International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 6 (2015) pp. 14797-14812

http://www.ripublication.com

An Analysis on The Performance Evaluation of Collaborative Filtering Algorithms Using Apache Mahout

Lakshmi Devi K[1], Amrita R[2], P Subathra[3], PN Kumar[4] (Corresponding author)
[1,2,3,4]Dept. of CSE , Amrita Vishwa Vidyapeetham, Ettimadai, Coimbatore, Tamil Nadu, 641 112, India
laksdevi115@gmail.com[1],anku329@yahoo.co.in[2],p_subathra@cb.amrita.edu[3] , pn_kumar@cb.amrita.edu[4]

Abstract

Recommendation systems are being widely adopted in many areas which include social networking, e-commerce etc. Long years of research have led to the proposal of many algorithms in order to effectively capture the real tastes of users and deliver the recommendations accurately. Collaborative filtering is considered to be one of the popular and successful approaches to provide recommendations. In this paper, we conduct a performance evaluation of three popular collaborative filtering algorithms viz. User based, Item based and Slope-one recommender. We illustrate a brief overview on the different approaches of collaborative filtering, their method of working, advantages and limitations. We demonstrate the results based on the evaluation metrics precision, recall, f-measure, fallout and reach. Our experiments revealed that the Slope-one approach outperformed the other two approaches based on the evaluation metrics. We also explored different kinds of similarity metrics and highlighted the effect of size of the neighbourhood on the evaluation metrics.

Keywords: Collaborative Filtering (CF), Recommendation systems, Apache Mahout, User based CF, Item based CF, Slope one.

Introduction

The world is overwhelmed with information beyond the ability to process them completely, due to the rapid growth of internet in the recent years. The information explosion has drastically reduced the use ratio of the information which resulted in what is known as information overload. Spurred by the growth, the internet users find it difficult to choose what they really want. Personalized recommendation is one of the most popular methodologies to resolve the problem of so-called information overload [1].Recommendation systems come into aid at this situation to help users to differentiate between relevant and irrelevant information. Recommender systems

recommend to users the relevant items which are unknown and are of interest to them, through which they can get rid of the overwhelmed information.

Recommendation systems are widely adopted by e-commerce websites like Amazon, Flipkart etc, social networking websites like Facebook, Twitter etc. The quality of the recommendations produced by the recommender system depends mainly on how it captures the real interests of a particular user and recommends items which shares the same characteristics of his/her preferred items. When a user views or buys an item, he/she is provided with a list of similar products using recommendation systems and thus the users get satisfied by obtaining his/her preferred items. An award of 1million US dollars was given by NetFlix for an improvement of ten percent accuracy of their recommendation system which clearly signifies the importance of user satisfaction provided by the recommender systems.

Recommendation systems mainly fall into two categories: Collaborative Filtering (CF) and Content based recommendation.

- Collaborative Filtering (CF): CF algorithms are based on a particular user's relationship to items. These algorithms do not consider the characteristics or features of the items and hence the CF algorithms work independent of domains i.e. the recommendation framework developed for one domain can be applied any other domain. Moreover, this approach can work well in cross-genre recommendations since domain knowledge is not required in this approach [2]. But there are some issues in this approach. The cold-start problem occurs when a new user/ new item is added and the user has not rated any items yet or the item has not been rated by any user [3]. Another issue associated with this approach is the data sparsity which occurs when there is insufficient data i.e. the user*item matrix contains less rating entries in order to make good recommendations [4]. Scalability is another issue in CF methods where it is difficult to scale up the similarity computation with the growth of both the number of users and items in the database [5].
- Content Based Recommendation: Content based recommendation works with respect to a domain i.e this recommendation framework takes into consideration the properties and characteristics of the items and hence a recommendation framework developed for a particular domain cannot be applied to any other [6]. This is a major disadvantage of this approach. For eg, consider a recommendation system for movie domain. This approach works on the basis of the properties of movie domain like user demographics, genre, actors etc which is not applicable to a clothing domain. Content based recommendation systems rely on the content data and in order to extract the content data, techniques like information indexing and feature extraction are employed. The main advantage of using content based recommendation is that this approach does not have the limitations that tend to occur in CF methods like cold start problem and sparsity issue since this approach does not require other users to produce recommendations [2]. The major disadvantage with this approach is that it is difficult to define features for certain domains which are not automatically machine parsable like movie domain [2].

In this work, we mainly focus on the performance evaluation of collaborative filtering algorithms viz. user based CF, Item based CF and slope one, which is a slight variation of Item based CF. The rest of the paper is organised as follows: Section 2 discusses the related work. Section 3 describes the proposed system, recommender algorithms and their advantages and limitations. Section 4 presents a brief discussion about Apache mahout. Section 5 discusses the various similarity metrics used in recommender algorithms. Section 6 provides the implementation details. Section 7 demonstrates the evaluation metrics. The results and analysis is given in section 8. The conclusion of the paper is given in section 9.

Related Work

Recommender systems have their applications in variety of fields like information retrieval, cognitive science, approximation theory, forecasting theory etc. Due to the abundance of these applications, recommender systems have become a popular and imminent research problem to tackle. Research in this area started in the mid-1990s with major focus on personalised recommendations and consumer choice modelling [7]. In order to help users to access to the relevant information from a pool of “information overload", personalized recommendation systems are essential which provide personalized service of recommending items to users based on their preferences or interests[8].

As stated earlier, personalised recommendation systems can be broadly classified in to two: Collaborative Filtering (CF) and Content based recommendation systems [9]. Collaborative filtering techniques depend on the similarity between users (User based or Neighbourhood based recommendation) or similarity between items (Item based recommendation) to produce personalised recommendations [9][10]. User based recommendation defines a set of neighbours by computing the similarity between the users and this method is simple since only defining the number of neighbours needs tuning[11]. Defining the neighbourhood is done in either of the two ways: Fixed size neighbourhood and threshold based neighbourhood. The parameter k is set which determines number of neighbours in fixed size neighbourhood and a threshold can be kept in the similarity value (between -1 and +1) which cuts off all the users having similarity value less than the threshold in the case of threshold based neighbourhood. Item based recommenders work by finding out items similar to the items the users prefers [12].

The ratings provided by the users are stored in a *user *item* matrix where the rows represent users, columns represent items and the entries represent ratings. The collaborative filtering is classified into two approaches based on how the preference matrix is processed: model-based method and memory based method [7],[8].The memory based methods are simple and generally produce accurate recommendations. This method uses the whole matrix to provide recommendations and also use various similarity metrics. But these algorithms suffer from issues like sparsity, scalability, cold start and privacy [1][13][14]. The sparsity problem occurs when the users have rated only less number of items or many users have not rated items at all which results in difficulty in finding neighbourhood and thus affecting the quality of the

recommendations produced. The model-based CF methods work by constructing a model from the preference matrix and predicting the ratings of new items using this model[15]. Model based CF is considered to be faster in prediction than memory based CF. However, it has certain limitations. The model construction and updation are time consuming and some models will have numerous parameters for estimation and show high sensitivity to data changes. Wrong recommendations can result when the assumptions made while model construction do not really fit the data [15].

There are many similarity metrics used in Neighbour-based collaborative filtering for providing accurate recommendations. Alejandro Bellogín and Arjen P. de Vries [16] have done performance analysis of twelve features and demonstrated that some of the features were able to correlate with the performance up to 90 percent. Another variation of item based recommendation is termed as slope one recommendation which depends on the linear relationship between the ratings given by the user to produce recommendations [17]. Guy Shani and Asela Gunawardana [18] discuss three different kinds of experiments to evaluate a recommender system : online, offline and user studies.

Proposed System

In this paper, we have implemented three widely adopted recommender algorithms in Apache Mahout and evaluated their performance using the evaluation metrics - precision, recall, f-measure, fallout and reach. The recommender algorithms implemented are :

- User based CF
- Item based CF
- Slope-one

We have also demonstrated the effect of neighbourhood size (used in User based CF) on the evaluation metrics used. We also evaluated the Absolute Difference Error and Root Mean square Error and depicted their sensitivity on the neighbourhood size. The datasets used are MovieLens datasets of two different sizes: 100K and 1M. The details of the datasets used are furnished at Section 6.1.

User Based CF

CF algorithms are classified under two categories: User based and Item based recommendation. User based recommendation is based on the similarity between users. This approach finds users who are similar to a given user based on the number of items they like in common and recommend to the user the top N items which are unknown to the user and similar users like. The preferences of the people are dynamic in nature since user preferences may change over time and hence this approach is difficult to scale[11].

The Classic User Based Algorithm

*for each item **x** that user **m** has not rated yet*
*for every other user **n** who has rated item **x***
*Compute similarity **s** between users **m** and **n** using any of the similarity metrics*

*add user **n**'s preference for item **x**, weighted by similarity **s**, into a running average*
*return the top **N** recommended items, ranked according to the decreasing order of weighted average.*

Since this approach looks for every item, this approach is slow. So in order to alleviate this problem, a notion of neighbourhood has been introduced where by a neighbourhood of similar users is retained and only the items known to the neighbours are taken into account for recommendation [19].

The Neighbourhood Algorithm

The classic algorithm changes to :

*for each item **x** that user **m** has not rated yet*
*for every other user **n** who has rated item **x***
*Compute similarity **s** between users **m** and **n** using any of the similarity metrics*
*Rank and retain the top **K** similar users according to the decreasing order of similarity with the user **m** either by fixed size neighbourhood or threshold neighbourhood.*
*for each item **x** that some user in **K** has given ratings for and unknown to user **m***
*for every other user **n** in **K** who has rated item **x***
*Similarity is computed between users **m** and **n** using any of the similarity metrics*
*Weight each preference value in average according to how close that user is to the active user **m***
*return the top **N** recommended items, ranked according to the decreasing order of weighted average.*

Item Based CF

Item based recommendation is based on similarity between items. It attempts to find the top N items which are similar to a particular user's preferred items. Even though they both work under the notion of similarity, they are notably different approaches. The execution time of user based algorithms scales up as the number of users increase whereas for item based approach, the execution time increases as the number of items increase[19]. The Item based approach is considered to have less running time since the number of items will be less than the number of users in most cases. The relationship between items will be mostly static, so the computation can be done offline and is relatively easy to scale. Another benefit of this static relationship is that the similarities can be pre-computed which results in faster execution of the algorithm. Consider a news website which recommends to users similar news articles that a user has viewed and rated for. This algorithm can be adopted which can recommend the similar news articles quickly by taking advantage of this pre-computation.

The classic item based algorithm

*for each item **x** that user **m** has not rated yet*
*for every other item **y** that user **m** has rated,*
*Similarity is computed between items **x and y** using any of the similarity metrics*
Weight each preference value in average according to how close that item is to the

active item ***x***

return the top ***N*** *recommended items, ranked according to the decreasing order of weighted average.*

Slope-One Recommender

Slope-one recommender is a slight variant of item based recommendation, which provides much simpler and faster recommendation. This approach works only when the users have given the rating scores explicitly. In this algorithm, the prediction of preferences for new items is done using the average difference in preference values between the items the user likes and the new item. This algorithm works under the assumption that for a pair of items that a user prefers, there exists a linear relationship between the ratings provided by the user for those items. For instance, let X and Y be the ratings given by the user for two items and as stated above the linear relationship can be denoted as Y= aX+b, which is a straight line equation where the slope 'a' is 1 and hence b = Y-X that is the difference between the ratings[19].

The main advantage of using slope-one recommender is that when the algorithm works online, the computation is very fast. The other advantage is that whenever a user changes the preference, updation can be easily done on the underlying data structure which includes updating only the differences in the ratings, but the limitation is that the memory requirement scales up proportional to the square of number of items. This approach is suitable in applications where there is constant change in the preferences of the users. In slope-one approach, run time computations are quick but it requires significant amount of time for the pre-computaion of its internal data structures.

The prediction equation is given below[15]:

$$p_{uj} = \bar{v}_u + \frac{1}{|R_j|}\sum_{j \epsilon R_j}\sum_{x \epsilon S_{ji}} \frac{V_{xi} - V_{xj}}{|S_{ji}|} \quad \text{..... (1)}$$

The prediction equation where weighting is employed[15] :

$$P_{uj} = \frac{\sum_{i \epsilon I_u - i_j}\left(\sum_{x \epsilon S_{ji}} \frac{V_{xi} - V_{xj}}{|S_{ji}|} + V_{ui}\right)|S_{ji}|}{\sum_{i \epsilon I_u - i_j}|S_{ji}|} \quad \text{..... (2)}$$

The classic Slope one algorithm

The slope- one algorithm consists of two phases :

- Preprocessing phase
- Recommendation phase

Preprocessing algorithm is given below:

for each item ***x***

for every other item ***y***

for each user ***m*** *who has rated for both items* ***x*** *and* ***y***

add the difference between the ratings in user ***m****'s preference value for items* ***x*** *and* ***y*** *to an average.*

Recommendation algorithm is given below:

for each item ***x*** *the user* ***m*** *has not rated yet*

for each item ***y*** *that user* ***m*** *has rated*

*the average preference difference between items **y** and **x** is computed*
*difference is added to user **m**'s rating for item **y**, then add to a running average*
return the top recommended items, ranked according to the decreasing order of these averages.

Apache Mahout

In this work, we have used Apache Mahout which is an open source machine learning library under the licence of Apache. The core algorithms, (known as the primary algorithms) that the Mahout implements are : Collaborative Filtering, Classification and Clustering [19]. Apart from these, Apache Mahout has implementations in Frequency pattern matching, topic modelling, genetic etc. The major advantage of Apache Mahout is that it is scalable and some of the algorithmic implementations can be distributed under Apache Hadoop which is a framework to perform MapReduce distributed computing. Hence the performance limitation of the collaborative filtering algorithms due to searching millions of users to find the nearest neighbourhood can be reduced. To support our research, we have used Apache Mahout because it provides many implementations in collaborative filtering that a recommendation system developer can make use of [20]. The collaborative filtering in Apache Mahout provides various similarity metrics and evaluation methods like Absolute Difference Error, MAE(Mean Absolute Error), RMSE(Root Mean Square Error) etc. Apache Mahout is readily extensible and also has a vibrant community [20].

Similarity Metrics

Similarity measures are employed in recommender systems in order to combine a multitude of preferences into a single preference value that can represent the similarity between two users or items. Apache Mahout supports many similarity metrics and they are described below.

Pearson Correlation Similarity

The Pearson Correlation Similarity coefficient between two users is defined as[19]:

$$s(a,u) = \frac{\sum_{i \epsilon I_a \cap I_u}(V_{ai} - \overline{V_u})}{\sqrt{\sum_{i \epsilon I_a}(V_{ai} - \overline{V_a})^2 \sum_{i \epsilon I_a \cap I_u}(V_{ui} - \overline{V_u})^2}} \quad \text{.....} \quad 3)$$

The value of the Pearson correlation is between -1 and +1 and it indicates the degree of linear relationship between the preference values: the value +1 indicates perfect positive correlation, the value 0 indicates no correlation and -1 indicates perfect negative correlation. Although Pearson correlation measure is often used in recommendation systems to find user or item similarity, there are some limitations associated with it which makes it not a good choice. First and foremost, this similarity measure does not consider the total number of items which two users like in common. For instance, two users who have read three books in common and given higher ratings has higher correlation ratio than two users who have read 300 books in common and given lesser ratings. Second, Pearson correlation cannot find correlation between users who have only one preferred item in common and hence this might be

a limitation in sparse datasets in which only a small number of users' items overlap. Moreover, the similarity cannot be computed if the preferences given by either user for all items are same values. Pearson correlation coefficient is best suitable for determining the similarity between ratings with values that are known to have linear relationship.

Euclidean Similarity

This similarity measure is based on the Euclidean distance between users where each user is represented as a point in n-dimensional space where there are n items and the preference values are represented as coordinates[21].

$$d(p,q)=d(q,p)=\sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$$

$$=\sqrt{\sum_{i=1}^{n}(q_i-p_i)^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

Cosine Similarity

The Cosine similarity is a similarity metric that depends on projecting user preferences as points in n-dimensional space. Similarity between users or items is found out by calculating the cosine angle between the lines projected from the origin to the corresponding points. Smaller the angle between them, the more similar the users /items are[19].

$$S(a,u)=\sum_{j\in I}\frac{V_{aj}}{\sqrt{\sum_{k\in I_a}V_{ak}^2}}\frac{V_{uj}}{\sqrt{\sum_{k\in I_u}V_{uk}^2}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5)$$

Spearman Correlation

This measure is a slight variant of Pearson correlation which first assigns relative ranking to all the preference values mostly assigning value 1 for the least preference and value 2 for next higher preference and so on. These transformed values are substituted in Pearson correlation formula to form the spearman correlation. In this, the ordering between the preferences is maintained but the real difference between the likeliness is lost[22].

Tanimoto Coefficient

Tanimoto coefficient, also known as Jaccard coefficient is given by ratio of total number of items two users have given ratings to the total number of items that either user has given preference for. The value is always greater than or equal to zero (no items in common) and the maximum value is +1 (all items are common). This similarity measure is employed when the users have not given explicit ratings but only liked the items[23].

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (6)$$

where A and B represent the item sets of user A and user B respectively.

Log Likelihood Similarity
This similarity metric works on the basis of determining how unlikely it is that two users have no interests in common, the more the probability of unlikeliness, the closer or more similar the users will be.

Implementation Details

In this paper, we have performed a comparative study of three popular memory based recommendation algorithms in Apache Mahout. The algorithms that we have implemented in this work are User based recommendation, Item based recommendation and Slope-one recommendation on Movie Lens dataset of two different sizes: 100K and 1M. We evaluated the performance of these algorithms on the basis of evaluation metrics precision, recall, fallout, reach and f-measure. We also analysed the sensitivity of these evaluation metrics on the neighbourhood size using Pearson Correlation similarity and Euclidean distance similarity. Further, we demonstrated the effect of Absolute Difference error and Root Mean Square error on the neighbourhood size in user based CF.

Dataset
The datasets used in this experiment are the MovieLens datasets from GroupLens Research . Two datasets of different sizes were used:

- The ML100K dataset which has 943 users, 1,682 movies and 100,000 ratings for these movies [24].
- The ML1M dataset which has 6,040 users, 3,900 movies and 1,000,209 ratings for these movies [24].

The rating scores given in these datasets consist of integers within the numerical range between 1 (least likeliness) to 5 (highest likeliness).

Collaborative Filtering in Apache Mahout
Recommendations can be done in either of the two ways:

- Using the browsing patterns of a user as the indicators of his/her likeliness.
- Using the explicit ratings given by the user against an item which will be within a numerical range.

In this paper, we are considering only the case where explicit ratings are provided by users for the items. In order to perform CF in Apache Mahout, we need to have an input which consists of User identity, Item identity and a rating score within a numerical scale. This triple is termed as 'preference' in Mahout, which indicates the strength of a user's likeliness towards an item. The preference object is a triple which consists of the attributes: User ID, Item ID, Rating and is stored in User Preference Array. The performance of recommendation systems depend on the quality and quantity of data. Since it accesses a huge amount of information, Mahout has chosen the data structures and their representation in such a way that they do not affect the run time performance of the recommender algorithms. Each rating given by a user to an item is stored as a preference object. In Mahout, the set of preferences is not represented using Collections in Java since they consume more memory. Instead

Mahout uses a different way of representation in which it maintains a particular user ID and all the associated items and the corresponding ratings, thus saving memory. Moreover Mahout uses data structures like Fast ID Set, Fast Map, and Fast By ID Map instead of Hash Map and Tree Set, which reduce memory space and improve performance of the recommender [19].

In-memory Data Models

Data models are used to access and use the preference objects and there are different kinds of in-memory data models which vary according to the input data. Generic Data Model (used when the data is in memory rather than from other external sources like file, database etc.), File Data Model(when the input data is taken from a file and stored in the memory), JDBC Data Model and My SQLJDBC Data Model (used when the input data is accessed through JDBC and MYSQL connector). Sometimes we may not have the preference for items for example, when the recommendations are based on the hyperlinks the user has clicked. In Mahout, it is termed as "boolean preferences" which can be accessed using Boolean Pref Data Model[19].

Evaluation

The performance of the recommender algorithms is evaluated for the better understanding of how the algorithms meet the requirements. Initially, the only criteria for the evaluation of the recommender algorithms were their accuracy: how well the algorithms were able to capture the user's interests. Although accuracy is very important in determining the prediction power of the algorithms, it is not sufficient for a good recommender algorithm. Generally, what users expect from recommender engines is capturing their exact tastes. Apart from this, there are many applications where users use recommender systems to discover/explore new and diverse items, to obtain fast responses of the recommendation system, privacy preservation, and other forms of interaction with the recommendation system. Therefore, Recommender systems should meet the properties specific to the application and evaluation must be done on the basis of specific properties of the application [18].

One of the ways of evaluating the recommendations of a recommender system is to determine how close the estimated preferences with the actual preferences are: this can be done by calculating the difference between the estimated preference and the actual preference. This difference is called as *'score'* and it determines the quality of the recommendations produced by the recommender systems. The smaller the value of *score*, higher is the quality of the recommendations. This type of evaluation is done by dividing the dataset into training set and testing set and then estimating the preferences for the test data which are then compared with the actual values. In this work, we have used 90 percent of the whole dataset as training set and rest 10 percent as test set.

There are three ways of measuring prediction accuracy:

- accuracy in terms of ratings predictions
- accuracy in terms of usage predictions
- accuracy in terms of item ranking [18].

Root Mean Squared Error (RMSE) and Mean Squared Error (MAE) are considered to be popular metrics used for evaluating predicted ratings accuracy which depend only on the magnitude of the difference between predicted and actual preferences [18]. The MAE and RMSE between the predicted and actual preferences are given by the equations[15]:

$$|\bar{E}| = \frac{\sum_i^N |p_i - v_i|}{N} \quad \text{.....} \quad (8)$$

$$|\bar{E}| = \sqrt{\frac{\sum_i^N (p_i - v_i)^2}{N}} \quad \text{.....} \quad (9)$$

There are slight variations of RMSE and MAE denoted as Normalized RMSE (NMRSE) and Normalized MAE (NMAE) in which the preferences are normalized within the range r-min and r-max. Another variation of RMSE and MAE are called as Average RMSE and Average MAE which are used to adjust test data that are unbalanced. In another type of evaluation, the accuracy depends on whether the recommender system will be able to effectively predict whether the user will use the items ie accuracy in terms of usage predictions [18]. The evaluation metrics used in information retrieval can be used here:

- Precision: Precision is the ratio of top recommendations to good recommendations.
- Recall: Recall is the ratio of good recommendations to the top recommendations.
- F-measure : F-measure is the harmonic mean of metrics precision and recall.
- Fallout: Fallout is the ratio of non-relevant items that are retrieved to the total non-relevant items available
- Reach: Reach will return the fraction of users out of total users for whom recommendations can be done.

Both precision and recall must be taken into consideration in order to evaluate the performance of recommender algorithm completely [25]. Precision and recall depends on the number of items retrieved. Higher the numbers of items returned, higher the recall and lower the precision. F-measure combines precision and recall into a single metric.

Results and Analysis

In this paper, we have done the performance comparison of UCF-P (User based Collaborative Filtering using Pearson correlation similarity), UCF-E (User based Collaborative Filtering using Euclidean distance similarity), ICF-P (Item based Collaborative Filtering using Pearson correlation similarity), ICF-E (Item based Collaborative Filtering using Euclidean distance similarity), and slope one recommender using the evaluation metrics Precision, Recall, F-Measure and fallout in Movie Lens 100K dataset. Figure 1 suggests that the slope one approach outperforms the other algorithms in terms of precision recall, f-measure and fallout. Figure 2 presents the performance comparison of the above stated algorithms in Movie Lens

1M dataset which clearly indicated the performance improvement of slope one approach over the other algorithms stated above.

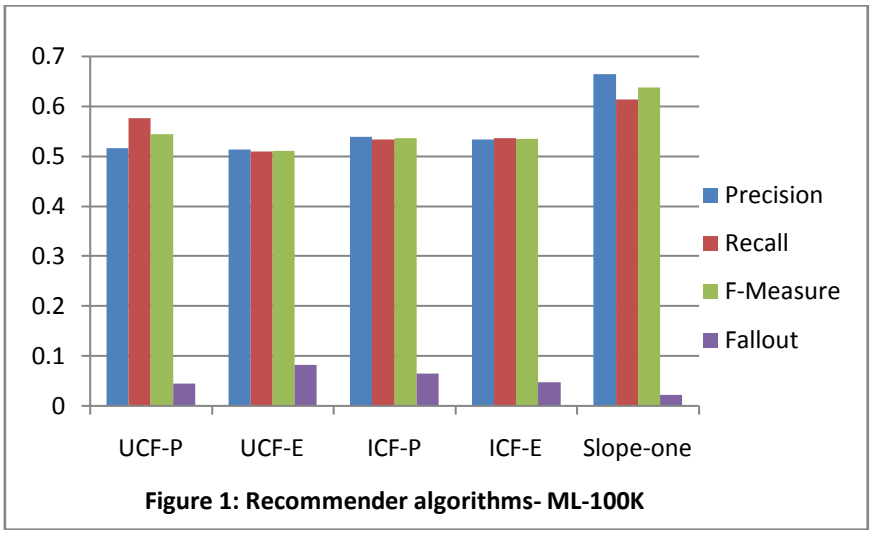

Figure 1: Recommender algorithms- ML-100K

The evaluation metrics Absolute Difference Error (ADE) and Root Mean Square Error (RMSE) greatly depend on the number of neighbours in the neighbourhood based approach. The Figure 3 depicts the sensitivity of neighbourhood size on Movie Lens 100k dataset.

The sensitivity of evaluation metrics on the neighbourhood size are given in the Figure 4, Figure 5, Figure 6 and Figure 7. The similarity metric to compute user neighbourhood in Figure 4 and Figure 5 is Euclidean distance similarity and in Figure 6 and Figure 7 is Pearson correlation similarity and the datasets used here are ML-100K and ML-1M respectively.

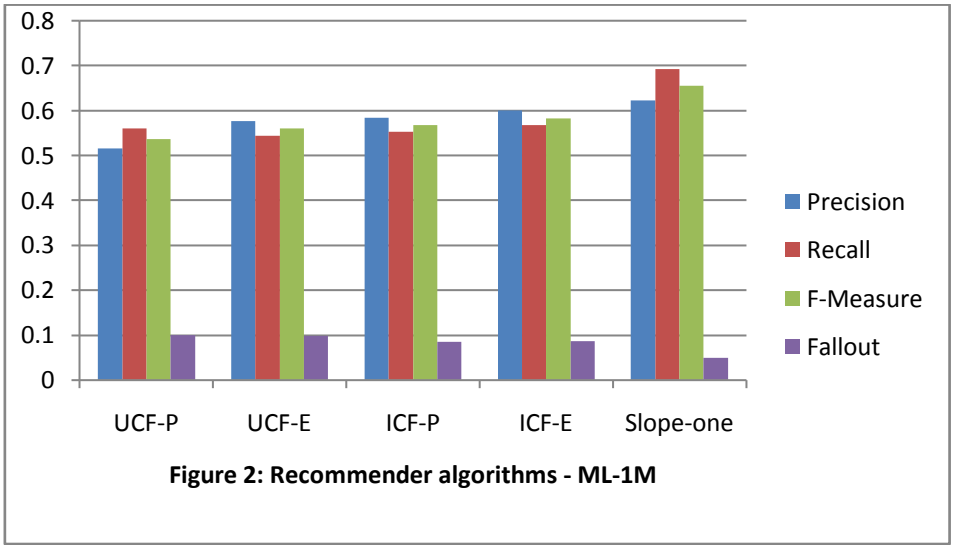

Figure 2: Recommender algorithms - ML-1M

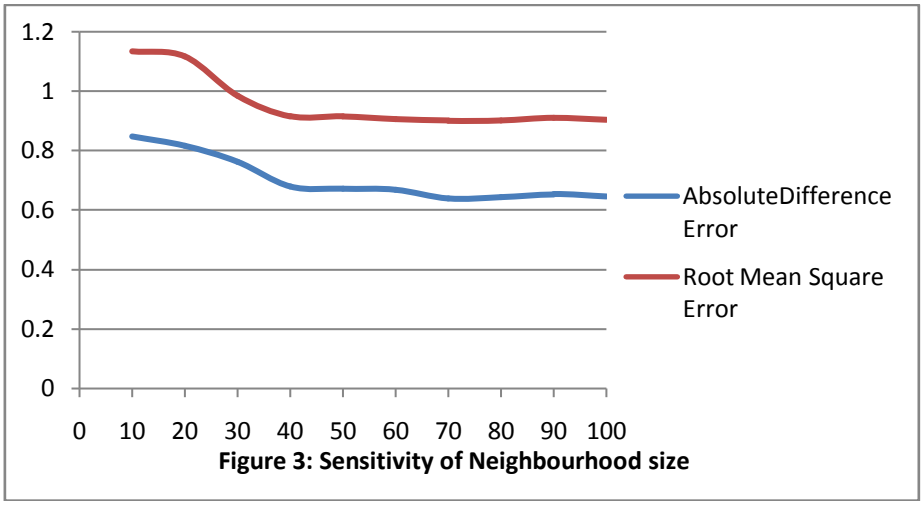

Figure 3: Sensitivity of Neighbourhood size

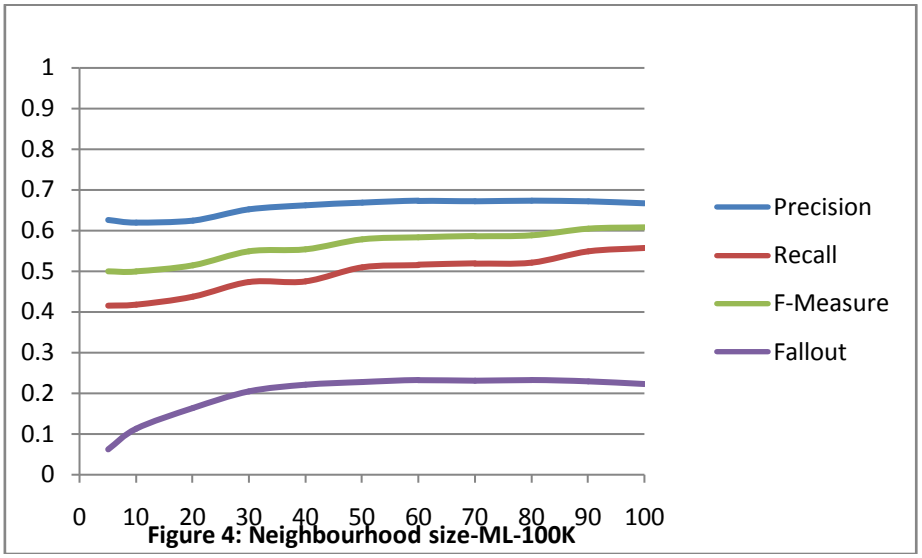

Figure 4: Neighbourhood size-ML-100K

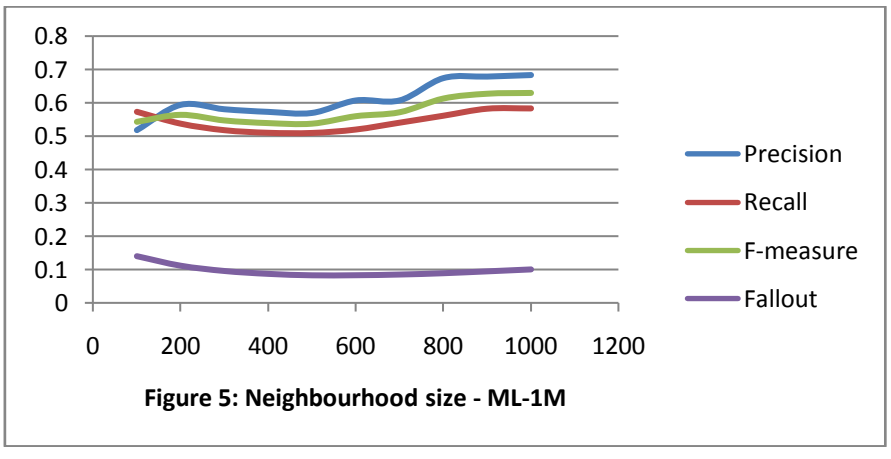

Figure 5: Neighbourhood size - ML-1M

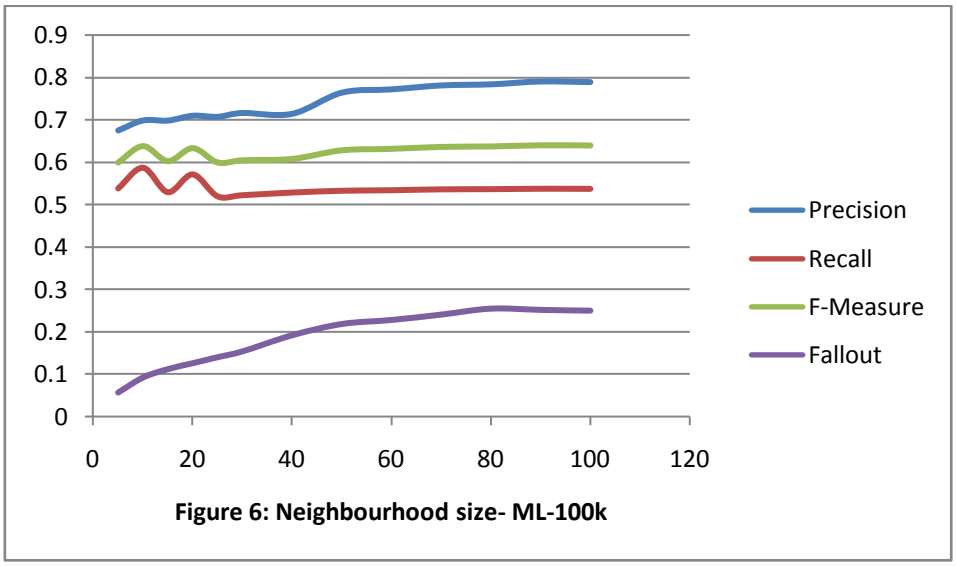

Figure 6: Neighbourhood size- ML-100k

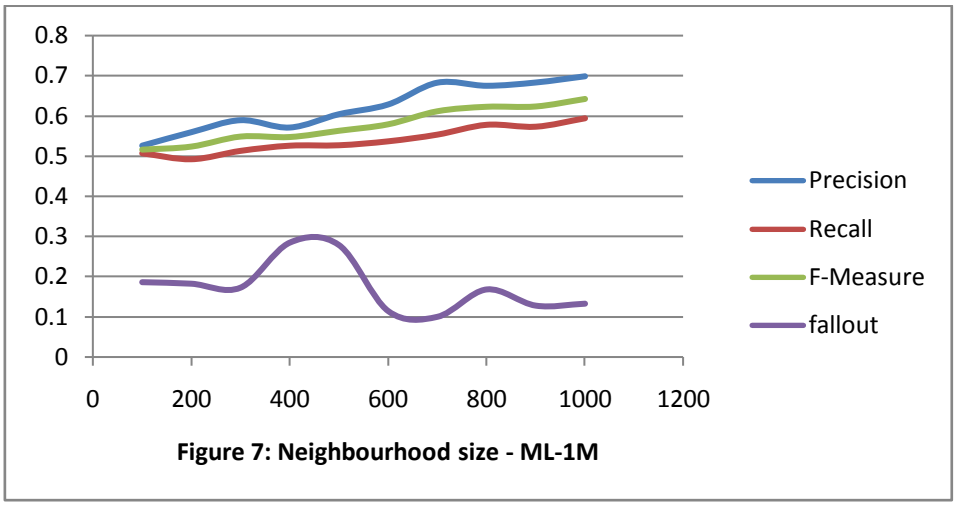

Figure 7: Neighbourhood size - ML-1M

Conclusion

In this paper, we presented a performance analysis of three CF-based recommender systems. We first presented a short overview of recommender systems and their importance in the present scenario. Then we discussed the various types of recommender systems, their methodology of working and advantages and disadvantages of each approach.

The three CF based algorithms taken into consideration in this work are User based CF, Item based CF and Slope one recommender. In this paper, we have implemented these CF algorithms in Apache Mahout using the datasets Movie Lens of two different sizes : 100K and 1M and also evaluated its precision, recall, f-measure, fallout and reach. The results demonstrate that slope-one recommender outperformed the classic user based and item based CF in both datasets. We have also depicted the

sensitivity of Absolute Difference Error and Root Mean square Error on the neighbourhood size. Further, we have analysed the effect of neighbourhood size on precision, recall, f-measure and fallout using Pearson correlation similarity and Euclidean distance similarity on the two datasets.

The research on recommender systems signifies the importance of improving recommendation performance. Due to the popularity of ecommerce websites and online social networks, more algorithms are required to better capture or mine the real interests of users. Most of the recommender algorithms work offline and steps must be taken to improve the quality of online recommendations.

References

[1]. Chen, Y., Wu, C., Xie, M., and Guo, X., 2011, "Solving the sparsity problem in recommender systems using association retrieval," Journal of computers., 6(9), pp. 1896-1902.

[2]. Rao, K. N., 2010, "Application domain and functional classification of recommender systems—a survey," DESIDOC Journal of Library & Information Technology., 28(3), pp. 17-35.

[3]. Bobadilla, J., Ortega, F., Hernando, A., and Bernal, J., 2012, " A collaborative filtering approach to mitigate the new user cold start problem," Knowledge-Based Systems, 26, pp. 225-238.

[4]. Papagelis, M., Plexousakis, D., and Kutsuras, T., 2005, " Alleviating the sparsity problem of collaborative filtering using trust inferences," In Trust management pp. 224-239.

[5]. Papagelis, M., Rousidis, I., Plexousakis, D., and Theoharopoulos, E., 2005, "Incremental collaborative filtering for highly-scalable recommendation algorithms," In Foundations of Intelligent Systems pp. 553-561.

[6]. Niloofar, R., and Mansoor Z. J., 2014, "Using Content Features To Enhance The Performance Of User-Based Collaborative Filtering," International Journal of Artificial Intelligence & Applications (IJAIA), 5(1).

[7]. Adomavicius, G., and Tuzhilin, A., 2005, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering," IEEE Transactions on., 17(6), pp. 734-749.

[8]. Schafer, J. B., Konstan, J. A., and Riedl, J., 2001, " E-commerce recommendation applications," In Applications of Data Mining to Electronic Commerce., pp. 115-153).

[9]. Lee, J., Sun, M., and Lebanon, G., 2012, "A comparative study of collaborative filtering algorithms," arXiv preprint arXiv.,1205.3193.

[10]. Yang, X., Guo, Y., Liu, Y., and Steck, H., 2014, " A survey of collaborative filtering based social recommender systems," Computer Communications, 41, pp. 1-10.

[11]. Desrosiers, C., and Karypis, G.,2011, " A comprehensive survey of neighborhood-based recommendation methods," In Recommender systems handbook, pp. 107-144.
[12]. Sarwar, B., Karypis, G., Konstan, J., and Riedl, J., 2001, " Item-based collaborative filtering recommendation algorithms," In Proceedings of the 10th international conference on World Wide Web, pp. 285-295.
[13]. Pagare, R., and Patil, S. A., 2013, " Study of Collaborative Filtering Recommendation Algorithm-Scalability Issue," International Journal of Computer Applications, 67(25).
[14]. Seitlinger, P., Kowald, D., Kopeinik, S., Hasani-Mavriqi, I., Ley, T., and Lex, E., 2015, " Attention Please! A Hybrid Resource Recommender Mimicking Attention-Interpretation Dynamics," arXiv preprint arXiv.
[15]. Cacheda, F., Carneiro, V., Fernández, D., and Formoso, V., 2011, " Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," 5(1).
[16]. Bellogín, A., and de Vries, A. P., 2013, " Understanding similarity metrics in neighbour-based recommender systems," In Proceedings of the 2013 Conference on the Theory of Information Retrieval, p. 13.
[17]. Wang, P., and Ye, H., 2009, " A personalized recommendation algorithm combining slope one scheme and user based collaborative filtering," In Industrial and Information Systems, pp. 152-154.
[18]. Shani, G., and Gunawardana, A., 2011, " Evaluating recommendation systems," InRecommender systems handbook, pp. 257-297.
[19]. Anil, R., Dunning, T., and Friedman, E., 2011, "Mahout in action".
[20]. Seminario, C. E., and Wilson, D. C., 2012, "Case study evaluation of mahout as a recommender platform," In workshop on recommendation utility evaluation Beyond RMSE, held in conjunction with ACM in Ireland.
[21]. http://en.wikipedia.org/wiki/Euclidean_distance
[22]. http://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient
[23]. http://en.wikipedia.org/wiki/Jaccard_index
[24]. http://grouplens.org/datasets/movielens/
[25]. Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T., 2004, "Evaluating collaborative filtering recommender systems," ACM Transactions on Information Systems (TOIS), 22(1), pp. 5-53.