Arizona State University

Project Title: Exploiting side information in Collaborative Filtering

(Increasing the quality of pure user based CF by using different methods)

Submitted in Partial Fulfilment of

Semantic Web Mining: CSE 591

Vaibhav Patel	Aakanxu Shah	Rutvik Patel
vppatel1@asu.edu	apshah7@asu.edu	rutvik.patel@asu.edu
ASU ID : 1208649516	ASU ID : 1208663179	ASU ID : 1208668184

Index Page No

1. Abstract

Today with the booming of use of internet, many services are provided by many different providers in different domains. With so many choices available in the terms of products or services available on particular website it becomes extremely important to direct the user to particular product or service in order to attract them to continue using the website. Recommender system can come to rescue in such situations. Recommender system help direct the user to particular item based on using different algorithms based on context. There are basically two types of recommender systems used i.e. content based and collaborative filtering based approach. Collaborative filtering has become the widely used algorithm nowadays. We planned to use side information of items to pure CF approach to generate better results than pure CF based algorithm. We have also included other method of improving the output of pure user-user CF by making the input matrix denser by adding the recommendations generated from pure item-item CF.

2. State of the Art

2.1 Recommender System

Motivation and introduction

- Growth of Internet in Our Lives

Today the internet has become an important aspect of our daily lives. Today Internet provides wide array of functionalities for different purpose. The functionalities provided by the internet and people utilizing them are increasing at a rapid rate. Total number of internet users have surpassed 3 billion count [1] and still growing at fast pace. The amount of data and information on internet is also simultaneously growing at tremendous speed.

- Growth of Ecommerce Sites

With this amount of information on internet it becomes extremely important to map and direct user to particular product or page they may be interested. Today ecommerce websites have also grown with tremendous amount. Hence it nowadays becomes extremely important to direct the users to the products they may like in order to make them continue to use the service. Hence recommender system can come to rescue in such scenarios where this type of systems can effectively and efficiently map the user to items they may like from wide array of items. Today many ecommerce websites like Amazon and Netflix have successfully implemented large scale recommender system to increase and manage their business efficiently. Most of the ecommerce websites are deploying recommender system nowadays to provide the users with products and functionalities they would like along with increase in that website's business.

Recommender System basically directs the users to their preference from the available choices by using various algorithms tailored for different purpose and context. Hence role of recommender system is to provide efficient and effective suggestions to the users such that users continue to use the service or buy product from that website creating a kind of win-win situation for both users and companies. These suggestions help the users in decision making process, like which products to purchase, which books or news to read, which music to listen and many more. Hence recommender system help the user to choose best item out of many alternatives.

2.2 Types of recommender system

Recommender system can be classified into two main categories depending upon the information they exploit to generate recommendation. This two main types are content based and collaborative filtering.

2.2.1 Content based recommender system

This type of recommender system uses the past preference of the user to suggest the items from the array of available items. Hence content based recommender suggest the similar products to the user that he/she have liked in the past. Content based recommender matches the attributes of any product with the attributes in the user profile obtained from the past preferences. Attributes for user profile can be obtained explicitly or implicitly. In explicit method the attributes are obtained from the feedback the user gives to any item in form of like, rating or comment. In implicit feedback methods the active user involvement is not required and the feedback can be obtained from monitoring and analysing user activities.

2.2.2 Collaborative Filtering Approach

In this type of recommender system instead of using the past preference of the user directly, preference of similar users (in the sense users who have shared similar taste in the past with that particular user) is utilized to generate the suggestions and predictions for items. This method works under the assumption that 'Users which share similar taste in the past will continue to share the similar taste in the future'.

There are various ways of finding out the similar users and generating recommendations which can be subcategorized into memory based and model based collaborative approaches.

Memory based collaborative approach.

This type approach is further classified as item based and user based. In user-user or user based collaborative filtering method the similarity between users is calculated to find the similar users which are then used to predict the rating for any item I for user U. Various well known approaches are available for finding similar users, like Pearson coefficient, Spearman coefficient and cosine similarity. After finding the similarity between the users the top k similar users or users above some threshold similarity value are considered for generating prediction for any item for particular user.

Item-Item Collaborative filtering

In contrast to user based collaborative filtering this approach deals with finding the similarity between each candidate items and the items that any particular user has rated. Generally Cosine similarity is used to find the similarity between items.

Drawbacks of memory based approach

Process of finding similarity between users is costly as it takes quadratic time complexity. And other drawback is that the results depends on approach used for finding similarities, which may depends on suboptimal relation between users in user based and items in item based collaborative approach.[2]

Model Based Collaborative Filtering

Model based approach deals with training the models based on whole or part of U-I matrix and generating predictions from this models. Conventional model based approach includes Bayesian network model, mixture model and latent semantic model. In recent past, matrix factorization approach for model based CF has gained wide attention due to Netflix challenge and its benefits in terms of scalability.

2.3 Main Problems in Recommender system.

Recommender system faces different limitations depending upon the type of approach used. Major types of problems faced during recommender system built-up can be classified as follows.

2.3.1 Cold Start Problem

Recommender System is dependent on the data of users, items or user-item interaction to generate recommendations. There are situations where sufficient data is not available for generating recommendations depending upon the algorithms used. This problem can be largely categorized into two types as follows.

1) New User in the system

When a new user enters the system there is very scarce information available about the user, as user may have rated very few items. So, in such situations it becomes difficult to generate realistic recommendations. Both the main approaches that is content based and collaborative filtering suffer problem of some amount due to cold start problem.

2) New Item in the system

Whenever new item is introduced in the system there are no ratings associated with it. Hence there is no feedback of users is available on that item. Than in such situations CF faces serious problems as no user is associated with this new item, it faces hard time recommending this item to any user.

While content based approach can tackle this problem as it can extract the features of the item based in the information of that item to match with the user’s preference features to generate recommendations.

2.3.2 Data or U-I Matrix Sparsity Problem

CF based approach faces this problem as there may be many items which are rated by only few users, which can make this product less recommended item than the items having high number of ratings. CF based approach finds difficult to find the similar users for the items having less number of ratings which results in less recommendation of items having less number of ratings even though this few ratings may be very high rated.

Content based systems does not face this problem generally.

2.3.3 Serendipity in Recommendations

Most of the effort has been paved in improving the accuracy of the recommender systems, but many times most accurate recommendations are not the best recommendations. [3, Konstan].

Content based recommendations are poor in terms of generating serendipity in recommendations as they are based on providing the recommendations of the products depending upon users past preferences only. While Collaborative filtering based recommendations can recommend the product to the user from the domain which he/she has not used or rated in the past.

For example, considering the scenario of book recommendation system, content based recommender system mostly generated recommendations from the same genre that the user have read and rated the books in the past leading to poor serendipity in recommendations. While CF based approach where similarity between users is taken into account while generating recommendation can recommend the book to the user from genre which he/she has not read in the past but may like it as others users with similar taste have like it.

2.3.4 Overspecialization in recommendation

In content based approach, the system tries to find the most similar match of product with the user profile. This may result in similar product suggestions even some cases user may have already rated that item. For example the books having the different versions or editions may get recommended frequently. Pure content based filtering generally faces this problem [4].

2.4 Evaluating Recommender System

Measuring the performance of recommender system is an integral part of recommender system which helps in analysing and making changes to the recommender system as needed according to context of the problem. There are various metrics available to measure the effectiveness of any recommendation algorithm from different perspective for different purpose.

2.4.1 Accuracy Metrics and Error Metrics

These set of matrices measure how well the recommender system is performing by predicting the ratings for already rated item by user and comparing the predicted rating with the actual rating to calculate the error or accuracy between predicted and actual rating. Hence in this approach certain set of already rated items by any user are hidden from the recommender system and ratings are predicted for this hidden items which are then used to find the error or accuracy.

For example 10% of ratings of each user are randomly selected any are taken out while calculating the computation for recommendation and rating for that item is predicted by using system and then difference between each predicted and actual rating is calculated. Than usually the mean of error for every rating for particular user is calculated which is then finally utilized to find the mean of errors for all users.

There are various variants are used like Mean Absolute Error(MAE), Root Mean Square Error(RMSE), Mean Square Error(MSE) for calculating the error metrics.

2.4.2 Decision Support Metrics

This set of metrics measures how well the recommender system performs while helping the users make good decisions. Good decisions in the sense that selecting the good items and rejecting the bad items. Hence in this kind of metrics we can decide threshold where mostly the decision of most of the users change. For example on 5 star rating scale for books ratings having 0.5 denominations can have probably change of decision from 3.5 i.e. most users considers books having ratings above 3.5 as books as good and below scale indicates books which are not so good or bad.

One type of decision support metrics is precision and recall, where defines percentage of items selected that are relevant to the user’s interest while recall defines the number of relevant items returned out total number of items. Hence precision defines that only useful items are shown to the user while recall measures that useful item does not get missed.

Another popular approach for decision support metrics is receiver operating characteristics curve (ROC curve) where performance of recommender filter for different threshold values can be measured. ROC curve uses graph to plot the true positives against the false positives that gets filtered through the system at different threshold levels. In ROC area under the curve is mostly used as the measure for the performance of recommender system. In other words we can say that recommendation predictor acts as a filter which includes the item which have high prediction of rating and rejecting the item which have low prediction of ratings. In other words ROC graph can be said to the plot of sensitivity versus specificity where sensitivity means probability that good item is accepted by filter and specificity means bad item is rejected by filter [6].

2.4.3 Ranking Metrics.

Rank metrics measures how well the recommender performs while measuring relative preference for items by the user. There many popular methods used in this segment like spearman rank Correlation, Mean Reciprocal Rank, discounted cumulative gain, fraction of concordant pairs and many others. Delving into details of this metrics mean reciprocal rank finds the reciprocal rank by formula of 1/I where I is the rank of the first good item returned by the recommender system and mean reciprocal rank is the average of reciprocal ranks. Spearman Correlation coefficient measures the correlation between the actual rank of the product and the one generated by the recommender and punishes the recommender for each misplaced item. Discounted Cumulative gain measures the utility of item for every position in the list, in other sense it considers things at front more important, and finally normalize the results to obtain the normalized discounted gain for recommender system. Fraction of concordant pairs tests the pairwise accuracy by measuring what fraction of pairs of predicted and actual ratings by user are in the correct relative order.

3. Work done under this project

3.1 Intended algorithm use

We planned to use content boosted collaborative approach to implement during our project for which we chose book crossing dataset available under group lens project website [7]. The dataset contained 3 tables where one table contained ratings for the books corresponding to its ISBN by the user corresponding to USER ID, second table contained the side information of books where it contained the information columns for ISBN no, author, publisher and year of publication, and third table contained the information about users like USER ID, age and location.

We planned to implement 4 algorithms using 2 tables of the dataset that is user-book rating and book information data tables.

Intended 4 algorithm we planned to implement were,

1) Pure Collaborative Filtering Recommender
2) Pure content based Recommender
3) Naïve Hybrid Algorithm Recommender
4) Content Boosted Collaborative Recommender

Note: We included the details of this algorithm during our project presentation in the class, but due to the problems we faced as discussed in section 3.2 we partially implemented the algorithms given in section 3.1 and instead completely implemented other algorithm discussed in section 3.3

3.1.1 Dataset: The Book-Crossing

Dataset comprises 3 tables.

I. Table Name: BX-Users
Table Columns: ID, Location, Age

II. Table Name: BX-Books
Table Columns: ISBN, Book-Title, Book-Author, Year-Of-Publication, Publisher

III. Table Name: BX-Book-Ratings
Table Columns: ID, ISBN, Book-Rating

3.1.2: Pure Collaborative Filtering on Book Crossing Dataset

Tools Used: Apache Mahout, JAVA

Table Used: BX-Book-Ratings from Book Crossing Dataset

Step 1: We first calculated the user-user similarity between the users by using the Pearson coefficient

$$P_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r_a}) \times (r_{u,i} - \bar{r_u})}{\sqrt{\sum_{i=1}^{m}(r_{a,i} - \bar{r_a})^2 \times \sum_{i=1}^{m}(r_{u,i} - \bar{r_u})^2}}$$

Where,

$P_{a,u}$ is the similarity between user u and a.

$R_{a,i}$ is the rating given to item I by user u.

M is the total number of items

Step 2: Than we find the top k users by using k-nearest neighbour method available to implement K-nearest neighbour algorithm, we have taken k = 20 for our calculation

Step 3: Than we predicted the rating for the item using the weighted methods as explained through relations below

$$\mathbf{P_{a,i} = \bar{r_a} + \frac{\sum_{u=1}^{n}(r_{u,i} - \bar{r_u}) \times P_{a,u}}{\sum_{u=1}^{n} P_{a,u}}}$$

Where,

$P_{a,i}$ is the prediction for item I for user a.

$P_{a,u}$ is the similarity between user u and a.

- We got the MAE around 1.1 on average

3.1.3: Pure content based Recommender

Tools Used: Apache Mahout, JAVA

Table Used: BX-Book-Ratings and BX-Books from Book Crossing Dataset

We decided to use the side information about the books available in the BX-Books dataset to generate the content based recommendations for items. Then our plan was to measure the result of this algorithm and add the results produced to the original BX-Books rating matrix (originally it's sparse) to make it dense matrix which was planned to be uses for the calculation in the algorithm 4(Content Boosted Collaborative Filtering).

The content information available for the books are Name of Author, Name of publisher and year of publication.

But as the proper features were not formed by using this available information the results we find using the name of author and name of publishers as feature were very poor.

Limitations and Conclusion we obtained

We find that our approach of selecting features was very poor and our dataset didn't contained proper information about features which can be used for calculation in the content based recommendations like genre of the book.

Due to improper selection of dataset for the content based algorithm we were unable to complete the 3 and 4th algorithm included in the planed work as they are dependent on the results of the content based algorithm.

3.1.4: Naïve Hybrid Approach – (Future Work)

This algorithm was basically intended to calculate the average of output of pure content based and pure collaborative filtering algorithm.

3.1.5: Content based Collaborative Approach – (Future Work)

Tools to be used: Apache Mahout, JAVA

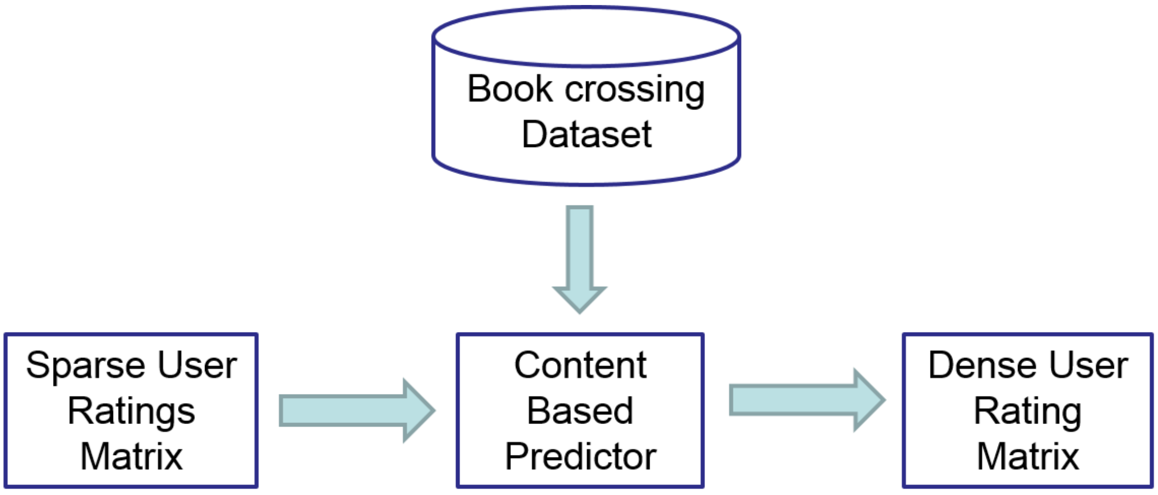
Table Used: BX-Book-Ratings and BX-Books from Book Crossing Dataset

We intended to make the sparse matrix of BX-Book-Ratings dense by using content based recommender before giving it as a input to the collaborative filter in order to get the benefits of both the content based and collaborative filtering while reducing inclusion of limitations. [6]

Step 1: To convert sparse U-I matrix into Dense pseudo U-I matrix by using content Based Recommender.

$V_{u,i}$: Dense U-I Matrix

$$v_{u,i} = \begin{cases} r_{u,i} : \text{if user u rated item i} \\ c_{u,i} : \text{predicted by content based system} \end{cases}$$



Step 2: Calculating Harmonic Mean Weighting

The output of pseudo user rating matrix depend upon the number of ratings the user has given. Hence the values of ratings added by content based recommender in U-I matrix are good if the user has given more number of ratings and the ratings added are of poor quality if the user has rated few items. Hence to prevent the misleading effect of this kind of ratings we can use the harmonic mean weighting [6].

$$hm_{i,j} = \frac{2m_i m_j}{m_i + m_j}$$

$$m_i = \begin{cases} \frac{n_i}{15} : if n_i < 15 \\ 1 : otherwise \end{cases}$$

Where, n_i is the total number of item rated by user i

Harmonic mean will bias the lower of two values m_i and m_j [6]

Calculating Hybrid Correlation Weight

$$hw_{a,u} = hm_{a,u} + sg_{a,u}$$

Significant weighting factor is added to the harmonic mean weighting to get the value of hybrid correlation weight.

Step 3: Calculating Self Weighting

$$sw_a = \begin{cases} \frac{n_a}{15} \times max : if n_a < 15 \\ max = 2 : otherwise \end{cases}$$

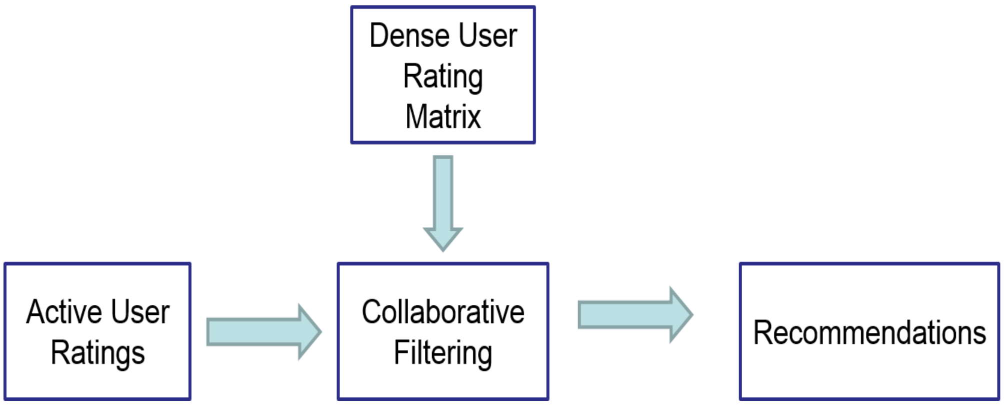
Where,

Na is the number of items rated by any active user.

Parameter max indicates the confidence placed in pure content based algorithm

Self-Weighting factor is used to increase the confidence placed in pure content predications for the active users. For accomplishing that the pseudo active user is considered to be of more importance than other neighbours.

Step 4: Producing Predictions



$$P_{a,i} = \bar{v}_a + \frac{sw_a(c_{a,i} - \bar{v}_a) + \sum_{u=1, u\neq a}^{n} hw_{a,u} P_{a,u} (v_{u,i} - \bar{v}_u)}{sw_a + \sum_{u=1, u\neq a}^{n} hw_{a,u} P_{a,u}}$$

Where,

Ca,i is pure content based prediction for the active user on item i.

Vu,i is the pseudo user rating for the user u for item i.

SWa is the self-weighting for user a

Hw,a is the hybrid correlation weight

N is the size of neighbourhood.

3.2 Transition from Intended to Final Algorithm

While working on the intended algorithms discussed in the section 3.1, we knew that our goal of using content based algorithm was to convert the sparse U-I matrix into dense pseudo matrix which was planned to be given as input to the collaborative filtering recommender.

Our failure to implement the content based recommender for the available dataset led us to ponder for other ways to make the sparse matrix dense in order to improve the pure collaborative filtering. While working we thought of trying to use predictions of item-item collaborative filtering to be included in the original U-I matrix to make it dense which than be given into User-User CF to generate better results compared to pure user-user CF. This journey led us to the work as discussed in section 3.3.

3.3 Final Algorithm

3.3.1 Dataset: Movie-lens

This data u.data file consists of

- 100,000 ratings (1-5) from 943 users on movies
- Each user has already rated at least 20 movies
- Dataset consists of columns for user id, item id, ratings

3.3.2 Pure collaborative filtering

This is the same algorithm as discussed in section 3.1.2 implemented on different dataset with same parameters used.

Step 1: We first calculated the user-user similarity by using Pearson coefficient having equation as follows,

$$P_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r_a}) \times (r_{u,i} - \bar{r_u})}{\sqrt{\sum_{i=1}^{m}(r_{a,i} - \bar{r_a})^2 \times \sum_{i=1}^{m}(r_{u,i} - \bar{r_u})^2}}$$

Where,
$P_{a,u}$ is the similarity between user u and a.
$R_{a,i}$ is the rating given to item I by user u.
M is the total number of items

Step 2: We calculated the n nearest neighbours by using k-nearest neighbours approach by using k value of 10

Step 3: We generate top 5 recommendations for each user U by using apache mahout.

Step 4: We measure MAE (Mean Absolute Error) value of this algorithm by taking 20% of ratings of each user as test ratings and 80% as base ratings.

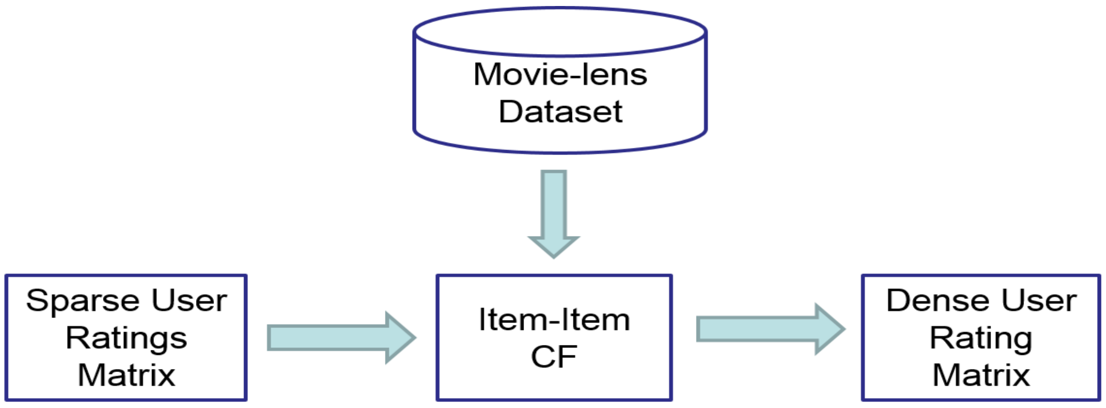
Note: Results and implementation steps of this algorithm are discussed in Section 4.1

3.3.3 User-User CF augmented by Output of Item-Item CF

Part 1: Goal: To use the predictions generated by item-item CF along with original sparse U-I matrix to make it more dense.

Steps followed with the particular parameter values we used is as per follows.

- We calculated the ratings for items that users have not rated by using loglikelihood item-item similarity collaborative filtering approach. We implemented this algorithm in apache mahout.
- We selected only 1 item (unrated item) for each item (item already rated) having the maximum similarity and generated the rating for the item based on the rating of the rated item and similarity value.
- We than **added this ratings produced from item-item CF having ratings value of 5 and 4 into original matrix**. In this way we make the original U-I matrix denser, which is used in the part 2 of this algorithm. We added approximately 55,000 ratings to original dataset.
- We have selected only items having rating value of 5 and 4 to be added into our pseudo matrix because in the end good items are mostly important for recommending items to the user.



Part 2: Goal: To use the dense user rating matrix to generate the predictions using User-User CF approach

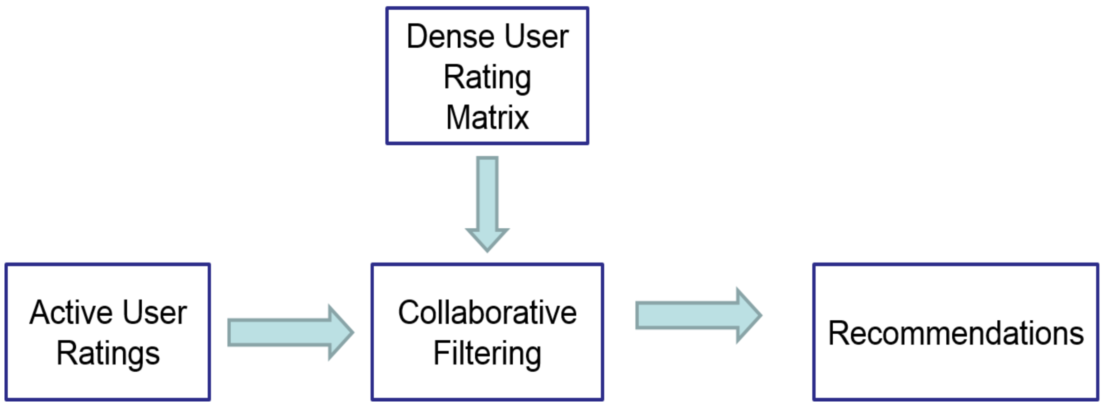
Algorithm Used: User-User Collaborative Filtering on pseudo U-I matrix

Steps followed for this algorithm are as follows,

- We applied the User-User collaborative filtering on the pseudo U-I matrix obtained from part 1 of this algorithm.
- For this algorithm we used the Pearson coefficient to find the user-user similarity. Equation used is same as discussed in first part of this algorithm.

- We calculated k similar users by using k nearest neighbour method where k=10
- We generated the top 5 recommendations for each user
- We calculated the value of MAE (Mean Absolute Error). We took 20% of ratings given by each user as test rating and 80% of ratings as base ratings.

In order to compare the results between algorithm 1 and 2 discussed in section 3.3.2 and 3.3.3 we used the same parameter and algorithm as used in algorithm 1 and second part of algorithm 2.(i.e. we used same parameters for calculating user-user CF in algorithm 1 and part 2 of algorithm 2).



Note: Results and implementation steps of this algorithm are discussed in Section 4.2

4. Implementation and Results Comparison

4.1 Implementation of Algorithm 1: Pure User-User CF (Algorithm discussed in section 3.3.2)

Algorithm Used: Pure Collaborative Filtering (Algorithm discussed in section 3.3.2)

Dataset Used: Movie lens (Discussed in section 3.3.1)

Tools Used: Java, Apache Mahout

Steps we have followed while implementing this algorithm are as follows

1) We converted the u.data file available from 'grouplens.org/datasets/movielens/' and converted it into movies.csv by trimming the extra space and adding commas.
   Program File Name: MovieDataConvert.Java
2) We sorted the movies.csv file by using USERID in order to get the ratings of any particular user together.
   Program File Name: Sorting.java
3) We implemented the Pure user-user Collaborative Filtering (as discussed in section 3.3.2)
   Program File Name: CF_Movies.java

4.2 Implementation of Algorithm 2: User-User CF augmented by Output of Item-Item CF (Algorithm discussed in section 3.3.3.)

Algorithm Used: Output of Item-Item CF + Original U-I as input to User-User CF (Algorithm discussed in section 3.3.2)

Dataset Used: Movie lens (Discussed in section 3.3.1)

Tools Used: Java, Apache Mahout

Steps we have followed while implementing this algorithm are as follows

1) We converted the u.data file available from 'grouplens.org/datasets/movielens/' and converted it into movies.csv by trimming the extra space and adding commas.
   Program File Name: MovieDataConvert.Java

4) We sorted the movies.csv file by using USERID in order to get the ratings of any particular user together.
   Program File Name: Sorting.java

2) We calculated item-item similarity by using Loglikelihood item-item similarity
   Program File Name: Item.java

3) Than we predicted the ratings for 1 best similar item for all already rated items.
   Program File Name: Predict.java

4) Than we added the prediction generated from step 4 if only rating value is 4 or 5 to the original sorted U-I matrix.

Program File Name: Merge.java

5) We sorted the final merged dense file
Program File Name: Sorting2.java

6) We computed the user-user CF on final sorted dense data file and generated top 5 recommendations for each user and calculated the MAE (Mean Absolute Error) for each item.
Program File Name: CF_Densed.java

4.3 Comparison of Results

1) We computed the output of both algorithms in single file in order to compare the results.
Program File Name: UserAccuracy.java

Values are the MAE values calculated for both Algorithm

Run Index	Test Users/Base Users (%)	Movies.csv (Pure User-User CF) Algorithm 1 (MAE Value)	Densed.csv (User-User CF augmented by Item-Item CF) Algorithm 2 (MAE Value)	Difference (Algorithm 1-algorithm 2)
Run 1	20/80	0.821016	0.789837	+0.031179
Run 2		0.811426	0.794007	+0.017419
Run 3		0.821884	0.798824	+0.023060
Run 4		0.796876	0.801848	-0.004972
Run 5		0.805718	0.784903	+0.020815
Run 1	10/90	0.800035	0.786234	+0.013801
Run 2		0.809574	0.798048	+0.011526
Run 3		0.791507	0.754425	+0.037082
Run 4		0.777731	0.779085	-0.001354
Run 5		0.817608	0.772358	+0.045250

Average Difference of MAE = 0.193806/10 = 0.0193806

5. Conclusion

5.1 Conclusion: Intended Algorithm

Intended algorithm discussed in section 3.1 included implementation of 4 algorithm namely pure collaborative, pure content based, hybrid naïve and content boosted collaborative filtering. In all 4 algorithms content boosted collaborative filtering (CBCF) was the main algorithm discussed in the paper [6] we tried to implement. The CBCF was designed to mingle the advantages of both CF based and content based approach and eliminate some of the disadvantages of both methods. CF based approach have advantages like domain independence, can work in domains where pure content information is not available and can generate serendipity in recommendations and disadvantage like Data sparsity problem and First rater problem. Content based approach have advantage like similar user independence, can handle first rater problem and is transparent and disadvantage like it requires content information and cannot generate serendipity in recommendations. CBCF can club the advantages of both hence it provides features like it can handle first rater problem, can handle first rater problem, can find better neighbours and better recommendations.

We first implemented the CF based approach and measured the performance by using MAE. We get the usual results. Than we tried to implement content based recommender by using the side information of books like author's name, publisher's name and year of publication. We tried to predict the recommendations using the formula's discussed in report, but as the side information of books didn't contained proper information which can be converted into feature vector our results were very poor than discussed in the paper[6]. Hence we concluded that to implement the content-based recommendations one requires proper data about the items which can be converted into proper feature vectors. While working on CBCF, based on intuition we decided to augment the original U-I matrix by using the recommendations generated from item-item recommender rather than content based recommender. Than this augmented pseudo U-I matrix is given to pure CF recommender. Conclusion of this second approach which we included in this project is discussed in next part.

5.2 Conclusion: Final Algorithm

We implemented two algorithms in second part of our project by using movie lens dataset. Two algorithm implemented are Pure collaborative Filtering based algorithm and second algorithm User-User CF augmented by Item-Item CF. On looking at the comparison of results we discussed in section **4.3**, we can conclude that by making the U-I matrix denser by using recommendations generated from item-item CF and using this pseudo dense U-I matrix to generate user-user CF based recommendations we can improve the results of recommender. As seen from the comparison we can conclude that on using pseudo dense U-I matrix populated by item-item based ratings we can increase the accuracy of prediction by around 2.2%. We have added 55373 pseudo ratings generated by Item-Item CF to the original U-I matrix containing 100,000 ratings. Our results generated from the second approach is better than pure CF based approach.

6. Future Work

6.1 Intended Algorithm

As due to improper selection of dataset we were able to complete only partial algorithms from 4 intended algorithm implementation. We look forward to crawl the dataset to get proper item features which can be handy while implementing content based recommender and implementing content boosted CF. We also look forward to explore other ways of including side information of users and items in different algorithm with different parameters to improve the recommender.

6.2 Final Algorithm

We implemented pure CF and user-user CF augmented by item-item CF outputs and got positive results of around 2.2% on MAE. We look forward to examine difference in performance of this two approaches by using other metrics like decision support and rank metrics. We also look forward to run this approach on different dataset to get onto some firm conclusion of benefits of using second approach.

7. Project Contribution

1) Vaibhav Patel: Algorithm Research, Implementation of Pure user-user CF and Evaluation, Report Preparation

Contribution Percentage: 34%

2) Rutvik Patel: Algorithm Research, Implementation of User-User CF augmented by Item-Item CF and Evaluation

Contribution Percentage: 33%

3) Aakanxu Shah: Algorithm Research, Implementation of User-User CF augmented by Item-Item CF and Evaluation

Contribution Percentage: 33%

8. References

[1] http://www.internetlivestats.com/internet-users/

[2] Yue shi, Martha Larson, Alan Hanjalic, Delft University of technology “Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges” ACM Computing Surveys Volume 47 Issue 1, July 2014 Article No. 3

[3] McNee, S. M., Riedl, J., and Konstan, J. A. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In CHI

'06 Extended Abstracts on Copyright is held by the author/owner(s).

Human Factors in Computing Systems. 2006

[4] Zeinab. A., Sihem. A.Y., Lakshmanan. V.S., Sergei. V., Cong. Y., “Getting recommender systems to think outside the box.” 285-288 RecSys, 2009.

[5] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating Collaborative Filtering Recommender Systems. ACM TOIS, 22(1):5–53, 2004.

[6] P. Melville, R. Mooney, R. Nagarajan, Content-boosted collaborative filtering, in: ACM SIGIR 2001 Workshop on Recommender Systems, 2001.

[7] Boddu Raja Sarath Kumar , Maddali Surendra Prasad Babu .An Implementation of Content Boosted Collaborative Filtering Algorithm. International Journal of Engineering Science and Technology (IJEST), April 2011

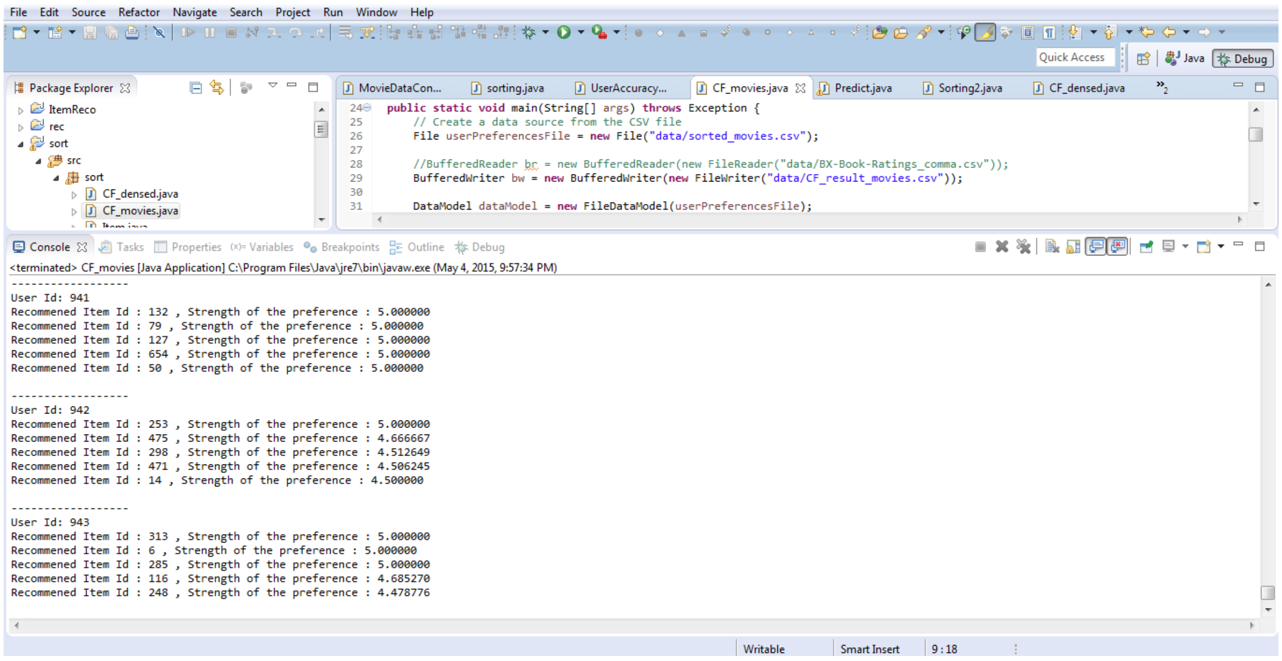
[8] Apache Mahout: https://mahout.apache.org/

[9] Database Link for intended algorithm: http://www2.informatik.uni-freiburg.de/~cziegler/BX/

[10] Database link for Final Algorithm implemented: http://grouplens.org/datasets/movielens

9. Some Screenshots of Results

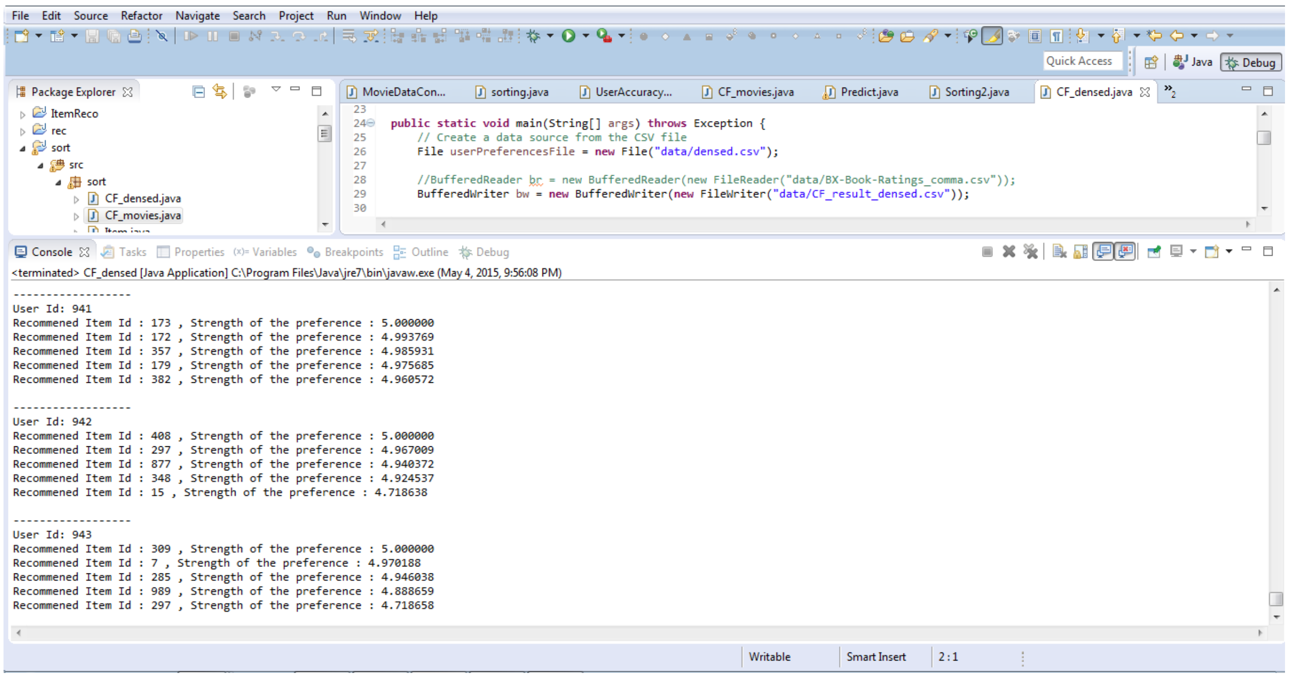
1) Screenshot Recommendations generated by Pure user based CF



```java
public static void main(String[] args) throws Exception {
    // Create a data source from the CSV file
    File userPreferencesFile = new File("data/sorted_movies.csv");

    //BufferedReader br = new BufferedReader(new FileReader("data/BX-Book-Ratings_comma.csv"));
    BufferedWriter bw = new BufferedWriter(new FileWriter("data/CF_result_movies.csv"));

    DataModel dataModel = new FileDataModel(userPreferencesFile);
```

```
<terminated> CF_movies [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (May 4, 2015, 9:57:34 PM)
------------------
User Id: 941
Recommened Item Id : 132 , Strength of the preference : 5.000000
Recommened Item Id : 79 , Strength of the preference : 5.000000
Recommened Item Id : 127 , Strength of the preference : 5.000000
Recommened Item Id : 654 , Strength of the preference : 5.000000
Recommened Item Id : 50 , Strength of the preference : 5.000000

------------------
User Id: 942
Recommened Item Id : 253 , Strength of the preference : 5.000000
Recommened Item Id : 475 , Strength of the preference : 4.666667
Recommened Item Id : 298 , Strength of the preference : 4.512649
Recommened Item Id : 471 , Strength of the preference : 4.506245
Recommened Item Id : 14 , Strength of the preference : 4.500000

------------------
User Id: 943
Recommened Item Id : 313 , Strength of the preference : 5.000000
Recommened Item Id : 6 , Strength of the preference : 5.000000
Recommened Item Id : 285 , Strength of the preference : 5.000000
Recommened Item Id : 116 , Strength of the preference : 4.685270
Recommened Item Id : 248 , Strength of the preference : 4.478776
```

2) Screenshot Recommendations generated by User-User CF augmented by Output of Item-Item CF



```java

public static void main(String[] args) throws Exception {
    // Create a data source from the CSV file
    File userPreferencesFile = new File("data/densed.csv");

    //BufferedReader br = new BufferedReader(new FileReader("data/BX-Book-Ratings_comma.csv"));
    BufferedWriter bw = new BufferedWriter(new FileWriter("data/CF_result_densed.csv"));

```

```
<terminated> CF_densed [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (May 4, 2015, 9:56:08 PM)
------------------
User Id: 941
Recommened Item Id : 173 , Strength of the preference : 5.000000
Recommened Item Id : 172 , Strength of the preference : 4.993769
Recommened Item Id : 357 , Strength of the preference : 4.985931
Recommened Item Id : 179 , Strength of the preference : 4.975685
Recommened Item Id : 382 , Strength of the preference : 4.960572

------------------
User Id: 942
Recommened Item Id : 408 , Strength of the preference : 5.000000
Recommened Item Id : 297 , Strength of the preference : 4.967009
Recommened Item Id : 877 , Strength of the preference : 4.940372
Recommened Item Id : 348 , Strength of the preference : 4.924537
Recommened Item Id : 15 , Strength of the preference : 4.718638

------------------
User Id: 943
Recommened Item Id : 309 , Strength of the preference : 5.000000
Recommened Item Id : 7 , Strength of the preference : 4.970188
Recommened Item Id : 285 , Strength of the preference : 4.946038
Recommened Item Id : 989 , Strength of the preference : 4.888659
Recommened Item Id : 297 , Strength of the preference : 4.718658
```