ISSN:0975-9646

An Implementation of the User-based Collaborative Filtering Algorithm

Maddali Surendra Prasad Babu
Professor, Dept. of CS&SE,
AUCE, Andhra University
Andhra Pradesh, Visakhapatnam

Boddu Raja Sarath Kumar
Professor, Dept. of CSE,
Lenora College of Engineering
Rampachodavaram-533288,

Abstract -Collaborative filtering algorithms (CFAs) are most popular recommender systems for collaborating one another to filter the documents they read from the last decade. CFAs have several features that make them different from other algorithms. The Classification accuracy is one among them. A user-based collaborative filtering algorithm is one of the filtering algorithms, known for their simplicity and efficiency. In the present paper a steady is conducted for its implementation and its efficiency in terms of prediction complexity

Key words – Collaborative Filtering Algorithm, Mean Absolute Error, Prediction Complexity

1. INTRODUCTION

The growth and availability of data on the internet has caused information overload and hence searching for a query is not an easy task for an individual in the sources of information available. The amount of information found in the Internet, is growing day by day. Personalized retrieval systems are becoming more interesting, especially when not limited to just searching for information but that also to recommend the items that would be more appropriate for the user's needs or preferences There are mainly two types of recommender systems, as a function of the algorithm used: Content-Based Filtering (CBF) and Collaborative Filtering (CF). CF is one of the most commonly used methods in personalized recommendation systems. Collaborative filtering algorithm recommend items based upon opinions of people with similar tastes. Collaborative filtering can also recommend items that are not similar and like-minded users have rated the items. Collaborative filtering faced some problems by traditional information filtering duly eliminating the need for computers to understand the content of the items. Recommender systems need to store certain information about the user preferences, known as the user profile to achieve this personalization. CF is also further classified as three main filtering techniques [1]: They are (i) Memory-based Collaborative Filtering (ii) Model-based Collaborative Filtering and (iii) Hybrid Collaborative Filtering. In recent years research in recommender system was carried out by R. Bell [2], Manos Papagelis [3], R. Salakhutdinov [4][6], Bhaskar Mehta[5][7], J. Sandvig[8].

The system will inform the user of what items are well recommended by other users with similar likes or interests. An analysis of the content by the system is not necessary and the quality or subjective evaluation of the items will be considered. However, these algorithms present problems in their computational performance and efficiency. These systems have different techniques, though some of them may be general enough to be applied to various techniques. *CF* techniques use a database of preferences for items by users to predict additional topics. Other technologies have also been applied to recommendation systems including Bayesian networks, Pearson Correlation, Cosine Correlation, Clustering and Horting.

2. SOME IMPORTANT CFAs WITH PREDICTION COMPLEXITY

2.1. Memory based CFA

User rating data is used to compute similarity between users or items. This is used for making recommendations [1] and is used in many commercial systems. It is easy to implement and is effective. Some important algorithms are: User-based collaborative filtering, Item-based collaborative filtering and Similarity fusion collaborative filtering. Memory-based *CFAs* use the entire or a sample of the user-item database to generate a prediction. By identifying the neighbors of a new user, a prediction of preferences on new items for him or her can be produced. When the task is to generate a top-*N* recommendation, we need to find the most similar users or items after computing the similarities, and then aggregate the neighbors to get the top-*N* most frequent items as the recommendation.

2.2. Model based CFA

These are used to make predictions for real data. To find patterns based on training data models are developed using data mining and machine learning algorithms [1]. Some important algorithms are: Bayesian belief nets collaborative filtering, Regression based collaborative filtering, Slope one collaborative filtering, Latent Semantic Indexing collaborative filtering and Cluster based smoothing collaborative filtering

The design and development of models can allow the system to learn to recognize complex patterns. Then make intelligent predictions for the collaborative filtering tasks for real-world data. The above mentioned, Model-based *CF* algorithms have been investigated to solve the drawbacks of memory-based *CF* algorithms.

2.3. Hybrid CFA

A number of applications combine the memory-based and the model-based CF algorithms. These overcome the limitations of native CF approaches [13]. It improves the prediction performance and overcomes the CF problems such as sparsity and loss of information. A hybrid CF also known as Content Boosted Collaborative Filtering, approach was proposed to exploit bulk information

designed for exact product classification to address the data sparsity problem of *CF* recommendations. Some important algorithm is: Personality diagnosis collaborative filtering
Hybrid CFAs combine *CF* with other recommendation techniques to make predictions. Content-based CF makes recommendations by analyzing the content of textual information. Many elements contribute to the importance of the textual content. A content-based recommender then uses classification algorithms to make recommendations. Content-based techniques have the cold-start problem, in which they must have enough information to build a reliable classifier. They are limited by the features explicitly associated with the objects they recommend, while collaborative filtering can make recommendations without any descriptive data. Content-based techniques only recommend items that score highly against a user's profile. Other recommender systems includes demographic information i.e. gender, postcode, occupation etc,. Utility-based recommender systems and knowledge-based recommender systems are the systems which require knowledge about how a particular object satisfies the user needs.

3. USER BASED COLLABORATIVE FILTERING ALGORITHM

User-based CF algorithm produces recommendation list for object user according to the view of other users. The assumptions are if the ratings of some items rated by some users are similar, the rating of other items rated by these users will also be similar [3]. CF recommendation system uses statistical techniques to search the nearest neighbors of the object user and then basing on the item rating rated by the nearest neighbors to predict the item rating rated by the object user, and then produce corresponding recommendation list. Collaborative Filtering component that uses a neighborhood-based algorithm is as follows. In neighborhood based algorithms, a subset of users are chosen based on their similarity to the active user, and a weighted combination of their ratings is used to produce predictions for the active user.
The algorithm can be summarized in the following steps:
Step: 1. all users are weighted with respect to similarity with the active user.
Similarity between users is measured as the Pearson correlation between their ratings vectors.
Step: 2. Select n active users that have the highest similarity.
Step: 3. Compute a prediction, $P_{a,u}$ from a weighted combination. Similarity between two users is computed using the Pearson correlation coefficient

$$P_{a,u} = \frac{\sum_{i=1}^{rn} (r_{a,i} - \bar{r}_a) \times (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{rn} (r_{a,i} - \bar{r}_a)^3 \times \sum_{i=1}^{rn} (r_{u,i} - \bar{r}_u)^3}} \qquad (1)$$

Where **r** $_{a,i}$ is the rating given to item i by user a;
and r_a is the mean rating given by user a.
In step 3, predictions are computed as the weighted average of deviations from the neighbor's mean:

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n} (r_{u,i} - \bar{r}_u) \times P_{a,u}}{\sum_{u=1}^{n} P_{a,u}} \qquad (2)$$

Where $P_{a,i}$ is the prediction for the active user a for item i. $P_{a,u}$ is the similarity between users a and u. n is the number of users in the neighborhood.

4. DATASET

Jester is a web based online joke recommendation system [12], which has been developing at University of California, Berkeley. This data has 73,421 users collected with a rating from -10 to +10. In the dataset missed ratings are represent as 99. We assumed that the rating value to its round value. In which we selected 500 users with complete rating pragmatically and generated results using JFREE Charts.

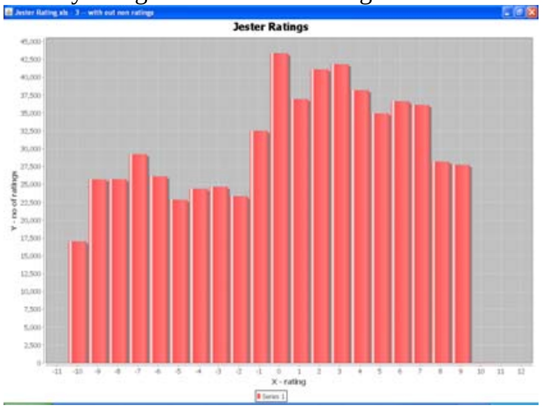

Figure.1 Jester dataset with its rating

5. METHODOLOGY

User-based collaborative filtering algorithm is evaluated and measured prediction times, as well as the quality of their predictions. All the experiments were performed on an Intel Pentium-IV Processor, 2 GB RAM. The algorithm has been implemented in Java and executed the results . The dataset is stored in database MySQL.

6. IMPLEMENTATION

The following java programs *CFA3.java, NBSSimblanceRow.java and XYSplineRendererDemoTest.java* used in achieving this task.
NBSSimblanceRow.java is the class responsible for holding the row number and its semblance with current row. i.e, if 30 users are taken, then semblance list for the 1st row (user) will contain 29 *NBSSimblanceRow* objects each one containing its row number and its semblance with the 1st row. Once this list is sorted based on the semblance, then best n number of semblance rows will be generated
CFA3.java is responsible for generating the MAE values for User-Based Collaborative Filtering Algorithm. It achieved through the steps Step 1, Step2, Step3 explained earlier in this document

```
CFA3 we = new CFA3();
List original = new ArrayList();
```

```
String fileName2 = "D:\\Excelwork\\jester-data-x.xls";
int requiredSize = 30;
we.populateExcelToList(original, fileName2, requiredSize);
```

In the above five lines, we populate the excel sheet data to an array list *'original'*. The implementation of above functionality (population) can be done in *populateExcelToList* method. Here '*fileName2'* is the path to the excel sheet we are testing. '*requiredSize'* is the no of users we are testing. We assumed this as 30.

```
List sheetData = new ArrayList();
```

'sheetData' is an array list which will hold the randomly picked values from the 'original' data.

```
we.initialize(sheetData, requiredSize);
```

In the above line we are initializing 'sheetData' with all zeros

```
 int[] randoms = we.getRandomUsers(original);
int[] lines = new int[randoms.length];
 int[] columns = new int[randoms.length];
Arrays.sort(randoms);
int counter = 0;
for(int d : randoms){
lines[counter] = d/100;
columns[counter] = d%100;
counter++;
}
 we.getRandomList(lines, columns, original, sheetData);
```

the above *getRandomList will* generate 500 random numbers from that random numbers which rating is to be picked up by using the division with 30 (number of users). For example, if one of the random numbers is 307, the user will be 10th user (307 / 30 = 10) where 10 is dividend and item will be 7 (307 % 30 = 7) where 7 will be remainder. Here 'lines' represents the users and 'columns' represents the items. Lines[i] and columns[k] together will represents ith user's rating on k item. The last line (we.getRandomList(lines, columns, original, sheetData);) is responsible for populating randomly picked 500 ratings from 'original' list to 'sheetData'

```
NBSSimblanceRow[] nbsSimilarRows = null;
NBSSimblanceRow oNBSSimblanceRow = null;
ArrayList<NBSSimblanceRow[]> listSimblances = new
ArrayList<NBSSimblanceRow[]>();

for(int i=0; i<30; i++){
 nbsSimilarRows = new NBSSimblanceRow[30];

for(int j=0; j<30; j++){
 oNBSSimblanceRow = new NBSSimblanceRow();

nbsSimilarRows[j] = oNBSSimblanceRow;
}
listSimblances.add(nbsSimilarRows);

}
```

In the above line of code, initialize arraylist which holds the semblances is performed. then 'listSimblances' will contain 30 entries in which each entry will contain 30 NBSSimblanceRow objects which are responsible for holding the row number and its semblance with current row. i.e, if 30 users are taken, then semblance list for the 1st row (user) will contain 30 NBSSimblanceRow objects (including the self) each one containing its row number and its semblance with the 1st row.

```
we.populateRows(listSimblances,sheetData);
```

In the above line of code, the semblances values are calculated and then populate in 'listSimblances'

```
for(int i=0; i<30; i++){
  nbsSimilarRows = listSimblances.get(i);
 Arrays.sort(nbsSimilarRows, we.new MyComparator());
}
```

In the above lines, we are sorting the obtained semblances.

```
List sheetData1 = ((List) ((ArrayList) sheetData).clone());
Hashtable table = new Hashtable();
for(int i=4; i<30; i=i+4){
List  s3 =  we.populatePredictUJ(sheetData1, listSimblances,  i);
 List  s4 =  new ArrayList();
 double mae = we.getMAE(sheetData, original, s3);
BigDecimal            z1            =            new
BigDecimal(mae).setScale(2,BigDecimal.ROUND_HALF_UP);
   mae = z1.doubleValue();
 System.out.println(" For neighbourset size -- " + i +" MAE is " +
mae);
table.put(new Double(i), mae);
  }
```

sheetData is cloned to sheetData1 which is used to obtain s3 which contains the predicted values for non rated ratings. The method **populatePredictUJ** is used for implementation for getting the predicted values. The parameters sent to this method are

sheetData1 -randomly picked ratings arraylist,
listSimblances - its calculated and sorted semblances list
i – size of neighbour set.

Once predicted values set is obtained, MAE values are calculated using the *getMAE* method for which the parameters are

original – actual data from excel sheet..
sheetData – ramdomly picked 500 ratings.
S3 – predicted ratings.

MAE is a commonly used recommendation quality measurement method. .The implementation of *getMAE* method is used to generate MAE values and inserted into a hash table with neighbour set size as key and its corresponding MAE value as the value. MAE to measure the recommendation quality and calculates the irrelevance between the recommendation value predicted by the system and the actual evaluation value rated by the user.

In this dataset with jester joke data set [12] with 73421 users but limited it to 500 users programmatically. The generated 500 random numbers through programmatically decide which rating is to be picked up by using the division with 30 (number of users). For example, if one of the random numbers is 307, the user will be 10th user (307 / 30 = 10) where 10 is dividend and item will be 7 (307 % 30 = 7) where 7 will be remainder. Here '*lines*' represents the users and '*columns*' represents the items. *Lines[i]* and *columns[k]* together will represents ith user's rating on kth item. The predicted values for non rated ratings will be generated from *sheetData* method which is further cloned into *sheetData1*. The method *populatePredictUJ* is used for implementation for getting the predicted values. Once predicted values set is obtained, MAE values are calculated using the *getMAE* method. The values are inserted into a hash table with neighbour set size as key and its corresponding MAE value as the value. The results obtained are tabulated in table-II.

XYSplineRendererDemoTest is responsible for generating the graph (using JFREE chart) for exposed neighbourset size on X axis and its corresponding MAE values on Y axis.

XYSplineRendererDemoTest xysplinerendererdemo1 = new XYSplineRendererDemoTest("JFreeChart: XYSplineRendererDemo1.java", table);
xysplinerendererdemo1.pack();
RefineryUtilities.centerFrameOnScreen(xysplinerendererdemo1);
xysplinerendererdemo1.setVisible(true);

7. EXPERIMENT RESULTS AND EVALUATION

The MAE values are calculated using the method **populatePredictUJ** and tabulated in table 1. The influence of various nearest neighbors set on predictive validity is tested by gradually increasing the number of neighbors. User based dataset predicts item rating of the users are evaluated as per the opinions of the users chosen ratings. The results are shown in the JFREE Chart representing MAE values and respective their neighbor set sizes. Here it is observed that when Nearest Neighbor Set value increases the corresponding MEA decreased.

TABLE 1

Neighbour Set Size	4	8	12	16	20	24
MA E	4.80	4.48	4.37	4.31	4.29	4.14

Nearest neighbor set and MEA on predictive validity

The Graphical representation is as follows:

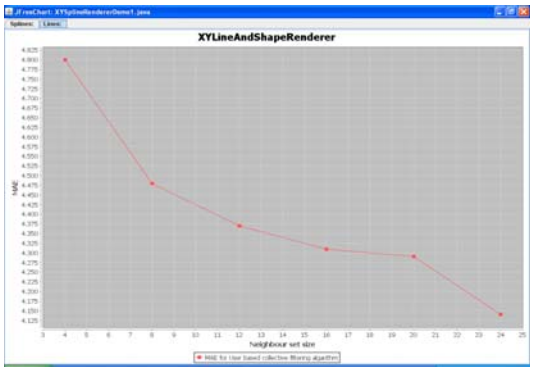

Figure 2 shows NNS vs MEA

8. CONCLUSIONS

In this paper, user based collaborative filtering algorithm has been implemented and evaluated. The quality of the predictions is evaluated with similar algorithms. It shows that it exhibits a behavior that is equivalent to that of the best algorithms. The main aim of this paper is to improve the quality of the results and make it easier for the user to find relevant information from the internet.

9. REFERENCES

[1] Xiaoyuan Su and Taghi M. Khoshgoftaar, "Review Article a Survey of Collaborative Filtering Techniques" published in Hindawi Publishing Corporation, Advances in Artificial Intelligence Volume 2009, Article ID 421425, 19 pages, doi:10.1155/2009/421425

[2] R. Bell and Y. Koren, "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights", IEEE International Conference on Data Mining (ICDM'07), pp. 43–52, 2007.

[3] Manos Papagelis and Dimitris Plexousakis "Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents" Engineering Applications of Artificial Intelligence 18 (2005) 781–789 www.elsevier.com/locate/engappai

[4] R. Salakhutdinov, A. Mnih and G. Hinton, "Restricted Boltzmann Machines for Collaborative Filtering", Proc. 24th Annual International Conference on Machine Learning, pp. 791–798, 2007.

[5] Bhaskar Mehta, Thomas Hofmann, and Peter Fankhauser, Lies and propaganda: detecting spam users in collaborative filtering, IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces (New York, NY, USA), ACM Press, 2007, pp. 14–21.

[6] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization", Advances in Neural Information Pro-cessing Systems 20 (NIPS'07), pp. 1257–1264, 2008.

[7] Bhaskar Mehta and Wolfgang Nejdl, Attack-resistant Collaborative Filtering, To appear: In Proceedings of the 31st ACM SIGIR Conference, ACM Press, 2008.

[8] J. Sandvig, Bamshad Mobasher, and Robin Burke, Robustness of collaborative recommendation basedon association rule mining, RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems (New York, NY, USA), ACM, 2007, pp. 105–112.

[10]. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[12]. Jester data, http://www.grouplens.org/.

[13]. X. Su, T. M. Khoshgoftaar, and R. Greiner, "A mixtureImputation-boosted collaborative filter," in Proceedings ofthe 21th International Florida Artificial Intelligence Research Society Conference (FLAIRS '08), pp. 312–317, Coconut Grove, Fla, USA, May 2008.