Alleviating Data Sparsity and Cold Start in Recommender Systems using Social Behaviour

Reshma R
Department of CSE
National Institute of Technology Karnataka
Surathkal, Mangalore, INDIA
r.reshma67@gmail.com

Ambikesh G
Department of CSE
National Institute of Technology Karnataka
Surathkal, Mangalore, INDIA
ambig45@gmail.com

P Santhi Thilagam
Department of CSE
National Institute of Technology Karnataka
Surathkal, Mangalore, INDIA
santhisocrates@gmail.com

***Abstract*—Recommender systems are used to find preferences of people or to predict the ratings with the help of information available from other users. The most widely used collaborative filtering recommender system by the e-commerce sites suffers from both the sparsity and cold-start problem due to insufficient data. Most of the existing systems consider only the ratings of the similar users and they do not give any preferences to the social behavior of users which shall aid the recommendations made to the user to a great extent. In this paper, instead of finding similarity from rating information, we propose a new approach which predicts the ratings of items by considering directed and transitive trust with timestamps and profile similarity from the social network along with the user-rated information. In cases where the trust and the rating details of users from the system is absent, we still make use of the social data of the users like the products liked by the user, user's social profile-education status, location etc. to make recommendation. Experimental analysis proves that our approach can improve the user recommendations at the extreme levels of sparsity in user-rating data. We also show that our approach works considerably well for cold-start users under the circumstances where collaborative filtering approach fails.**

Keywords—Recommender system, Collaborative filtering, Sparsity, Cold-start problem, Social behaviour

I. Introduction

Since internet has become part of daily life of millions of users, most of the sellers are trying to make internet as their selling platform to reach more and more customers. Thus many e-commerce websites are springing up day by day. All of these websites want to provide better services and experiences to their customers to attract more and more of them and thereby increase their revenue. Understanding the interests of users and recommending products to the users based on their individual interests have become a major challenge which has given birth to the recommender systems.

Mass marketing in e-commerce is very expensive and is no more an efficient way to advertise the products. Targeted marketing is a better way to attract more users, in which people will get recommendation about only those products which he or she may be interested in. Nowadays personalized recommendations have more chances because of the increase in availability of information from internet. All popular websites implementing recommender system like Movie Lens, Last.fm, Netflix, Amazon, YouTube, etc. make use of all possible information in order to collect maximum user preferences so that they can provide better recommendations to their users.

Generally a recommender system recommends items by learning user's profile, user's previous activities, and the kind of items available in the system, etc. There are various approaches used by the e-commerce platform to recommend products to their users. Major approaches include content-based, collaborative filtering and hybrid methods. Out of these approaches, collaborative filtering (CF) based approaches build the most efficient recommender systems. Even though there are many algorithms in CF based recommender system, user-based collaborative filtering (UBCF) and item-based collaborative filtering (IBCF) are widely implemented. UBCF recommender systems have gained so much attention because this approach uses the opinion of similar minded people from their rating pattern. However, major challenges of this approach are sparsity of ratings and cold-start problem. When the information reported (like items rated) by a user is very less compared to the available data (items in the system), then it is very difficult to get the similarity between the like-minded people accurately. This problem is termed as sparsity of ratings. Prediction will be absolutely impossible when a new user or an item is introduced into the system which is termed as cold-start problem.

With the social networking websites like Facebook, Google Plus, LinkedIn, etc. becoming more and more popular, social data of common people are becoming easily available on the internet. E-commerce websites have started making these social networks as their platform in-order to get access to the social data of users so that they can understand the user's behavior and interests using which they can market their products in a much better way. The traditional recommender systems avoid the trust and/or the social relationship of the users. It is quite common that the activities of a person be greatly influenced by his or her friends. In most of the cases decisions taken by the user may be a balance of his opinion and his/her friend's tastes. Social networking sites provide a better way to share interests, comments, likes, etc. This data is very helpful in personalized recommendations. For example while watching movies or listening music, people will give more preference to those movies or music which are highly rated or liked by their close friends. Thus it is quite important to

978-1-4673-9802-2/16/$31.00© 2016 IEEE

analyze not only the user's behavior, but also the interactions and activities of user's friends to get better understanding of the user's likes.

II. RELATED WORK

Recommender system plays a very important role in e-commerce. Most of the commercial websites use the recommender systems in order to recommend products which are close to the user's preference. Even the social networking websites now-a-days use recommender systems in order to recommend friends, movies, places to visit and so on. In order to achieve recommendation, it is quite important to understand the preferences of users. Some of the approaches taken up to understand the user preference include learning the user's profile, finding out products related to the previous products preferred by the user, finding users who were having similar interests, etc. Existing approaches fall under three broad classifications such as Content-based recommender Systems, Collaborative filtering based recommender system and Hybrid recommender systems

With the social networks like Facebook, Google Plus, etc. becoming more and more popular in everyday life of millions of people, Social filtering based recommender system pitched in to provide better recommendations to the users. We represent this classification using a taxonomy diagram shown in Figure 1.

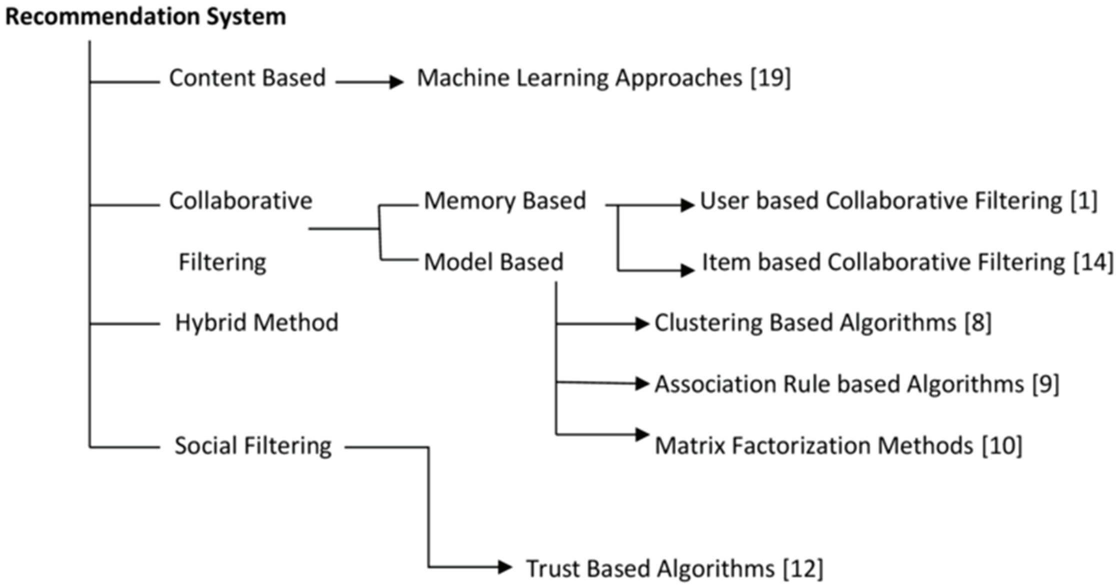
Recommendation System
Content Based
Machine Learning Approaches [19]
Collaborative Filtering
Memory Based
Model Based
User based Collaborative Filtering [1]
Item based Collaborative Filtering [14]
Hybrid Method
Clustering Based Algorithms [8]
Association Rule based Algorithms [9]
Social Filtering
Matrix Factorization Methods [10]
Trust Based Algorithms [12]

Figure 1: Proposed System Design

2.1 Content based Recommender Systems

Content based recommender systems[1] make recommendation based on user profile, user's previous preferences and the characteristics of previously preferred items in an assumption that better personalized recommendations can be made by gathering more information about users and items (like demographic information, movie genres, etc.). However one of the major problems of Content based recommender systems is that it under performs when there is lack of contents, and it always recommends items which are similar in characteristics and is already known to the users.

2.2 Collaborative Filtering Recommender Systems

Collaborative filtering based recommender system considers the previous user ratings or the history of previous purchases made by the user and applies the information filtering technique on it [2, 3, 4]. Finding a set of like-minded people or to identify a set of similar items closely related to the targeted item forms the most important step in this approach. Hence, the fundamental principle behind collaborative filtering approach is that the interests of active users in the system will more likely be similar with those of the existing users.

2.2.1 Memory based collaborative systems

Memory based algorithms make use of the entire user-rating database to make single prediction. Since it uses entire information in the database to make every single prediction, the time taken is too huge. User-rating data is analyzed to group similar users or similar items [5, 6, 7].

2.2.2 Model based Collaborative Filtering

In model based approach [8, 9], models developed using data mining and machine learning approaches are used to make predictions. The model based approaches are faster in predictions than the memory based approaches because the latter uses the entire information present in the database. But this efficiency comes at the cost of time spent in learning the model.

2.3 Hybrid Recommender Systems

Hybrid recommender systems combine two or more recommendation techniques to overcome the drawbacks of each of them so that a better performance is obtained [10]. Most of the time, collaborative filtering is combined with other recommendation methods to overcome the problems of collaborative filtering [11].

2.4 Social Filtering

Social network allows people to share their favorites with the online world or with the people to whom they are connected. Main objective of recommender system is to make recommendations more and more personalized and these social platforms opened a new way to recommender system by allowing it to find the actual trust value between people. [12]

2.5 Challenges of Recommender System

The e-commerce websites are used by millions of people and hence the amounts of data generated by these websites are too huge. As a result of this, recommender systems are becoming more and more computationally expensive with the increase in availability of data. Due to the huge number of users and products, computational cost for prediction is becoming very high. This introduces the scalability problem. IBCF recommender system will be a solution for this problem since item similarity is comparatively more stable than user similarity. But IBCF suffers from many other problems like cold-start problem, sparsity, etc. [13, 14]. When a new user is added into the system, user would not have rated enough number of items. Therefore it is impossible for the system to find similar users. So neighborhood of this new user will not be sufficient enough to make any predictions. Same is the scenario when a new item is added into the system, since the item will not have any user ratings associated with it. Therefore such newly introduced items cannot be recommended to any user. This problem is termed as cold-start problem in recommender system [15].

In any recommender system, it is impossible to assume that each user will rate every item present in the system. It is even possible that some users will not rate any item. All these cause the user-rating matrix to be sparse. This is termed as sparsity problem. This further weakens the recommendations. For example, if we consider common collaborative filtering based recommender systems which use Pearson correlation as a similarity measure to compute the similarity between two users, it considers two users with single common rated item also as similar which is logically incorrect.

III. PROBLEM DESCRIPTION

The objective of this proposed approach is to increase the accuracy of prediction and alleviate problems due to sparsity and cold-start in recommender systems. To achieve this, our approach will incorporate the characteristic of Social Network Graph (SNG) along with User-Rating Matrix (URM).

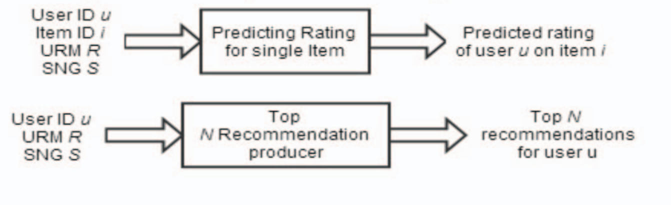
User ID *u*
Item ID *i*
URM *R*
SNG *S*
Predicting Rating for single Item
Predicted rating of user *u* on item *i*
User ID *u*
URM *R*
SNG *S*
Top *N* Recommendation producer
Top *N* recommendations for user u

Figure 2: Block Diagram of Recommender System

Figure 2 represents black box view of recommender system with mapping between inputs and output. Let *U = {u1, u2, ..., um}* be a set of users and *I = {i1 , i2 , ..., in}* be a set of items (e.g. books, movies, or CDs rated by users). Each user ui, i = 1, 2... m has rated a subset of items. The rating of user ui for items ij, j = 1, 2... n is denoted by $r_{i,j}$.

In social network, users {u1, u2 ... um} are considered as nodes and each node is associated with 9 attributes which are nothing but the information available about the users. Attributes are defined over a set:

A = {a1, a2, a3, a4, a5, a6, a7, a8, a9}

where {*a1: Gender, a2: Home Town, a3: Language, a4: Educationalstatus, a5: Visited place, a6: Music, a7: Movies, a8: Activities/Interests, a9: Books*}.

Friendship among users is represented by directed edges. Each edge may also have certain attributes which indicates the different interactions among users. Interaction attributes are defined as a set:

IN = {*in1, in2, in3, in4*} where {*in1: Private message, in2: Wall Post, in3: PictureTagging, in4: Comments/likes/share*}.

Each interaction type ***in*** in the set *IN* takes two values; one is the number of interactions over the edge and the second one is the timestamp of last interaction over that edge.

IV. PROPOSED METHOD

Recommender algorithms generally use rating pattern from the user-rating matrix in which each user will rate items according to his/her interests. Rating may vary from 1 to 5. This information is not sufficient to get the actual trusted similarity between people. Decision taken by the user will be balance of his/her opinion and friend's influence.

In the proposed approach, first step is computation of nearest neighbors from the user-rating matrix. But unlike in the other algorithms, here a threshold is set on the number of co-rated items between users. Only those set of users who have the number of co-rated items crossing the threshold is considered further for similarity calculations. The second step is to analyze the characteristics of social network data to get the interaction intensity and attribute similarity of all other users with the active user. Further, the weighted sum of these values are calculated to get the social similarity and thereby a set of neighbors from the social network. Predicted rating of an item is calculated separately using each set of neighbors obtained in the above two steps and a weighted sum of these predicted ratings are returned as the final predicted rating. We discuss each of these steps in detail along with algorithms in the following subsections.

4.1 Finding Nearest Neighbors from Rating Pattern

The first step is computation of nearest neighbors of each user using their rating patterns. To perform this, the ratings expressed by each user are extracted from the user-rating matrix. This is called the rating pattern of user in the system. Each user might have rated only few items in the system or he/she may have rated comparatively large number of items. The user under current consideration is referred as the active user. Rating pattern of the active users of the system will be compared against that of all other users in the system. This comparison is done with a goal to identify the like-minded people in the system. Pearson correlation co-efficient is used to find the degree of similarity. Normally this will lead to wrong predictions. This issue becomes even more significant as the number of items in the system increases. One of the possible ways to avoid this problem is to have a check on the number of co-rated items between the active user and each other users. Hence a threshold value is introduced for the number of co-rated items between each pair of user.

We calculate the similarity between only those set of users whose co-rated value exceeds the pre-set threshold. Another threshold value called similarity threshold is introduced on the calculated similarity values. We consider only those set of neighbors whose similarity value crosses this threshold which is pre-defined. Similarity threshold can vary according to the level of accuracy. If similarity threshold is very high, near to one, then the nearest neighborhood will be very small. If it's very small, say very close to zero, then there are chances to get inaccurate result. So in order to get optimum result we need to set a similarity threshold in between 0 and 1. Algorithm 1 shows the detailed steps to find the set of nearest neighbors from rating patterns.

Algorithm 1: Finding nearest neighbors from rating pattern

```
Input: User ID u, URM R
Output: A list of nearest neighbors and their similarity with u
for each row v ! = u ∈ R do
    Calculate C = the set of co-rated items of u and v and ncr = |C |
    If ncr>min_threshold then
        user_sim (u, v) = $\frac{\sum c \in C(r_{uc} - \bar{r}_u)(r_{vc} - \bar{r}_v)}{\sqrt{\sum c \in C(r_{uc} - \bar{r}_u)^2}\sqrt{\sum c \in C(r_{vc} - \bar{r}_v)^2}}$
        If user_sim (u, v) > μ then
            Add (v, user_sim (u, v)) to a list N N _RS (u)
        end if
    end if
end for
return N N _RS (u)
```

In Algorithm1, min_threshold is the threshold on the number of co-rated items as explained previously. min_threshold can vary according to the size of the dataset. μ is the similarity threshold. Value of μ varies between 0 and 1. Variable N N _RS represents the set of nearest neighbors from rating pattern. It is easy to observe the rating patterns of the users when the number of items in the entire rating system is less. But if there are many items, making accurate predictions will be a difficult task. If the user is new to the rating system, then that person will not have any rating for any of the items.

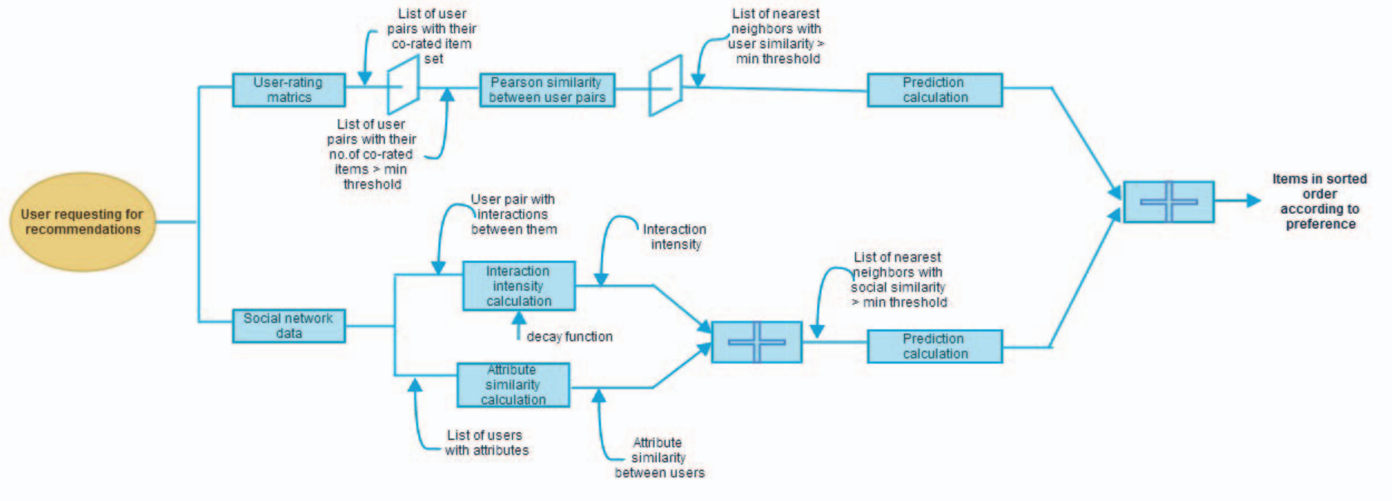
User requesting for recommendations
User-rating matrics
List of user pairs with their co-rated item set
List of user pairs with their no.of co-rated items > min threshold
Pearson similarity between user pairs
List of nearest neighbors with user similarity > min threshold
Prediction calculation
Items in sorted order according to preference
Social network data
User pair with interactions between them
Interaction intensity calculation
Interaction intensity
decay function
Attribute similarity calculation
List of users with attributes
Attribute similarity between users
List of nearest neighbors with social similarity > min threshold
Prediction calculation

Figure 3: Social Recommender System

4.2 Finding nearest neighbors from social network

In the second step, social graph of users are introduced in order to reduce the existing problems of sparsity and cold-start in collaborative filtering approach. Social network of the user is analyzed to compute the interaction similarity and attribute similarity separately and theses two similarities are combined together to get the new social similarity. We discuss the calculation of interaction similarity, attribute similarity and there by the social similarity values in further sub-sections.

4.2.1 Calculating the interaction intensity

Interaction intensity between two users is calculated based on the timestamps of the interactions that has happened between them i.e. only those interactions which are quite recent are counted while the interactions which happened long back are given less or no importance. Interaction type could be private messages, wall post, picture tagging, comments, likes, share, etc

Friendship is nothing but a relation represented as directed edge between users. Hence the interaction between two users need not be symmetric. Interactions are thus considered as strictly asymmetric. Even though there is an edge between users, if they are not communicating then their intensity of interaction is zero. Same is the case if the interactions have timestamps which are very old. A time decay function is introduced to attach a decay value for every interaction between any user pairs based on the timestamp of interactions to give higher weight-ages to most recent interactions.

Time Decay function: In time decay function, current timestamp is compared with the most recent interaction timestamp. Decay value keeps on decreasing as the difference between current timestamp and the most recent interaction time keeps increasing; finally time decay function reaches zero when the difference is very high. Beyond that, time decay function remains zero. Let t_c and t_i be the current timestamp and the timestamp of the most recent interaction between two friends' u and v. For a given interaction in, let the variable δ indicates the difference in number of days between the two timestamps tc and tin ; Time decay function td(in) gets a value 1 when delta is less than or equal to 180. If δ is greater than 360 then t_d(in) returns 0. t_d(in) is defined when δ is in between 180 and 360 as shown below, with values of limit equal to 360 and base equal to 180.

$$t_d \text{ (in)} = \frac{limit - \delta}{base - \delta} \qquad (1)$$

Integrating time decay function and number of interactions: Interaction intensity is the sum of interactions calculated over a set of interactions associated, each with their respective time decay function value. In Algorithm 2, a detailed procedure to calculate the set of neighbors based on their interactions has been shown. In this algorithm, formula given in line number 7 denotes the interaction intensity, intr (u, v) between active user u and his/her direct neighbors' v. Sometimes two users may have many common friends, so that there are chances that these two users become friends in future.

So it is important to predict the chances of their interaction. When there is no edge in between two users, system will predict their chances of interaction and return this value as interaction intensity. This is called transitive interaction intensity. Line number 11 calculates the transitive interaction intensity, intr (u, v) between two users' u and v who have common friends.

In Algorithm 2, E (u) is the list of nodes in the directed edges starting from u.

4.2.2 Calculating the Attribute Similarity

Attribute similarity between users is also considered to find the set of similar behavior. Some people may not be friends with each other but there are chances that they are similar minded. Here we consider nine major attributes of users as defined in section 3. Each attribute must be associated with a weight factor which indicates the importance of that attribute in similarity calculation. Sum of the weights should be equal to one. This approach assigns equal weight-age to all the nine attributes. So the weight-age of individual attribute is one by nine. All the attribute similarity values are combined by assigning equal weight-age as explained previously to get the final attribute similarity. Algorithm 3 describes the entire procedure for finding the attribute similarity.

Algorithm 2: Calculation of interaction intensity from social network

```
Input: User ID u, SNG S
Output: A list of users with their interaction intensity with u
1: for each user v 2 E (u) in S do
2: for each interaction in 2 IN) do
3: Compute $n_i$= number of interaction in from u to v
4: Compute t (in, u, v) =$t_d$ (in)
5: intr (u, v) = 1/IN × $\sum_{in \in IN} ni \times t(in, u, v)$
6: Add (v, intr (u, v)) to a list Interaction (u)
7: end for
8: for each user w∈ $E(u) and\ v \in E(w)\ in\ S\ do$
9: intr (u, v) =$\frac{\sum_{w \in E(u) and v \in E(w)} intr(u,w) \times intr(v,w)}{\sqrt{\sum_{w \in E(u) and v \in E(w)} (intr(u,w) \times intr(v,w))^2}}$
10: Add (v, intr (u, v)) to a list Interaction(u)
12: end for
13: end for
14: return Interaction (u)
```

4.2.3 Combining Interaction Intensity and Attribute Similarity

Finally, the social similarity from social network is calculated by finding the weighted sum of interaction intensity and attribute similarity. Algorithm 4 describes the procedure to find the social similarity by combining these values. Here, α is the weighted moving average co-efficient which decides the weight-age for interaction intensity and attributes similarity. The variable N N _SN represents the set of nearest neighbors from the social network.

Algorithm 3: Finding the attribute similarity between users

```
Input: UserID u, SNG S
Output: A list of users with their attribute similarity
1: for each user v ∈ S do
2: for each attribute $a_i$ ∈ A do
3: if $a_i$ = $a_1$ then
4: Let $N_u$ = total number of friends of u and let $N_v$=
Total number of friends of v
5: Let $M_u$ = number of male friends of u and let $M_v$ =
Number of male friends of v
6: gender_sim (u, v) = 1-| $(M_u/N_u) - M_v/N_V)$|
7: end if
8: if $a_i = a_2||a_i = a_3||a_i = a_4\ then$
9: $a_i = \{1\ if\ common\ else\ 0\}$
10: end if
11: if $a_i = a_5|| a_i = a_6||a_i = a_7||a_i = a_8||a_i = a_9$
then
12: $a_i = \frac{a_i = |a_i(u) \cap a_i(v)|}{|a_i(u) \cup a_i(v)|}$
13: end if
14: end for
15: attr_sim (u, v) = 1/9× $\sum_{i=1} 9\, a_i$
16: Add (v, attr_sim (u, v)) to a list Attribute_relation (u)
17: end for
18: return Attribute_relation (u)
```

Algorithm 4: Finding nearest neighbors from social network

```
Input: UserID $u$, List Interaction ($u$),
List $Attribute\_Relation$ ($u$)
Output: A list of nearest neighbors from social network with the social similarity with $u$
for each key v $\in$ $Attribute\_relation(u)$ do
    Search v in the lsitInteraction (u)
    if found then
        SN_Sim (u, v) = $\alpha \times Intr_{Sim(u,v)} + (1-\alpha) \times$ Attr_Sim (u, v)
    else
        SN_Sim (u, v) = (1- $\alpha$) × Attr_Sim(u,v)
    end if
    if SN_Sim (u, v)> $\chi$ then
        Add (v, SN_Sim (u, v)) to NN_SN (u)
    end if
end for
return $N N\_SN (u)$
```

4.3 Assigning Weight-ages to Two Nearest Neighborhood Sets

Third step is to assign weight-age to the nearest neighborhood sets N N _RS (u) and N N _SN (u) obtained from the first and second step explained previously. For a given user u, we find the number of elements in each nearest neighborhood set. Let *ratingNeighbors* represent the size of N N _RS (u) and *socialNeighbors* represent the size of N N _SN (u). Let the average number of ratings of user be R. If value of ratingNeighbors is very much greater than the value of socialN eighbors then more weight age is given to preferences of members of N N _RS (u). If the value of *socialNeighbors* is very much greater than the value of ratingNeighbors and the number of ratings done by the user u is greater than or almost equal to R then equal weight age is given to both the neighbor sets. When the value of *socialNeighbors* is very much greater than the value of *ratingNeighbors* but number of ratings of user u is very much lesser than R then more weight age is given to preferences of members of N N _SN (u). The complete steps of the above procedure are given in Algorithm 5.

Algorithm 5: Assigning weight-ages to two nearest neighborhood sets

```
Input: UserID u, URM R,N N_RS(u), N N_SN(U), Number of ratings of u=num_R, average number of ratings per user=avg_R
Output: $\beta$
if social neighbors>> ratingNeighbors and (num_R≈ Avg_R||num_R>avg_R) then
    $\beta$ = 0.5
else if ratingNeighbors >> socialNeighbors then
    $\beta$ =0.8
else if socialNeighbors>> ratingNeighbors and num_R<<avg_R then
    $\beta$ =0.2
else
    $\beta$ =0.5
end if
return $\beta$
```

4.4 Prediction and recommendation

Final step is to compute the predicted rating and recommendation. The rating given by a user for an item is predicted finally by finding the weighted sum of the ratings of neighbors from the rating pattern and social network. Weight-age is given by finding the value of β as described in the previous step. Prediction of the rating given for an item i by a user u is given by $P_{u,i}$ as shown below

$$P_{u,i}{=}r{-}_{u}{+}\beta{\times}\frac{\sum_{v\in N\,N_{RS(u)}}(r_{vi}-\bar{r}_{v})\times user_{sim(u,v)}}{\sum_{v\in N\,N_{RS(u)}}\left|user_{sim(u,v)}\right|}+$$

$$(1\text{-}\beta)\times\frac{\sum_{v\in N\,N\_SN(u)}(r_{vi}-\bar{r}_{v})\times SN\_sim(u,v)}{\sum_{v\in N\,N\_SN(u)}|SN_{sim(u,v)}|}$$

Top-N recommendations can be effectively done by ranking the items according to the prediction.

V. EXPERIMENTAL ANALYSIS

5.1 Dataset

We performed our experiment with the Movie Lens dataset and synthetic social network dataset. Movie Lens dataset is collected by the GroupLens Research Project at the University of Minnesota. We performed our experiment with the dataset containing 100,000 ratings on a scale of 1 to 5 from 943 users on 1682 movies (items). We adjusted the size of the dataset in order to achieve different levels of sparsity. The social network data is created synthetically to perform the experiment. However, we made sure that the data created matches closely with the characteristic of information available on the real social networks. We created 943 user nodes each with 9 attributes and added 15000 directed edges along with the interaction attributes on each edge.

5.2 Metrics

Statistical Accuracy metrics are used to evaluate the accuracy of prediction. In this paper we use Statistical Accuracy metric Mean absolute error (MAE) [7] to identify the drawbacks of the user based collaborative filtering recommender system. MAE is calculated as follows:

$$\text{MAE} = \frac{\sum_{i=1}^{N}|p_i - r_i|}{N} \tag{2}$$

Here, N is the number of actual ratings in an user-rating data, p1, p2, p3, p_n is the prediction of users rating and r1, r2, r3,...,r_n, is the corresponding real rating of users. Accuracy of prediction increases with decrease in value of MAE

We use one more metrics called coverage to measure the percentage of items for which algorithm can give predictions. Coverage is computed as follows:

$$\text{Coverage}=\frac{N_p}{N} \tag{3}$$

Where N denotes the set of available items and $N_p$denotes the set of items for which predictions can be made.

5.3 Experiment 1

First we conducted our experiment with traditional UBCF with the Movie Lens 100K dataset. The sparsity of ratings given to items by users present in the dataset is estimated to be $1 - (100,000/(943 * 1682))$ which is equal to 0.937 or 93.7%. Our experiments show that less than 0.05% of the total users have rated large number of items. Some users have rated average number of items. And there are more than 32% of the users who have rated very few items. In that case in our experiment on user based collaborative filtering, 32% of the users are not getting correct nearest neighborhood and prediction. To find the accuracy MAE is calculated on user based collaborative filtering. This experiment shows that MAE of users who rated very less item is high (0.87) compared to the MAE of people who have rated more number of items (0.78). The results reveal that UBCF performs comparatively well for users with more number of ratings.

5.4 Experiment 2:

In this experiment we show that the impact of two similarity thresholds used in user similarity calculation μ and social similarity calculation χ. We performed all the experiments by setting the threshold on minimum number of co-rated items (ncr) as 50. In each experiment we varied μ by keeping χ as constant. The results show that percentage of coverage at different values of μ and χ in Figure 4 to Figure 9. The conclusion is that by taking μ = 0.5 and χ = 0.25 maximum number of items are covered in a dataset with an average of 80% sparsity.

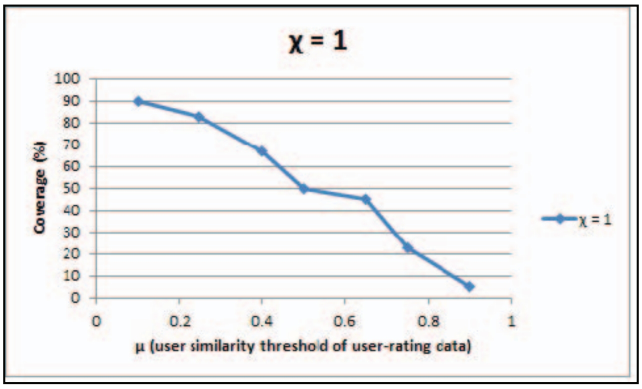

Figure 4: Percentage of coverage when χ = 1

5.5 Experiment 3

In this experiment, we introduced different levels of sparsity in the user-rating dataset. Initially the dataset was 95% sparse. We decrease the level of sparsity step by step till 35% in steps of 10%.

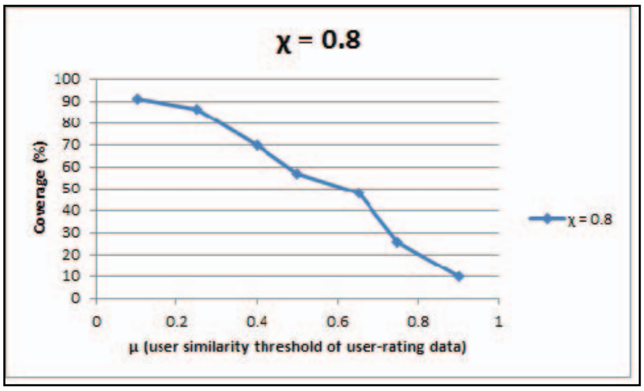

Figure 5: Percentage of coverage when χ = 0.8

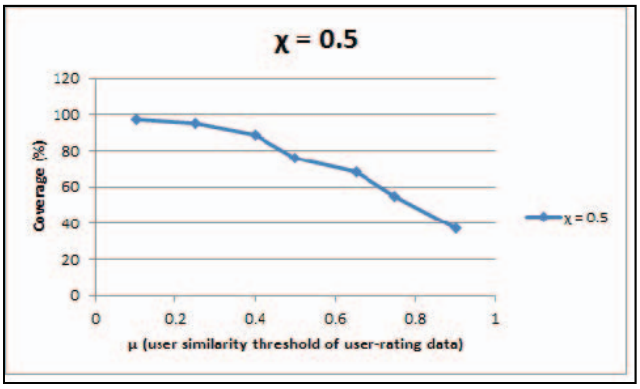

Figure 6: Percentage of coverage when χ = 0.5

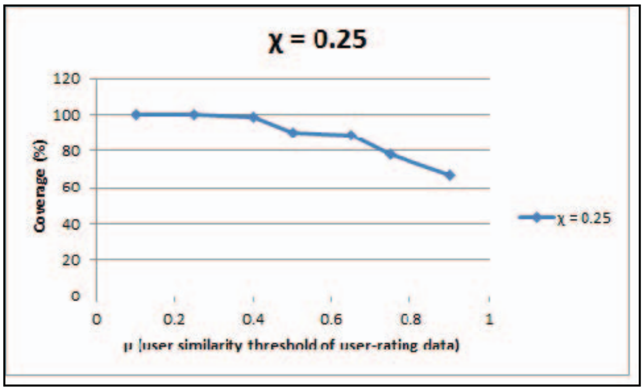

Figure 7: Percentage of coverage when χ = 0.25

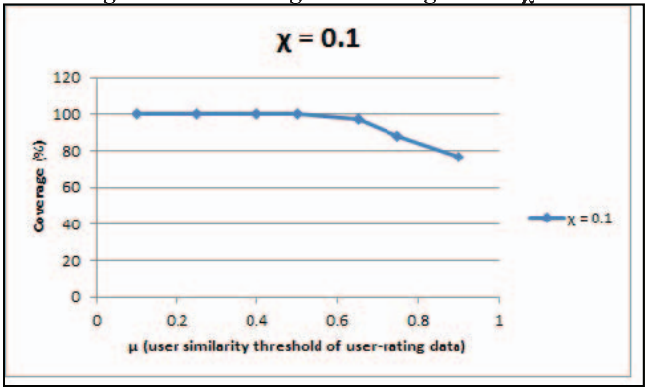

Figure 8: Percentage of coverage when χ = 0.1

We performed this experiment by setting the parameters ncr = 50, μ = 0.5 and χ = 0.25 in our proposed method. We compared our approach with UBCF. Results show that in both the algorithms, initially MAE decreases with decrease in sparsity, but after certain level MAE start increasing with decrease in sparsity. This is due to the impact of some additional neighbors which decreases the accuracy of prediction. Performance comparison of our approach with the traditional UBCF using Pearson correlation is shown in Figure 10. The result shows that MAE of our approach is less than UBCF at each level of sparsity. Our approach performs the best when sparsity is in range of 50-55%. If we still decrease the sparsity MAE start increasing again, but still our approach performs better than the traditional one.

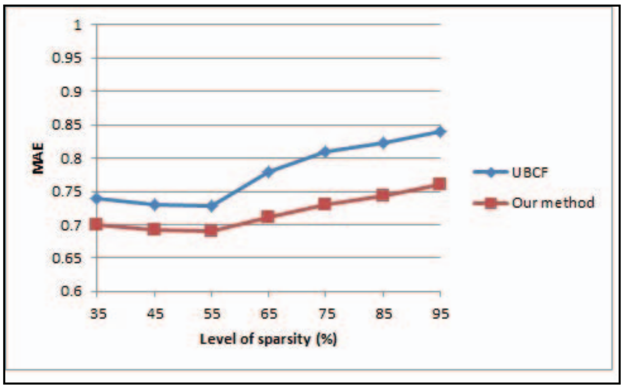

Figure 9: Mean Average Error

VI. CONCLUSIONS AND FUTURE WORK

In this paper we presented a novel approach to recommender systems by combining the collaborative filtering with social behavior of users for alleviating sparsity and cold-start problems. Our results prove that how data obtained from social network is useful under the circumstances where user-based collaborative filtering systems fail. This method employs a computational model which makes prediction by calculating the weighted sum of ratings obtained from socially related people and also the similar minded people from the rating patterns. The proposed approach acquires knowledge about social relations by extracting the different types of interactions between users, the frequency of interaction, timestamps and user profile similarities. The experimental result shows that the proposed approach works well under extreme level of sparsity and provides maximum coverage and works quite well for cold-start users.

The proposed approach does not consider trust beyond the second level. As a future work, we would consider propagation of trust in multiple levels. By incorporating efficient trust propagation in social network, we would be able to cover more people who would have rated a product for which we are trying to predict the ratings. In our approach rating data and social network data are stored in a centralized manner. As a second future work we would consider powering the recommendations with distributed storage and computation which will make recommendations much more efficient in terms of time to conduct predictions.

REFERENCES

[1] M. J. Pazzani. “A framework for collaborative, content-based and demographic filtering”. Artificial Intelligence Review, 13(5-6):393–408, 1999.

[2] Y.-Y. Shih and D.-R. Liu. Product recommendation Approaches: Collaborative filtering via customer lifetime Value and customer demands. Expert Systems with Applications, 35(1):350–360, 2008.

[3] B. Xie, P. Han, F. Yang, R.-M. Shen, H.-J. Zeng and Z. Chen Dcfla:” A distributed collaborative-filtering Neighbor-locating algorithm”. Information Sciences, 177(6):1349–1363, 2007.

[4] J. S. Breese, D. Heckerman and C. Kadie. “Empirical Analysis of predictive algorithms for collaborative filtering”, In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, pages 43–52, Morgan Kaufmann Publishers Inc., 1998.

[5] A. Umyarov and A. Tuzhilin: “Improving collaborative Filtering recommendations using external data”. In Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on, pages 618–627, IEEE, 2008.

[6] J. L. Herlocker, J. A. Konstan, A. Borchers and J. Riedl. “An Algorithmic framework for performing collaborative filtering”, In Proceedings of the 22nd annual international ACM SIGIR Conference on Research and development in information Retrieval, pages 230–237, ACM, 1999.

[7] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. “Using collaborative filtering to weave an information tapestry”, Communications of the ACM, 35(12):61–70, 1992.

[8] D. Billsus and M. J. Pazzani. “Learning collaborative information filters”. In ICML, volume 98, pages 46–54, 1998.

[9] B. Mobasher, R. Burke, and J. J. Sandvig. “Model-based Collaborative Filtering as a defense against profile Injection Attacks”. In AAAI, volume 6, page 1388, 2006.

[10] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. “Collaborative filtering by personality diagnosis: A hybrid Memory-and Model-based approach”. In Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence, Pages 473–480, Morgan Kaufmann Publishers Inc., 2000.

[11] G. Badaro, H. Hajj, W. El-Hajj and L. Nachman. “A hybrid approach with collaborative filtering for recommender systems”, in Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International, pages 349–354, IEEE, 2013.

[12] J. Golbeck. “Personalizing applications through integration of inferred trust values in semantic web-based social networks”. In Proceedings of Semantic Network Analysis Workshop, 2005.

[13] B. Sarwar, G. Karypis, J. Konstan and J. Riedl. “Item-based Collaborative filtering recommendation algorithms”. In Proceedings of the 10th international conference on World Wide Web, pages 285–295, ACM, 2001.

[14] M. Papagelis, D. Plexousakis and T. Kutsuras. “Alleviating the sparsity problem of collaborative filtering using trust Inferences”, In Trust management, pages 224–239, Springer, 2005.

[15] H. J. Ahn. “A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem”, Information Sciences, 178(1):37–51, 2008.