Movie Recommendation Engine

Mohammed Rizwan Rawani (11111030), Nitesh Bagmar(11111013)

***Abstract*— In this work, we analyze the basic collaborative filtering algorithm to predict user ratings on a set of movies. We also analyze and compare some well known modifications to the basic algorithm that make the predictions accurate. We then provide a hybrid algorithm of our own that uses content information (movie genre information) by modifying some steps in the well known collaborative filtering algorithms. We provide experimental results to show that our algorithm outperforms the other algorithms discussed.**

***Index Terms*— Recommendation Engine, Collaborative Filtering, Content Based Filtering, Mean Absolute Error**

I. INTRODUCTION

TODAY, the world is moving at a very fast pace. Almost every consumable product or service is available online. Consumers now don't want to step out of their homes when they can get everything online. But the large variety of product offerings also confuses the users in decision making. When buying products personally, users had options to ask salespersons to recommend them products based on their needs and interests and help them decide which product is right for them. In e-commerce, this task has now been taken up by recommendation systems. The goal of a recommender system is to generate meaningful recommendations to a collection of users for items or products that might interest them. Online users are overloaded by options and it is not possible for them to evaluate all the possible options themselves. Thus a good recommendation system is very helpful in the decision making and buying process and is appreciated by the users.

Recommender systems have equally proved useful to the producers. Enterprises collect large volumes of transactional data to analyze the interaction of a customer base to a set of product offerings. Recommender systems are developed based on this analysis. This analysis helps the enterprises to make future strategies to attract more users and to compete with their rivals. Thus recommendation systems have become an important part of e-commerce.

When browsing online, if we notice carefully, we will see the use of recommendation systems very often. Some popular websites widely use recommendation engines to achieve user satisfaction and to attract users. The examples include:

1. Amazon, the most popular online retailer, suggests different products like books, movies, games, electronic and household appliances etc, to users based on their previous purchases and purchases by similar users.
2. YouTube, the most popular online video streaming website, recommends videos to users based on their viewing history, video likes & dislikes and channel subscriptions.
3. Facebook, the most popular social networking website, suggests friends to users by comparing their mutual friends. It also recommends pages to users based on their page likes, likes by their friends and similar users.
4. Netflix, an on demand online movie streaming and DVD rental service, recommends movies, videos, TV shows to its users based on their likes, their previous rented videos, and likes & previously rented information of similar users.
5. Google, the most widely used search engine, provides search recommendations while user is entering the search query, based on most frequent searches, user's past searches and user's geographical location.

Netflix had announced an open competition called 'Netflix Prize' for improvement in their recommendation system in 2006. For this, they released a training data set containing one million ratings given by about half a million users to thousands of movie titles. The prize money for the competition was $ 1 Million. The competition was won by the team called "Bellkor's Pragmatic Chaos" in 2009, which managed to improve the recommendation system of Netflix by 10.06 %. This team includes two AT&T research staff, Bob Bell and Chris Volinsky; Yahoo's Israel Lab Yehuda Koren; Austrian researchers Andreas Töscher and Michael Jahrer; and Quebec-based Martin Chabbert and Martin Piotte.

Development of good recommendation systems requires knowledge of various disciplines such as data mining, machine learning, artificial intelligence, statistics, information retrieval etc.

Currently, recommender systems remain an active area of research, with a dedicated ACM conference (ACM International Conference on Recommender Systems) which is held annually.

II. BACKGROUND

There are two major entities of a recommendation system. These are users (who use the system) and items (for which recommendations are given). In context of movie recommendation system the items are 'movies'. There may be various attributes of these entities. The attributes required mainly depend on the domain of the recommendation system and the filtering algorithm used. The filtering algorithm used also depends somewhat on the domain of the system. Different algorithms are found to be suitable for different domains. These algorithms are mainly classified into two groups depending on how they generate the recommendations. These are:

1. Content Based Filtering

These algorithms focus on the content information of items and users to learn what items a user may like. The idea is to compare representation of content contained in an item and representation of content that may interest the user. In context of movie recommendation system, the content information will be the various attributes of movies such as, genre, writer, director, actors etc. Also users' attributes like age, gender, occupation etc. may be utilized in some way.

2. Collaborative Filtering

These algorithms mainly depend on users' past interactions with the system. User's feedback on the items previously purchased by the user is collected. Then depending on this feedback, users with similar interests are identified. The recommendation algorithm works on the principle that if a user likes a particular item, then another user who has similar interests is also likely to like that item. In context of movie recommendation system, the feedback is taken in the form of user ratings (say on a scale of 1-5). While recommending movies to a user, the idea is to find a set of similar users (similar in the sense that the ratings given by these users to different movies are similar to the ratings given by the user in consideration), and to recommend movies that these similar users have liked. Collaborative filtering algorithms have been found to perform better than content based filtering algorithms. Often a hybrid algorithm (a mixture of these two techniques) is used and it performs better than both of these as it incorporates the advantages of both.

The input to a recommendation system is a set of ratings on a particular set of items by a set of users. The input can be considered to be the set of 3-tuple (user_id, movie_id, rating) where 'rating' is the rating given by the user 'user_id' to the movie 'movie_id'.

There are two types of output possible from a recommendation system. A recommendation system may give as output, predicted ratings on unrated items or may give a list of recommended items to a particular user. We restrict our discussion to those systems where rating predictions are generated (This output of course can be converted into second type of output by selecting items that get higher predicted ratings by active user and recommending those items. There are other methods also given in [3]).

A. Basic collaborative filtering algorithm ([1] and [3]):

A.1. Representation:

The input is converted into a proper format. Suppose there are 'n' users and 'm' items (movies), then a rating matrix R (n x m) is created where the entry $r_{i,j}$ contains the rating given by i^{th} user to j^{th} movie as shown in Fig. 1. This matrix is usually very sparse as a user rates only a small subset of items. (A user is not likely to have seen or rated all the movies in the dataset)

		Items			
		1	2	…	m
Users	1	3		1	
	2		5		4
	:	2			3
	:			4	
	n		2		5

Fig. 1. Rating Matrix

A.2. Find similarity between users:

The similarity between two users can be calculated using Pearson Correlation. The similarity between the users u_i and u_k is calculated as:

$$sim_{i,k} = \frac{\sum_{j \in I}(r_{i,j} - \overline{r_i})(r_{k,j} - \overline{r_k})}{\sqrt{\sum_{j \in I}(r_{i,j} - \overline{r_i})^2 \sum_{j \in I}(r_{k,j} - \overline{r_k})^2}} \quad (1)$$

Here, $\overline{r_i}$ is the mean rating given by the user u_i and $r_{i,j}$ is the rating given by user u_i to j^{th} item. The summations are over the set I of items (movies) for which both the users u_i and u_k have given ratings.

A.3. Neighbourhood generation:

A similarity matrix of size n x n is made which stores the similarity between users. The cell at i^{th} row and j^{th} column stores the value $sim_{i,j}$ as calculated from (1). The diagonal elements are made zero as a user should not be treated a neighbour of itself.

Thus k maximum values in i^{th} row of similarity matrix, represent the similarity weights of the k most similar users to

the user u_i. Here k represents the neighbourhood size. Thus top k neighbours to the active user are found by selecting top k values and their indices in the row of similarity matrix which corresponds to the active user. Hence a set of k nearest neighbours to the active user can be created.

A.4. Prediction

The next step is to predict user ratings for unrated items. Suppose a user u_a has not rated the j^{th} item (movie). Then the prediction of rating on the j^{th} item by the user u_a is calculated as:

$$pr_{a,j} = \overline{r_a} + \frac{\sum_{i \in N}(r_{i,j} - \overline{r_i}) * sim_{a,i}}{\sum_{i \in N} |sim_{a,i}|} \quad (2)$$

Here $\overline{r_a}$ is the mean rating given by the user u_a and $r_{i,j}$ is the rating given by user u_i to j^{th} item. $sim_{a,i}$ is as calculated from (1). N is the neighbourhood set of the user u_a. The summations are taken only over those users in N who have actually rated the j^{th} item.

A prediction matrix P (n x m) can now be created as follows:

$$p_{i,j} = \begin{cases} r_{i,j} & : \quad if\ user\ u_i\ has\ rated\ j^{th}\ item. \\ pr_{i,j} & : \quad if\ user\ u_i\ has\ not\ rated\ j^{th}\ item. \end{cases}$$

If our aim is to generate a set of recommendations for some user u_i, then the items which have higher value of $p_{i,j}$ and were not previously rated by the user u_i can be recommended.

B. Problems in the basic collaborative filtering algorithm:

B.1. False neighbourhood:

Suppose two users have very common interests and should form a neighbourhood together, but if they have not rated any common item, then the Pearson Correlation between them can't be calculated using (1) and they would not be considered while creation of each other's neighbourhood. On the other hand if two users have very few commonly rated items, they may be considered very strong neighbours based on these items only which again may create a false neighbourhood.

B.2. First rater problem:

Suppose an item has not been rated by any user till now, then it won't be possible to predict its rating by some user using (2). Therefore, an item won't be recommended unless it has been rated by at least one user.

C. Solutions suggested for these problems in [3]:

The major cause of these problems is the sparsity of matrix. Hence various methods are suggested to preprocess the rating matrix R and fill the empty cells. These are:

C.1. Fill the empty cells in the rating matrix with mean ratings by the corresponding users. A modified rating matrix Q can be created from R as follows:

$$q_{i,j} = \begin{cases} r_{i,j} & : \quad if\ user\ u_i\ has\ rated\ j^{th}\ item. \\ \overline{r_i} & : \quad if\ user\ u_i\ has\ not\ rated\ j^{th}\ item. \end{cases}$$

Here, $\overline{r_i}$ is the mean rating given by the user u_i.

C.2. Fill the empty cells in the rating matrix with mean ratings of the corresponding items. A modified rating matrix Q can be created from R as follows:

$$q_{i,j} = \begin{cases} r_{i,j} & : \quad if\ user\ u_i\ has\ rated\ j^{th}\ item. \\ \overline{r_j} & : \quad if\ user\ u_i\ has\ not\ rated\ j^{th}\ item. \end{cases}$$

Here, $\overline{r_j}$ is the mean rating given to j^{th} item.

C.3. Composite method of filling the sparse matrix: First mean ratings by all the users are computed. Then we proceed as follows. If user u_i has not rated j^{th} item, we look at the j^{th} column in the rating matrix R. For all those users who have rated j^{th} item, we calculate a correction term. If user u_k has rated j^{th} item, then correction term is calculated as $\delta_k = r_{k,j} - \overline{r_k}$. The modified rating matrix Q is created from R as follows:

$$q_{i,j} = \begin{cases} r_{i,j} & : \quad if\ user\ u_i\ has\ rated\ j^{th}\ item. \\ \overline{r_i} + \dfrac{\sum_{k \in L} \delta_k}{|L|} & : \quad if\ user\ u_i\ has\ not\ rated\ j^{th}\ item. \end{cases}$$

Here, $\overline{r_i}$ is the mean rating given by the user u_i. L is the set of users who have rated j^{th} item. δ_k is calculated as mentioned above. If no user has rated j^{th} item, then we can fill $q_{i,j}$ with $\overline{r_i}$.

After the modified rating matrix has been created using any of the above three methods, the similarity between users can be calculated using (1), but instead of using rating matrix, we use modified rating matrix:

$$sim_{i,k} = \frac{\sum_{j \in I}(q_{i,j} - \overline{r_i})(q_{k,j} - \overline{r_k})}{\sqrt{\sum_{j \in I}(q_{i,j} - \overline{r_i})^2 \sum_{j \in I}(q_{k,j} - \overline{r_k})^2}} \quad (3)$$

Here, $\overline{r_i}$ is the mean rating given by the user u_i and $q_{i,j}$ is the rating in the complete modified rating matrix. The summations are over the set I of all the items (movies). Similarly, (2) can be modified to calculate predicted ratings from the modified rating matrix:

$$pr_{a,j} = \overline{r_a} + \frac{\sum_{i \in N}(q_{i,j} - \overline{r_i}) * sim_{a,i}}{\sum_{i \in N} |sim_{a,i}|} \quad (4)$$

Here $\bar{r_a}$ is the mean rating given by the user u_a and $q_{i,j}$ is the rating in the complete modified rating matrix. $sim_{a,i}$ is as calculated from (3). N is the neighbourhood set of the user u_a.

All the above methods are likely to solve the false neighbourhood problem and the first rater problem.

Another modification can be in the neighbourhood generation step. Instead of selecting the top k neighbours of the active user, an aggregate neighbourhood scheme can be followed. The first neighbour to the active user is the user closest to the active user. This user and the active user now form the current neighbourhood. The selection of the next neighbour at any stage depends on the current neighbourhood. This is done as follows. Suppose at any stage, we have a set H of h users in the current neighbourhood including the active user (h < k+1). The centroid of the current neighbourhood is calculated as:

$$\vec{C} = \frac{1}{h}\sum_{j \in H} u_j \tag{5}$$

Here u_j is the j^{th} row in the similarity matrix (row that corresponds to j^{th} user.) Next neighbour selected is the one which is closest to the centroid. Of course, users already present in the neighbourhood are ignored in this process.

This method of neighbourhood generation is likely to solve the false neighbourhood problem.

III. OUR APPROACH

Apart from known user ratings to movies, we also use movie genre information in our approach. Hence the input to our algorithm is a set of ratings where each element is a 3 tuple (user_id, movie_id, rating) as discussed in the previous section. Another input is the movie genre information (list of genres a movie belongs to). Using this genre information we slightly modified some of the steps in the algorithm discussed in previous section,

A. Our algorithm with justification:

A.1. Preprocessing:

1. Create the rating matrix R as discussed in previous section (Fig. 1).

2. Suppose there are 'm' movies and 'g' genres. Create the movie genre matrix G (m x g) such that

$$g_{i,j} = \begin{cases} 1 & : \ if\ i^{th}\ movie\ belongs\ to\ j^{th}\ genre \\ 0 & : if\ i^{th}\ movie\ does\ not\ belong\ to\ j^{th}\ genre \end{cases}$$

It should be noted that a movie may belong to more than one genre.

3. Calculate the mean ratings by all the users using the rating matrix R.

4. Suppose there are 'n' users and 'g' genres of movies. Create a matrix M (n x g) such that the cell $m_{i,j}$ stores the mean rating that i^{th} user gives to j^{th} genre:

$$m_{i,j} = \frac{\sum_{k \in J} r_{i,k}}{|J|} \tag{6}$$

Here J is the set of movies that belong to j^{th} genre (k^{th} movie belongs to j^{th} genre if $g_{k,j} = 1$) and have been rated by the user u_i (i^{th} user). $r_{i,k}$ is the rating given by i^{th} user to k^{th} movie. If the ith user has not rated any movies of j^{th} genre, then $m_{i,j}$ can be taken as the overall mean rating of that user ($\bar{r_i}$).

5. Create the expected rating matrix E (n x m), that stores the expected ratings of all the movies by all the users. By expected rating, we mean the rating expected from the movie genre information and users' mean rating for various genres. $e_{i,j}$ stores the average of the mean genre ratings by i^{th} user (considering only the genres to which j^{th} movie belongs.)

$$e_{i,j} = \frac{\sum_{k \in Z} m_{i,k}}{|z|} \tag{7}$$

Here, Z is the set of those genres to which j^{th} movie belongs (j^{th} movie belongs to k^{th} genre if $g_{j,k} = 1$). $m_{i,k}$ is as calculated from (6). If a movie does not belong to any genre (which is very unlikely), then $e_{i,j}$ can be taken as the overall mean rating of i^{th} user ($\bar{r_i}$).

6. Create the complete modified rating matrix Q (n x m) as follows. If user u_i has not rated j^{th} item, we look at the j^{th} column in the rating matrix R. For all those users who have rated j^{th} item, we calculate a correction term using the expected rating matrix E. If user u_k has rated j^{th} item, then correction term is calculated as $\delta_k = r_{k,j} - e_{k,j}$. The modified rating matrix Q is created from R and E as follows:

$$q_{i,j} = \begin{cases} r_{i,j} & : \quad if\ user\ u_i\ has\ rated\ j^{th}\ item. \\ e_{i,j} + \frac{\sum_{k \in L} \delta_k}{|L|} & : \ if\ user\ u_i\ has\ not\ rated\ j^{th}\ item. \end{cases}$$

Here, $e_{i,j}$ is the corresponding entry in the expected rating matrix as calculated from (7). L is the set of users who have rated j^{th} item. δ_k is calculated as mentioned above. If no user has rated j^{th} item, then we can fill $q_{i,j}$ with $e_{i,j}$.

Thus we get the complete modified rating matrix Q which can be used for further steps in the algorithm. The main aim

of obtaining the matrix Q is to get a complete initial rating matrix which is not sparse. In the previous section, we had discussed filling the empty cells in R with corresponding mean user ratings to get Q. It is obvious that the more accurate these initial guesses about the missing ratings are, the more accurate will be the final results. So instead of directly using the mean user ratings, we have calculated expected rating for each user-movie pair. A user may like the movies of some genre more than other genres depending upon his interests. Suppose, a user likes comedy movies very much but does not like science fiction movies. Then filling all the empty cells with mean rating of that user is not desirable. It is desirable that in the initial approximation, comedy movies should get a higher rating and science fiction movies should get a lower rating. So we have utilized the genre information of movies. From the already rated movies by the user, we calculate the mean genre rating of a particular user using (6). Then we utilize this information to calculate the expected rating a user is likely to give to a particular movie. This is calculated as the average of those mean genre ratings by the user to which the movie belongs. This calculation is done in (7).

Finally, we calculate a correction term to be added to the expected rating to get the corresponding entry in Q. The significance of this correction term lies in the fact that although a user may like or dislike movies of a particular genre in general, a particular movie may be independently good or bad, and depending on this fact, its initial approximation of rating should deviate from its expected rating. This correction term is calculated by analyzing the ratings given to a movie by the users who have actually rated that movie. We calculate the deviations of the actual ratings of that movie (by the users who have rated it) from the corresponding expected ratings. We take the average of all these deviations which gives us the idea that on an average, how much the rating of a particular movie deviates from its expected value. Thus we add this calculated term to the expected rating value of this movie (by a user who has not rated it) to fill the corresponding entry in the modified rating matrix Q.

A.2. Find similarity between users:

We calculate the similarity between two users using Pearson Correlation with a little modification to incorporate the genre information of the movies by using expected ratings as calculated from (7) instead of mean user ratings.

$$sim_{i,k} = \frac{\sum_{j \in I}(q_{i,j} - e_{i,j})(q_{k,j} - e_{k,j})}{\sqrt{\sum_{j \in I}(q_{i,j} - e_{i,j})^2 \sum_{j \in I}(q_{k,j} - e_{k,j})^2}} \quad (8)$$

Here, $q_{i,j}$ is the rating in the complete modified rating matrix Q and $e_{i,j}$ is the corresponding entry in the expected rating matrix E. The summations are over the set I of all the items (movies). The idea behind using $e_{i,j}$ instead of mean user rating is same as described earlier. The expected rating of an item by a user as calculated from (7), gives a better approximate than the corresponding mean user rating. The reasons behind this are same as given earlier while calculating the modified rating matrix Q.

A.3. Generate Neighbourhood

Form the similarity matrix as described in previous section and generate neighbourhood by using either the normal method (selecting top k neighbours of the active user) or the aggregate neighbourhood method (selecting the user closest to the centroid of current neighbourhood), as discussed in previous section.

A.4. Prediction:

The prediction of rating on the j^{th} item by the user u_a is calculated by slightly modifying (4) to incorporate the genre information of the movies by using expected ratings instead of mean user ratings:

$$pr_{a,j} = e_{a,j} + \frac{\sum_{i \in N}(q_{i,j} - e_{i,j}) * sim_{a,i}}{\sum_{i \in N} |sim_{a,i}|} \quad (9)$$

Here $e_{a,j}$ is the expected rating of j^{th} item by the user u_a as calculated from (7) and $q_{i,j}$ is the rating in the complete modified rating matrix. $sim_{a,i}$ is as calculated from (8). N is the neighbourhood set of the user u_a.

A prediction matrix P (n x m) can now be created as follows:

$$p_{i,j} = \begin{cases} r_{i,j} & : \quad if\ user\ u_i\ has\ rated\ j^{th}\ item. \\ pr_{i,j} & : \quad if\ user\ u_i\ has\ not\ rated\ j^{th}\ item. \end{cases}$$

The idea behind using expected rating instead of mean user rating of the corresponding user is same as described earlier that expected rating as calculated from (7) gives a better initial approximate.

In our approach, we focused on completing the initial sparse matrix as accurately as possible and therefore we came up with the idea of utilizing movie genre information to calculate expected ratings. In [2], the authors have focused on the same issue, but they use various attributes of movies and do a textual analysis on these attributes to get initial approximations for missing ratings and after that they continue with the pure collaborative filtering algorithm, while we have used only the mean ratings for various genres by the users to approximate missing ratings, and we have also modified the equations of Pearson Similarity (Correlation), and Prediction of ratings, to incorporate users' tastes while

calculating similarity and while predicting the ratings by using expected ratings in the equations.

IV. DATASET DESCRIPTION AND EXPERIMENTAL RESULTS

A. Dataset Description

The data set that we have used in this work has been donated by MovieLens (a movie recommendation website). MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota. This data set contains 100,000 ratings on 1682 movies rated by 943 users. Each user has rated at least 20 movies. The ratings are given on the scale of 1-5. The user details include user id, gender, age and occupation of the users. The movie details include movie id, movie name and various genres to which the movie belongs.

For the analysis of various algorithms, we made four different groups of disjoint training and testing sets. These training sets contained 80,000 ratings each and the corresponding testing sets contained the remaining 20,000 ratings. The performance of the algorithms was averaged over all the four groups.

We also made two more groups of disjoint training and testing sets such that the testing set in each group contained 10 ratings by each user making a total of 9430 ratings in the test set. The corresponding training sets contained the remaining 90570 ratings. The performance of the algorithms was averaged over these two groups.

B. Performance Evaluation

We used the training set to generate the rating matrix and to predict the ratings of the unrated item. Then for each rating in the test set, we measured the error in the corresponding prediction (absolute difference from the actual rating), and calculated the mean of the errors over the entire test set. This performance metric is widely used to evaluate recommendation systems and is known as 'mean absolute error'. This metric is suggested for performance evaluation in [1], [2] and [3]:

$$MAE = \frac{\sum_{i \in T} |actual\_rating_i - predicted\_rating_i|}{|T|} \quad (10)$$

Here T is the test set, $actual\_rating_i$ is the actual rating as present in the test set and $predicted\_rating_i$ is the corresponding predicted rating present in P. MAE refers to the mean absolute error.

C. Experimental Results

The experimental results have been summarized in Table I

TABLE I

PERFORMANCE EVALUATION OF DIFFERENT ALGORITHMS

Algorithm	MAE for training-testing sets of I type	MAE for training-testing sets of II type
Basic Collaborative Filtering Algorithm.	0.8442	0.8563
Sparse Matrix Filled With Mean User Ratings.	0.7580	0.7637
Sparse Matrix Filled With Mean Item Ratings.	0.7647	0.7963
Sparse Matrix Filled With The Composite Method.	0.7225	0.7499
Sparse Matrix Filled With The Composite Method. Aggregate Neighbourhood Used.	0.7230	0.7524
Our Algorithm.	0.7165	0.7467
Our Algorithm Using Aggregate Neighbourhood.	0.7174	0.7474

V. CONCLUSION

From the results observed as summarized in Table I, it can be seen that our method performs better than the methods discussed in section II. The intuitive reason is that, we generate better initial approximates for missing ratings by incorporating movie genre information as we calculate the expected ratings of all the movies by all the users, using the mean genre ratings by the user. We use these expected ratings throughout the algorithm, even while calculating similarity between users and while generating final predictions.

VI. FUTURE WORK

We plan to work on item based collaborative filtering in future (these algorithms work on the similarity between items), analyze and compare its performance with user based collaborative filtering, and then analyze a weighted combination of both.

REFERENCES

[1] Prem Melville and Vikas Sindhwani, "Recommender Systems", Encyclopedia of Machine Learning, 2010, Available: http://www.prem-melville.com/publications/recommender-systems-eml2010.pdf

[2] Prem Melville, Raymond J. Mooney and Ramadass Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations", Eighteenth National Conference on Artificial Intelligence, July 2002, Available:http://www.cs.utexas.edu/users/ml/papers/cbcf-aaai-02.pdf

[3] Emmanouil Vozalis, Konstantinos G. Margaritis, "Analysis of Recommender Systems' Algorithms", presented at HERCMA, Athens, 2003, Available:http://users.uom.gr/~mans/papiria/hercma2003.pdf