Movie Recommendation System

Anshul Jain
ajain89@asu.edu

Arunkumar Subramanian
asubra19@asu.edu

Rahul Jain
rjain29@asu.edu

Ronak Dhalawat
rdhalawa@asu.edu

Vivekanand Nolastname
vivek.anand@asu.edu

***Abstract*- The concept of recommendation system dates back to as early as 1970s. It found its first use in the internet in the early 1990s, since then its importance in the e-commerce market scaled to new heights due to the hatful of choices that a customer faces. They bring about huge revenue to organizations by accentuating out right customers for a set of products by using several information filtering techniques. Movie recommendation systems use pretty much the same techniques, they identify similarity of choices among likeminded individuals and suggest movies accordingly. This paper presents the detailed implementation of a movie recommendation system named 'MovieRec'. The system uses collaborative filtering technique to suggest movies to users based on movies previously rated by them. The paper also presents the evaluation techniques and experimental results obtained in detail when the algorithm was run on a very large data set. Generated results shows appreciable efficiency compared to existing systems.**

I. INTRODUCTION

The invent of the first automatic recommendation system dates back to the early 1970s, when a computer based librarian named Grundy [1] grouped users as "stereotypes" based on a short interview. In the early 1990s [2] Xerox came up with its manual collaborative filtering system named "Tapestry", it allowed user to query in the information domain space in corporate emails based on their actions and opinions. Soon automatic collaborative filtering came into play and "GroupLens" [3] was the first to adapt to this technique to recommend articles that are likely to be preferred by a particular user based on their previous ratings on similar documents.

The commercial deployment of recommendation systems started evolving around in the late 1990s. Large e-commerce websites like Amazon started using recommendation systems to predict products for users based on their previous ratings for similar products. Netflix, the giant that provides users with movies which they can watch online also resorted to the same. Perhaps in the year 2006, Netflix launched the Netflix prize to improve the state of online movie recommendation. It announced a prize money of one million dollar for people who could come up with an algorithm that would beat theirs by a margin of 10%. The above story also reveals the value that these companies put on their recommendation system. Movie recommendation systems have gained great popularity ever since then.

Many researches have been carried out in the field of Online Movie Recommendation Systems, each outperforming the correctness of the other. Techniques like collaborative filtering, content based filtering, hybrid techniques have become the buzz words in the field of recommendation systems. Our paper focuses on a simple collaborative filtering based technique used to design our movie recommendation system 'MovieRec'. The system was tested on a large dataset of around one million of lines of data, each representing a user, a movie and the rating for that movie by the user. The results generated leads to interesting observations that later discussed in the paper. Run time complexity, memory usage and complexity of coding has been considered to develop a simple but robust recommendation system.

The paper is divided into six sections. Section II enunciates the background research work done before developing the system. Section III gives a brief overview of the methodologies and terminologies. The section moreover discusses the motivation behind choosing Collaborative Filtering over other methodologies (such as Content based). Section IV deals thoroughly with the implementation details and the experimentation carried out on the system. Section V is devoted to the outcome of the experimentation carried out. Section VI discusses on the future work to take this project further.

II. BACKGROUND

IBM developed SmartPad, a Personalized Digital Assistant based system. The system used content based filtering along with collaborative filtering technique to recommend products to users. This system was introduced by Lawrence et al 2001[4] where suggestion were based on users shopping demeanor.

Jung, Harris, Webster and Herlocker (2004) [5] designed a system where ratings from previous users were used to make recommendations to users with similar taste. The system introduced logger and informative search queries where ratings were collected from users to make recommendation to users with similar needs.

The website MovieLens also works on similar principle were users are used to rate movies that they had previously seen. The ratings are captured and stored in a large rank matrix and machine learning algorithms are used to predict movies the user would probably end up preferring. The MovieLens website also provides data for researchers to work on better techniques of implementing recommendation systems.

Several techniques have been used and researched to implement online movie recommendation systems till date. Chen and Aickelein (2004) [6] used Artificial Immune System to implement online movie recommendation. Chen et al described candidates in the database as candidate antibodies, and the user who viewed the movies as an antigen. Moreover the paper introduces the use of two interesting correlation techniques: Weighted Kappa [7] and Kendall tau method [8] to find the correlation between antigens and antibodies.

Even though several techniques were used to build recommendation systems, the Collaborative filtering is still considered to be one of the most popular and widely used of all. Even in the Netflix Prize competition (2006), which attracted over 20,000 participation across 167 countries, the top papers selected were all based on Collaborative filtering. Jinlong Wu and Tiejun Li suggested a “Modified Fuzzy C- Means Algorithm for Collaborative Filtering” [9], they combined the ideas of Matrix Factorization and Fuzzy C- Means to propose a new clustering model. The algorithm when applied to the Netflix prize data acquired accuracy comparable with that of Matrix Factorization. Gavin Potter in his paper “Putting the Collaborator back into collaborative filtering” [10] considered psychological decision making process to achieve significant improvements in results. This approach brought in important implications for both design and analysis of the data from collaborative filtering systems. Oscar Celma and Pedro Cano [11] presented a music recommendation system based on total playcounts and the Long Tail model. They defined a term “popularity” which is based on the total playcounts value. Their work reveal that collaborative filtering is prone to popularity biased.

Our work hovers around the use of Collaborative filtering with K Nearest Neighbor algorithm. Cosine similarity technique is used (as the matrix generated was a sparse matrix) to calculate similarity among users and K Nearest similar users are considered to predict the rank of a particular user. Our motivation behind choosing Collaborative filtering technique over other available techniques (such as Content based filtering) has been discussed in great details in the subsequent sections.

III. METHODOLOGY

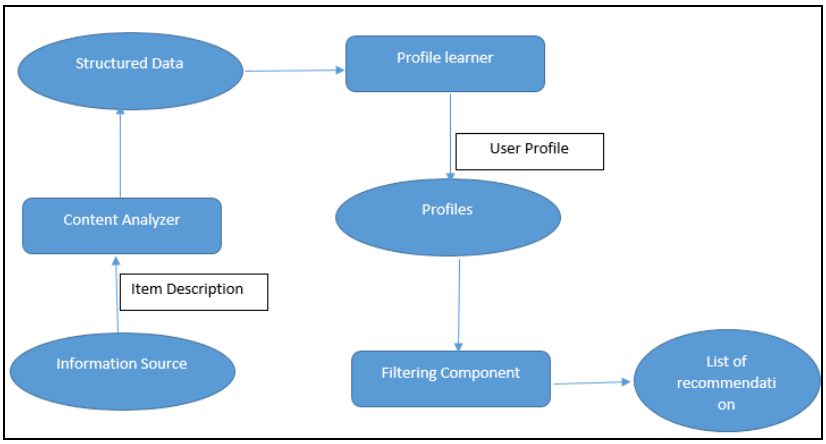
A. Content Based Technique

Content based filtering also known as cognitive filtering has its roots in information filtering system. Systems using a content-based recommendation technique analyze a set of descriptions of movies previously rated by a user, and build a model or profile of user interests based on the features of the objects rated by that user. The user profile represents the interests of the user, and built in such a way to recommend the new movies. The basic process of the content based recommendation is matching up the features of the user profile with the features of the content object.

Overview of content based filtering:

Content based filtering can be divided in three steps, techniques needed to represent the user ratings, techniques to learn user profile and finally the techniques to compare the user profile and content object. The flow chart represents the recommendation process for the content based filtering.

1. Content analyzer: This component is concerned with preprocessing of data. Usually the dataset for the movie recommendation system is not represented in structured way. The main aim of this component is to represent the data coming from various information sources in a form i.e. suitable for the next processing steps. This representation is the input to the Profile Learner and Filtering Component.
2. Profile learner: This module collects data representative of the user preferences and tries to generalize this data, in order to construct the user profile. This module is responsible for building user profile through machine learning techniques which are able to infer a model of user interests starting from items liked or disliked in the past.
3. Filtering Component: This module is responsible for suggesting the movie recommendation by matching the profile representation against that of items to be recommended.



Limitations of Content Based Technique

Though content based has some advantageous like it is user independence, transparent, capable of recommending items not yet rated by any user but it fails to overcome the following shortcomings.

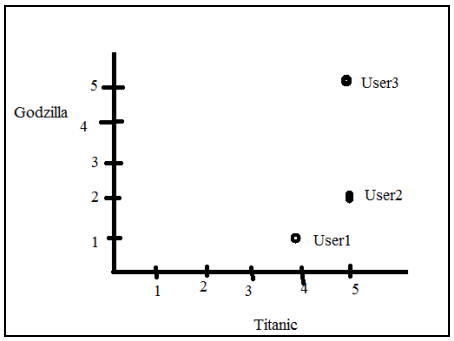
1. Limited Content analysis: Content based techniques are limited in number and types of features that are associated with the objects they recommend. Domain knowledge is often required in content based filtering e.g., for movie recommendations the system needs to know the actors and directors. If the analyzed content does not have enough information to classify movies the user likes from movies the user does not like then no content-based recommendation system can provide suitable suggestions.
2. Over Speciation: Content based recommendation does not contain method for the outliers. The system suggests items whose scores are high when matched against the user profile, hence the user is going to be recommended items similar to those already rated. This drawback is also called serendipity problem to highlight the tendency of the content-based systems to produce recommendations with a limited degree of novelty.
3. New User: A large amount of user rating has to be collected before the system can really understand user profile and provide accurate recommendation. Therefore, when few ratings are available, as for a new user, the system will not be able to provide reliable recommendation.

B. Collaborative Filtering

The recommendation method we are using in our project is called collaborative filtering because it makes recommendations based on other similar people's preferences. In other words people collaborate to come up with movie recommendations. The general architecture of Collaborative Filtering in movie recommendation can be described as:
"Suppose the task is to recommend a movie to one person. We search among the users of one particular hosting website that maintains movie ratings by users to find other users that are similar based on the movie they like. Once we find those similar users, we can recommend movies based on their good rating"

The first step is to find similar user has been explained here in two dimensional spaces as shown in diagram. Suppose users rate movies on a 5 star system- zero stars means that the movie is

terrible and 5 star means that the movie is greatly liked by the user. Since we are explaining the 2D case, we are going to take 2 movies ratings into our consideration. Let's assume we consider 2 movies – *Titanic* and *Godzilla.*



The above graph shows the ratings given by 3 users User1, User2 and User3 for 2 different movies Titanic and Godzilla. Following is the table describing the users with their corresponding ratings.

User	Godzilla	Titanic
User3	5	5
User2	2	5
User1	1	4

Now, I would like to recommend a movie to a mysterious Mr. X who rated Godzilla 4 stars and Titanic 2 stars. The first task is to find those persons who are most similar to Mr. X. We do this by computing distance. The following section describes different types of distance metric:

1. Manhattan Distance:

   The easiest distance measure to compute is called Manhattan Distance. In the 2D case, each person is represented by an (x, y) point. We will add a subscript to the x and y to refer to different people. So (x_1, y_1) might be User3 and (x_2, y_2) might be the Mr. X. Manhattan Distance between two users is then calculated by

$$| x1 - x2 | + | y1 - y2 |$$

   So, after computing the Manhattan distance between Mr. X and other users, the following result is obtained

Users	**Distance from Mr. X**
User3	4
User2	5
User1	5

The above table shows that User3 is closest to Mr. X. Now, we can look User3's history and find what all movies User3 has rated great (like 4 or 5). Based on User3's rating movies to Mr. X.

2. Euclidian Distance:
   Similarly, we can also look for Euclidean Distance to compute the similarity of different users with MR X.

$$\text{Euclidean Distance} - \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

3. N-Dimensional Approach
   Let's now look at higher dimensional space. Suppose we now look for more than 2 movies. Let's say users can rate movies on a star system of 1-10 stars.

		Users			
		1	2	3	4
Movies	a	2	0	8	9
	b	1	6	5	0
	c	0	2	8	7
	d	6	8	0	2
	e	0	7	8	5

The 0 in the table indicates that a user didn't rate that particular movie. For now we are going to compute the distance based on the number of movies they both rated. So, for example, when computing the distance between User 1 and User 2, we will use the ratings for movies b and d only.
Manhattan Distance and Euclidean distance work best when there are no missing values in the dataset. Dealing with missing values is an active area of scholarly research. Murkowski Distance is a generalization of Euclidean Distance –

$$dist = \left(\sum_{k=1}^{n} |p_k - q_k|^r\right)^{\frac{1}{r}}$$

4. Pearson Coefficient
   User behavior also comes handy when one user always rates a movie with extreme ratings. 2 users may rate differently to a movies they like. One user may always rate very high to movies they like. There may be other users who always rate low to movies they don't like.
   One way to fix this problem is to use Pearson Correlation Coefficient.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

The similarity between two users can be measured by treating each user as a vector of movie ratings and computing the cosine of the angle formed by the ratings vectors. This formulation can be adopted in collaborative filtering, which uses users or items.

Formally, if R is the m x n user-item matrix, then the similarity between two users, i and j, is defined as the cosine of the n dimensional vectors corresponding to the i^{th} and j^{th} row of matrix R.

5. Cosine Similarity:
   Vector cosine similarity between movies i and j is given by

$$w_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \bullet \vec{j}}{\|\vec{i}\| * \|\vec{j}\|},$$

So, which similarity measure should we use?

- If the data is subject to grade-inflation (different users may be using different scales), then use pearson.
- If the data is dense (almost all attributes have nonzero values) and the magnitude of the attribute values is important, use distance measures such as Euclidean or Manhattan.
- If the data is sparse, consider using Cosine Similarity.
  In our movie dataset, we have sparse data. It makes sense as many users might not have rated all the movies which we are considering at a particular point of time.
  Therefore, we have used Cosine similarity in our project to find out the similar users in order to recommend movies to some arbitrary users. The problem we were facing in implementing collaborative filtering is that we were relying on a single "most similar" user. We need to extend this to more than just one similar user. For this, we have considered the k-nearest neighbor approach.

C. K-Nearest Neighbor

In the k-nearest neighbor approach for collaborative filtering, we use k most similar user to determine movie recommendations. The best value for k is application specific. We have used weighted similarity for predicting ratings.

Four things that make a memory based learner:

1. A distance metric: Euclidian (and many more)
2. How many neighbors to look at?: K
3. A weighting function (optional): Unused

4. How to fit with the local points?: Just predict the average output among the k nearest neighbors.

D. Challenges in Collaborative Filtering

There are many challenges for collaborative filtering tasks. CF algorithms are required to have the ability to deal with highly sparse data, to scale with the increasing numbers of users and items, to make satisfactory recommendations in a short time period, and to deal with other problems like synonymy (the tendency of the same or similar items to have different names), shilling attacks, data noise, and privacy protection problems.

Collaborative filtering systems use the user rating data to calculate the similarity or weight between users or items and make predictions or recommendations according to those calculated similarity values. This approach is known as so-called memory-based CF methods.
There are several limitations for the memory-based CF techniques, such as the fact that the similarity values are based on common items and therefore are unreliable when data are sparse and the common items are therefore few. To achieve better prediction performance and overcome shortcomings of memory-based CF algorithms, model-based CF approaches have been investigated. Model-based CF techniques use the pure rating data to estimate or learn a model to make predictions. The model can be a data mining or machine learning algorithm. Well-known model-based CF techniques include Bayesian belief nets (BNs) CF-models, clustering CF models and latent semantic CF models .

To evaluate CF algorithms, we need to use metrics according to the types of CF application. Instead of classification error, the most widely used evaluation metric for prediction performance of CF is Mean Absolute Error (MAE). Precision and recall are widely used metrics for ranked lists of returned items in information retrieval research. ROC sensitivity is often used as a decision support accuracy metric.

E. Characteristics of Collaborative Filtering

For CF-systems, producing high-quality predictions or recommendations depends on how well they address the challenges, which are characteristics of CF tasks as well.

<u>Data Sparsity:</u>
In practice, many commercial recommender systems are used to evaluate very large product sets. The user-item matrix used for collaborative filtering will thus be extremely sparse and the performances of the predictions or recommendations of the CF systems are challenged.

The data sparsity challenge appears in several situations, specifically, the cold start problem occurs when a new user or item has just entered the system, it is difficult to find similar ones because there is not enough information. New items cannot be recommended until some users rate it, and new users are unlikely given good recommendations because of the lack of their rating or purchase history.

Technique used to resolve Data Sparsity issue.

- Dimensionality reduction techniques, such as Singular Value Decomposition, can be used to remove unrepresentative or insignificant users or items to reduce the dimensionalities of the user-item matrix directly.

Scalability:

Traditional CF algorithms will suffer serious scalability problems, when numbers of existing users and items grow tremendously, with computational resources going beyond practical or acceptable levels. Example, with tens of millions of customers and millions of distinct catalog items , a CF algorithm is already too large and complex. Also many existing systems have to respond to high user requirements and new user requirements irrespective of their purchase rate and user history which demands for highly scalable systems.

Technique used to resolve Scalability issue.

- Dimensionality reduction techniques such as SVD can deal with the scalability problem and quickly produce good quality recommendations, but they have to undergo expensive matrix factorization steps.
- We have used Memory-based CF algorithms, such as the item-based Pearson correlation CF algorithm which can achieve satisfactory scalability. The similarity we calculated was based only on item co-rated by the users.

Synonymy:

Synonymy refers to the tendency of a number of the same or very similar items to have different names or entries. Most recommender systems are unable to discover this latent association and thus treat these products differently. For example, the seemingly different items "children movie" and "children film" are actual the same item, but memory-based CF systems would find no match between them to compute similarity. Indeed, the degree of variability in descriptive term usage is greater than commonly suspected. The prevalence of synonyms decreases the recommendation performance of CF systems.

Technique used to resolve Synonymy issue.

The SVD techniques, particularly the Latent Semantic Indexing (LSI) method, are capable of dealing with the synonymy problems. SVD takes a large matrix of term-document association data and construct a semantic space where terms and documents that are closely associated are placed closely to each other. SVD finds and relates major associative pattern in the data and ignore the less important ones.

Gray Sheep:

Gray sheep refers to the users whose opinions do not consistently agree or disagree with any group of people and thus do not benefit from collaborative filtering. Black sheep are the opposite group whose idiosyncratic tastes make recommendations nearly impossible. Although this is a failure of the recommender system, non-electronic recommenders also have great problems in these cases, so black sheep is an acceptable failure.

Technique used to resolve Gray Sheep issue.
An hybrid approach combining content-based and CF recommendations by basing a prediction on a weighted average of the content-based prediction and the CF prediction. In this approach, the weights of the content-based and CF predictions are determined on a per-user basis, allowing the system to determine the optimal mix of content-based and CF recommendation for each user, helping to solve the gray sheep problem.

Shilling Attacks

In cases where anyone can provide recommendations, people may give tons of positive recommendations for their own materials and negative recommendations for their competitors.
Recently, the shilling attacks models for collaborative filtering system have been identified and their effectiveness has been studied. Item-based CF algorithm is much less affected by the Shilling attacks than the user-based CF algorithm, and they suggest that new ways must be used to evaluate and detect shilling attacks on recommender systems.
Technique used to resolve Shilling attacks.
By removing global effects in the data normalization stage of the neighbor-based CF, and working with residual of global effects to select neighbors.

F. Pros of Collaborative Filtering

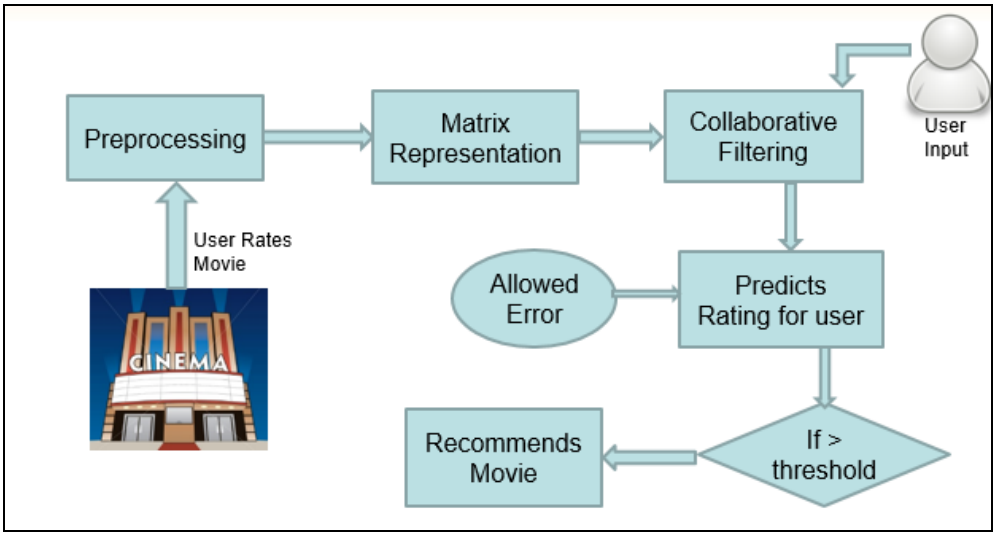
- Collaborative filtering takes into account object's quality or defect into account when suggesting objects particularly in explicit customer rankings.
- It prevents deficient suggestions and recommendations by taking the actual precedence of customers into account.
  This filtering approach is very beneficial particularly in the domains where the analysis of content is very expensive or difficult.
  e.g.: Recommending a movie without any knowledge of content of movies or any kind of domain knowledge.
- The Memory based collaborative filtering is
  1 Easy to implement
  2 New data can be added easily and incrementally
  3 Need not consider the content of the items being recommended, Scales well with co-rated items.
- The Model based collaborative filtering,
  1 Better addresses the sparsity, scalability and other problems
  2 Improves prediction performance
  3 Give an intuitive rationale for recommendations.
- The Hybrid based collaborative filtering
  1 Overcome limitations of CF and content-based or other recommenders
  2 Improves prediction performance
  3 Overcome CF problems such as sparsity & gray shape.

G. Cons of Collaborative Filtering

- The most important problem is the phase of startup in recommendation system, since there are many objects int the system and the items are provided in the system while there are few customers and few or no rankings. This problem is as named “cold start” and means that recommendation system cannot produce any suggestion or recommendations.
- Second problem named “gray sheep” problem, which defines the hardship for recommendation system for the people who do not belong to the part of an obvious group.
- Scalability is also a challenge of CF, when number of customers increases. High scalability problem generally arises for online demands.
- Another challenge that Collaborative Filtering (CF) is faced is synonymy. This problem related to inclination of numerous of very similar objects to have distinctive names.
- The memory based collaborative filtering has issues involving following:
  1 They are dependent on human ratings,
  2 Performance decreases when data are sparse.
  3 Cannot recommend for new users and items.
  4 Have limited scalability for large datasets.
- The memory based collaborative filtering,
  1 Have expensive model-building.
  2 Have trade-off between prediction performance and scalability.
  3 Lose useful information for dimensionality reduction techniques.
- The memory based collaborative filtering,
  1 Have increased complexity and expense for implementation
  2 Need external information that usually is not available.

IV. Implementation

A. Design



The dataset downloaded from "Movie tweeting dataset" is first pre-processed to form the structured dataset (described in next section). Then this structured data is fed as the input to the collaborative filtering technique. Now, to predict the recommendation for the new user. The new user will rate some movies then we will find the similar users based on these rated movies. Then the collaborative technique will predict the ratings for the new user and allowable errors are (+-0, +-1, +-2, +-3, +-4). If the resultant values are greater than the threshold then those movies are recommended to the user.

B. Data-set

The project was implemented in Java and the results were formulated using MS Excel. We have used the dataset from Movie Tweeting Dataset obtained from the research conducted by Simon Dooms. The dataset consisted of the following files:

1. movies.dat: This is a double colon (::) separated file that contains <Movie ID, Movie Name, and Genre>. Following is a screenshot that describes a few entries from this file.
2.

```
1  0000008::Edison Kinetoscopic Record of a Sneeze (1894)::Documentary&&Short
2  0000010::La sortie des usines Lumière (1895)::Documentary&&Short
3  0000091::Le manoir du diable (1896)::Short&&Horror
4  0000417::Le voyage dans la lune (1902)::Short&&Adventure&&Fantasy
5  0000439::The Great Train Robbery (1903)::Short&&Western
```

We have not used the complete dataset because of the memory constraints and so the above file did not have all the movies and Movie ID's. It is clear from the above screenshot that many movies ID's were missing from this file and so we assigned a sequential value for all the movies in our dataset. Following is a screenshot of the data we have used in our implementation. First column is the Movie Id and second column represent the sequential number we have given to the movies.

8	0
10	1
91	2
417	3
439	4
628	5

We have used only the Movie ID's in our implementation. Final results were formulated using the movie names and genre.

3. users.dat: Contains the mapping of the users id's on their true Twitter id in the following format: user id::twitter id. For example:

```
1  1::995885060
2  2::40501255
3  3::138805259
4  4::2452094989
5  5::391774225
```

The user id is used in the ratings.dat file.

4. ratings.dat: This is a double colon (::) separated file that contains <user id, movie id, rating, and time stamp>. Following is a screenshot that describes a few entries from this file.

USER	MOVIE	RATING	Timestamp
1	68646	10	1381620027
1	113277	10	1379466669
2	422720	8	1412178746
2	454876	8	1394818630
2	790636	7	1389963947

The ratings contained in the tweets are scaled from 0 to 10, as is the norm on the IMDB platform. We have ignored the time stamp in our case as it does not have any implementation impact.

Because of the memory constraint, we will be using 6140 movies and 36146 users.

C. Implementation

All the data were converted in CSV format for implementation. We have used the hash map for storing the movie id and their sequential id. Using the dataset, we have created the rating(coll) matrix which contains ratings by a particular user for a particular movie. The rows represent the movies and columns represent the users. Each entry $coll_{i,j}$ represent rating by user j for movie i. This matrix will be used in calculating the similarity between users using collaborative filtering. Following is a screenshot of this matrix.

		Users			
		1	2	3	4
Movies	a	2	0	8	9
	b	1	6	5	0
	c	0	2	8	7
	d	6	8	0	2
	e	0	7	8	5

We have used 70% of the ratings for training purpose and the remaining 30% for testing. For each user, number of movies for training is different. This is done so that we get exactly the 70% of the movies rated by that user. For example, if our dataset contains 20 movies and user1 and user2 have rated only 10 movies from this set. User1 has provided the rating for the movies {1, 10, 11, 12, 13, 14, 15, 16, 17, 18} and User2 has provided the rating for {1,2,3,4,5,6,7,8,9,10}. In our implementation, the first 15 movies will be used for training and the remaining 5 for testing for User1. For user2, first 7 movies will be used for training and the remaining 13 for testing. This ensures that we have seen exactly 70% movies rated by each user. In this way, some of the

users were ignored because they have rated very few movies and so our final result does not contain recommendation for all users.

For the training data obtained, we have then calculated the similarity between the users using the cosine similarity metric. This metric was used as the data was sparse and by our survey, cosine similarity metric works best in case of sparse data. We have used the following formula for calculating the cosine similarity between two users.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Below is the sample of the result obtained after calculating the similarity between two users. First column denotes the first user and the second column denotes the second user. The third column represents the similarity between the first and the second user, 1 represents the most similar user and -1 denotes no similarity between users.

20	407	0.752577
20	7352	0.1849
20	9209	0.473029
20	9639	1
20	14924	1
30	220	0.232628
30	340	0.108629
30	405	0.211534
30	467	0.304479
33	854	0.556022
33	1252	0.472866
33	1364	0.574367
33	1809	1

Using the above cosine symmetry matrix, we created the list of the similar user for a particular user sorted in decreasing order of their similarity. We have used Hash Map for this purpose in the form <user id, List of similar user in decreasing sorted order>. Following is a screenshot of the hash map we created using the cosine symmetry matrix. The first column represents the user id and rest of the columns represent the similar users, column 2 representing the most similar user.

User	Similar User				
20	9640	14925	408	34039	28600
30	22376	12191	12042	613	2224
33	1810	3799	4445	7205	10314
36	5655	34449	16806	11414	16422
40	1057	2026	2113	3490	4561
46	2628	4741	4872	8005	9293
53	7242	10395	14391	19755	20980
76	28657	2109	27039	21708	25982

This concludes the implementation of collaborative filtering. In the next step, we have created a data representing the predicted rating based on user similarity and the actual rating. This will be used to formulate the results of our experiment. Following is the screenshot representing the data for formulating the results. First column represent the user id, second column represent the movie id, third column represent the actual rating and forth column represent the predicted rating according to similar users. Finally the fifth column represents the modulus of the difference of the actual and the predicted rating. This column will be used in our results.

User ID	Movie ID	Actual Rating	Predicted Rating	Difference
30	108399	7	8	1
30	102803	4	7	3
30	96256	7	9	2
30	112818	8	9	1
30	98253	8	8	0
33	87544	7	9	2
36	68646	8	10	2
36	105695	8	9	1
36	57590	5	5	0
40	109830	8	8	0
40	108399	6	5	1
46	78748	7	10	3
46	94226	6	8	2
46	78788	9	10	1

D. Execution and Code

Following are the java files we have used in our implementation and will be shared along with this report:

1. CollaborativeItem.java: This file is used to create the user vs movie rating matrix. This is the most important part of collaborative filtering implementation. It takes MovieIDs.txt and DataSet.txt as input files and creates CollaborativeDataset.csv file as the output result. Cosine similarity will be calculated using this matrix.

2. CosineSimilarityCalculation.java: This file is used for calculating the cosine similarity between users. It takes MovieIDs.txt and DataSet.txt as input files and creates CosineSymmetryMatrix.csv file as the output result.
3. CollaborativeFiltering.java: This file is used for creating the data that is used for generating the results. It takes MovieIDs.txt and DataSet.txt as input files and creates TestOutput.csv file as the output result. The output file contains {USER 1,USER 2,MOVIE,RATING 1,RATING 2,DIFFERENCE}
4. Recommendor.java: Finally using the above file, Recommendor.java will be recommending movies based on user similarity. This file can be used as a recommender system.

E. Problem Faced

The dataset was very large and so scaling the data was a major problem. We tried to run our code on the complete dataset but it ran out of heap memory. Finally we restricted our data and assigned 2GB RAM to java for running the implementation and we were successful in obtaining the results.
Also we created various files containing the matrix we used at the intermediate step and files thus obtained were having size greater than 5GB. Thus, reading and understanding those files was not difficult as the Excel itself was not able to display the complete file.

V. Results

This section will describe the results obtained from our experiment. We have worked only on collaborative filtering and the results obtained were quite accurate. Because of the accuracy of our results, we have used our metric to finally predict the movies a user may like based on his similarity with other users.

A. Result 1

The following table represent the count of the difference between actual and predicted ratings.

Difference of actual and predicted rating	Count of DIFFERENCE
0	2147
1	3279
2	1883
3	839
4	374
5	174
6	84
7	30
8	17
9	13
(blank)	
Grand Total	**8840**

Following table represent the summary of the above table.

	Error = +-0	Error = +-1	Error = +-2	Error = +-3	Error = +-4
Matched	2147	5426	7309	8148	8522
Un-Matched	6693	3414	1531	692	318
Total	8840	8840	8840	8840	8840
Error%	75.71266968	38.6199095	17.31900452	7.828054299	3.5972851
Success Rate	24.28733032	61.3800905	82.68099548	92.1719457	96.402715

From the above result we can say that our implementation of collaborative was quite accurate. We see that 25% of the ratings were accurately predicted. The main aim of this project is not accurately predicting the rating for a user. Our project aims at recommending movies and so we focus more on +-2 difference in the actual and predicted rating. We are using +-2 because a user who likes a movie and gave it rating 10 is also assumed to like a movie for which he has given rating 8. We saw that for error +-2, the accuracy was close to 83% which is a good success rate for predicting the movies for a user. For more difference in the actual and predicted rating, the success rate improves which is quiet obvious. For error +-3, the accuracy was 92% but +-3 may not be a good movie recommender.

B. Result 2

In this section, we will be providing the recommended movies using the above implementation of collaborative filtering. Following are the tables describing the recommended movies with variation in the number of similar users and movie rating.

Recommendation based on 1 similar user who gave rating more than 5 to a movie

USER	RECOMMENDED MOVIES		
20	108002		
30	71615	95809	103247
33	59578	105323	
36	34248	38109	38787

Recommendation based on 2 similar user who gave rating more than 5 to a movie

USER	RECOMMENDED MOVIES		
20	108002	108002	
30	71615	95809	103247
33	59578	105323	59578
36	34248	38109	38787

Recommendation based on 2 similar user who gave rating more than 6 to a movie

USER	RECOMMENDED MOVIES		
20	108002	108002	
30	71615	95809	103247
33	59578	105323	59578
36	34248	38109	38787

Recommendation based on 2 similar user who gave rating more than 7 to a movie

USER	RECOMMENDED MOVIES		
20	108002		
30	71615	95809	103247
33	105323	59578	
36	34248	38787	51454

Recommendation based on 2 similar user who gave rating more than 8 to a movie

USER	RECOMMENDED MOVIES		
20	108002		
30	71615	103247	79501
33			
36	34248	38787	51454

Recommendation based on 2 similar user who gave rating more than 9 to a movie

USER	RECOMMENDED MOVIES		
20	108002		
30	71615	103247	79501
33			
36	53125	54215	56592

Recommendation based on 3 similar user who gave rating more than 7 to a movie

USER	RECOMMENDED MOVIES		
20	108002	108002	109830
30	71615	95809	103247
33	105323	59578	
36	34248	38787	51454

Recommendation based on 3 similar user who gave rating more than 8 to a movie

USER	RECOMMENDED MOVIES		
20	108002	109830	
30	71615	103247	79501
33			
36	34248	38787	51454

Recommendation based on 3 similar user who gave rating more than 9 to a movie

USER	RECOMMENDED MOVIES		
20	108002		
30	71615	103247	79501
33			
36	53125	54215	56592

Recommendation based on 4 similar user who gave rating more than 8 to a movie

USER	RECOMMENDED MOVIES		
20	108002	109830	99685
30	71615	103247	79501
33	59578	78788	
36	34248	38787	51454

Recommendation based on 4 similar user who gave rating more than 9 to a movie

USER	RECOMMENDED MOVIES		
20	108002		
30	71615	103247	79501
33			
36	53125	54215	56592

From the tables above, we can see that varying the number of similar user may not have much impact on the movies recommended. Varying the rating may affect the recommended movies for a user. Using 2 or 3 similar users who gave rating more than 7 is a good option as movies with rating more than 7 can be assumed to be good movies.

C. Result 3

In the following results, we will be showing the movies we have recommended to a user based on our implementation of collaborative filtering. In our implementation, we will be recommending three movies to a user. Following is a snapshot of the movies we have recommended the user based on symmetry matrix.

User	Recommended Movies		
177	64115	73486	92965
191	68646	113277	68646
198	82198	87078	89893
221	68646	113277	68646
280	89218	89218	93437
290	109830	99685	112641

The above table contains the user id and the recommended movie id's. The above result is not very informative and so we will be using original movies.dat file and MS excel to make the results more informative. Following are the final recommendations obtained after performing all the experiments.

USER	Movies Liked By User		Recommended Movies	
	MOVIE	Genere		
191	The Godfather (1972)	Crime&&Drama	Heat (1995) Action&&Crime&&Drama&&Thriller	Underworld (1927) Crime&&Drama&&Romance
198	How to Train Your Dragon 2 (2014)	Animation&&Action&&Adventure	Conan the Barbarian (1982) Action&&Fantasy&&Adventure	Red Sonja (1985) Action&&Adventure&&Fantasy
221	The Godfather	Part II (1974)	The Godfather (1972) Crime&&Drama	Heat (1995) Action&&Crime&&Drama&&Thriller
280	Forrest Gump (1994)	Drama&&Romance	The Goonies (1985) Adventure&&Comedy&&Family	Wings (1927) Drama&&Romance
	Se7en (1995)	Crime&&Mystery&&Thriller	The Goonies (1985) Adventure&&Comedy&&Family	Wings (1927) Drama&&Romance
	Saving Private Ryan (1998)	Action&&Drama&&War	The Goonies (1985) Adventure&&Comedy&&Family	Wings (1927) Drama&&Romance

From the above results, we can see that our recommender system has performed well as the genre of the movies liked by user and the recommended movies share same genre. Moreover the accuracy for the collaborative filtering was good to make a good recommendation. The experiment was performed on a very large dataset and we obtained an accuracy of 83%. For a much larger dataset, accuracy is assumed to increase and thus making a good recommender system.

VI. FUTURE WORK

We have implemented the technique of collaborative filtering and achieved an appreciable accuracy compared to some of the existing systems. But there are still few bottlenecks that are associated with Collaborative Filtering, which can be taken care of and performance can be further boosted.

There are many techniques that already exist and some techniques which are still under research.

In order to overcome the problems that are associated with the collaborative technique implemented by us and improve the recommendation for users we have to delve into the different techniques and analyze each technique with respect to dataset used by us. We can also analyze combination of different techniques which can prove beneficial in many cases.

Thus one of the existing techniques that can be investigated further involves an hybrid approach of combining content-based and collaborative filtering approach. One approach that falls under this category is "Content-Boosted Collaborative Filtering" which performs better than a pure content-based predictor and a pure collaborative filter. This technique uses components from both the methods and thus provides advantages of both. Here we use a content-based predictor to enhance existing user data, and then provide personalized suggestions through collaborative filtering.

One of the techniques under research that can be used to improve performance involves Sentimental analysis technique. Here recommendation to the users can be made based on analysis of users review and comments for the movies.

REFERENCES

[1] E. Rich, User modeling via stereotypes,Cognitive Science, vol. 3, no. 4,pp. 329?354, October 1979.

[2] Goldberg, David, et al. "Using collaborative filtering to weave an information tapestry." Communications of the ACM 35.12 (1992): 61-70.

[3] Konstan, Joseph A., et al. "GroupLens: applying collaborative filtering to Usenet news." Communications of the ACM 40.3 (1997): 77-87.

[4] Lawrence R.D., Almasi G.S., Kotlyar V., Viveros M.S., Duri S., Personalization of supermarket product recommendations (2001)

[5] Jung S., Harris K., Webster J., Herlocker, J.L. SERF -Integrating Human Recommendations with search (2004)

[6] Chen, Qi, and Uwe Aickelin. "Movie Recommendation Systems using an artificial immune system." arXiv preprint arXiv:0801.4287 (2008).

[7] Cohen, Jacob. "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit." Psychological bulletin 70.4 (1968): 213.

[8] Lindskog, Filip, Alexander Mcneil, and Uwe Schmock. Kendall?s tau for elliptical distributions. Physica-Verlag HD, 2003.

[9] Wu, Jinlong, and Tiejun Li. "A modified fuzzy C-means algorithm for collaborative filtering." Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition. ACM, 2008.

[10] Putting the Collaborator back into collaborative filtering

[11] Celma, scar, and Pedro Cano. "From hits to niches: or how popular artists can bias music recommendation and discovery." Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition. ACM, 2008.

[12] Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In Recommender systems handbook (pp. 73-105). Springer US.

[13] Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In The adaptive web (pp. 325-341). Springer Berlin Heidelberg.

[14] Van Meteren, R., & Van Someren, M. (2000, May). Using content-based filtering for recommendation. In Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop (pp. 47-56)

[15] Ekstrand, M. D., Riedl, J. T., & Konstan, J. A. (2011). Collaborative filtering recommender systems. Foundations and Trends in Human-Computer Interaction, 4(2), 81-173.

[16] Ron Zacharski. A Programmer's Guide to Data Mining, Chapter 2 – 2012

[17] Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000, December). Explaining collaborative filtering recommendations. In Proceedings of the 2000 ACM conference on Computer supported cooperative work (pp. 241-250). ACM.

[18] Park, S. T., & Pennock, D. M. (2007, August). Applying collaborative filtering techniques to movie search for better ranking and browsing. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 550-559). ACM.

[19] Umyarov, A., & Tuzhilin, A. (2008, December). Improving collaborative filtering recommendations using external data. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on (pp. 618-627). IEEE.

[20] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM.
[21] Hu, Y., Koren, Y., & Volinsky, C. (2008, December). Collaborative filtering for implicit feedback datasets. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on (pp. 263-272). IEEE.
[22] Rafsanjani, A. H., Salim, N., Aghdam, A. R., & Fard, K. B. (2013). Recommendation Systems: a Review. International Journal of Computational Engineering Research, 3(5), 47-52.
[23] Huang, Z., Chen, H., & Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Transactions on Information Systems (TOIS), 22(1), 116-142.
[24] Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009, 4.
[25] Melville, P., Mooney, R. J., & Nagarajan, R. (2002, July). Content-boosted collaborative filtering for improved recommendations. In AAAI/IAAI (pp. 187-192).