Final Project Report

Empirical Analysis of Predictive Algorithms for Collaborative Filtering

Dhayalini Nagaraj

A20359686

Abstract:

Collaborative filtering or recommender system uses a database about user preference to predict the products and topics that a new user might like. In this project, we use different algorithms for this task based on correlation coefficients, vector similarity calculations to predict the user preferences. We analyse and compare different algorithms based on accuracy and mean absolute error in a set of representative problem domains.

1. Introduction:

The automated search over a corpus of items is based on a query identifying intrinsic features of the items sought. Most content retrieval methodologies use some type of similarity score to match a query describing the content with the individual titles or items, and then present the user with a ranked list of suggestions. The system depends on the usage or patterns of other users. Collaborative filtering is a method of making automatic predictions about the interests of a user by collecting preferences or taste information from many users. It is also known as recommender systems. The assumption is that if a person "A" has a opinion in an issue "C" and a person "B" has a similar opinion in the same issue "C", then more likely both the person "A" and "B" will have a similar opinion in a different issue. In this project, we use different algorithms for this task based on correlation coefficients, similarity calculations to predict the users preferences. We analyse and compare different algorithms based on accuracy in a set of problem domains. We will use books data and apply collaborative filtering techniques to recommend books to users and compare the predictive accuracy of each algorithms and decide the efficient algorithm. Books Dataset associated with Amazon Product data is used for the experiments which has reviews and ratings of the user for each data items.

2. Data

2.1 Data Source

Books datasets is used for this project. The dataset is obtained from Amazon Product data and it can be accessed from the site http://jmcauley.ucsd.edu/data/amazon/links.html. We have a two types of datasets for books. They are 'K-core' and 'Ratings only' dataset. These datasets contains reviews spanning May 1996 - July 2014. The 'K-core' data set contains 8,898,041 reviews and the 'Ratings only' dataset contains 22,507,155 ratings. The 'K-core' is dense and is approximately 3GB. The 'Ratings only' dataset includes no reviews but only ratings and the size is approximately 1 GB. Since the 'K-Core' dataset is more, it cannot be processed on disk. So we need to use an instance of AWS to process the data. The 'Ratings only' data can be processed on disk and we planned to implement different algorithms using this dataset.

2.2 Data Format

Ratings: The dataset includes no metadata or reviews. It includes only User, item, rating and timestamp tuples.

- reviewerID - ID of the reviewer
- asin - ID of the product
- overall - rating of the product
- unixReviewTime - time of the review (unix time)

Sample:

{"reviewerID": "AH2L9G3DQHHAJ",
"asin": "0000013714",
"overall": 4.0,
"unixReviewTime": 1019865600}

K-core:

This dataset is a reduced subset, in which each user and items have k reviews each. It has reviewerID, asin, reviewerName, reviewText, overall, summary, unixReviewTime, reviewTime.

2.3 Programming Language, packages

Python language is used to implement and analyse the algorithms. The packages that we will use for this project are Scipy, Pandas, nltk, scikit-learn, Numpy, countVectorizer.

2.4 Preliminary Experiments:

Data Pre-process:

a) Load Dataset: The pandas framework is used to load the dataset. The books rating only file is downloaded from the above mentioned site and is loaded and read.

b) Matrix data: The data set is loaded and then converted into user-item matrix. It is a sparse matrix where each row represents the user and each column represents the book and its rating.
The item for which the ratings are empty is filled by 0.The User- Item matrix would be:

	B1	B2
R1	4	3.6
R2	3	0

c) Filtering data and data used: The ratings dataset has 22,507,155 ratings. In order to perform experiments I have used 500,000 ratings. This was filtered out based on the user and books who have reviewed less than 10 and which has more than 10 ratings.

d) Normalization: The normalization was done using the sklearn package. This uses min-max normalization in which we fit the data in a predefined boundary.

Then cosine similarities are found and the recommendation system is further built based on that.

3. Approaches

Recommender systems typically produce a list of recommendations in one of two ways through collaborative and content-based filtering or the personality-based approach.

- Collaborative filtering approaches building a model from a user's past behaviour as well as similar decisions made by other users. This model is then used to predict items that the user may have an interest in.

- Content-based filtering approaches uses a series of discrete characteristics of an item in order to recommend additional items with similar properties.

We focus on collaborative filtering techniques. Collaborative filtering uses implicit or explicit user votes to build the system**.** Explicit voting refers to a user consciously expressing his or her preference for a title, usually on a discrete numerical scale. In book dataset, the ratings scale is from 1 to 5, where the user rate each book explicitly based on the scale. Implicit votes are based on other browsing data, purchase history and other types of information. Since we are using explicit votes for our project, the amount of missing vote will be nil and the dataset is also huge.

A variety of algorithms have been reported and evaluated empirically. Some of the most popular algorithms in collaborative filtering use a correlation-based approach. We will be evaluating three types of algorithm in this project and they are

1. **Memory-based algorithms**
    a) **Correlation**
    b) **Vector Similarity**

2. **Extensions to memory based algorithms**
    a) **Default Voting**
    b) **Inverse-User Frequency**

3. **Model based algorithms**
    a) **Matrix Factorization**
    b) **Cluster Model Collaborative Filtering**

Memory-based algorithms operate over the entire user database to make predictions. In Model based collaborative filtering, the user database is used to estimate or learn a model, which is then used for predictions.

4. Implementations and Experiments:

4.1 Memory Based Algorithm

In memory based algorithms in order to measure similarity correlation between two user is found. This gives us a value from -1 to 1 which determines who alike two users are. A value of 1 means that they both rate in the exactly the same manner, whereas a value of -1 means that they rate things exactly opposite. The votes are represented as $v_{i,j}$. It means the vote of user ' i ' for item ' j '. The Items set that is voted by the user ' i ' is represented as I_i. The mean vote of the user is

$$\overline{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$$

Based on partial previous preference and similarity we predict votes of a user. It is calculated using the following formula.

$$p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i)$$

Here K is normalizing factor, n represents the unique users in the database, the correlation or similarity between the users is weight w. The weight calculation into two different approaches.

4.1.1 Correlation Similarity

Pearson correlation coefficient: The basic correlation algorithm used to measure the ratings and compare them. They will get a positive correlation if they vary in the same way else negative correlation. The correlation between user 'a' and ' i ' is found using

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

Here the summations over j are over the items for which both users a and i have recorded votes.

4.1.2 Vector Similarity

The second similarity measurement is called vector similarity. Treat two users as vectors in n-dimensional space where n is the number of items in the dataset. Using the two vectors, compare the angle between them. If it points to the same direction it gives positive similarity else negative similarity. The vector frequencies of user item and votes are calculated and compared. In this case, the votes indicate positivity and no role for negativity. The items with no votes takes zero as values.

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$

The vector similarity between active user and other users are found based on cosine values.

4.2 Extenions to Memory based Algotithm

Some modifications are done to the standard algorithms for better performance.

4.2.1 Default Voting

It simply adds a number of imaginary items that both have rated in common in order to smooth the votes. When only few votes are present for active user or the matching user, the correlation algorithm will not perform well. This is because it uses only the votes in the intersection of the items that both individuals have voted on (Ia ∩ Ij). If we assume some default value as a vote for items for which we do not have explicit votes, then we can form the match over the union of voted items,(Ia ∪ Ij), where the default vote value is inserted into the formula for the appropriate unobserved items.

4.2.2 Inverse-User Frequency

In this, we just transform each vote when weighting two users, such that commonly rated items are given less importance. The idea used in this algorithm is to reduce weights for commonly occurring items, capturing the intuition that they are not as useful in identifying the topic, while items that occur less frequently are more indicative of topic. Analogous transformation is done to the votes which is known as inverse user frequency. The transformed vote is simply the original vote multiplied by the fj factor.

$$w(a,i) = \frac{\sum_j f_j \sum_j f_j v_{a,j} v_{i,j} - (\sum_j f_j v_{a,j})(\sum_j f_j v_{i,j}))}{\sqrt{UV}}$$

where,

$$U = \sum_j f_j (\sum_j f_j v_{a,j}^2 - (\sum_j f_j v_{a,j})^2)$$

$$V = \sum_j f_j (\sum_j f_j v_{i,j}^2 - (\sum_j f_j v_{i,j})^2)$$

Case Amplification:

A transform applied to the weights of the user, used in the basic collaborative filtering prediction formula is known as Case Amplification. The weights are transformed using the below fomula.

$$w'_{a,i} = \begin{cases} w_{a,i}^{\rho} & \text{if } w_{a,i} \geq 0 \\ -(-w_{a,i}^{\rho}) & \text{if } w_{a,i} < 0 \end{cases}$$

4.3 Model Based Algorithm

Model based algorithm is building a model from the dataset of book ratings and use that model to build the recommendation system without using the complete dataset. We use Probability of predicting method.

4.3.1 Matrix Factorization

Matrix factorization is the process of breaking down a matrix into a product of multiple matrices. Singular value decomposition is used to break the matrix into three parts. SVD is used to find the high level features from users and books. The singular vectors and corresponding singular values produced by SVD allow users and books to be mapped into the same "latent semantic space".

D - Diagonal matrix of eigen values.

U - Represents the relationship between users and features

V_t -Represents the relationships between features and movies.

The product of the matrix has the original values appromixates and is used to predict the ratings for other users.

4.3.2 Cluster Model Collaborative Filtering

Cluster Model algorithm is building a cluster model from the dataset of book ratings and use that model to build the recommendation system without using the complete dataset. We use Probability of predicting method and clustering is used for this algorithm.

I have worked on Cluster Model. I have used K-means clustering algorithm to build the cluster model. The idea I have used in this is there are certain groups or types of users who have common set of preferences. So I have used K-means clustering algorithm to cluster similar users who has similar rating for books.

K is an input to the algorithm that specifies the desired number of clusters. I have used k =3 for the clustering algorithm. In the first pass the algorithm takes the first k users as the centroid of k unique clusters. The remaining users are then compared to the closest

centroid. The cluster centroids are recomputed based on cluster centroids formed in the previous pass. The N number of users has to be clustered. The ratings dataset has 22,507,155 ratings. In order to cluster the users I have used 500,000 ratings. Clustering results of the users U={u1, u2, u3, u4, u5} are represented as {C^1, C^2, C^3, ..., C^k}.

In order to cluster similar users, I calculated the similarities between pairs of users and identify their neighbourhood. I used Pearson correlation coefficient function for measuring the similarity between the users.

$$sim_{u,u'} = \frac{\sum_{t \in T(u) \wedge T(u')} (R_u(t) - \overline{R_u}) \cdot (R_{u'}(t) - \overline{R_{u'}})}{\sqrt{\sum_{t \in T(u) \wedge T(u')} (R_u(t) - \overline{R_u})^2} \sqrt{\sum_{t \in T(u) \wedge T(u')} (R_{u'}(t) - \overline{R_{u'}})^2}}$$

In this equation t is the items and $R_u(t)$ is the ratings given by the user for the item t.

Data Smoothing.

Based on the cluster results, the data smoothening methods to items with no ratings.

$$R_u(t) = \begin{cases} R_u(t) & \text{if user } u \text{ rate the item } t \\ \hat{R}_u(t) & \text{else} \end{cases}$$

$$\hat{R}_u(t) = \overline{R_u} + \Delta R_{C_u}(t)$$

The $R_{Cu}(t)$ is average deviation rating for all the users in the cluster.

Based on the similarities, the neighbour of the user is found. For this we can search the whole database. But when the number of users are more, it is difficult. So I used the cluster concept. The centroid of a cluster is represented by average rating of all users in the cluster. The similarity between each group and the active users are again calculated.

Now we have two ratings, the user ratings and the group ratings. The confidential weight for the user u to the item t is

$$w_{ut} = \begin{cases} 1-\lambda & \text{if user } u \text{ rate the item } t \\ \lambda & \text{else} \end{cases}$$

Here λ is the parameter for tuning the weight between original rating and group rating. The value of λ is set to 1 for the cluster based filtering algorithm.

The votes are integer valued with a range for 0 to m we have

$$p_{a,j} = E(v_{a,j}) = \sum_{i=0}^{m} \Pr(v_{a,j} = i | v_{a,k}, k \in I_a)\ i$$

The probability expression is the probability that the active user will have a particular vote value for item j given the previously observed votes.

$$\Pr(C = c, v_1, \ldots, v_n) = \Pr(C = c) \prod_{i=1}^{n} \Pr(v_i | C = c)$$

The parameters of the model, the probabilities of class membership Pr(C = c), and the conditional probabilities of votes given class Pr (vi |C = c) are estimated from a training

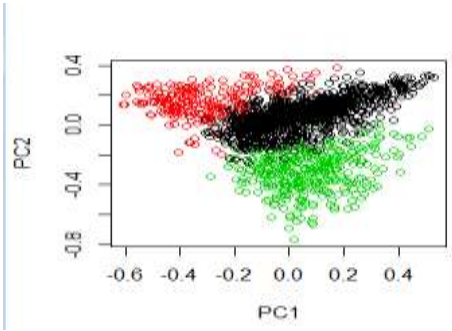
set of user votes, the user database. Based on this Cluster Probability Model, we then build a recommendation system and trained the test data.

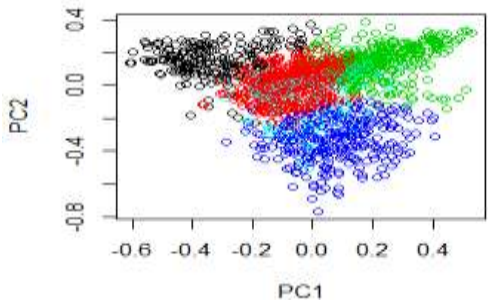
5. Experiments and Results:

The cluster model algorithm was implemented by me and the following experiments are done with the algorithm.

Cluster1 **Cluster2**

The users are grouped using the similarities found. K-Core data set is used to find the frequency of words. The features, reviews and ratings are also considered and the word cloud for each cluster is done which shows the frequency of the word in each cluster.



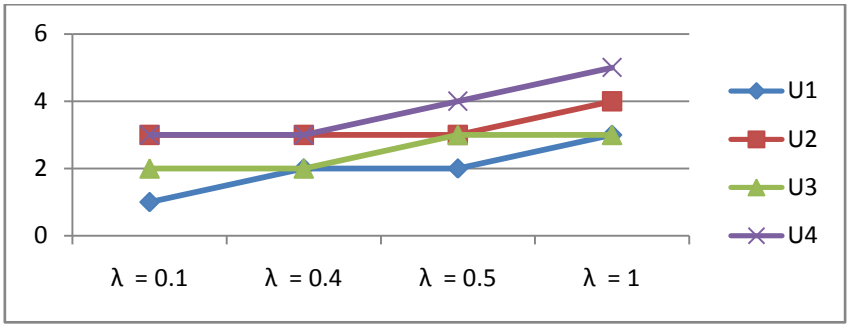


K=3 k=5

To build a cluster model the similarities of the users were calculated and the users are grouped together. In order to divide the similar users k value was altered from k= 1 to 10. The best result occurred when k =3. From the visuals it is seen that when the users are divided into three cluters, the similarity between the three clusters are nil. The cluster points are not scattered. Each cluster represent group of user who has similarities. When k=5, it is seen that four clusters are seen visibly but the cluster points are scattered and the fifth cluster is not seen clearly. This shows that the users are not grouped into similar groups and there are variations. So we took k as 3 and grouped users into 3 to build probability model.

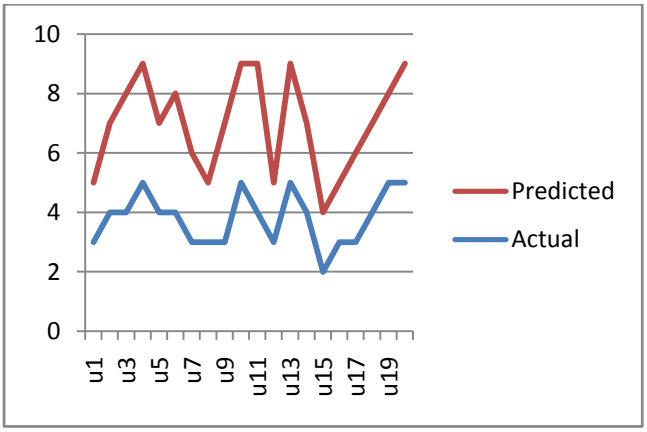
Effect of λ variation:

Users	λ = 0.1	λ = 0.4	λ = 0.5	λ = 1
A1KLCGLCXYP1U1	1.0	2.0	2.0	3.0
A2EIPZNHAEXZHJ	3.0	3.0	3.0	4.0
A14A5Q8VJK5NLR	2.0	2.0	3.0	3.0
A1MC4E00RO5E9T	3.0	3.0	4.0	5.0



Varying lambda value the predicted ratings

The dataset is taken as test dataset and passed to the model and the results of the book dataset is analysed. I varied the number of rated items provided by the active user and analysed the results. When the parameter λ was varied from 0.1, 0.4, 0.5 and 1 the prediction performance increased. This shows when the cluster model is formed, the neighbours has to be selected with high similarities which would in turn affect the predictions.



User	Actual	Predicted
U1	4.0	2.0
U2	4.0	3.0
U3	5.0	4.0
U4	4.0	4.0
U5	4.0	3.0
U6	3.0	4.0
U7	3.0	3.0

Actual Vs Predicted

Evaluation Crietria:

The performance of the algorithm is evaluated using mean absolute error (MAE). MAE has the same unit as the original data, and it can only be compared between models whose errors are measured in the same units. The difference between predicted rating and actual rating are used to find the error and accuracy.

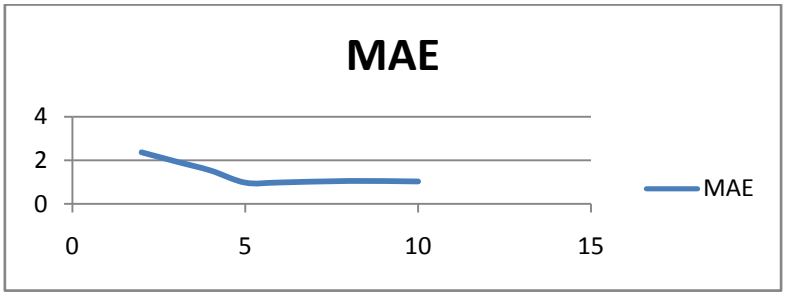
$$MAE = \frac{\sum_{i=1}^{n} |p_i - a_i|}{n}$$

Effect of K variation:

The dataset is divided into training and test data and then passed through the implemented cluster model filtering for predicting the books and ratings for other users.

K	MAE
2	2.367
3	1.945
4	1.532
5	0.978

6	0.986
8	1.056
10	1.032



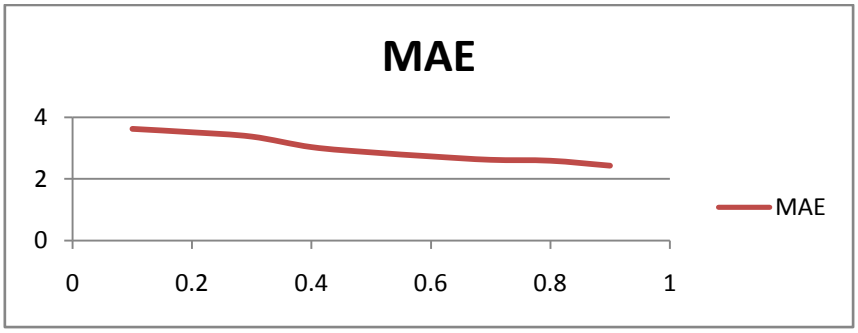
K vs MAE

When the k value vary, the predictions varied. When the cluster is more it provides the information more specific while the small clusters represent general difference among dissimilar users. The MAE decreases gradually when k value is increased. When K=5 the error was less and increasing the k values, the error was increased

Effect of varying training and test data:

When the training and test data was varied from 20, 40 and 60% for a fixed k=5 the mean absolute error decreased.

Data Ratio	MAE
0.1	3.62
0.2	3.51
0.3	3.37
0.4	3.03
0.5	2.86
0.6	2.73
0.7	2.62
0.8	2.59
0.9	2.43



Data vs MAE

From the above result it is clearly seen that when the training data is increased the MAE. When more data is included in the training set, then the recommendation for other new users are predicted with less MAE. This shows that the model works correctly when more data is used to build the cluster and training set.

Advantages and Disadvantages:

The accuracy of the prediction is increased. It also solves the scalability problem, prediction speed is also faster because the time required to query the model is faster than to query the whole dataset. It is difficult to add a new data to the model based system which makes it inflexible. Quality of predictions depends on the way the model is built so the prediction quality would be poor.

Recommendations for a User:

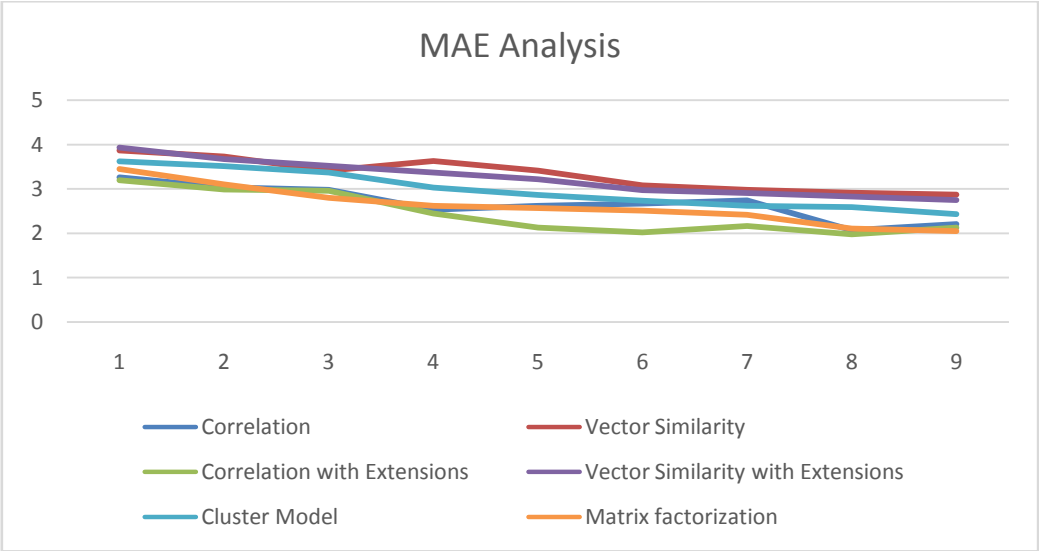
The system is to recommend books for a user based on his previous selections and similarities of the user. I selected A1MC4E00RO5E9T user and based on the ratings the recommended items for the user are : **Book ID: 13714, 29831, 41696, 230022, 913154**.

When compared with actual data the user has given good ratings for these books.

6. Analysis of all the algorithms

The following table and graph shows the various algorithms and the MAE values of each algorithm when the sample data ration is altered. Same data set is taken for the experiments and the errors are calculated to compare them.

Data Ratio	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Correlation	3.26	3.04	2.98	2.53	2.62	2.67	2.74	2.07	2.21
Vector Similarity	3.87	3.73	3.41	3.63	3.41	3.08	2.98	2.91	2.87
Correlation with Extension	3.2	2.99	2.96	2.45	2.13	2.02	2.17	1.98	2.13
Vector Similarity with Extension	3.93	3.67	3.52	3.37	3.22	2.97	2.91	2.83	2.75
Cluster Model	3.62	3.51	3.37	3.03	2.86	2.73	2.62	2.59	2.43
Matrix factorization	3.45	3.1	2.8	2.62	2.57	2.51	2.42	2.11	2.05



Collaborative Filtering Algorithms and MAE Analysis.

The above table and chart shows the data ratio and MAE values for each algorithms. It is clearly seen that when the data is increased the MAE values for all the algorithms is decreased. That is when the dataset size is more and sparsity of data is less, the algorithms works well.

While comparing Correlation and Vector Similarity Algorithms with its Extensions Algorithms, it is seen that the extensions algorithms performs well. Instead of using the similarities of the user-item adding default value and using inverse frequency will recommend the books to user correctly.

Comparing the model based algorithms, though both algorithm performs well, matrix factorization using SVD is better because the cluster model is dependent on the similarities of users in cluster and if the cluster is not accurate then the performance of the system will decrease.

By analysing all the algorithms, matrix factorization and the correlation algorithm with extensions outperforms the other four algorithms.

7. Conclusion

The various collaborative filtering algorithms are implemented and the performance of the recommendation system are analysed by varying parameter values. It is clearly shown from the MAE values of algorithms that recommendation system built using the Matrix Factorization and Correlation with Extensions algorithms perform well when compared to other algorithms.
I learnt about different algorithms and it's performance. The difficulties I faced was to build a proper cluster model and to recommend items for a new user. The Memory based algorithms requires the entire database and predicts slow, but the predictions are good and easy to update the database. The Model based algorithms requires small memory, predicts faster, reduces scalability problems, but the prediction quality is affected since it depends on the model.

References:

http://jmcauley.ucsd.edu/data/amazon/links.html
http://www.haas.berkeley.edu/Courses/Spring2000/BA269D/BreeseHeckermanKadie99.pdf
http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/modelbased.html
https://ashokharnal.wordpress.com/2014/12/18/worked-out-example-item-based-collaborative-filtering-for-recommenmder-engine/
http://aimotion.blogspot.com/2009/11/collaborative-filtering-implementation_13.html
https://www.codementor.io/jadianes/build-data-products-django-machine-learning-clustering-user-preferences-du107s5mk
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.227.5662&rep=rep1&type=pdf
http://www.cis.upenn.edu/~ungar/Datamining/Publications/clust.pdf