User Trust Aided Recommender System

Course Project: CSCI 59000 – Recommender Systems

Vachik Dave, Arvind Nair and Hitendra Rathod

Objective:

In literature there are two major ways for recommendation of any item to any user.

1. Build a system which directly recommend list of items based on some similarity and priorities.
2. Build a system that can predict user's view (rating) over different (new/unrated) items and suggest most interesting items.

Here, we are using the second method where we are predicting user's rating for specific item for better recommendation for the user.

> *Problem definition:* "For the given set of ratings of different items by users, try to predict his/her rating for new item and use that information to recommend unrated items to the user."

Pre-Processing of Data:

1. The Epinion data set (664824) is divided as 3 parts
 i. Training Dataset (70%)
 ii. Validation Dataset (15%)
 iii. Testing Dataset (15%)
2. First we take 1000 values at a time, and we store them in an array.
3. We shuffle the contents of the array randomly.
4. We then divide the contents of the array as 70%, 15% and 15% and store them into training, validation and testing dataset for every 1000 times.
5. In the end, we take the last 824 values and divide in 70%-15%-15% also.
6. We then sort the training, validation and testing arrays and then output the values into the training, validation and testing files.

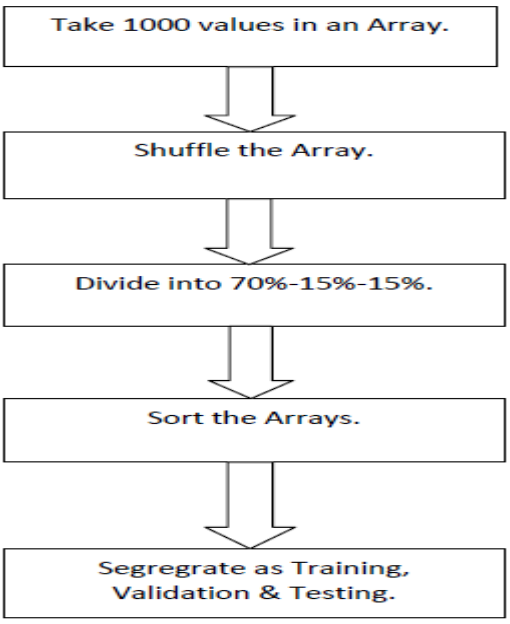

Figure 1: Flow chart for Pre-Processing of Epinion Dataset

Baseline Method 1:

Item-Item Similarity Method:

In this method we find the top items for recommendation using the cosine similarity of items with respect to one another by considering users who have co-rated the items.
How the algorithm works:
1. First we load the Training data in CSR representation by reading from the training file.
2. Then we take its transpose using CSR format and store the transpose in CSR format.
3. From the transpose, at a time in order we select 2 items to be compared.
4. For comparison, we look at the users who co-rated these two items and the total number of users who rated these items separately.
5. After that using the formula, $sim(a,b) = \frac{a.b}{|a||b|}$
where |a|=a.a and |b|=b.b using dot product.
6. Here the ratings are considered as 1 and if users have not rated those items then zero.
7. The similarities of all items with respect to all other items are stored in a cosine table.
8. This is done by first calculating the similarity of one particular item with all other items and then storing in a temporary array.
9. This array is then sorted using merge sort as per the cosine similarities in descending order.
10. In the cosine table we store the top similar items with respect to a particular item and a limit i.e., 200 items are specified to be stored in the table to use memory effectively and for speed.
11. Thus in the table top 200 (limit) items which are similar to each of the items is stored (from item 1 to 140000).
12. We now read the testing data into the CSR format.
13. For the top K unpurchased items, we store the K unpurchased items from the cosine table into a temporary cosine table for that particular user.

14. So if a user has brought 3 items, a temporary cosine table of 3XK will be created where the similarities between the 3 purchased items will not be stored.
15. From the temporary cosine table the total similarity of unpurchased items is calculated and the candidate set is generated, sorted using merge sort algorithm and stored in descending order of cosine similarity values.
16. Then we look at the testing data and store all the items the user purchased in the testing data and check with the candidate set of that particular user.
17. If the each of the items are present in the top N items of the candidate set then hits++ and the ARHR is calculated by considering the position of the item in the candidate set.
18. $Hit\ Rate\ (HR) = \frac{\text{hits}}{\text{number of users}}$

$Average\ Reciprocal\ Hit\ Rate\ (ARHR) = \text{ARHR} + (\frac{1}{1+\text{i}})$

where i is the index/position of the hit in the candidate set. ARHR is calculated and summed for all users.

19. Thus the Item-Item Similarity is implemented in baseline form.

Note:
1. If users or items are missing they are handled appropriately in the CSR representation. In CSR when we consider the difference in the row pointer to find out the users who rated items or items which are rated by users the difference is zero for that particular item or user if item is not rated by any user or if user has not purchased any item.
2. Limit is the columns or the number of items similar to each of the items present in the dataset. For each particular user then the purchased items are filtered out from the cosine table into the cosine table temporary for finding out similarity of Top K unpurchased items with respect to each purchased item.

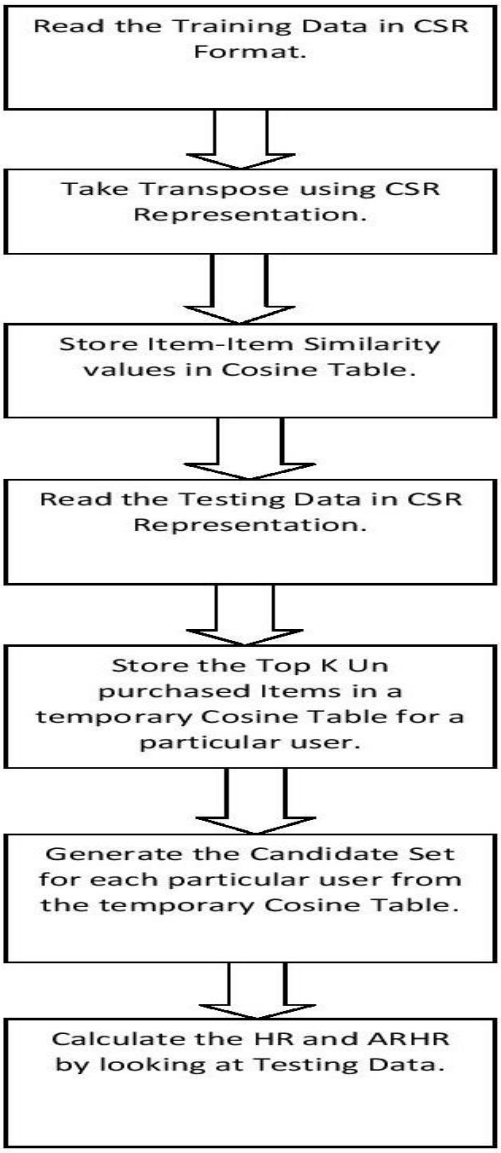

Figure 2: Flowchart of Item-Item Similarity

Top Un purchased Similar Items (K)	Top Candidate Items (N)	Cosine Similarity Calculation Time (ms)	Candidate Set Calculation Time (ms)	Hit Rate (HR)	Average Reciprocal Hit Rate (ARHR)
10	10	4687999.00	5160.00	0.0015188	0.0003236
20	15	3148942.00	32085.00	0.0020167	0.0002738
20	20	4085727.00	15577.00	0.0025894	0.0002738
25	15	4011800.00	79018.00	0.0021661	0.0002987
25	25	3095425.00	49557.00	0.0030874	0.0002987
30	15	3005803.00	67923.00	0.0018922	0.0002738
30	20	3185459.00	80173.00	0.0023902	0.0002738
30	30	3175794.00	71910.00	0.0033115	0.0002738

Table 1: Item-Item Similarity Results

Analysis Of Item-Item Similarity Results:

1. For increasing combinations of K and N values, the HR should be increasing which is the case.

2. Mainly it is seen to be increasing as N increases.

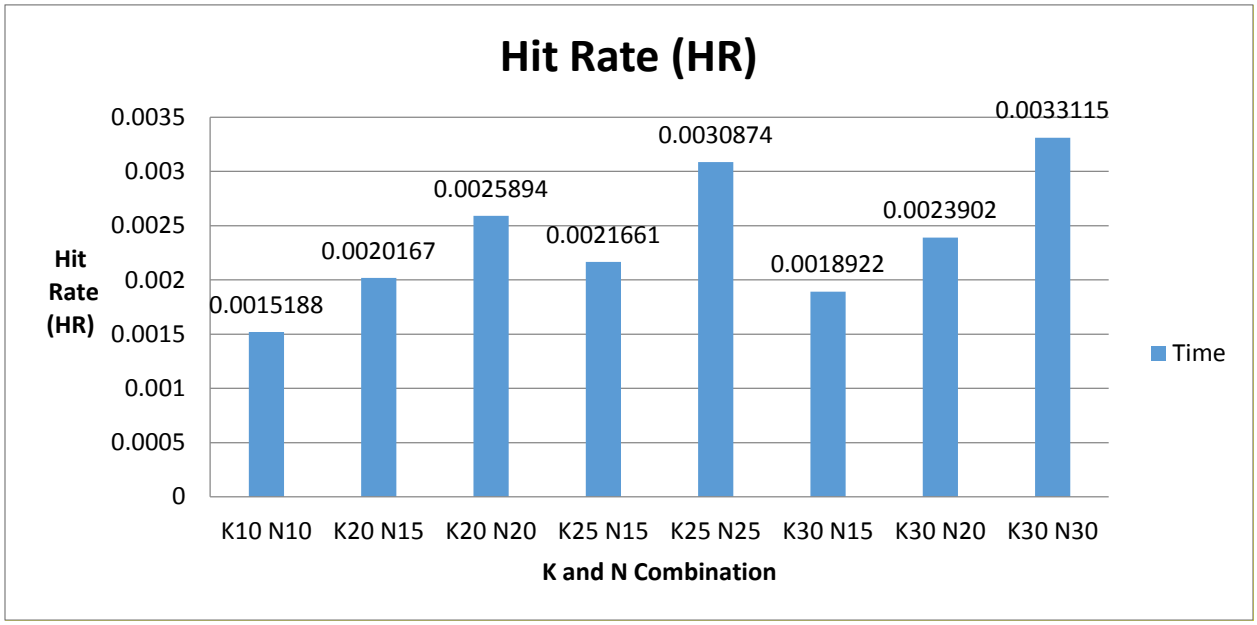

Figure 3: Hit Rate

The Hit Rate is the highest for K=30 and N=30 and lowest for K=10 and N=10.

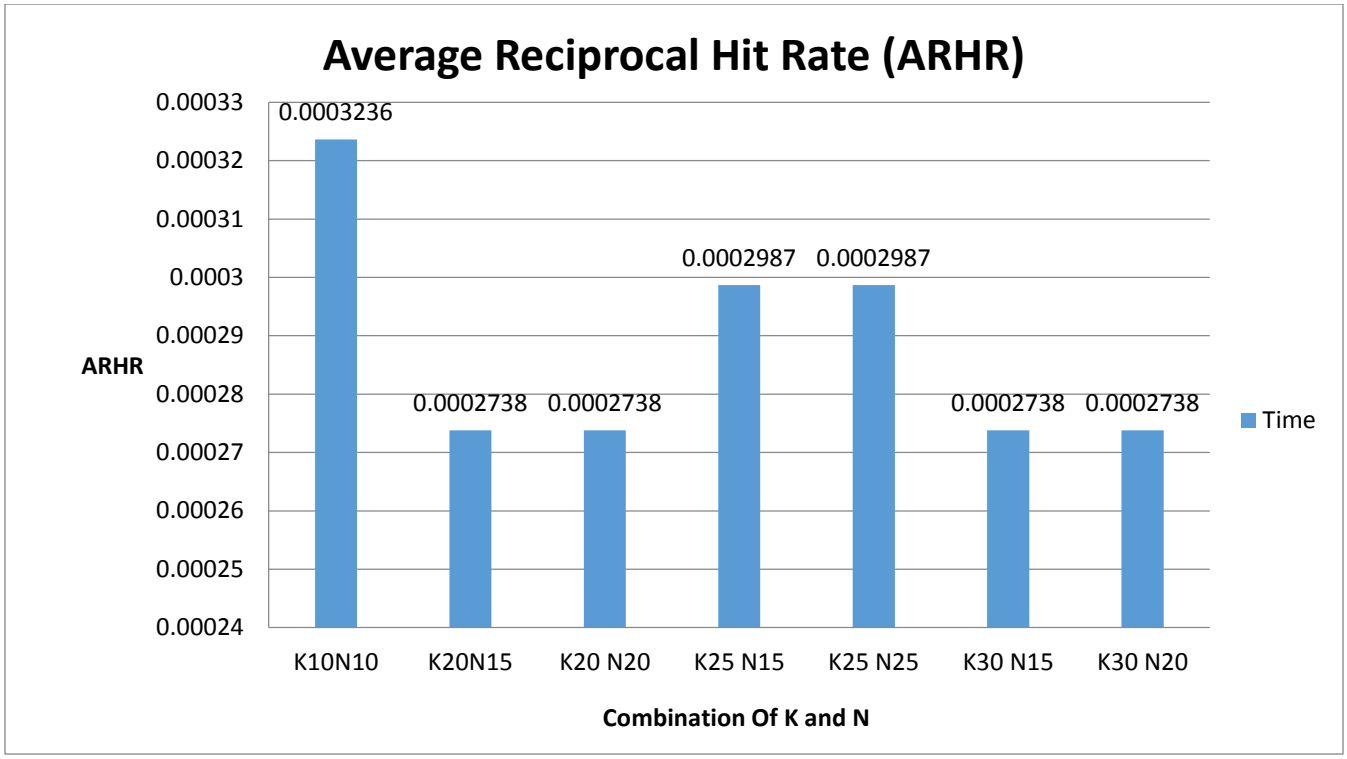

Figure 4: Average Reciprocal Hit Rate

ARHR is highest for K=10 and N=10 and remains same for the values of K. For values of N it changes slightly in the last decimal places.

Problems faced during the Item-Item Similarity Method:

1. The main problem faced was to optimize the speed of the algorithm so that we would get results in proper time.
2. The dataset of Epinion was huge enough to be time consuming to calculate the results.
3. The implementation of the algorithm had to be changed so that it would run in an acceptable speed of at least within 1 hour to 1 hour 18 minutes.
4. This took a lot of time and effort.
5. The limit is specified for cosine table as 200 which we may vary till higher values but the algorithm may be slower and may result in Java Heap Error.

Baseline Method 2:

Matrix Factorization Method:

In this method we optimize the objective function and learn the matrices P and Q at each iteration where P is the user factor matrix and Q is the item factor matrix using the Alternating Least Squares method (ALS).

How the algorithm works:

1. First we load the Training data in CSR representation by reading from the training file.
2. Then we take its transpose using CSR format and store the transpose in CSR format.
3. The objective function:

$$f(P,Q)= \sum_{(u,i)\in S} (r_{ui} - p_u q_i^T)^2 + \lambda(||P||_F^2 + ||Q||_F^2)$$

where r_{ui} is the rating given by user u for item i,

p_u is the k X 1 user factor for user u,

q_i is the k X 1 item factor for item i,

$p_u^T q_i = \sum_{h=1}^{k} p_u(h) \text{ X } q_i(h)$,

F is the Frobenius norm of P and Q,

$||P||_F^2 = \sum_{h=1}^{k} \sum_{j=1}^{m} P(h,j)^2$,

$||Q||_F^2 = \sum_{h=1}^{k} \sum_{j=1}^{n} Q(h,j)^2$,

$\sum_{(u,i)\in S}$ is sum over all the user-item ratings where $R(u,i)\neq 0$ and k is the latent dimension.

4. We solve the optimization function by using the alternating least squares method.
5. We first fix Q and solve for P using equation,

$$p_u= (\sum_{i|(u,i)\in S} (q_i q_i^T + \lambda I))^{-1} \sum_{i|(u,i)\in S} q_i r_{ui}$$

where $\sum_{i|(u,i)\in S}$ means sum over all the items that user u has ever rated,

λ is the control parameter

and q_i^T is the transpose of q_i .

6. On similar lines, we then fix P and solve for Q.
7. The LS_Closed function is called to execute step 5 and 6.
8. The objective function is calculated and checked to see if the difference is less than a specified value epsilon.
9. If it is less than epsilon which is 0.000001 then it will break out and if not it will continue in the iterations.
10. The iterations continue until a specified maximum number of iterations maxIters which is 100.
11. After either condition 9 or 10, the P and Q matrices have been learnt.
12. Now, the testing data is read in CSR Representation.
13. We look at the testing data, and for each particular user we try to use prediction to calculate MSE and RMSE.

14. For Prediction,

Prediction=P(u,.)*Q(.,i);

where P(u,.) is the 1 X k matrix for user u in P and

Q(.,i) is the 1 X k matrix for item i in Q.

5. The Mean Square Error (MSE) is calculated as,

$$MSE \mathrel{+}= (T(u,i)-Prediction)^2$$

for each user in the Testing dataset.

16. The total MSE is calculated as

MSE=MSE/Number of users in Testing Dataset

17. The Root Mean Square Error (RMSE) is calculated as

RMSE= √MSE

18. Then we call the hrarhr function to calculate the HR and ARHR values.

19. For each user, we calculate the prediction value using the dot product as discussed in step 14 for each of the un purchased items of user u.

20. Then we store the items in a candidate set and sort the items as per the descending prediction values using Merge-Sort algorithm.

21. Then, we look at the testing data for that particular user and check if the items purchased by him match those in the candidate set which we have generated.

22. In the candidate set, we consider only the top N values (we have done for N=30).

23. If the each of the items are present in the top N items of the candidate set then hits++ and the ARHR is calculated by considering the position of the item in the candidate set.

24. $Hit\ Rate\ (HR) = \frac{\text{hits}}{\text{number of users}}$

$Average\ Reciprocal\ Hit\ Rate\ (ARHR) = \text{ARHR} + (\frac{1}{1+\text{i}})$

where i is the index/position of the hit in the candidate set. ARHR is calculated and summed for all users.

25. Thus, the Matrix Factorization Method is implemented in baseline form.

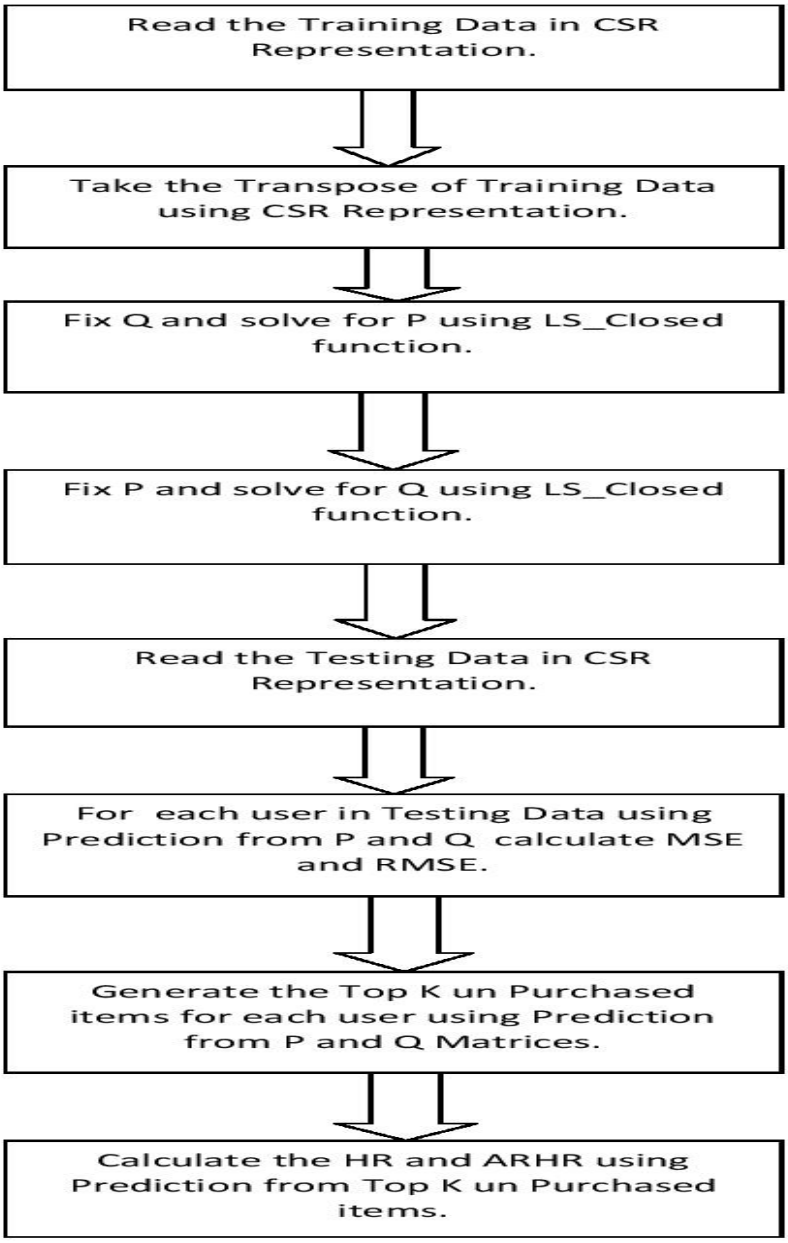

Figure 5: Flowchart for Matrix Factorization

Matrix Factorization Results on Epinion Dataset:

Latent Dimension (K)	Control Parameter (Lambda)	LS_Closed Time (ms)	Mean Square Error (MSE)	Root Mean Square Error (RMSE)	MSE RMSE Calculation Time (ms)	Hit Rate (HR)	Average Reciprocal Hit Rate (ARHR)	Hit Rate Time (ms)
1	3	5237445	1.5960	1.2633	47.00	0.0008465	0.00002489	616902
1	5	2971475	1.5544	1.2467	31.00	0.001269	0.00002489	731200
3	3	5462518	1.6234	1.2741	52.00	0.0008714	0.00004979	1086632
3	5	5442586	1.5546	1.2468	47.00	0.001518	0.00004979	590792
5	7	1696161	1.5209	1.2332	45.00	0.002589	0.00004979	636134
6	10	1002407	1.4811	1.2170	46.00	0.005726	0.00019918	1344894

Table 2: Matrix Factorization Results

Analysis of Matrix Factorization Results:

1. The HR values increase as the MSE reduces which validates the algorithm as for lower MSE values the HR should be higher.

2. For higher MSE values, the HR should be lower which is given by our program.

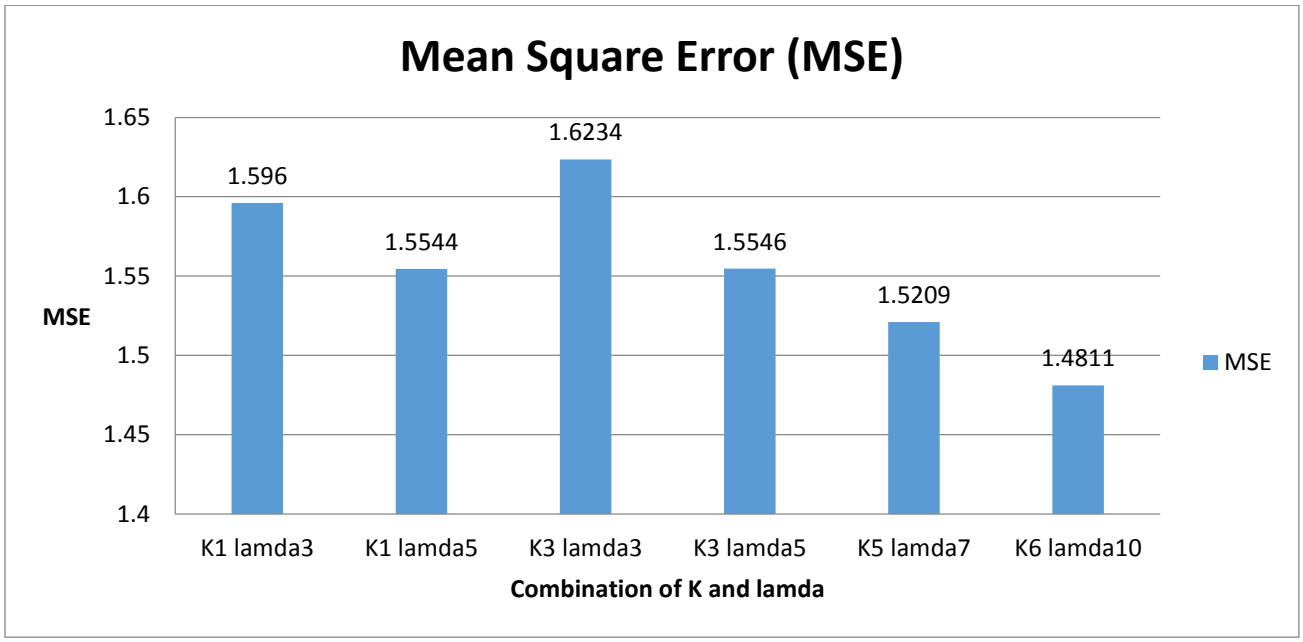

Figure 6: Mean Square Error

The MSE varies as per combinations of K and Lambda but is the lowest for K=6 and Lambda=10.

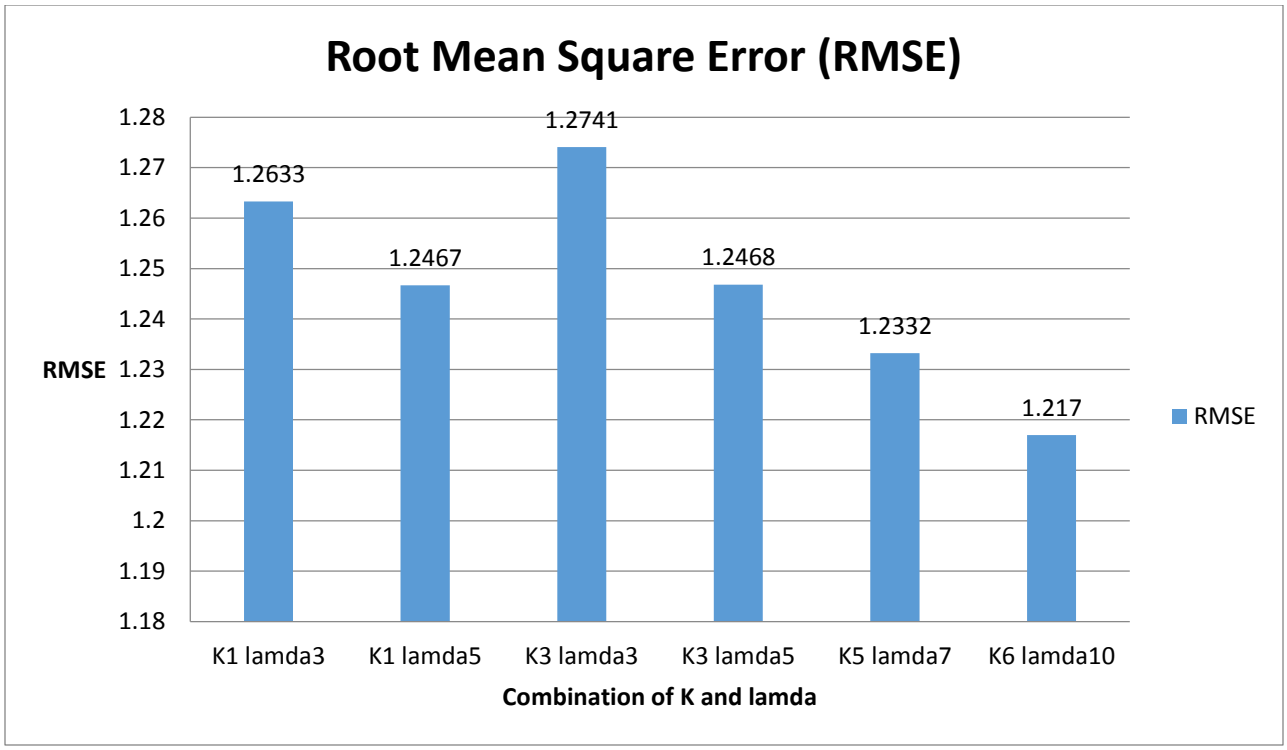

Figure 7: Root Mean Square Error

The RMSE varies as per the various combinations of K and Lambda and is lowest for K=6 and Lambda =10.

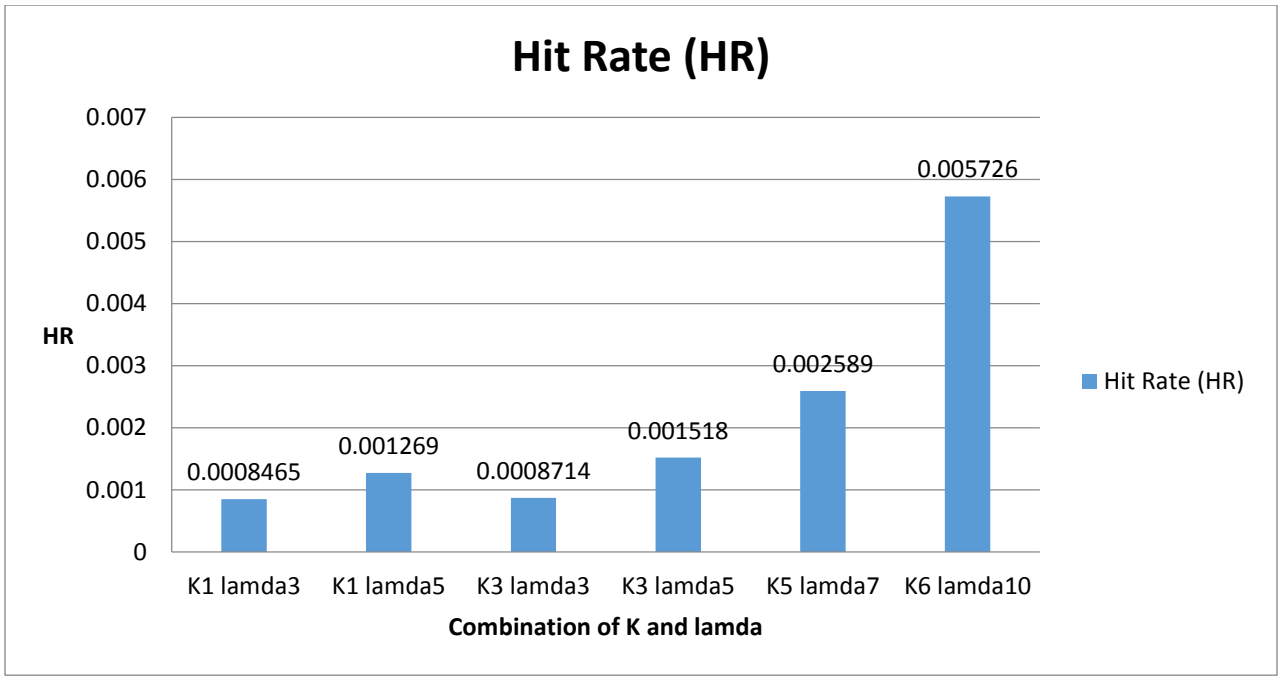

Figure 8: Hit Rate

The Hit Rate is the highest for K=6 and Lambda=10 which gives the lowest MSE.

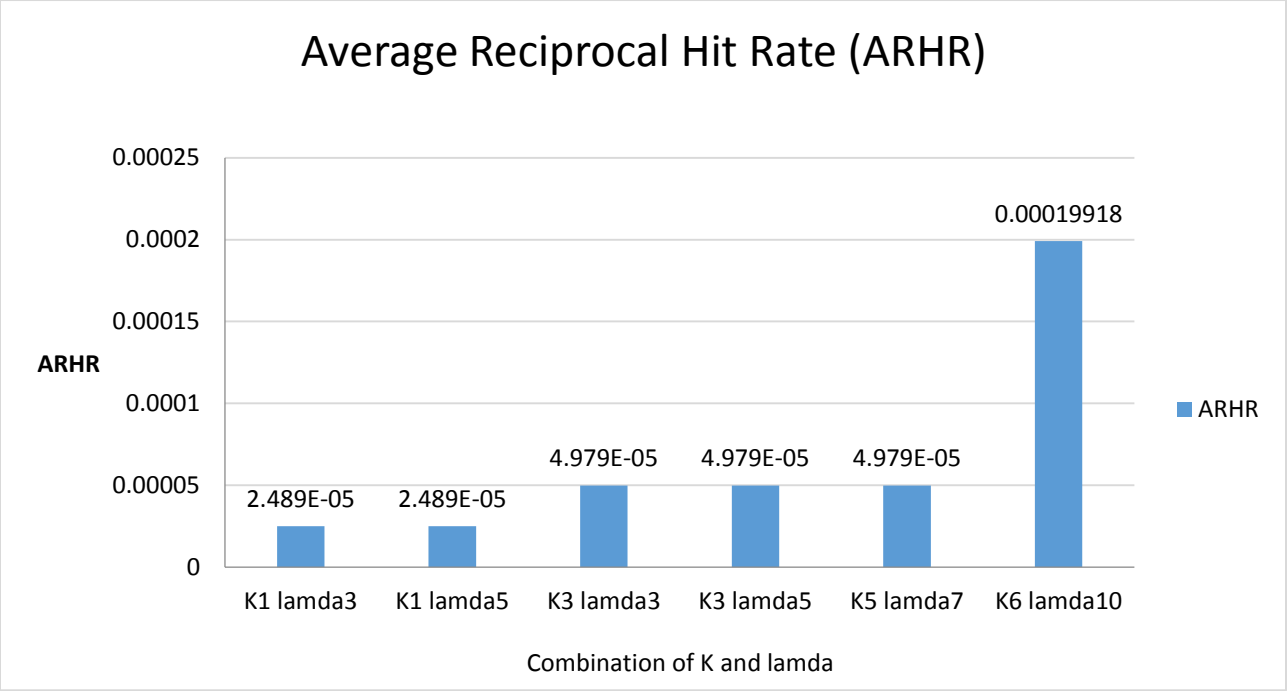

Figure 9: Average Reciprocal Hit Rate

The ARHR is the highest for K=6 and Lambda=10 which gives the lowest MSE.

Problems faced during the Matrix Factorization Method:

1. The main problem faced was to optimize the speed of the algorithm so that results will come in proper time.
2. The dataset of Epinion was huge enough to be time consuming to calculate the results.
3. The implementation of the algorithm (mainly the transpose representation and some loops) had to be changed so that it would run in an acceptable speed of at least within 1 hour to 1 hour 30 minutes.
4. This took a lot of time and effort.
5. The JAMA Matrix implementation was difficult to debug due to the size of the dataset.

Motivation:

Generally traditional recommender system methods only use historical information of different users to predict his/her rating and suggestion in items. But in real world, there are many other influences for a user to give specific rating to a particular item. Now out of all, one important and highly influential feature is friends and other known users review. It is highly probable that a friend of the user gave positive review about an item them the user may rate that item positively with good rating. Thus we are planning to incorporate some more information in the standard recommendation method and here we use trust among the users as additional information. This information provides a user trusts some set of other users. Now this information is really helpful in recommendation as well as prediction of rating, because in real world scenario generally a person trust his friend or other person who has similar interest/test in items. Hence rating of a trustworthy person is very important feature to use in any statistical or logical prediction model.

Additionally we have reviewed many related articles which used trust information for recommendation. Such as, Massa. P et.al. [1] and Ray, S. et.al. [3] have used trust information with collaborative filtering and Ma, H. et.al.[2] have used trust information in matrix factorization as regularization. There are many other researchers have used trust information in their articles [5] & [6] to improve the recommendation results.

Proposed Method:

For this project we used matrix factorization method for prediction of rating and build recommendation list. Matrix Factorization(MF) is very basic but powerful technique for this prediction. However, if we include more information as we explained above then it can improve its accuracy in prediction and generate better recommendation list. There are two possible ways to incorporate the trust information into MF:

1. Add the trust information in loss function.
2. Add the trust information as regularization term.

After reviewing few related articles [1], [2] and [3].We found out that trust information in loss function is never used by researcher, that encourage us to use trust information in loss function of MF. From related article by Jamali, M [7] have used trust propagation information in their recommendation model. Similarly article by Sanjog Ray et. al.[3], we observed interesting improvement can be done by trust propagation information. They have used trust propagation with specific threshold based on cosine similarity and incorporate this information in collaborative filtering. We have thought of trust propagation based prediction can be good base for predicting missing values to make rating matrix little less sparse. However, that predicted values wouldn't be very accurate, we decided to incorporate trust propagated values into objective function as loss function. Now in this new matrix factorization objective function the factors(user[P] & item[Q]) are learnt from rated values as well as propagated rating values with α weightage.

Trust Propagation:

Trust propagation is very simple method for information propagation in directed graph. There can be multiple ways to propagate the information among the network. We have used simple average propagation method, where we get mean of the propagated value among the friends (trusted users). For example as shown in figure 10(a) we have 4 users (A,B,C,D) out of which user A & user D have rated a certain item i_1 with value 4 & 5 (shown as bold value besides the vertex) respectively. However user B and C haven't rated the item i_1. Now propagation applies only for those users who haven't rated the item, here in our case user B & C. For single iteration propagation user B trusts user A & user C, but user C haven't rated the item hence user B receives only one value from user A which is 4(shown in italic font). Same for user C trusts user A & user D, hence mean propagated value is 4.5(shown in italic) [(4{A}+5{B}) /2]. Now for second propagation we again consider user without rating for the item i_1 (b & C). However for calculation we consider propagated valued also, that means for user B rating would be 4.25(shown in italic) in figure 10 (b). This value is coming from 4 of user A rating and 4.5 of user C propagated rating in 1st iteration, that is (4 + 4.5)/2 = 4.25. On the other hand as we don't have any user who haven't rated item i_1 , the value for user C would be same as previous iteration.

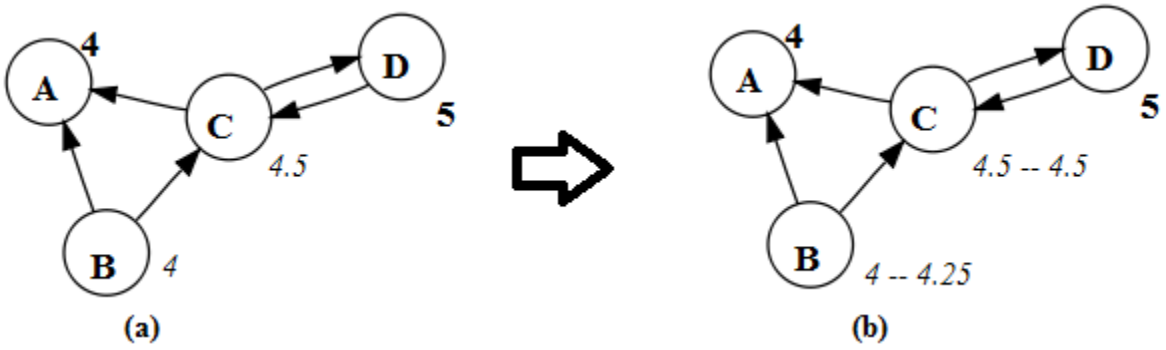

Figure 10. (a) Single iteration propagation; (b) 2 iterations of propagation

Here, first of all we discussed basic matrix factorization method and considered it as baseline method. Its results are compared with our proposed method.

- Matrix Factorization:

$$R \approx PQ^T \ldots [1]$$

Where,

R = Input Rating Matrix $(m \times n)$

P = User factor $(m \times k)$

Q = Item factor$(n \times k)$

- Objective function would be:

$$f(P,Q) = \frac{1}{2}\sum_{u,i}\left(I_s\left(r_{u,i} - p_u q_i^T\right)^2 + \alpha \times \hat{I}_s\left(\widehat{r_{u,i}} - p_u q_i^T\right)^2\right) + \frac{\lambda}{2}\|P\|_F^2 + \frac{\lambda}{2}\|Q\|_F^2 \ldots [2]$$

Where,

$r_{u,i}$ = value at row *u* and column *i* in input rating matrix *R*.

I_s = function, returns 1 if dataset has non-zero value for current P_u and Q_i rating, else 0.

$\hat{I}_s$ = function, returns 1 if propagated matrix has non-zero value for current P_u and Q_i rating, else 0.

$\widehat{r_{u,i}}$ = value at row *u* and column *i* in propagated estimated matrix $\hat{R}$.

$\|P\|_F$ = Frobenius norm of matrix *P*.

$\|Q\|_F$ = Frobenius norm of matrix *Q*.

Here, the objective is to minimize the value of $f(P,Q)$. For that we used alternating least square (ALS) method where we first fixed values of matrix *Q* and try to find minimum value for *P* and then fixed *P* and find minimum value for *Q*.

- Fix *Q* and solve *P* , the equation 2 becomes

$$f_p(P) = \frac{1}{2}\sum_{u,i}\left(I_s\left(r_{u,i} - p_u q_i^T\right)^2 + \alpha \times \hat{I}_s\left(\widehat{r_{u,i}} - p_u q_i^T\right)^2\right) + \frac{\lambda}{2}\|P\|_F^2 \ldots [3]$$

- Taking partial derivative of equation 3 with respect to p_u and set it to 0.

$$\frac{\partial f_p(P)}{\partial p_u} = -\sum_i \left[I_s\left(r_{u,i} - p_u q_i^T\right)q_i + \alpha \times \hat{I}_s\left(\widehat{r_{u,i}} - p_u q_i^T\right) q_i\right] + \lambda p_u = 0$$

$$p_u = \left(\sum_i \left[I_s q_i r_{u,i} + \alpha \times \hat{I}_s q_i \widehat{r_{u,i}}\right]\right) \times \left(\lambda \mathrm{I} + I_s q_i^T q_i + \alpha \times \hat{I}_s q_i^T q_i\right)^{-1} \ldots [4]$$

Equation 4 is a closed form solution for problem in equation 3. Now if we fixed values of *P* and solve *Q* we get similar equation, which would be as below:

$$q_i = \left(\sum_i \left[I_s p_u r_{u,i} + \alpha \times \hat{I}_s p_u \widehat{r_{u,i}}\right]\right) \times \left(\lambda \mathrm{I} + I_s p_u^T p_u + \alpha \times \hat{I}_s p_u^T p_u\right)^{-1} \ldots [5]$$

Second Version Proposed:

After getting good result for this modified objective function, we started thinking how can we improve the results little more? In this equation we are incorporating trust network information to estimate zero rating (not predicted) values. However there is no change or impact of the trust information on already rated items, hence we actually partially ignoring trust information for rated values. This thought leads us to our second proposed method where we used trust information generally with all rated and non-rated items. The idea is we can extend the use the trust information by including it into regularization term and let the model learn in more

constrained environment. That means initially model doesn't use trust network for improving non-zero rating, for which I added one more regularization term. The new objective function is:

$$f(P,Q) = \frac{1}{2}\sum_{u,i}\left(I_s\left(r_{u,i} - p_u q_i^T\right)^2 + \alpha \times \hat{I}_s\left(\widehat{r_{u,i}} - p_u q_i^T\right)^2\right) + \frac{\lambda}{2}\|P\|_F^2 + \frac{\lambda}{2}\|Q\|_F^2 + \frac{\beta}{2}\sum_{f \epsilon F_u^+} Sim(u,f)\left\|P_u - P_f\right\|_F^2 \quad \ldots [6]$$

Where,

F_u^+ = user P_u trusts this set of users.

- Here, again we used alternating least square method for finding closed form solution of parameters. Fix Q and solve P , the equation 6 becomes

$$f(P,Q) = \frac{1}{2}\sum_{u,i}\left(I_s\left(r_{u,i} - p_u q_i^T\right)^2 + \alpha \times \hat{I}_s\left(\widehat{r_{u,i}} - p_u q_i^T\right)^2\right) + \frac{\lambda}{2}\|P\|_F^2 + \frac{\beta}{2}\sum_{f \epsilon F_u^+} Sim(u,f)\left\|P_u - P_f\right\|_F^2 \quad \ldots [7]$$

- Taking partial derivative of equation 3 with respect to p_u and set it to 0.

$$\frac{\partial f_p(P)}{\partial p_u} = -\sum_i \left[I_s\left(r_{u,i} - p_u q_i^T\right) q_i + \alpha \times \hat{I}_s\left(\widehat{r_{u,i}} - p_u q_i^T\right) q_i\right] + \lambda p_u + \beta \sum_{f \epsilon F_u^+} Sim(u,f) \times (p_u - p_f) + \beta \sum_{g \epsilon F_u^-} Sim(g,u) \times (p_u - p_g) = 0$$

$$p_u = \left(\sum_i \left[I_s q_i r_{u,i} + \alpha \times \hat{I}_s q_i \widehat{r_{u,i}}\right] + \beta \sum_{f \epsilon F_u^+} Sim(u,f) \times p_f + \beta \sum_{g \epsilon F_u^-} Sim(g,u) \times p_g\right) \times \left(\left[\lambda + \beta \sum_{f \epsilon F_u^+} Sim(u,f) \times p_f + \beta \sum_{g \epsilon F_u^-} Sim(g,u) \times p_g\right] \mathbf{I} + I_s q_i^T q_i + \alpha \hat{I}_s q_i^T q_i\right)^{-1} \quad \ldots [8]$$

This equation 8 is again a closed form solution for objective function in equation 7. If we look at equation 6 carefully partial derivative of the equation with respect to q_i would be same as equation 5 as trust information doesn't make any change in item factor values.

Dataset:

Here for the project experiments we have used well-known and freely available "Epinion dataset". It has mainly two kind of information:

1. User – Item rating values
2. User –User trust network

Statistics:

users = 49,290 → 40,163 (active)

items = 139,738

rating = 664,824 very very sparse (0.00965% dense)

Avg. number items rated by user = 4.7576

Avg. degree = 11.0252 (active users)

Max out degree = 1723

Max in degree = 2500

Training data 70% = 465376 ratings

Validation / Testing 15% (both) = 99723 ratings

Experimental Results:

For evaluation of out experiments we used standard error matrixes Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\tilde{R}_i - R_i)^2$$

RMSE is square root of the MSE value.

Another evaluation criteria used for accessing good recommendation are Hit Rate (HR) and Average Reciprocal Hit Rank (ARHR).

$$HR = \frac{\#\ correctly\ recommended\ items}{\#\ of\ total\ test1ing\ items}$$

$$ARHR = \frac{1}{n}\sum_{i=1}^{h} 1/_{rank_i}$$

Parameter Selection:

For given model we need to adjust many parameters to get best result out of the model. For parameter selection we evaluated validation set and run the set of parameters for which model is performing best on validation set.

With the help of Prof. Xia Ning we came to know that intrinsic dimension of our training dataset is 8. Hence we prefer to select K value between 1-8 for optimal results of the model. However just for confirmation we also check results for K = 10. In fugure 1y-axis shows values

of MSE, while x-axis labels are showing values of Lemda (λ) of equation 2 with preceding by letter 'L and after hyphen value of K is shown. Different values of alphas (α) are plotted accordingly, where MF_basic is simply matrix factorization value by putting alpha value as zero.

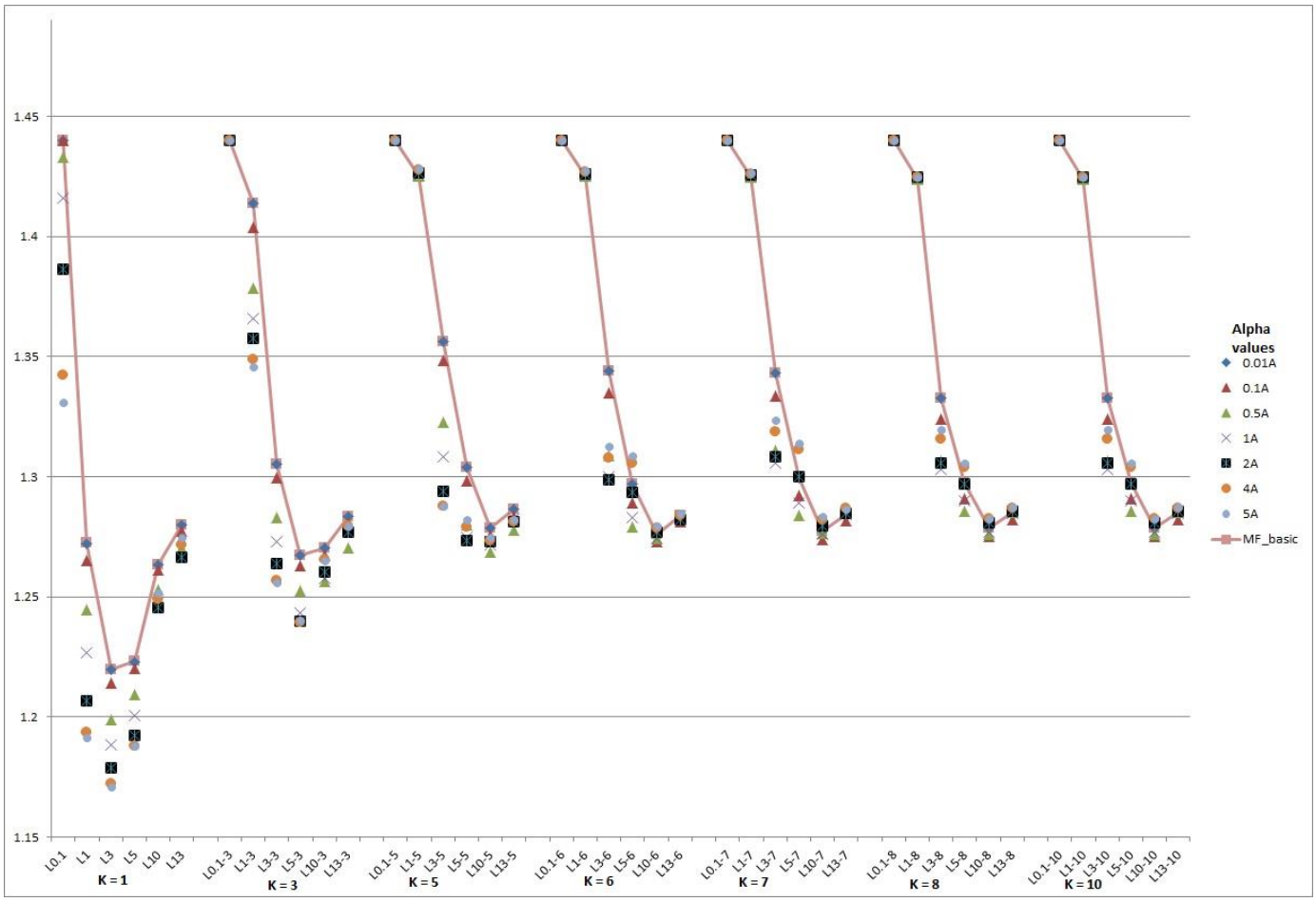

Figure 11: MSE results on Validation

Figure 11 shows clear improvement of proposed method with best alpha basic MF method on validation data for best selected alpha value. From these results we continue comparing Basic MF method result produced by Arvind for different values of K & Lemda (λ). For comparison on testing data we used only best alpha derived from validation data experiment.

Here in figure 12 we show the standard trend that proves validation always give better results than testing data. That is the reason to use validation for parameter setting for fair evaluation of the model.

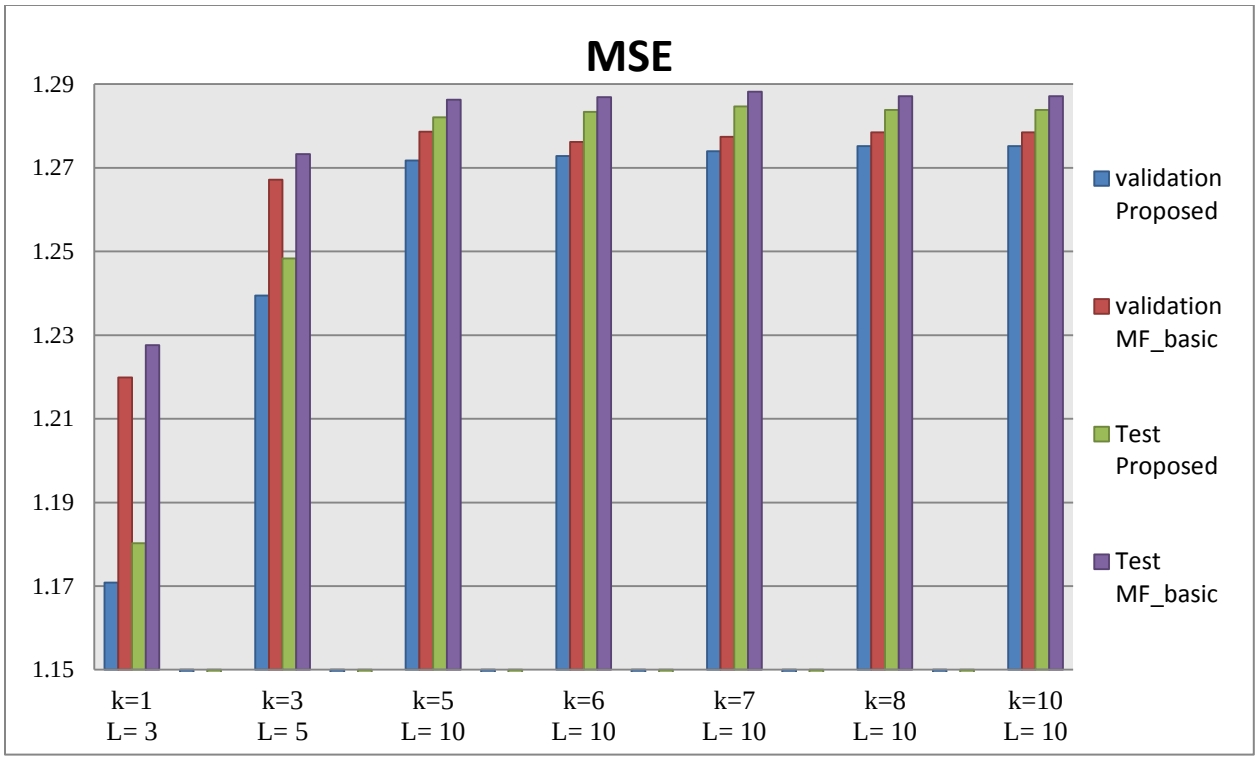

Figure 12: Results on validation and testing datasets

We also have results from Arvind's implementation of MF, figure 13 shows comparison of proposed method and Basic MF results. It clearly outperforms the results of basic matrix factorization method for both evaluation criteria MSE & RMSE.

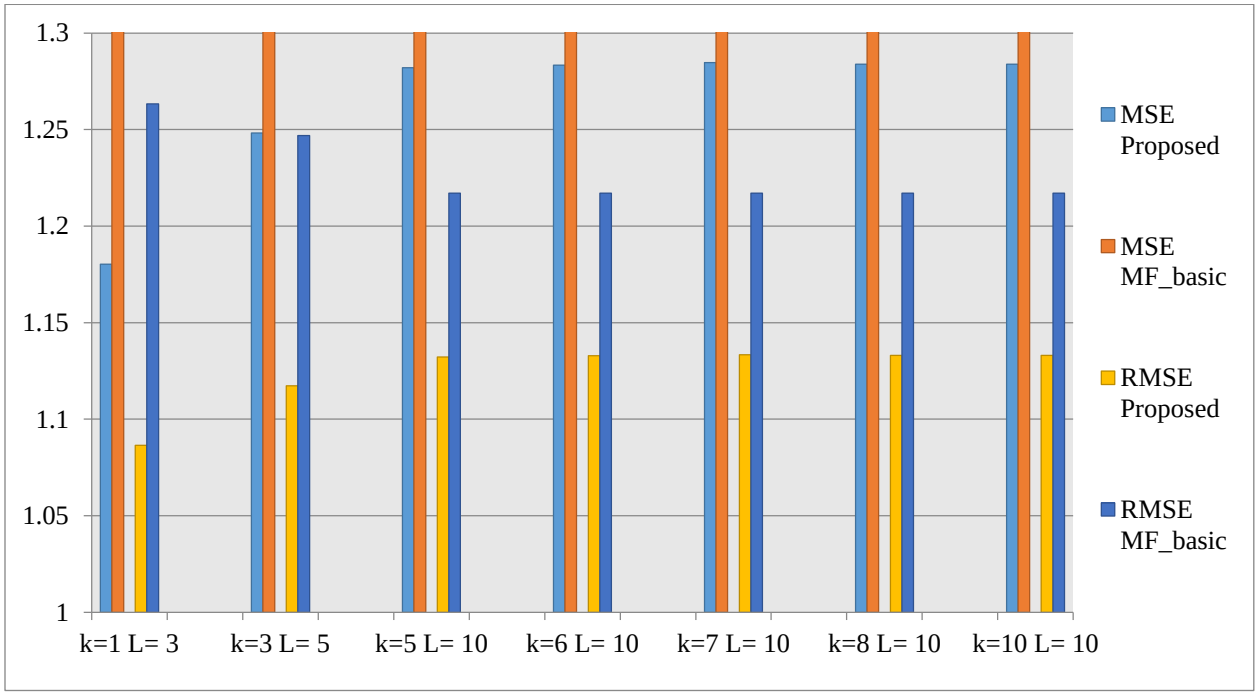

Figure 13: MF vs Proposed (MSE / RMSE).

	MSE		RMSE		Hit Rate		ARHR	
	Proposed	**MF_basic**	**Proposed**	**MF_basic**	**MF_basic**	**Proposed**	**MF_basic**	**Proposed**
k=1 L= 3	1.180252	1.596018	1.086394	1.263336	0.000847	0.005896	0.00002489	0.000367
k=3 L= 5	1.248303	1.554639	1.117275	1.246852	0.001518	0.016666	0.00004979	0.002172
k=5 L= 10	1.282036	1.481154	1.13227	1.217027	0.005727	0.024157	0.0001992	0.003662
k=6 L= 10	1.283319	1.481156	1.132837	1.217028	0.005726	0.024538	0.0001992	0.00362
k=7 L= 10	1.284687	1.481156	1.13344	1.217027	0.005727	0.024578	0.0001992	0.003653
k=8 L= 10	1.283793	1.481156	1.133046	1.217027	0.005727	0.024017	0.0001992	0.003717
k=10 L= 10	1.283793	1.481155	1.133046	1.217027	0.005727	0.024017	0.0001992	0.003717

Table 3: Results of Proposed Method vs Basic MF

Table 3 shows all results for proposed method and matrix factorization. For all these experiment finding HR & AHR value N = 50 that is top 50 recommendation are considered. We can also see by comparing table 3 results for HR & ARHR that it has improved both of these values by notable amount. We can see the improvement of HR & ARHR in bar chart in figure 14.

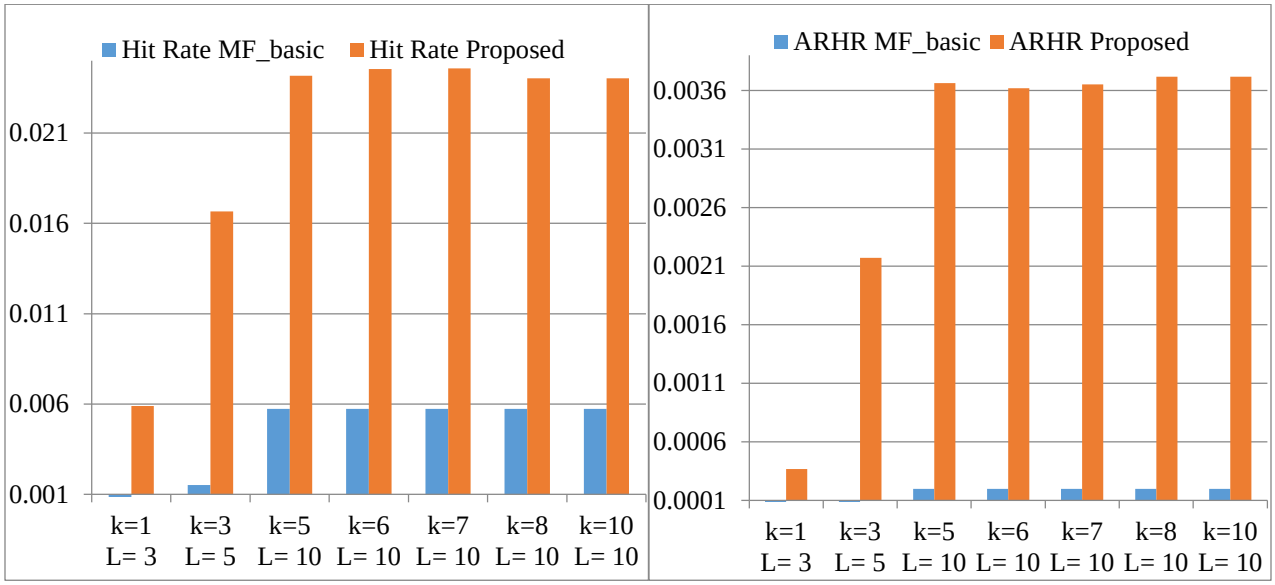

Figure 14: Proposed Method Vs Basic MF (HR & ARHR)

If we compare our proposed method with item-item similarity (collaborative filtering) method then best result for HR from table 1 is 0.0033115 while proposed method can give HR 0.024578 (table 3), which is an order of better result than collaborative filtering method. If we compare ARHR then also 0.003717(table 3) is an order of higher value than collaborative filtering method for which best value is 0.0003236(table 1).

Discussion & Future Work:

The results clearly show that incorporating trust information in loss function has clear benefit on result by all evaluation criteria. Hence we as per our assumption trust network gives really important information for recommendation and prediction of rating by any user. However,

Second version of proposed work is still need to verify and prove its better. I believe that incorporating trust in both loss function and regularization helps to improve result and may perform better than first proposed method. Because limitation of time, I could not completely implement and make it working correctly. However I submitted the code for that in my submission.

We can clearly see that HR & ARHR value is not too high or is better than base line models still not good in general. The reason I believe is too sparse matrix with very high number of items in recommendation list, user in test set cannot rate all, hence if we look at average number of items rated by users that is less than 5 which shows user cannot rate all those items which he/she likes hence recommender system put those unrated but user liking items in front. This may lead to very poor hit rate and ARHR.

Parts Done by Each one of the Team Members:

- The data pre-processing was formulated by Vachik Dave and the missing users were removed by Vachik Dave and pre-processing program was implemented by Arvind Nair.
- All data Statistics are collected by Vachik Dave.
- The Item-Item Similarity method (excluding Item CSR Transpose) was implemented by Hitendra Rathod.
- The Item-Item Similarity CSR Transpose was implemented by Arvind Nair.
- The Matrix Factorization Method (excluding CSR Input Representation) was implemented by Arvind Nair.
- The CSR Input Representation for Matrix Factorization Method was implemented by Hitendra Rathod.
- Finally, Vachik Dave has proposed both methods and all related experiments, documentation and presentation.

Reference:

[1] Massa. P. and Avesani, P. (2004). *Trust-Aware Collaborative filtering for Recommender Systems*, Lecture notes in Computer Science, Vol. 3290, pp. 492-508.

[2] Ma, H., Zhou, D. et. al. (2011). *Recommender Systems with Social Regularization*, ACM international conference on web search and Data-mining, pp. 287-296.

[3] Ray, S. and Mahanti, A. (2010). *Improving Prediction Accuracy in Trust-Aware Recommender System*, Hawaii International conference on System Sciences, pp. 1-9.

[4] Epinions Dataset: http://www.trustlet.org/wiki/Epinion_dataset.

[5] Massa. P. and Avesani, P. (2007). *Trust-Aware Recommender Systems*, ACM conference on Recommender systems, pp. 17-24.

[6] Ray, S. (2012). *Identifying Influential Taggers in Trust-Aware Recommender Systems*, International Conference on Advances in Social Network Analysis and Mining.

[7] Jamali, M. and Ester, M. (2010) *A matrix factorization technique with trust propagation for recommendation in social networks*, ACM conference on Recommender systems, pp. 135-142.