Movie Recommendation System

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

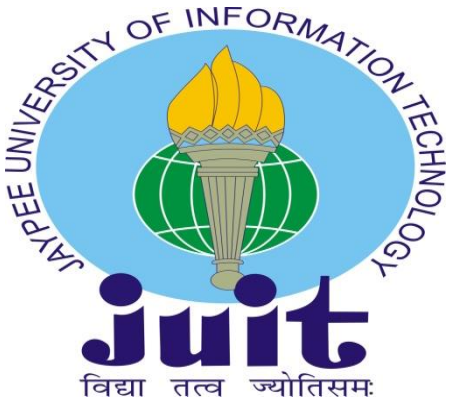
By

Rashi Gupta (161351)

Under the supervision of

(Dr. Vivek Sehgal)

to



Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **Movie Recommendation System Using Machine Learning"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from May 2020 to June 2020 under the supervision of **Dr. Vivek Sehgal** (Associate Professor, Computer Science).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Rashi Gupta, 161351

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Vivek Sehgal

(Associate Professor)

(Department of Computer Science & Engineering and Information Technology)

Dated:

ACKNOWLEDGEMENT

I am highly indebted to our supervisor **Dr. Vivek Sehgal** for his guidance and constant supervision as well as providing necessary information regarding the project and also for his support in completing the project.

We would like to express our gratitude towards other faculty members for their kind co-operation and encouragement which helped us in completion of this project and for giving us such attention and time.

Our thanks and appreciation also goes to the faculty of Computer Science Department of Jaypee University of Information Technology, Waknaghat for their constant support and motivation.

LIST OF FIGURES

Figure Number	Title	Page Number
3.1	Content based recommendation system	12
3.2	Vectors for movies ratings	13
3.3	Collaborative filtering based recommendation system	16
3.4	User based collaborative filtering	18
3.5	Item based collaborative filtering	18
3.6	Model based collaborative filtering	20
3.7	Graph plotted for four user's ratings	21
3.8	Cosine similarity plot	22
3.9	Model Design	26
3.10	Flow chart for System architecture	27
3.11	Flow chart for proposed system	28
4.1	Dataset Snippet for *movies.csv*	32
4.2	Dataset Snippet for *tags.csv*	33
4.3	Dataset Snippet for *ratings.csv*	34
4.4	Latent Matrix snippet for Content based RS	35
4.5	Latent Matrix snippet for Collaborative based RS	36
4.6	Function snippet	36
4.7	Output based on Content based RS	37
4.8	Output based on Collaborative based RS	37
4.9	Output based on Hybrid based RS	37

LIST OF TABLES

Figure Number	Title	Page Number
3.1	Content features of two movies	14
3.2	Content feature based profile computation	15
3.3	Matrix factorization	23

LIST OF ABBREVIATIONS

Abbreviation	Full form
SVD	**Single Value Decomposition**
RS	**Recommendation System**
CB	**Content based**
CF	**Collaborative Filtering**
ML	**Machine Learning**
RMSD	**Root Mean Square Deviation**
TF-IDF	**Term frequency- inverse document frequency**
IMDb	**Internet Movie Database**
CSV	**Comma Separated Values**

ABSTRACT

Predicting the unknown information by gaining knowledge about the user and the items related to the user is the work of recommendation system. Such type of system uses an algorithm to predict the next best movies for user or next best books that the user is most probably to like. In the world of technology, we want somebody to tell us what we should watch or see or read or buy that our based on our likings, everyone is so fast that they do not want to spend time in searching what suits them according to their interest rather they want somebody to tell them. Ecommerce websites uses recommendation system to increase and personalize their websites and also to increase user's interaction and involvement. Such type of websites will display the items the user going to like in accordance with what the user has bought or liked or most interested in.

Our project i.e. recommendation based on movie aims on predicting movies to users according to what are they watching, what they don't like to watch and all these factors. We primarily focus on two types of recommendation systems i.e. content based recommendation systems and collaborative based recommendation systems. To perform this evaluation, predicting ratings are designed for a movie from a user past experience and similar tastes of users. We build our recommendation system and them compare their evaluation and analyze their different results. We also compared the different algorithms that can be use in making or building a recommendation system. Movie Lens is one such dataset that we took advantage of. For the start we have worked on smaller datasets in order to get efficiency in the result. We have created content based, collaborative based and a hybrid model of recommendation system for predicting movies. We have mainly concentrated on user based collaborative filtering under the collaborative based recommendation engine. Evaluating accuracy of such type of model can be difficult because this comes under unsupervised learning part of machine learning. Hence it is very difficult for us to day how our model preformed. Such type of system need to be constantly optimized in order to meet customer expectations and can generate good revenues.

Table of Contents

Chapter-1 1
INTRODUCTION 1
1.1 Overview 1
1.2 Problem Statement 1
1.3 Objective 2
1.4 Methodologies 2
1.4.1. Data Pre-Processing 2
1.4.2 Feature Selection and Feature Generation 2
1.4.3 Implicit rating: 3
1.5 Organization 4
Chapter-2 5
LITERATURE SURVEY 5
2.1 Summary of Papers 5
Chapter-3 11
SYSTEM DEVELOPMENT 11
3.1 Methods or Algorithms 11
3.2 Types of Recommender System 11
3.2.1 Popularity based RS 11
3.2.1.1 Disadvantages of Popularity based RS 12
3.2.2 Content based RS 12
3.2.2.1 Content features 14
3.2.2.2 Content feature based User profile 15
3.2.3 Collaborative based RS 16
3.2.3.1 User based CF 18
3.2.3.2 Item based CF 18
3.2.3.3 Memory based CF 19
3.2.3.4 Model based CF 19
3.2.4 Hybrid Recommender system 20
3.3 Abstraction based extraction 21
3.3.1 Finding users similar to U who have rated item I 21
3.3.2 Calculating ratings based on ratings of user that we have recorded. 22
3.4 Dimensionality Reduction 23
3.5 SVD Algorithm 24
3.6 TF-IDF Vectorizer 24
3.7 Model Design 25
3.8 System architecture for Extractive approach 26
3.9 Proposed System for Extractive approach 27
Chapter-4 29
PERFORMANCE ANALYSIS 29
4.1 Proposed solutions 29

4.2 Analysis 30
4.2.1 Prominent features based analysis 30
4.2.2 Prediction analysis 30
4.3 Dataset 30
4.4 Dataset Snippet 31
4.5 Building Model 34
4.5.1 Building latent matrix for content RS 35
4.5.2 Building latent matrix for collaborative RS 35
4.5.3 Calculating cosine similarity 36
4.5.4 Output 36
Chapter-5 38
CONCLUSION 38
5.1 Future scope of improvement 39
References 40

Chapter-1

INTRODUCTION

This chapter is all about the introduction to our project, what is the aim, what are the methodologies we have used and a brief overview of what e are going to do in next chapters. Also, the technology on which we are going to work has been listed below.

.

1.1 Overview

In the world of artificial intelligence, recommendation system is just another result. It understands user's behaviour and recommend the things a user is going to like or dislike. This is one of the most popular way of doing business and generating high revenues. Amazon declared recently that 70% of their revenues are generated through recommendation systems. So the overall work of this project is to build a recommendation engine that can work on user preference and content of items and gives best possible recommendations of movies. When we talk about what interests an user, we generally focus on user's browsing history, clicking history, geographical coordinates, similar traits etc. and how likely they are going to watch those movies. This is attained through heuristics and predictive modeling by working on our dataset or data available. The major turn or work in recommendation systems is they are personalize to each individual because if it is going to recommend same things to everyone then it cannot be termed as recommendation based on user's preference.

1.2 Problem Statement

Our project aim is to read about and understand the how different recommendation systems work. Then the primary aim is to build such system. A system that can recommend the movies to user based on his interest. People who love to watch movies online, such system can be very beneficial to them. We study user profile, see what type of ratings they have given to other movies and then based in that we are going to recommend them the movies that will interest them. Such types of system help websites like Netflix or Prime Video to generate revenues and to attract large and larger no of customers.

1.3 Objective

a. Data extraction and cleaning: We will use Movie Lens dataset and will extract it using python script. Data cleaning is always necessary before using it because raw data is of no use.
b. Feature extraction using fundamental analysis: The required features are extracted from the dataset and are used in our project.
c. Building the Machine Learning Model: The reduced dataset is applied to build and teach the Machine Learning model for recommendation system.
d. Evaluating accuracy: Accuracy evaluation is difficult but we can only improve our system..
e. Analyzing the work that we have done and see what could further be done for future improvement.

1.4 Methodologies

1.4.1. Data Pre-Processing

The pre-processing stage formally involves

a. Data reducing: Reducing the data but with our importance and in accordance with the algorithms.
b. Data transformation: filtering out the unnecessary information and adding some.
c. Data Cleaning: Cleaning i.e. removing all the unnecessary elements from the dataset keeping the ones we actually needed.
d. Data Integration: Integration of data files

After we have done performing all the pre-processing stages on our dataset, the dataset is used to train model and further for predictive analysis. In such systems, we have only focus on training dataset because testing dataset would make no sense in recommendation engines.

1.4.2 Feature Selection and Feature Generation

In order to get rid of unwanted or noisy features, feature selection method is the most useful and has been used in the field of machine learning for very long. The unwanted features in our dataset generally create chaos in dealing with the dataset. After treating or cleaning such features, dataset looks more simpler and model can be trained for such dataset in a more efficient way. For the movies that we are going to take into consideration are implicit ratings of movies that the user has seen previously. Therefore, we cannot use directly correlation method to calculate the information mutually. We have to take each user's database differently because

recommendation systems are personalized so we have to take into account each user individually. It may not appear important for others but it can be important for one. For the content based recommendation system, user profile $wk_{(u,j)}$ is directly proportional to user u who has watched movies that has feature j and what are the implicit ratings.

$$Rel_{(u,j)} = wk_{(u,j)} \quad \text{....eqn(1)}$$

Generally in many of the classification problems, one feature of two features is omitted if they are correlated to each other. Correlation on two features can be defined as how $wk_{(u,j)}$ and $wk_{(u,j')}$ close are based on the two features j and j′ but for movie recommendation systems, this happen somewhat differently. In content recommendation system, we don't get rid of irrelevant features but we disregard the irrelevant ones. If by chance, a movie contains only some of the important words, that can't be disregard but ahs to be kept in user's profile personally. In our experiment using SVD algorithm we get rid of a features having less variance in order to improve the performance. We performed prediction only on topmost features that explained most of the variance. Average test performance is considered for taking the best ratio of features. If we have discrete ratings or explicit ratings then our recommendation system can use these ratings also.

1.4.3 Implicit rating:

Average test performance is considered for taking the best ratio of features. If we have discrete ratings or explicit ratings then our recommendation system can use these ratings also. Some users do not provide any feedback to the movies they have watched nor hey give any ratings. For such users, an implicit rating has to be calculated based on the ratings provided by other users to that movie in order to recommend new movies to that user. Many of recommendations completely depend on explicit feedback or ratings of users. But if ratings are available then they have to be generated implicitly. Using implicit ratings will however, produce different results with variations. Sometimes we take average of all the ratings that movie has been given by the users and sometimes we use the formula:

$$r_{(u,i)} = t_{(u,i)} / t_i \quad \text{....eqn(2)}$$

.

Here, u is the user, i is the movie and $t_{(u,i)}$ is the duration for which the user watches the movie and t_i is the total time or duration of the movie. This is the normalizing viewing duration or implicit rating of a movie.

1.5 Organization

Chapter 1: This chapter is all about the introduction to our project, what is the aim, what are the methodologies we have used and a brief overview of what we are going to do in next chapters. Also, the technology on which we are going to work has been listed below.

Chapter 2: This chapter lists all the literature survey that has been done in making of this project. All the research papers listed have been read thoroughly in order to have deeper knowledge of the topic.

Chapter 3: This chapter contains all the algorithms and methods that we have studied and used in order to develop our system. Several flow chart and diagrams have been made to have clear understanding of what we have done in our project.

Chapter 4: This chapter contains analysis done in the project. We have also explained step by step execution of our project. Also, it includes snippets from various stages and snippets of output also.

Chapter 5: This chapter is all about the concluding part of the project. It also mentions what are the issues we faced and what further can be done for future improvement.

Chapter-2

LITERATURE SURVEY

This chapter lists all the literature survey that has been done in making of this project. All the research papers listed have been read thoroughly in order to have deeper knowledge of the topic.

2.1 Summary of Papers

2.1.1

Title	**Feature selection for movie recommendation[1]**
Authors	Zehra CATALTEPE Mahiye ULUYAGMUR Esengul TAYFUR
Year of Publications	23 March 2016
Publishing Details	Turkish journal of Electrical Engineering & Computer Sciences
Summary	This paper is research based for the Turkish movie recommendation system by implementing systems based on content and collaborative methods. They have used different features for making the predictions. Each users has their own profile which is used to store their implicit ratings for several features spaces. Therefore, the past ratings of users given are used to produce the content. Natural language processing is used to convert words into vectors. They have used the Turkish movie descriptions since this paper is about the Turkish systems. Profiles are made reliable by selecting features based on the profile of the user. They have checked their performance by performing analysis on different number of movies. An aggregation of both content based and collaborative based has also been performed in this research paper. The concluding part of this research paper includes that success of recommendation can surprisingly improves if selection of features is done properly and accurately.

2.1.2

Title	**An improved collaborative movie recommendation system using computational intelligence[2]**
Authors	Zan Wang School of computer software, Tianjin University, Tianjin 300072, China Xue Yu College of Management and Economic, Yianjin University, Tianjin 300072, China Nan Feng College of Management and Economic, Yianjin University, Tianjin 300072, China Zhenhua Wang American Electric Power Gahanna, OH 43230, United States
Year of Publications	14 October 2014
Publishing Details	Journal of Visual Languages & Computing
Summary	This paper has developed various methods for making the hybrid system. One method we have adopted for our project is making linear combination hybrid recommended system from both the content-based and collaborative-based to try to suggest the best movie combinations they could. This research paper has laid stress more on the collaborative movie recommendations. Ratings from history of profile of user has been captured to cluster together similar neighbors.

2.1.3

Title	**A personalized movie recommendation system based on collaborative filtering[3]**
Authors	V. Subramaniyaswamy School of Computing, SASTRA UNIVERSITY, Thanjavur, India R. Logesh School of Computing, SASTRA UNIVERSITY, Thanjavur, India M. Chandrashekhar School of Computing, SASTRA UNIVERSITY, Thanjavur, India Anirudh Challa School of Computing, SASTRA UNIVERSITY, Thanjavur, India V. Vijayakumar School of Computer Science and Engineering, VIT University, Chennai, India
Year of Publications	24 March 2017
Publishing Details	International Journal of High Performance Computing and Networking
Summary	This research paper has also done research on similar dataset that we have been working on i.e. MovieLens dataset which provides a precise information and is from reliable resource. They have explained the importance of data in today's world through their theory and conducted various experiments to recommend the movies. They have tried to understand the user's behaviors and patterns of watching movies and their tastes in them and tried to recommend the movies which are rated most positively or highly by the users. They have exploited the dataset according to the preferences of the users.

2.1.4

Title	**Exploring Movie Recommendation System Using Cultural Metadata[4]**
Authors	Shinhyun Ahn Graduate School of Culture Technology, KAIST, Republic of Korea Chung-Kon-shi School of Humanities & Social Sciences, KAIST, Daejeon, Republic of Korea
Year of Publications	Feb 2018
Publishing Details	Transactions on Edutainment II
Summary	They have used Word of mouth to recommend the movies and cultural contents. They have mostly worked on metadata of 'word of mouth'. They have tried to improve the recommendation systems that are already available on the web by harnessing their features for a low cast movie recommendation system. They have also evaluated their performance and its strength. They have tried to other things as well like books, articles, new content according to taste and preferences of the audiences.

2.1.5

Title	**Performance Improvement of a Movie Recommendation System based on Personal Propensity and Secure Collaborative Filtering[5]**
Authors	Woon-hae jeong Se-jun-Kim Doo-soon Park Jin Kwak
Year of Publications	March 2013
Publishing Details	J Inf Process Syst, Vol.9
Summary	This research broadly talks about the different recommender systems available as of now. their diferences and their pros and cons. they have briefly mentioned the issues or problems faced due to improper data availability, scalablity, running out of memory, slow running time an dtranparency. They have also emntioned some of the security issues in gathering the data that forms the basis for forming the prfiel of the users and Hence, predicting the movies. They have talked more about the collaborative filtering and less about the content-based recommendation system because that is used more nowadays and is popular. They have tried to solve the glitches faced during making recommendations an dselected the proper propensity of variables in proper utilization of collaborative filtering technique. the 'Push Attack' prinicple is adopted to cope with the issue of security and vulnerability. MovieLens database is used for the framework

	and improvement as well. They have sucessfully developed a personalized and embodied movie recommendation system for optimum an dsecure use in future.

Chapter-3

SYSTEM DEVELOPMENT

This chapter contains all the algorithms and methods that we have studied and used in order to develop our system. Several flow chart and diagrams have been made to have clear understanding of what we have done in our project.

3.1 Methods or Algorithms

We are overloaded with thousands of new articles and new blogs each day. Millions of movies are made. The work of recommender system is to tell which among thousands are of interest to us. This system will help in solving the problem of finding the things of our interest. In any recommender system, there are items and users. Items are movies and users are people who watch movies. So the most basic work of recommender system is on a given set of items and users, is to match the items to users most appropriate items based on their preference of they might or might not like. Forex: Linkedin or Facebook, in such systems, users are the members who are using it and items are also the people who they are matched with. For Amazon, users are members and items are products. Similarly, in the case of movie recommender system which Netflix or Amazon Prime Video mostly uses, users are members and items are movies. Recommendation system shows aspects which relies on individual interest. The propensity of information will increase if anybody relies on it. Recommender systems are getting popular because:

a. Identify products most relevant to the user
b. Personalized content
c. Help website improve user engagement

A match between user and items has created in order to exploit the relationship of similarity between users and items so that proper recommendations of movies can be made.

3.2 Types of Recommender System

3.2.1 Popularity based RS

Popularity based RS recommends movies based on their popularity. From zero probability of cross sell, system is recommending movies without even knowing what kind of users are and what are their preferences. The main advantage such system offer is that we do not need to know what kind of users we have. The popularity-based recommendation system eliminates the need

for knowing other factors like user browsing history, user preferences, the star cast of the movie, genre, and other factors. Hence, the single-most factor considered is the star rating to generate a scalable recommendation system. This increases the chances of user engagement as compared to when there was no recommendation system. It is the same as we say we don't know you but we know what others like. This system will only take care of star ratings not how many people have rated it. And for the items that are now new in database or are not rated by any of the user, we calculate their average rating. The average rating is calculated by sum total of all ratings provided by user to other items divided by total no of items. This average rating will be rating of that new item.

3.2.1.1 Disadvantages of Popularity based RS

a. These types of recommendation systems are not personalized according to user attributes.
b. All users get the same type of recommendations irrespective of their tastes, likes, dislikes, age etc.
c. The mean star rating will have discrepancies based on the number of reviews a movies gets.
d. This type of system does take into account the age of person or viewer, it is going to recommend same things to everyone based on what is trending. The problem is such sytem is that what is liked by a teenager may not be liked by a 50 year old person.

Such type of systems are not useful and can be little bit tweaked. Such systems when tweaked according to the regional dialects, demographic profile of a user or according to business requirement, then they become hybrid recommendation system. They are modified according to the audience.

3.2.2 Content based RS

Figure 3.1:. Content Based Recommendation System

Content based recommended system recommend movies based on the content. It will look for the content watch by the user and then recommend movies based on that content. If a user has watched a particular movie, it will look for movies similar to that movie by extracting features from that movie and then recommend similar movies to watch for that user.
The most common assumption this system makes is people like to watch similar content always by virtue of their taste.

Below is an example to have better understanding.
Let's say Aashi loves to watch movies and has given good ratings to movies like 'James bond' and 'Mission Impossible' which comes under the 'Action' Genre and she does not like the movie 'Toy Story' which falls under 'Children' Genre.
Based on this information, a user vector of Aashi has been created based on her 3 ratings. Aashi loves to watch 'Action' Genre movies so we assign value of 9 to 'Action' Genre, Aashi does not like to watch Animated movies so we assign 0 to the 'Animation' Genre and now, since, Aashi has given a bad rating to the movie belongs to 'Children' Genre, we assign -6 to 'Children' Genre. We are working here, on a scale of -10 to 10.

Thus, the user vector of Aashi in order of (Action, Animation, Children) is (9,0,-6).

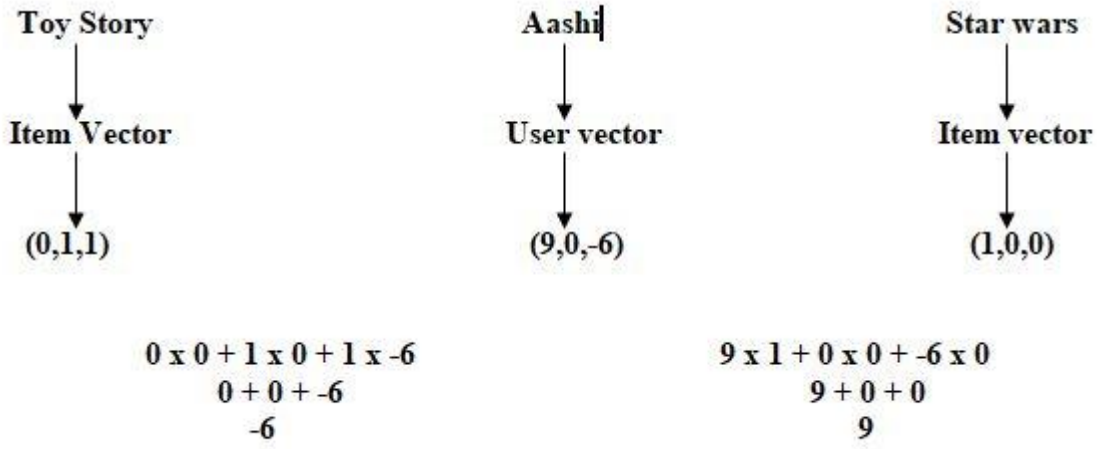


Figure 3.2 : Vectors for movies ratings

As written above, in order of (Action, Animation, Children), (0, 1, 1) and (1, 0, 0) are the item vectors for the movies Toy story and Star wars respectively.
And as we have calculated, the dot product for 'Star wars' is 9 and that of 'Toy story' is -6.

So according to our calculation, 9 score is something we can recommend so we will be recommending ‘Star wars’ to her as she also likes ‘Action’ Genre. Also, she hates or dislikes ‘Children’ Genre.

Similarly, We can create item vector of all the movies that we have in our database product based on viewer dislikes or likes an d can calculate dot product of those item vectors and thus, can recommend top 10 movies to the viewer, here Aashi.

3.2.2.1 Content features

Computer does not understand the language of words so we have to convert the words coming in the content of the movies or the words that are describing the features of movies into numbers. So we will create a token vector of each word and for each movie making it understandable to machine. So in order to do that, tf-idf weight matrix is used. Tf-idf stands for term frequency-inverse document frequency. Tf-idf weights are used to extract the features of movies or keyword features of movies.
The formula for computing tf-idf weight keyword:

$$tf - idf\ (j, i) = tf\ (j, i) \times \log(\ N / nj) \qquad \text{....eqn(3)}$$

Tf(j,i) is the frequency of occurrence of a keyword j in a particular movie i. The idf i.e.inverse document frequency is calculated as log (N / nj). Here, N is total number of movies that are being watched and nj is the number of movies that have the keyword j.

Table 3.1: Content Features of two movies

Movie	Genre	Actor	Keyword
Muppets from Space	Family	Dace Goelz	Plane, puppet, love, curiosity, space
The Shawshank Redemption	Crime drama	Tim Robins	Prison, friend, murder, slavery

3.2.2.2 Content feature based User profile

Content feature based profile of a user is made by looking at the features of the movies that have been watched by the user in the past and the ratings of the movie that can be implicit or explicit depending on what is available for a particular movie. The work of recommendation system is also to lay stress on the content that user might actually want to watch. In our training set, we have used the words that described the movie features, also the genre of the movie. 36 genres and 6000 keywords have been made and categorized into accordingly. The implicit ratings of the user are decided according to the weight of the each feature of the movie for all the current working dataset. The tag of movie may contain action, comedy, drama and lines that express the movie plot.

For calculation weight of each feature of the movie for a particular user u, training dataset has been used for ratings of movie.

Let's say, features $j_{0,\ldots\ldots} j_4$ are the features of movies that have been watched by the users $i_{0,\ldots\ldots} i_8$. The feature set k for user has been showed. Within the period, all the movies that have been watched by the user contain features that are from $j_{0,\ldots\ldots} j_4$. The column of rating in table are the ratings of movies $i_{0,\ldots\ldots} i_8$ for user u .

Table 3.2: Content feature based profile computation

User u	j_0	j_1	j_2	j_3	$j_{4\ldots\ldots\ldots}$	r (u,i)
i_0	1	1	0	0	0	0.5
i_1	0	1	0	0	0	0.3
i_2	1	1	1	0	0	0.9
i_3	1	0	0	1	0	0.7
i_4	0	0	0	1	0	0.2
i_5	1	0	0	1	0	1.0
i_6	1	0	0	1	0	0.44
i_7	0	1	0	0	0	0.67
i_8	1	0	0	1	0	0.2
W_k (u,j)	0.42	0.26	0.1	0.26	0	

The feature j in feature set k holds weight for user u is :

$$W_k(u,j) = \frac{1}{|I_u^{train}|} \times \sum_{i \in I_u^{train}} x_{k,u}(i,j) \times r(u,i) \qquad \ldots.eqn(4)$$

I_u^{train} is the set of the movies that are within the period watched by the user. k stand for the type of feature set used like genre, keywords, director, release year etc. r(u,i) stand for the implicit ratings of movie given by the user u of movie i. $x_{k,u}$ (i,j) is the j^{th} feature of the movie i. if supposedly, movie I has found a feature j, then user rating for the movie will contribute to the sum where $x_{k,u}$ (i,j) will be equal to 1.

We have shown the genre, director, year of release, director for three different users for content based profile. Thus, different number of elements will be present in user profiles. When type of the feature changes from release year to director to genre of the movie, the number of features in the profile of the user decreases. We can thus, based on the user's profile reduce some of the information that we think might not important to us.

After each user based on each feature weight has been calculated, we will finally use them to predict the contents recommendation ratings. The weights we have obtained from our current working dataset will be sued separately to calculate the rating of each feature of each user based on movies watched by the user.

$$r_k(u,i) = \sum_{j \in F_{kj}} W_k(u,j) \qquad \text{...eqn(5)}$$

In the above equation, $F_{k,j}$ stands for the features of all the feature set k related to movie i. r_k (u,i) stands for the rating movie I gets by the user u based on the feature set k.

3.2.3 Collaborative based RS

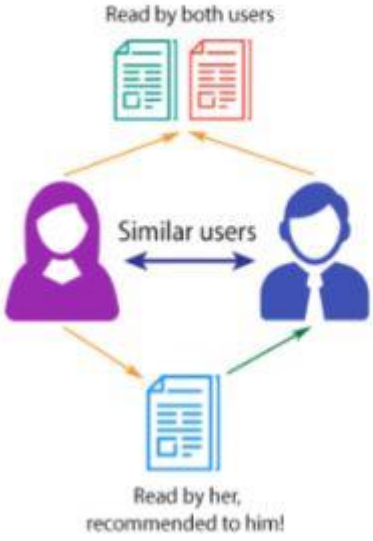


Figure 3.3: Collaborative Filtering based Recommendation System

Collaborative filtering makes recommendations for a given user based on aggregate rating information of similar users in a stored database. Sometimes one user has seen some movie and another user has seen the same movie. Now according to collaborative filtering, these both users profile will be similar based on their tastes. So next time, the movies seen by first user will be recommended to second user and movies seen by second user will be recommended to first user. This is the very basic idea about collaborative filtering that even if user has not seen the movie and we need to predict the rating of a particular movie for a particular user, then we have to check the similar users based on their similar ratings of past movies have liked the movie or not and thus, based on this we can form the recommendations

Content based recommended systems characterize each movie very uniquely based on the features but collaborative filtering technique have become more and more successful these days because they do not require content of the movie to predict movies. They can provide recommendations based on similar users (i.e. users neighbors) without containing content from the user's profile. These are the reason we have primarily focused to work on collaborative filtering technique of recommending movies. But for instance, we have worked on both content based recommendation system and collaborative based recommendation system in order to see the difference in their predictions and their working.

$$R_c(u,i) = \frac{1}{C} \text{ x } \sum_{v \in D_{i,u}} c\ (u,v) x\ r\ (v,i) \qquad \textit{...eqn(6)}$$

In the above equation, r (v,i) stands for the implicit rating for movie I by the user v. c (u,v) ≥ 1 stands for the number of movies that both the user (user u and user v) have watched(in order to find similar users) within training period. C Here, stands for the normalization constant, which is defined as:

$$\text{C} = \sum_{v \in D\ i,u} c\ (u,v) \qquad \textit{...eqn(7)}$$

Collaborative filtering technique generally faces problem like slow running time, scalability, comparable accuracy and the large number of parameters. In order to deal with the issue of large parameters and their vectors, we have used matrix factorization on our working dataset. Also, we have preferred to use the simple neighborhood-based collaborative filtering method.

The most common assumption this method makes is people with similar interests tend to like behave or like similar things. Websites like Amazon uses collaborative filtering to sell their products and it has been surveyed that 70% of their sell is because of their recommender system.

Steps involved in CF

a. Finding how a user is similar to another user in order to find group of users sharing similar interests so that recommendations can be made.
b. After knowing which users share similar interests or are similar to each other in terms of likes, find how a user gives rating to a movie based on rating given by the other users that share the similar interests.
c. Finding the accuracy of the model or to see, to what extent predictions made are correct.

Based on the recommendations, collaborative based recommendation systems are classifies into two types:

a. User-based collaborative filtering
b. Item-based collaborative filtering

3.2.3.1 User based CF

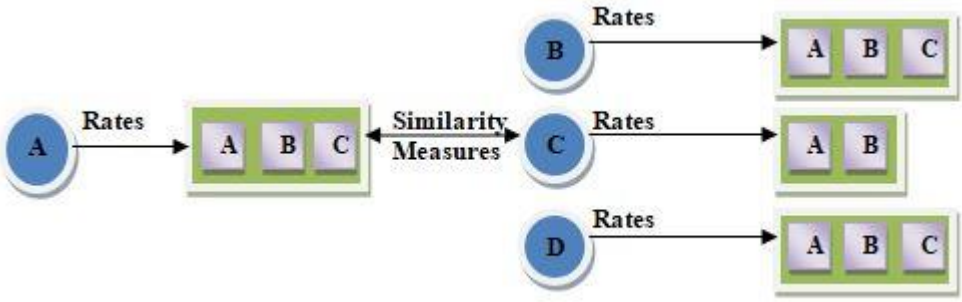


Figure 3.4: User based Collaborative Filtering

"Similar choice" people are identified and their choices are being compared through the users database. This type of system is called User-based collaborative filtering. User rating matrix is made and is used to find users that are similar to each other based on the ratings they give to the movies. After identifying such clusters or groups, content which is rated highly by the users is recommended to our target user.

3.2.3.2 Item based CF

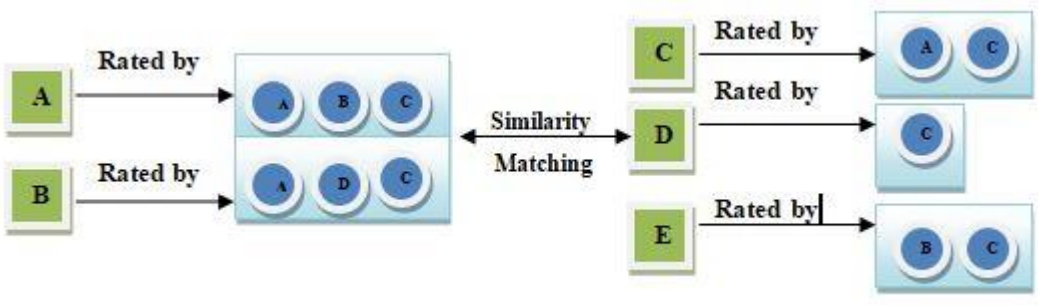


Figure 3.5 : Item based Collaborative Filtering

Each item is examined from the target user's list and similar items are then found from the item choice set. This method is known as Item-based collaborative filtering recommendation system. In this method, item matrix is used in which similar items are found out on the basis of the ratings given to them by the users. Generally, correlation between items is calculated but that means deduction of features, so we have also used predefined attributes like director, movie genre, movie summary etc. to find the similarity. The computation of recommendation is case of item- based collaborative filtering is much faster because based on the explicit features of the movies, items can be easily categorized or pre-scored. This is the mail advantage of Item-based collaborative filtering over User-based collaborative filtering as it increases scalability.
Collaborative filtering is also categorized into two more parts:

a. Memory-based Collaborative filtering
b. Model-based Collaborative filtering

3.2.3.3 Memory based CF

In memory based collaborative filtering, we create a matrix and using that matrix we predict the ratings. User's rating on a set of items are recorded and then compared and similarity between them is calculated and often the entire database is used in this technique. To calculate the similarity, we have used in our project "Cosine Similarity". We can also use other methods like Vector similarity based approach, Pearson- correlation based approach, Euclidean distance or extended generalized vector- space model.

3.2.3.4 Model based CF

In the model-based collaborative filtering technique, users are grouped together into small number of clusters poor groups based on their rating patterns so that similar users can be found. Every user is differentiated into one or more predefined clusters and based on these clusters and ratings they have given on particular items, predictions are made. To form clusters or to find classes of users, different machine learning algorithms are used on an item to predict the user's ratings.
Below drawn flow chart explained algorithms that can be used

a. We can use KNN (k-Nearest neighbours) to find the clusters of similar users.
b. We can use matrix- factorization based model like SVD (single Value decomposition)

c. We can also use deep learning techniques like auto-encoders or embedding as low dimensional hidden factors for users and items.

Other algorithms like Bayesian network approach, aspect models or mixture models can also be used.

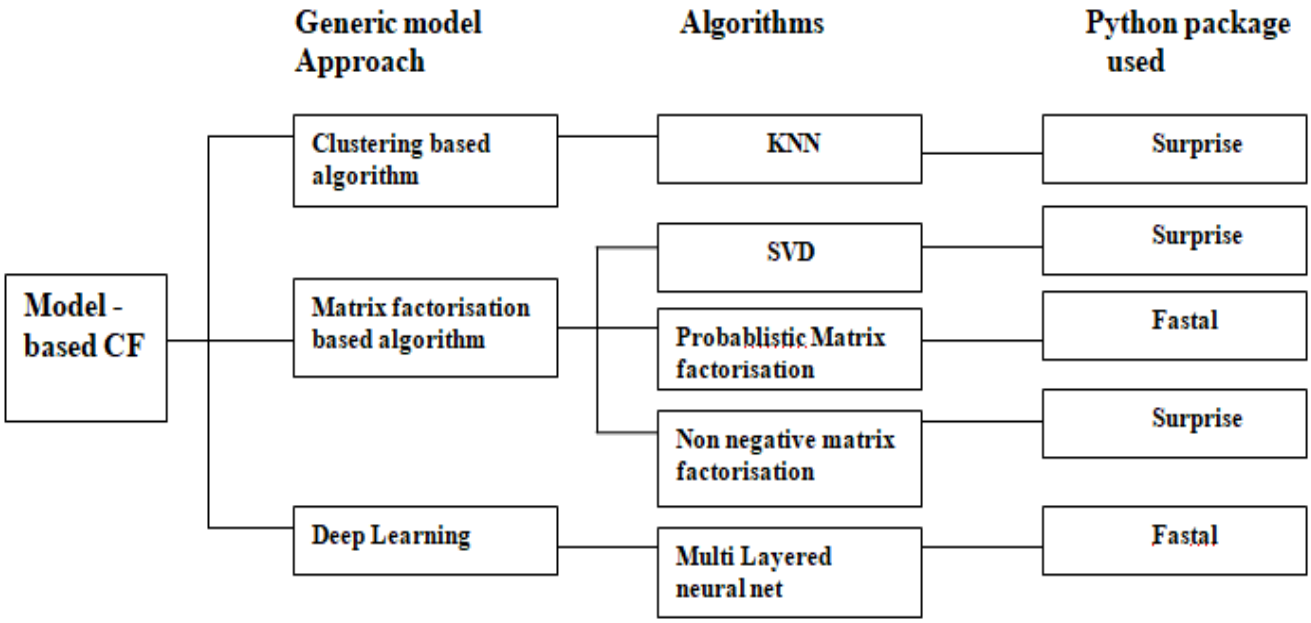


Figure 3.6 : Model based Collaborative Filtering

3.2.4 Hybrid Recommender system

We have also worked on Hybrid Recommendation System in our project. The hybrid recommendation system can be made in many ways but our hybrid system is just a linear combination of the ratings that are calculated form collaborative method and content based method.

Cold start problem is the major drawback of recommendation system and using the hybrid recommendation system, this problem can be avoided. The combination of these two methods (content based and collaborative based) can be proceeded in many ways:

a. Join the results obtained from separately implemented algorithms.
b. Both approaches can be brought together to create a unified recommender system.
c. Using some rules and techniques of collaborative filtering in content based system.
d. Using some rules and techniques of content based system in collaborative filtering approach.

Current Implementations

We have studied different datasets specifically related to movie recommendation systems like Yahoo! Movie, Netflix and Movie Lens. We have finally arrived to work on Movie Lens dataset with 100,000 ratings. The systems execution is elaborated in depth.

3.3 Abstraction based extraction

Techniques based on statistics are used over entire dataset in which we have calculated the rating (R) based on the user (U) that would be given to an item I and predictions are calculated.
Two things are mainly focused in this.

3.3.1 Finding users similar to U who have rated item I

We have created a sample dataset. It consists of four users A, B, C, D who have rated two movies. Their respective ratings are stored in the lists and each lists containing 2 numbers indicates rating of each movie.

a. Ratings by A [1.0, 2,0]
b. Ratings by B [2.0, 4.0]
c. Ratings by C [2.5, 4.0]
d. Ratings by d [4.5, 5.0]

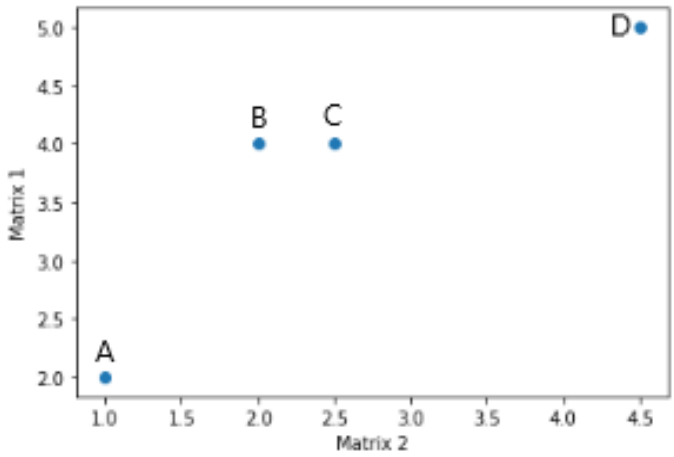


Figure 3.7: Graph plotted of four user's ratings

Using Euclidean distance to find the similarities between these four people, we found out that most similar users are c and because their Euclidean distance is less. But using cosine similarity here, because points are of uneven size and cosine similarity proves good for document of uneven size whereas Euclidean distance gives wrongs result in such cases. Also for points which

are along the same line, Euclidean distance gives wrongs answer because it works on the distance whereas cosine similarity gives good output because it works on the angle. It gives high similarity for low angle and low similarity for high angle.

The cosine similarity is cosine of an angle which is a function that decreases from 1 to -1 as angle increases form 0 degrees to 180 degree.

The formula for calculating cosine similarity:

$$u(c,s) = \cos(\overrightarrow{w_c}, \overrightarrow{w_s}) = \frac{\overrightarrow{w_c} . \overrightarrow{w_s}}{||\overrightarrow{w_c}|| \text{ x } ||\overrightarrow{w_s}||} \quad \text{...eqn(8)}$$

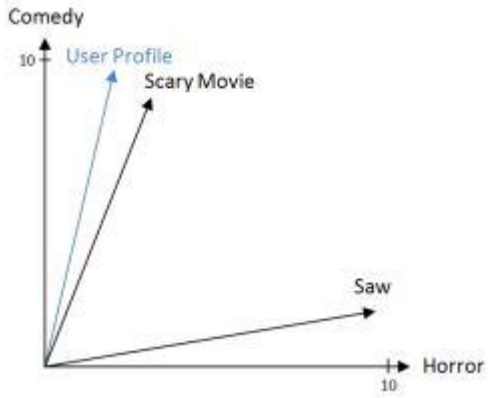


Figure 3.8: Cosine similarity plot

The angle between the two items will be small if two items are similar to each other. Thus, if items are described more properly or accurately, then their recommendations would be better because searching requires information about the items in detail.

After calculating cosine similarity between four users, we found that 'A' and B' are most similar because their cosine distance is almost 0.

3.3.2 Calculating ratings based on ratings of user that we have recorded.

For calculating the ratings:

a. Calculating the rating of all top 10 users that are similar to our particular user and taking average of them.

$$R_v = \left(\sum_{u=1}^{n} R_u\right)/_n \quad \text{....eqn(9)}$$

b. Taking weighted average because in a recommendation list always, first user will be

similar to our particular user but as we go down in that list, similarity decreases so the last user would not be that much similar or not similar at all. To save this, we multiply ratings with similarity factors.

$$R_v = \left(\sum_{u=1}^{n} \boldsymbol{R_u} * \boldsymbol{S_u} \right) \Big/ \left(\sum_{u=1}^{n} \boldsymbol{S_u} \right) \qquad \text{....eqn(10)}$$

3.4 Dimensionality Reduction

In the user item martrix there are tow dimensions:

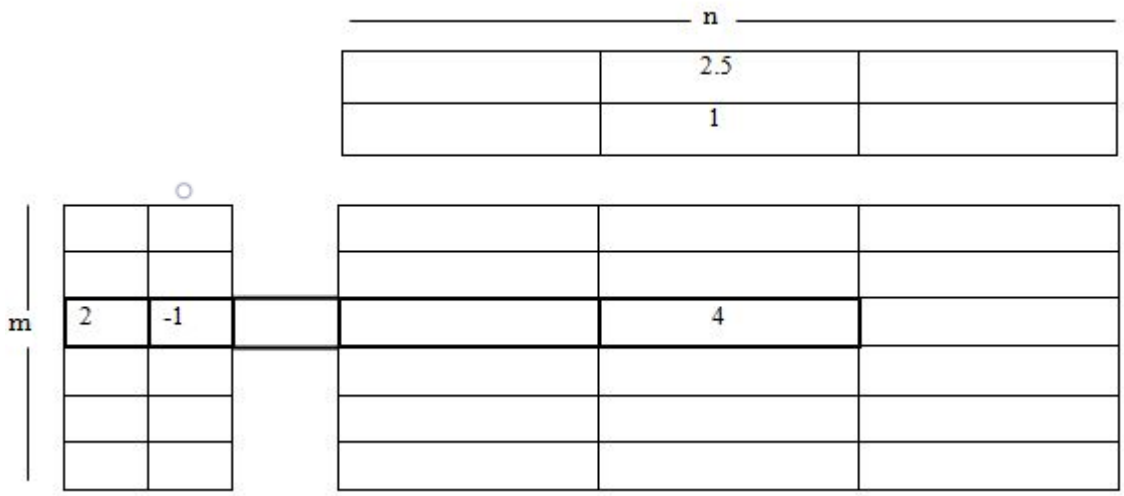
a. The number of suers
b. The number of items.

If the matrix is mostly empty, reducing dimensions can improve the performnace of the algorithm in terms of both space and time.

Matrix factorisation can be seen as breaking down a large matrxi into a product of smaller ones. This is similar to the factorisation of intergers, where 12 can be written as 6 x 2 or 4 x 3. In the case of matrices, a matrix A with dimensions m x n can be reduced to a product of two matrices X and Y with dimensions m x p and p x n respectively.

The reduced matrices actually represent the users and items individually. The m rows in the first matrix represent the m users, and the p columns tell you about the features or charactersitics of the users. The same egoes for the item matrix with n items and p characteristics. Here's an example of how matrix factroisation looks:

Table 3.3 : Matrix Factorization



In the image above, the matrix is reduced into two matricws. The one on the left is the user matrix with m users and the one on the top is the item with n items. The rating 4 is reduced or factorised into:

a. A user vector (2,-1)
b. An item vector (2.5,1)

The two columns in the user matrix and the two rows in the item matrix ar called latemt factorisatioon and are indication of hidden characteristics about the users or the items. A possible inbterpretation of the factorisation could look like this:

a. Assume that in user vector (u,v) u represnets how much u user likes the horror genre, and v represents how much they like th e romance genre.
b. The user vector (2,-1) thus represents a user who likes horror movies and rates them positively and dislikes movies that have romance and rate the negatively.
c. Asuume that in item vector(I,j) I represnts how much a movie belongs to hrror genre and j represnets how much a movie belongs to the romance genre.
d. The movie(2.5,1) has a horror rating of 2.5 and a romance rating of1. Multiply it by the user vector sing matrix multiplication rules gives you (2*2.5)+(-1*1)=4

So the movie belonged to the horror genre, and user cou,d have rated it 5 but the slight inclusion of romance caused the final rating to drop it by 4.

The number of latent factors affestcs the recommendations in a manner where the graetor the number of factors, the more personalised the recommendations become. But too many factiors can leads to over fitting in the model.

3.5 SVD Algorithm

With SVD, it depends on programmer to decide how many features to keep. The amount of variance keeps on decreasing with amount of features. The first feature contains most of variance and as we go down, amount of variance also gets decreasing. For dimensionality reduction, SVD is a algorithm and works well for sparse datasets. We will fit SVD on our matrix tf-idf. Because it has large no of columns so we need to reduce the dimensions. Compress it with SVD because if we plot a graph, almost 50% variance is explained by first 200 components..

3.6 TF-IDF Vectorizer

The TF-IDF stands for Term Frequency-Inverse Document Frequency. The TF-IDF is an algorithm that is used to calculate the frequency of a particular keyword in a document. The importance of that word is then assigned depending upon the number of times a keyword

appears. In a more simpler way, the term having higher tf-idf score, that term will be important and rarer and vice-versa. Every word or content has its tf-idf core or value or weight and the product of these two terms is collectively called tf-idf weight of that particular keyword. Inverse Document Frequency (IDF) tells how importance that term holds in a whole collection of document (corpus).

TF (t) = (Number of times term t appears in document) / (Total number of terms in the document)

IDF (t) = log_e (Total number of documents / Number of documents with term t in it)

$$W_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right) \qquad \text{...eqn(11)}$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

The above formula is sued for calculating the tf-idf score in a document. In python, Scikit library has already provided a tf-idf vectorizer that easily calculates this score for each word description. For our project, we have used this in case of content based recommendation system. This tf-idf matrix contains each word and simultaneously its tf-idf score also for each movie (in our case). We have also used stopwords of English which will remove the frequently occurring words from the list like ‘a’, ‘is’, ‘are’, ‘the’ etc.

3.7 Model Design

Below flow chart depicts the flow of our model.

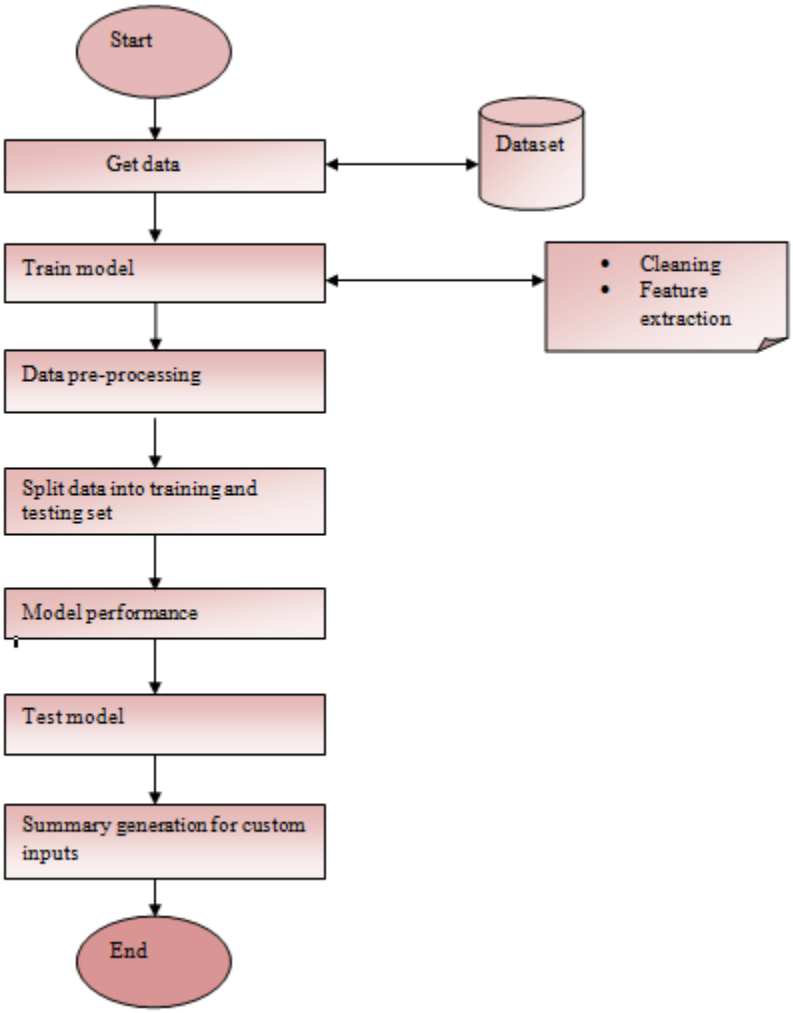


Figure 3.9 : Model Design

3.8 System architecture for Extractive approach

We are using Collaborative filtering for our movie recommendation system project. CF algorithm will group together the users depending upon their choices or preference of movies or the ones who enjoy same set of movies. A certain movie will be given to the user every time a new user enters in order to find similar users like him and prediction or recommendation can be made easily. These clusters changes time to time as the preferences of users changes in order to optimise our recommendation system performance.

Below flow chart describes how the execution in our project takes place.

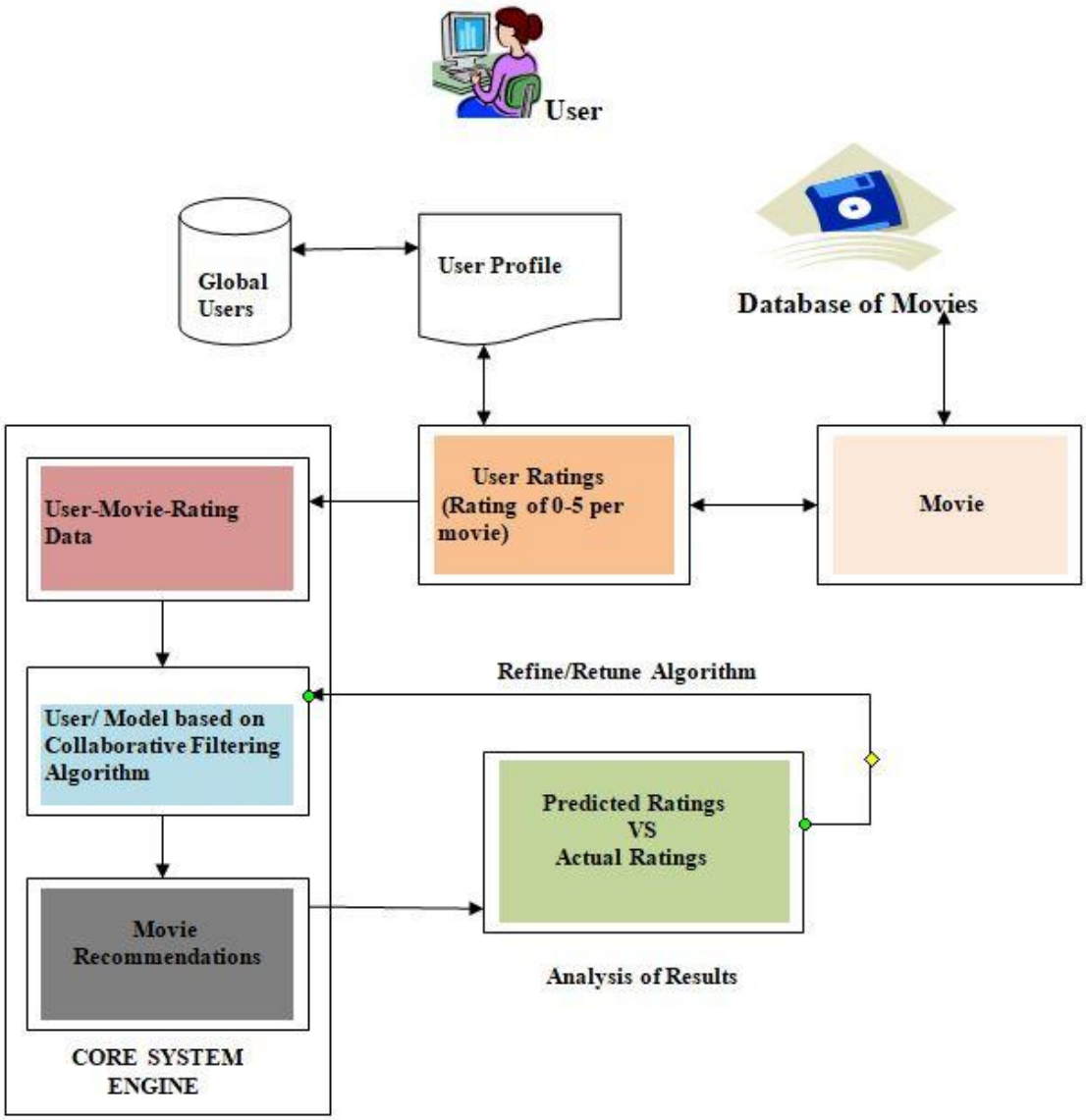


Figure 3.10 : Flow chart for System Architecture

3.9 Proposed System for Extractive approach

This phase tells that we have made two latent matrix , one of user and one of movies and both are feed into the three systems (content based, collaborative filtering based and hyrbid based) to make recommnedations

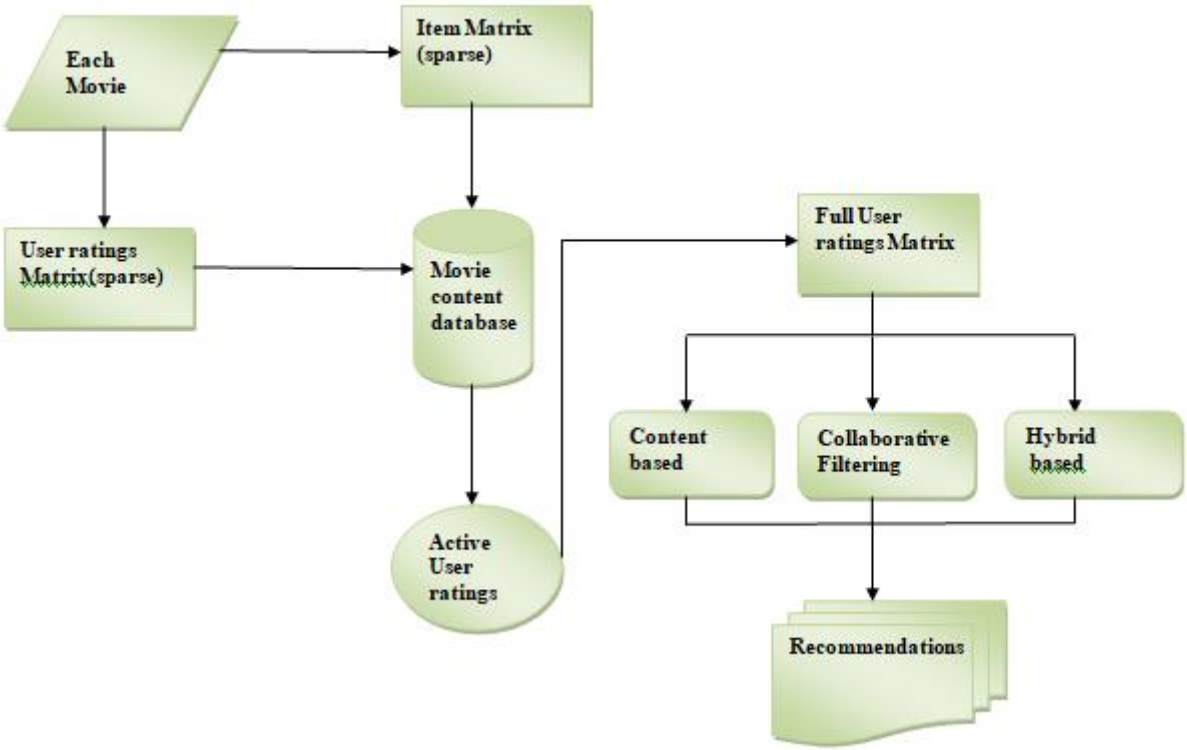


Figure 3.11: Flow chart for proposed system

Chapter-4

PERFORMANCE ANALYSIS

This chapter contains analysis done in the project. We have also explained step by step execution of our project. Also, it includes snippets from various stages and snippets of output also.

4.1 Proposed solutions

This project essentially categorizes the comparisons between content based recommendation system, collaborative recommendation system and hybrid based recommendation system between their users across the entire database of Movie Lens. The process of implementation can be categorized into two stages:

a. Developing Algorithm

   We have chosen to implement Memory based implementation of collaborative filtering technique so there is no proper implementation of algorithm but the cleaning nad extraction of important features from the database. We have chosen to run the made algorithm on our entire dataset which is left after processing and cleaning. We have removed the less relevant features and also the less important keywords from the content in the content based recommendation system. Doing this step has created the chances of having useful features in dataset than already existing ones. This also helps in improving our algorithm and revising it by focusing more.

b. Evaluation

   Testing or Evaluation of such type of projects is difficult because we cannot evaluate accuracy of our model. However, we can evaluate accuracy in terms of the ratings predicted by the model but we cannot check if the movie recommendations made by the model are correct or not. The only chance is to optimize and improve the performance by scaling it over the more dataset. Over-fitting and under-fitting are avoided.

c. Improvement

   Machine learning model have always space for improvement. They cannot be 100% correct always in making prediction. Making system scalable is the ultimate aim here in this project because as the number of users in the database will increase, problem of scalability will increase. All the features need to be given proper weightage in order to have better results.

4.2 Analysis

Analysis of presented work has been done through entering different movie names and seeing the results.

4.2.1 Prominent features based analysis

a. Coding in a respectful manner: Coding has been done in a way it can help other understanding.
b. Design or Modularity: The design of model has been kept minimum because we are working for small dataset for now. The maximum flexibility has been tried to achieve in the code as well as in the design.
c. Memory utilisation: Handling large dataset is difficult when you have to keep pace with memory also. Initially we were working with 20M dataset but that result in too slow running time of hours and hours. We reduced it foe now by reducing considerable amount of our dataset.

4.2.2 Prediction analysis

a. Working on small dataset- There are many kind of movie lens dataset. One is of 20M ratings, one is of 1M but for now, we have decided to choose 100,000 ratings of database. We were not able to handle large dataset. We chose smaller one for now because we want to check our performance on smaller dataset for now.
b. Using hybrid CF model: After performing content based and collaborative based techniques for recommending movies, we have decided to implement hybrid by combining the efforts of both the algorithms (content based and collaborative based). User based/ memory based CF algorithm is implemented for clustering process. This algorithm despite having its own disadvantage, we improve the parameters on our dataset.

4.3 Dataset

Our experiments based on the dataset provided by the Movie Lens of 100K ratings. We have first chosen to work on the 20M database but that is occupying lots of memory and slowing down the running time. Besides this, it significantly imposes scalability issues. For the short durstion of time, we want our hands experienced firstly on small dataset to see the accuracy and performance of the model. Our version of database contains 100,004 (100K) ratings from 671 users for almost 9125 movies. The scale of ratings is from 0.5 to 5 on a scale of 0 to 5. The users

who have not given ratings to any movies or movies that have not been given any rating, we have decided to fill them with 0 for now. Our dataset has also the information about the genre of movie. Also, it contains tags for movies. Genre and tags of movies are used only for content based technique and we have decided to ignore them for collaborative filtering technique. For content based, tf-idf vectorizer is created from the genre and tags of the movies by combining them. Also, we have filtered out the user who has not rated movies less than 55 to have better results. Demographic information about the user is not includes. Users have their information in the form of user Id and for now, no other information is included. Three files of Movie Lens dataset have been used in building up of a model – ratings file which contains all the information about the ratings of movies along with the movie id and user id, movies file which contain information about the movie name, their id and genres and tags file which contains information about the tags of each movie.

4.4 Dataset Snippet

We have worked on three files for our project..

a. Our first file is movies.csv having three attributes namely:
    1. MovieId- Id of the movie assigned uniquely.
    2. Title: Title or name of the movie.
    3. Genres: the genre of the movie.

b. Our second file is tags.csv containing four attributes:
    1. UserId- The Id assigned to each user uniquely
    2. MovieId- Id of the movie assigned uniquely.
    3. Tag- One word that describe the movie.
    4. Timestamp- time at which movie is rated.

c. Our third file is ratings.csv having four attributes:
    1. UserId- The Id assigned to each user uniquely.
    2. MovieId- Id of the movie assigned uniquely.
    3. Rating- rating given to the movie.
    4. Timestamp- Time at which movie is rated.

Snippets of dataset have been posted below.

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller
10	11	American President, The (1995)	Comedy|Drama|Romance
11	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
12	13	Balto (1995)	Adventure|Animation|Children
13	14	Nixon (1995)	Drama
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
15	16	Casino (1995)	Crime|Drama
16	17	Sense and Sensibility (1995)	Drama|Romance
17	18	Four Rooms (1995)	Comedy
18	19	Ace Ventura: When Nature Calls (1995)	Comedy
19	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller
20	21	Get Shorty (1995)	Comedy|Crime|Thriller
21	22	Copycat (1995)	Crime|Drama|Horror|Mystery|Thriller
22	23	Assassins (1995)	Action|Crime|Thriller
23	24	Powder (1995)	Drama|Sci-Fi
24	25	Leaving Las Vegas (1995)	Drama|Romance
25	26	Othello (1995)	Drama
26	27	Now and Then (1995)	Children|Drama
27	28	Persuasion (1995)	Drama|Romance
28	29	City of Lost Children, The (Cité des enfants p...	Adventure|Drama|Fantasy|Mystery|Sci-Fi

9742 rows × 3 columns

Figure 4.1 : Dataset snippet of file *movies.csv*

	userId	movieId	tag	timestamp
0	2	60756	funny	1445714994
1	2	60756	Highly quotable	1445714996
2	2	60756	will ferrell	1445714992
3	2	89774	Boxing story	1445715207
4	2	89774	MMA	1445715200
5	2	89774	Tom Hardy	1445715205
6	2	106782	drugs	1445715054
7	2	106782	Leonardo DiCaprio	1445715051
8	2	106782	Martin Scorsese	1445715056
9	7	48516	way too long	1169687325
10	18	431	Al Pacino	1462138765
11	18	431	gangster	1462138749
12	18	431	mafia	1462138755
13	18	1221	Al Pacino	1461699306
14	18	1221	Mafia	1461699303
15	18	5995	holocaust	1455735472
16	18	5995	true story	1455735479
17	18	44665	twist ending	1456948283
18	18	52604	Anthony Hopkins	1457650696
19	18	52604	courtroom drama	1457650711
20	18	52604	twist ending	1457650682
21	18	88094	britpop	1457444500
22	18	88094	indie record label	1457444592
23	18	88094	music	1457444609
24	18	144210	dumpster diving	1455060381
25	18	144210	Sustainability	1455060452
26	21	1569	romantic comedy	1419805413
27	21	1569	wedding	1419805419
28	21	118985	painter	1419805477

199 rows × 3 column

Figure 4.2 : : Dataset snippet of file *tags.csv*

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
5	1	70	3.0	964982400
6	1	101	5.0	964980868
7	1	110	4.0	964982176
8	1	151	5.0	964984041
9	1	157	5.0	964984100
10	1	163	5.0	964983650
11	1	216	5.0	964981208
12	1	223	3.0	964980985
13	1	231	5.0	964981179
14	1	235	4.0	964980908
15	1	260	5.0	964981680
16	1	296	3.0	964982967
17	1	316	3.0	964982310
18	1	333	5.0	964981179
19	1	349	4.0	964982563
20	1	356	4.0	964980962
21	1	362	5.0	964982588
22	1	367	4.0	964981710
23	1	423	3.0	964982363
24	1	441	4.0	964980868
25	1	457	5.0	964981909
26	1	480	4.0	964982346
27	1	500	3.0	964981208
28	1	527	5.0	964984002

199 rows × 3 columns

Figure 4.3 : : Dataset snippet of file *ratings.csv*

4.5 Building Model

For this project we considered movie lens dataset of 100,000 ratings which is small dataset and focused on three files i.e. movies.csv, ratings.sv, tags.csv.

Steps involved in building our recommendation system:

a. We Imported all the required libraries.

b. Importing ratings files in a dataframe.

c. Importing tags file in dataframe.

d. Importing movies file sin dataframe.

e. we use title of movie and genre of movie from movies file and tag of movie from tags file to create a vector for each movie id b. We limit ratings to user ratings.

f. We have already filter data on the base of user ratings. Users which have rated more than 55 movies will be the part of our system.

g. we created the mixed dataframe of tags, movie title and movie genre and named it metadata and all user tags given to each movie.

h. For movies that have not been rated for now, we have replace their values with 0 to fill null a values in our dataframe.

4.5.1 Building latent matrix for Content RS

a. For conetent based rs, we created a content latent matrix from movie metadata we just created.

b. For creating content latent matrix, we used the TF-IDF algorithm.

c. Using a tf-idf vectors, a sparse matrix is created. This matrix is sparse because it has more number of zeros than actual numbers.

	0	1	2	3	4	5	6	7	8	9	...	190	191	192	
Toy Story (1995)	0.026804	0.052404	0.019479	0.003361	0.004004	-0.024907	0.115090	0.012749	-0.001627	0.103939	...	-0.034889	0.140789	0.064087	-0.02
Jumanji (1995)	0.010512	0.010537	0.025846	0.000882	0.013254	-0.001140	0.068154	0.012675	0.006184	0.050531	...	0.010494	-0.032140	0.036050	0.05
Grumpier Old Men (1995)	0.038827	0.073948	-0.004930	-0.001323	0.031760	0.002476	-0.003612	-0.001630	-0.000253	0.000645	...	-0.001348	-0.011979	0.006573	-0.01
Waiting to Exhale (1995)	0.136327	0.077169	-0.020606	-0.002103	0.101073	0.012071	-0.012106	-0.002062	-0.004896	-0.002583	...	-0.024807	-0.022650	0.002608	0.05
Father of the Bride Part II (1995)	0.038644	0.084084	0.000456	0.000029	-0.013619	0.000075	0.013469	-0.000218	0.008462	0.013971	...	-0.006709	-0.003465	-0.033741	0.02
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Kein Bund für's Leben (2007)	0.383834	0.889127	-0.089173	-0.019586	-0.217577	0.002209	-0.033704	-0.043409	-0.016034	-0.009939	...	-0.000204	0.000476	-0.000306	-0.00
Feuer, Eis & Dosenbier (2002)	0.383834	0.889127	-0.089173	-0.019586	-0.217577	0.002209	-0.033704	-0.043409	-0.016034	-0.009939	...	-0.000204	0.000476	-0.000306	-0.00
The Pirates (2014)	0.006151	0.003922	0.014204	0.001669	0.012996	-0.021347	0.058406	-0.000141	-0.001525	0.015742	...	-0.007593	-0.008035	0.005662	-0.00
Rentun Ruusu (2001)	0.000028	0.000019	0.000157	0.000417	0.000186	0.000027	0.000401	-0.000411	0.004754	0.000053	...	-0.000267	0.000754	0.000452	-0.00
Innocence (2014)	0.056550	0.044639	0.334947	-0.016995	0.040865	0.321463	0.348913	0.189100	-0.033612	-0.041348	...	0.001049	0.000943	-0.000115	0.00

26694 rows × 200 columns

Figure 4.4 : Latent Matrix snippet for Content based RS

d. Now for dimensionality reduction, we had used SVD algorithm. Which work well for sparse datasets.

e. We have kept only 20 dimensions that are explaining our 50 % dataset.

4.5.2 Building latent matrix for Collaborative RS

We created a collaborative latent matrix from the movie metadata we just created.

a. For collaborative based, we merge the movies and ratings dataframe and created a pivot table.

b. That pivot table is the actual latent matrix of our collaborative system.

c. We used SVD to reduce the dimensions and used 200 components because they explained 50% of our dataset.

	0	1	2	3	4	5	6	7	8	9	...	190	191	192	
Toy Story (1995)	36.391240	-4.958788	14.331820	1.844157	-3.077922	-1.351172	8.341426	-0.561091	-1.898198	1.657859	...	3.815175	-1.518232	2.904773	0.
Jumanji (1995)	20.447499	0.675231	11.354480	-7.325453	-3.505242	3.585610	4.393396	-5.564558	0.434478	0.351438	...	-1.976575	-0.862297	-0.145221	-2.
Grumpier Old Men (1995)	8.407816	-5.190801	4.378314	-6.189039	-0.333121	1.924149	-1.548362	0.795947	-3.196046	2.121787	...	-0.487350	-0.422989	0.218700	0.
Waiting to Exhale (1995)	0.408037	-0.664700	0.739903	-0.202676	-0.498374	1.055409	0.184081	-0.508293	-0.429504	0.086539	...	0.016899	0.000638	-0.205163	-0.
Father of the Bride Part II (1995)	5.755079	-1.757369	3.961287	-3.673278	-3.688047	3.237859	1.140586	-0.690787	-2.171826	1.365967	...	-0.383190	0.092377	-0.420799	0.
Heat (1995)	21.147710	-6.162329	7.098517	2.507193	6.830614	8.125510	-5.282010	-0.279932	5.225557	0.826434	...	1.914424	0.603509	-1.633706	0.
Sabrina (1995)	6.168381	-4.654849	3.343397	-3.640084	-4.642767	5.257256	-0.176210	-0.764384	-0.495277	-1.106928	...	0.606525	1.460002	-0.085679	-0
Tom and Huck (1995)	1.561056	-0.488986	0.337722	-2.133252	-0.306774	2.064745	-1.066103	0.140543	-0.333948	0.510197	...	-0.183539	0.194142	0.369061	-0.
Sudden Death (1995)	0.934632	-0.872525	1.150692	-0.673230	0.571192	1.006842	-0.395729	-0.166296	0.037785	-0.091724	...	-0.504168	-0.524163	0.712095	1.
GoldenEye (1995)	20.793424	-5.617003	14.540853	-5.142048	3.584061	2.751457	-2.621010	-1.027200	1.848644	-0.688415	...	-0.567197	-1.260712	0.311721	-0.
American President, The (1995)	11.695284	-9.641686	7.035483	-3.435145	-4.943520	8.478412	2.946287	2.028756	-0.780606	-0.092433	...	0.617965	-1.006106	0.216956	0.

11 rows × 200 columns

Figure 4.5 : Latent Matrix snippet for Collaborative RS

4.5.3 Calculating cosine similarity

a. For collaborative filtering technique, we decided to choose cosine similarity in order to calculate the relation between the movies.

b. Calculated the cosine similarity with our particular movie with others in the list

c. For hybrid system, we took the average of both content and collaborative scores.

We also build a function to calculate the similar movies

```
In [18]: def get_similar(movie_name,rating):
             similar_ratings = corrMatrix[movie_name]*(rating-2.5)
             similar_ratings = similar_ratings.sort_values(ascending=False)
             #print(type(similar_ratings))
             return similar_ratings
```

Figure 4.6 : Function snippet

4.5.4 Output

The three images below shows three different outputs for a particular movie.

a. Shows movie recommendation based on content-based RS.

	content	collaborative	hybrid
Bug's Life, A (1998)	0.900620	0.561640	0.731130
Toy Story 2 (1999)	0.767566	0.662620	0.715093
Up (2009)	0.562634	0.387043	0.474838
The Lego Movie (2014)	0.452659	0.206402	0.329531
Guardians of the Galaxy 2 (2017)	0.431004	0.082114	0.256559
Wild, The (2006)	0.425950	-0.410423	0.007763
Antz (1998)	0.425950	0.377384	0.401667
Asterix and the Vikings (Astérix et les Vikings) (2006)	0.425950	-0.242559	0.091695
Adventures of Rocky and Bullwinkle, The (2000)	0.425950	-0.076344	0.174803
The Good Dinosaur (2015)	0.425950	-0.163090	0.131430
Monsters, Inc. (2001)	0.425950	0.579733	0.502842

Figure 4.7 : Output based on Content based RS

b. Shows movie recommendation based on collaborative-based RS.

	content	collaborative	hybrid
Jurassic Park (1993)	0.082496	0.686880	0.384688
Forrest Gump (1994)	0.039534	0.665154	0.352344
Toy Story 2 (1999)	0.767566	0.662620	0.715093
Shrek (2001)	0.255463	0.654041	0.454752
Apollo 13 (1995)	0.066382	0.646312	0.356347
Star Wars: Episode VI - Return of the Jedi (1983)	0.014288	0.640795	0.327541
Star Wars: Episode IV - A New Hope (1977)	-0.006326	0.636537	0.315105
Shawshank Redemption, The (1994)	-0.006718	0.619983	0.306632
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	-0.008436	0.617190	0.304377
Lion King, The (1994)	0.101872	0.617144	0.359508
Pulp Fiction (1994)	0.059598	0.616864	0.338231

Figure 4.8 : Output based on Collaborative based RS

c. Show movie recommendation based on both system content and collaborative i.e. hybrid RS.

	content	collaborative	hybrid
Bug's Life, A (1998)	0.900620	0.561640	0.731130
Toy Story 2 (1999)	0.767566	0.662620	0.715093
Monsters, Inc. (2001)	0.425950	0.579733	0.502842
Up (2009)	0.562634	0.387043	0.474838
Shrek (2001)	0.255463	0.654041	0.454752
Aladdin (1992)	0.229818	0.615551	0.422685
Toy Story 3 (2010)	0.348890	0.460485	0.404688
Who Framed Roger Rabbit? (1988)	0.344131	0.464340	0.404236
Groundhog Day (1993)	0.192450	0.613678	0.403064
Ice Age (2002)	0.373100	0.432997	0.403048
Antz (1998)	0.425950	0.377384	0.401667

Figure 4.9 : Output based on Hybrid based RS

Chapter-5

CONCLUSION

Traditionally, the content based system has been widely used. In order to overcome the issues of scalability and computation, we have tried to develop a user based / memory based cf for our mini project work. The algorithm that we used has been refined by us from time to time according to the sense of parameters and need of dataset. However, we chose to work for small dataset for now due to time and resources limitations. There are many disadvantages that still occurs in our system and we will keep on working to improve that. The uniqueness of our movies is not accountable in our project for now. For content based model, we have worked on genre of movie and its tags. For further improvement, we would like to improve it by including more features like actors, directors, critics reviews etc. additional social media tags will also be helpful if included in the project. We have taken advantage of both content-based and collaborative-based and try to make a more better recommendations for movies.

We have included very less features in our project due to time and resource limitation but there are many ways to improve the project. We can add more features like directors, actors, writers etc. the expand project will obviously, give better recommendations. Different databases can be created on the basis of year of release of movie in order to know if a user likes to watch popular (blockbusters) movies or movies with small business. Depending on the user taste, different ideas can be fit to the model. If user has variable taste, then this might cause over-fitting also. Apart from this, we have tried to develop hybrid model also but that is just a sum total of predictions made by both the systems (content-based and collaborative-based).

We have not included in our project any kind of geographical information about the user. We can ask the question form the user about his tastes, choice or preferences or what he dislikes which will further helps in filtering out the more information. We can also include the information about, for what duration user watches movie which will eventually, helps us in filtering out the movies with long duration (in case he does not liked to watch long duration movies). Further, more direction of research will be to develop the proper method of evaluation. Since we are training more and more complex, it is very important for us to develop a proper method in order to find the best amongst them. There are many methods for selecting features with many evaluation techniques for features and searching methods. Navigation of users from one social media to anther should be taken into considerations.

5.1 Future scope of improvement

There are many ways through which the work presented can be expand in the future with a scope of improvement:

a. The hybrid system can be made more powerful by not just taking the linear combination of both content- based recommendation system and collaborative-based recommendation system.
b. User-based collaborative system gave better performance than content-based so we can do more work on item-based to compare the results of user-based and item-based.
c. This work can be further expanded to use in other fields like recommending books, movies or other items.
d. We have worked on dataset with 100K ratings for now, in future we can increase our memory capacity and can work on dataset with 20M ratings.

References

[1] Cataltepe, Zehra & ULUYAĞMUR, Mahiye & TAYFUR, Esengül. (2016). Feature selection for movie recommendation. TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES. 24. 833-848. 10.3906/elk-1303-189.

[2] Wang, Z., Yu, X., Feng, N. and Wang, Z., 2014. An improved collaborative movie recommendation system using computational intelligence. Journal of Visual Languages & Computing, 25(6), pp.667-675.

[3] Subramaniyaswamy, V., Logesh, R., Chandrashekhar, M., Challa, A. and Vijayakumar, V., 2017. A personalised movie recommendation system based on collaborative filtering. International Journal of High Performance Computing and Networking, 10(1-2), pp.54-63.

[4] Ahn, S. and Shi, C.K., 2009. Exploring movie recommendation system using cultural metadata. In Transactions on Edutainment II (pp. 119-134). Springer, Berlin, Heidelberg.

[5] Jeong, W.H., Kim, S.J., Park, D.S. and Kwak, J., 2013. Performance Improvement of a Movie Recommendation System based on Personal Propensity and Secure Collaborative Filtering. JIPS, 9(1), pp.157-172.

[6] Debnath, S., Ganguly, N. and Mitra, P., 2008, April. Feature weighting in content based recommendation system using social network analysis. In Proceedings of the 17th international conference on World Wide Web (pp. 1041-1042).

[7] Lekakos, G. and Caravelas, P., 2008. A hybrid approach for movie recommendation. Multimedia tools and applications, 36(1-2), pp.55-70.