In most recommender systems calculating similarity measure is done by the use of traditional vector similarities which the most common ones include Pearson Correlation Coefficient and Cosine similarity [9]. Although systems using these similarity measures have been successful in predictions, there are various situations in which these systems cannot work well. Most important problems which Pearson and Cosine suffer from are insufficient of data and cold-start users. Considering an electronic store in which there are millions of items, the most active users hardly rate more than tens of items. Therefore in the user-item matrix most of entries are empty. This situation is considered as insufficient data. A cold-start user is a user whose number of items rated by him is not enough. In a typical electronic store there are many users who are not much active and have rated a small number of items. Consequently it is difficult to find other users who have enough ratings in common with them. Also in [10] it is mentioned that in order to be able to apply Pearson correlation with guarantees, the following assumptions must be met:

- Linear relationship between variables.
- Continuous random variables.
- Both variables must be normally distributed

These conditions are not normally met in real conditions, and Pearson presents some significant cases of erroneous operation that should not be ignored [10]. In this paper a new similarity measure among users is defined which reduces influence of above problems, and works well specially for cold-start users.

B. The Use of Time

Adding the timestamp of each rating in calculations is mostly done through defining a weight for each rating. These weights will change the influence of ratings in calculations. In [11] authors have pointed out that users' requirements change over the time and model of change differs based on the kind of products and the time elapsed from the purchase of each product. For example when a user has just bought a washing machine, it is foreseeable that he will not need another one until the average lifetime of that kind of product has passed. Therefore recommending such items just a short time after user's purchase could not be helpful. But some other kinds of products which a user could buy daily such as books, music,

etc could be recommended more frequently. Thus clothes, household items, entertainment items etc must be weighted differently. In [5] a collaborative filtering system is designed which tries to detect changes in customers' patterns by using item dimensionality reduction, and finding neighbors for each user based on his recent interests. Authors have used a movie data set for evaluation in which genres of movies are used in dimensionality reduction. In [6] authors believe that recent ratings of users are more relevant to his interests in future. Therefore ratings given by a user in different times must not be considered equally, i.e. ratings with higher timestamps must receive higher weights than ratings with lower timestamps. In [12] ratings given by the target user are weighted in an ascending order. Then users having more similarity with the target user will be chosen as his neighbors and predictions will be done using ratings from his nearest neighbors. Each algorithm has pros and cons, and their superiority is shown based on empirical evaluations and using special forms of data sets.

III. THE PROPOSED METHOD

In our method, we calculate similarity between users like user-based methods. But opposite them, in order to find similarity we do not limit ourselves to common ratings given by the target user and others. In our method we assume that all items are classified in predefined groups, thus we calculate similarity between users based on their interest to the *group* of each item. We name this kind of similarity *group similarity* between the target user and others. We calculate group similarity between users based on the measure of interest each user has to each group. So we must define a measure to show how much a user is interested to a group. We name it *group interest*. Then, for each user we cluster groups based on the measure of group interest that we have calculated for him. Then, in order to find group similarity between two users, we compare interest value of users to corresponding groups. Based on the cluster that each group is located in, we calculate similarity between groups. Notice that items can belong to more than one group. So first we must define a criterion to specify how much an item belongs to each group. We name this measure *group membership* for items. Then we can calculate group interest of each user. Group membership is calculated for items rated by each user; therefore it is calculated separately for the items rated by each user because it is calculated by the use of rating value and sequence of ratings a user has dedicated to an item.

A. Group Membership

Finding group membership of items is based on the hypothesis that the intense to which a user is interested to an item cannot equally be divided among groups which that item belongs to. For example, when a user rates a high value to a movie which belongs to comedy and crime genres, it does not necessarily mean that the user is interested in both genres equally. It might turn out that the user likes comedy movies and that was the reason why he rated it a high value, but he is not much interested in crime movies. In such conditions we say that for that user, comedy aspect of the movie is more interesting than crime aspect. So for that user we assume that this movie is a member of comedy group with a higher value, and is a member of crime group with a lower value. For another user, membership of that movie could be different. For calculating group membership of the items which the target user has rated, we use sequence of ratings. First we order target users' ratings ascending base on the ratings' timestamp. Then, starting from the first rating, we calculate group membership of each item rated by him to its groups, using the pseudo code shown in Fig. 1.

```
CalculateGroupMembership (Rating rt)
{
  targetUser.Ratings = targetUser.Ratings.OrderAscending();
  array of key-value pairs GroupsCount<key, value>;
  for ( i from 1 to targetUser.Ratings.Count)
  {
    Rating rt = targetUser.Ratings[i];
    totalGroupsCount = 0;
    foreach (Group g in rt.Groups)
    {
      If  (!GroupsCount.Contains(g))
          GroupsCount.Add(g, 0);

      GroupsCount[g] += rt.RatingValue;
      totalGroupCount += GroupsCount[g];
    }
    If  (rt.Groups.Count > 1)
    {
      foreach (Group g in rt.Groups)
      {
          membership = GroupsCount[g] / totalGroupsCount;
          rt.GroupMembership.Add(g, membership);
      }
    }
    else
      rt.GroupMembership.Add(g, 1);
  }
}
```

Figure 1. Pseudo code for calculating items' group membership

Group membership value is a real number between 0 and 1, and sum of all the values of all groups belonging to an item is equal to 1. Again it is good to mention that group membership of an item for different users is calculated separately, so it is not necessarily equal for different users, because it is affected by the sequence of items rated by a user, and the rating value which a user gives to that item and items rated earlier.

B. Group Intererst

For getting time weights involved and finding users' interests to each group in different periods of time, we assume that current interests or requirements of each user have more relevance to his recent ratings. Therefore we must add time weights to the process of finding user's interest to each group, in a way that recent ratings receive a higher weight than ratings given in the past. An ascending function of time, named forgetting function, for setting the measure of influence each rating makes in calculating group interest can be used. Some ways for defining a proper function can be found in [11] and [12]. In this research we have used (1),

$$w(r) = (t_r - t_{max} + 1)^b \quad (1)$$

where t_r shows current rating's timestamp, and t_{max} is the timestamp of the last rating given by user u. Variable b adjusts

intense of forgetting, the way whatever its value to be higher, recent ratings will have more influence in calculations.

Measure of interest that user u has to group g is calculated as shown in (2),

$$GF_u[g] = \sum_{r \in R_u} GPM_r[g] * w(r) \quad (2)$$

where R_u is set of all ratings given by user u, and $w(r)$ shows weight for the rating r.

C. Group Similarity

After group interests of users are found, similarity between users can be calculated. In standard collaborative filtering methods and many new ones, similarity between users is defined as a single number. But in this research we define similarity between users as *group similarity* which is an array of similarity values between items of each group. For calculating group similarity, for each user we cluster groups based on the interest value of the user to each group, derived by (2). In this way we can have a clustered model of users' interests in a given time. Reason for clustering groups is that some of groups have many similar items in common. For example a considerable number of movies belonging to comedy group also belong to drama group, or many films belong to both animation and children's groups. We define group similarity between two users u and v as shown in (3):

$$sim_{u,v}[g] = (GF_u[g] + GF_v[g]) * \left(||Clusters||^2\right) - \left(|GF_u[g] - GF_v[g]|| * (|ClsInv(u,g)^2 - ClsInv(v,g)^2| + 1)\right) \quad (3)$$

where $GP_u[g]$ shows the measure of interest user u has to group g calculated by (2), $|c|$ is number of clusters, and function $ClsInv(u,g)$ returns an inverse form of cluster number of group g for user u. For example having number of clusters equal to 5, for cluster number 5, value 1 is returned or for cluster number 4, value 2 is returned.

D. Prediction

For predicting the rating value which user u might give to item m, first we suppose that item m is the last item rated by user u with highest value (e.g. value 5 in scale of 1 to 5). Then we calculate group membership for item m using pseudo code shown in Fig. 1. Then we choose those users that have a higher group similarity for groups which m belongs to. We consider an aggregation of similarity values for the groups of m in order to compare neighbors of u. Then using (4), we calculate a single value as amount of help a neighbor i can give in predicting item m rating value.

$$use_m(i) = \sum_{g \in G_m} GF_i[g] * GRP_{rNew}[g] \quad (4)$$

In (4) $rNew$ is the temporary rating for item m which we assume to have the highest rating value, and timestamp for it will be current time of system. Array GF_i shows the measure of interest to group g by neighbor i. Then rating value for item m by user u can be predicted using (5):

$$p_{u,m} = \bar{r}_i + \frac{\sum_{i \in I}(r_{i,m} - \bar{r}_i) * use_m(i)}{use_m(i)} \quad (5)$$

where I is the set of all neighbors for target user u, $r_{i,m}$ is the rating to item m by neighbor i, and $\bar{r}_u$ is the average of all ratings given by user u.

IV. Experimental Results and Evaluation

A. Data Set

For evaluating our proposed algorithm and comparing it with standard collaborative filtering systems we have used MovieLens data set which is one of the most widely used data sets in evaluating recommender systems [6]. MoveiLens data set contains 1000209 ratings given by 6040 users to 3900 movies. Rating values are integers from 1 to 5 and have a timestamp represented in seconds since a starting time. Also genres of each movie are specified which we use as groups of items.

This research is focused on finding users' interests in the future and is tracking this goal through using sequence of ratings given by the target user and weighting his recent ratings. Accordingly for a better prediction of users' interests in different periods of time, ratings given by a user for different items should be submitted in system in the same order he has used those items. Unfortunately in MovieLens data set there are many users for whom all or a great number of ratings are submitted in only one day. In such circumstances the order of ratings and the time distance between them cannot help finding users' time activity pattern. So in order to have a more precise evaluation we have selected those users whose ratings have higher *dispersion*. We define dispersion as a measure for each user, as number of ratings divided by number of unique days in which at least one rating is submitted. This value will be a real number between 0 and 1. Whatever dispersion to be close to 1 it means that few ratings are submitted in the same day. Therefore we have first calculated dispersion measure for each user's ratings and then we have selected top 300 users having highest dispersion.

B. Evaluation Methods and Metrics

Several methods and metrics have been employed to evaluate recommender systems so far which some of them are described in [13]. What is common in most of methods is breaking data set into a training set and a test set. Training set is used as input to the system, and the system tries to predict ratings available in test set. Then predicted ratings are compared with real ratings and predictions precision will be calculated through different metrics. In this paper we have used MAE (Mean Absolute Error) for specifying precision which returns an average of differences between the real and the predicted ratings. Equation (6) shows how MAE is calculated for each user [14]:

$$MAE = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \quad (6)$$

where p_i is the predicted rating and r_i is the real rating for item i given by a user. Also we have used coverage metric in order to calculate what portion of ratings in test set can be predicted by our algorithm. Coverage is calculated as shown in (7):

$$coverage = \frac{\text{Number of predicted ratings in test set}}{\text{Number of all ratings in test set}} \quad (7)$$

Since our algorithm uses sequence of ratings and their timestamp for predicting users' interests in the near future, time is also used as a weight to predictions' errors. Whatever the timestamp of a rating in the test set to be a lower value, it will receive a higher weight. Time weighting function for evaluation is shown in (8):

$$H(t) = k * (\frac{t_{max} - t}{t_{max} - t_{min}})^2 + 1 - k \quad (8)$$

where t is the timestamp of current rating, t_{max} is the last rating's timestamp in the test set given by current user, t_{min} is the first rating's timestamp in the test set given by current user, and k is a real number between 0 and 1. Whatever k to be closer to 1, ratings with lower timestamp will receive higher weight. Notice that ratings in test set are ordered ascending based on ratings' timestamp. Therefore (7) is changed as shown in (9). We have used (9) for calculating predictions errors.

$$MAE' = \frac{\sum_{i=1}^{N} |p_i - r_i| * \mathrm{H(t)}}{N} \quad (9)$$

To begin evaluation, first we order all ratings given by users in the training set ascending based on their timestamp. We have evaluated our algorithm with three different divisions of data set into training set and test set which includes 25, 50, and 75 percent of ratings with timestamp within the selected range of time. For example in the case 75 percent, after ordering data set, we select ratings for which timestamp is shorter than 75 percent of the whole range of time (range of time is the time distance between the first rating and the last one). An advantage to this way of division is that we can simulate a schema like what the system confronts in real conditions. In such circumstance each user's number of ratings differs. There are some users whose ratings are not in selected range so there are no ratings by them in the training set. Some users have few ratings that we consider them as cold-start users. Some users' ratings are all in the selected period of time so there are not any ratings by them in the test set. Finally some others have a normal distribution of their ratings in training set and test set.

C. Evaluation results

Evaluation results are represented in table 1. Due to lack of space we were obligated to eliminate experimental tests for adjusting parameters, therefore only final results achieving under the best state are represented.

In our proposed algorithm we have considered 100 users as number of neighbors, and 5 clusters for clustering groups. Also parameter b in (1) is considered equal to 2. We have implemented user-based and item-based collaborative filtering algorithms and after applying changes to provide coordination with our evaluation method, the best state based on what is declared in [15] is selected. For User-based algorithm we have considered 300 users as number of neighbors and Pearson weighted by Jaccard as similarity measure. For Item-based algorithm we have considered 100 items as number of neighbors, 400 as size of model, and adjusted Cosine as similarity measure.

TABLE I. COMPARING MAE AND COVERAGE OF PREDICTIONS. TC-GF IS THE NAME OF OUR PROPOSED ALGORITHM; UB AND IB STAND FOR USER-BASED AND ITEM-BASED COLLABORATIVE FILTERING ALGORITHMS RESPECTIVELY.

Algorithm	Training set percent	MAE (all users)	Average of MAE (cold-start users)	Coverage (cold-start users)
TG-CF	25	0.794	0.805	0.893
UB	25	0.799	1.042	0.618
IB	25	0.798	1.088	0.536
TG-CF	50	0.796	0.952	0.957
UB	50	0.814	1.177	0.851
IB	50	0.831	1.128	0.51
TG-CF	75	0.781	0.779	0.955
UB	75	0.786	0.966	0.861
IB	75	0.881	1.079	0.645

We have evaluated precision and coverage for cold-start users separately. Since there is not a clear number of a ratings dedicated for cold-start users, we have selected users having less than 10 ratings in training set. As we mentioned before, the reason why we have evaluated our algorithm for cold-start users separately is that when a user keeps on his activities in a new group of items, he can be treated like a new user to that group and therefore his previous ratings and neighbors cannot be much useful in predicting items of the new group. As we can see in table 1, our proposed algorithm has been able to increase precision of predictions in each level in comparison with other algorithms. But results for cold-start users were considerably better. The most important reason for behaving well for cold-start users could refer to the new similarity measure that we have defined.

V. CONCLUSION

Collaborative filtering has been considerably successful in improving recommender systems both in the literature and commercial applications. Most of algorithms designed up to now consider users ratings equally and do not pay attention to the fact that users interests or needs might change over the time. In this research we have shown that in order to consider changes in users' interests or requirements we can use predefined groups of items as well as the time and order each user rates an item without increase in time complexity. Predefined groups can be used to calculate group similarity between users rather than a single value as similarity measure. We have empirically shown that our algorithm works better than standard user-based and item-based collaborative filtering methods for users whose ratings have a proper dispersion. Also our algorithm works better for cold-start users because there might be more than one cold-starting condition for each user based on the times he changes his interests.

REFERENCES

[1] X. Su , and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in Artificial Intelligence, vol. 2009, no. 4, January 2009.

[2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of Netnews," Proceedings of the ACM Conference on Computer Supported Cooperative Work, pp. 175–186, New York, NY, USA, 1994.

[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, January/February 2003.

[4] 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, no. 6, pp. 734–749, June 2005.

[5] S. H. Min, and I. Han, "Detection of the customer time-variant pattern for improving recommender systems," Expert Systems with Applications, vol. 28, pp. 189-199, February 2005.

[6] Y. Ding, and X. Li, "Time weight collaborative filtering," in Proceedings of the 14th ACM international conference on Information and knowledge management, pp. 485-492, Bremen, Germany, November 2005.

[7] R. J. Mooney, and L. Roy, "Content-based book recommending using learning for text categorization," Proceedings of the 5th ACM conference on Digital libraries, pp. 195-204, New York, NY, USA, 2000.

[8] J.S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," Proceedings of the 14th conference on Uncertainty in artificial intelligence, pp. 43-52, San Francisco, CA, USA, 1998.

[9] H. J. Ahn, "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem," Information Sciences, vol. 178, pp. 37-51, January 2008.

[10] J. Bobadilla, F. Serredilla, J. Bernal, "A new collaborative filtering metric that improves behavior of recommender systems" Knowledge-Based Systems, vol. 23, pp. 520-528, August 2010.

[11] Y. Blanco-Fernández, M. López-Nores, J. J. Pazos-Arias, A. Gil-Solla, and M. Ramos-Cabrer, "**Personalizing e-commerce by semantics-enhanced strategies and time-aware recommendations**," Semantic Media Adaptation and Personalization, Third International Workshop on, pp. 193-198, Prague, Czech Republic, December 2008.

[12] Z. Chen, Y. Jiang, and Y. Zhao, "A collaborative filtering recommendation algorithm based on user interest change and trust evaluation," International Journal of Digital Content Technology and its Applications, vol. 4, pp. 106-113, December 2010.

[13] A. Gunawardana, and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," The Journal of Machine Learning Research, vol. 10, pp. 2935-2962, December 2009.

[14] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," ACM Transactions on Information Systems (TOIS), vol. 22, pp. 5-53, January 2004.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," Proceedings of the 10th International conference on World Wide Web, ACM New York, NY, USA, pp. 285-295, 2001.