MOVIE RECOMMENDATION SYSTEM

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

in

Information Technology

by

Mukul Chugh(161468)

Under the supervision of

Dr. Rakesh Kanji

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Student's Declaration

I hereby declare that the work presented in this report entitled "**MOVIE RECOMMENDATION SYSTEM**" in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **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. Rakesh Kanji** (Assistant Professor in the Department of CSE).

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

[illegible]

Mukul Chugh, 161468

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

Rakesh Kanji

Dr. Rakesh Kanji
Assistant Professor (SG)
Department of CSE

Dated: 21/6/2020

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my project guide Dr. Rakesh Kanji for his exemplary guidance , monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him from time to time shall carry me a long way in the journey of life on which I am about to embark.

I am also obliged to staff members of JUIT college , for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Lastly, I thank almighty, my parents ,my classmates for their constant encouragement without which this project would not have been possible.

LIST OF CONTENTS

Student Declaration……………………………………………………………I

Acknowledgement…………………………………………………………..II

List of Contents………………………………………………………………III

List of Figures……………………………………………………………...V

Abstract……………………………………………………………………..VII

Chapter 1-Introduction……………………………………………………1-7

1.1 Introduction……………………………………………………………....1

1.2 Problem Statement………………………………………………………5

1.3 Objective…………………………………………………………………5

1.4 Methodology…………………………………………………………......6

1.5 Cosine Similarity………………………………………………………..7

Chapter 2-Literature Survey………………………………………….....9-11

Chapter 3-System Development……………………………………….....12-19

3.1 Demographic filtering…………………………………………….......12

3.2 Content-based filtering system………………………………………14

3.2.1 Credits, Genres and Keywords Based Recommender……………………………………...……..…15

3.2 Collaborative filtering system……………………………………………17

Chapter 4-Result……………...…………………………………………...20-23

4.1 Dataset………...………………………………………………………...20

4.2 Demographic Filtering…………………………………………………..21

4.3 Content-based filtering system…………………………………………...22

4.4 Collaborative filtering system…...……………………………………...23

Chapter 5-Conclusion...……...…………………………………………...24

References………………………………………………………………25

List of Figures

Fig 1. Product recommendation by Amazon……………………………………...1

Fig 2. Difference between catalogue and recommendations

between (a)Netflix India and (b)Netflix Japan…………………………………4

Fig 3. Graphical Representation of cosine similarity…………………………..7

Fig 4. Code for Demographic Filtering…………………………………………...13

Fig 5. Content Based Filtering………………………………………………………14

Fig 6.1. Credits, Genres and Keywords Based Recommender…………………16

Fig 6.2. Credits, Genres and Keywords Based Recommender…………………16

Fig 7.1. Collaborative filtering system……………………………………………18

Fig 7.2. Collaborative filtering system……………………………………………18

Fig 7.3. Collaborative filtering system……………………………………………19

Fig 8.1. Movie Table……………………………………………………………...20

Fig 8.2. Ratings Table……………………………………………………………..20

Fig 9. Output for Demographic Filtering…………………………………………21

Fig. 10.1. Content-based filtering system using only title and keywords………22

Fig. 10.2. Content-based filtering system using title, cast,

director, keywords and genres………………………………………………22

Fig 11.1. Collaborative Filtering System (Example 1)..23

Fig 11.2. Collaborative Filtering System (Example 2)..23

.

ABSTRACT

A recommendation motor helps to filter the data utilizing various algorithms and suggests the users most relevant items. It analyses an user's previous activities and gives suggestions that the consumers are prone to buy based on it. In case a new visitor with no previous user history visits the website this will not be able to recommend him products.In that case it will recommend most popular items or most profitable items for the business to the user.

Our recommendation system primarily uses three techniques:

1. Demographic filtering :
   - This program suggests the same films to users with similar backgrounds.
2. Content-Based filtering :
   - It aims to profile user preferences using collected information, and suggested products based on profile.
3. Collaborative filtering :
   - This attempt to band similar users together and utilize community comprehension to provide suggestions.

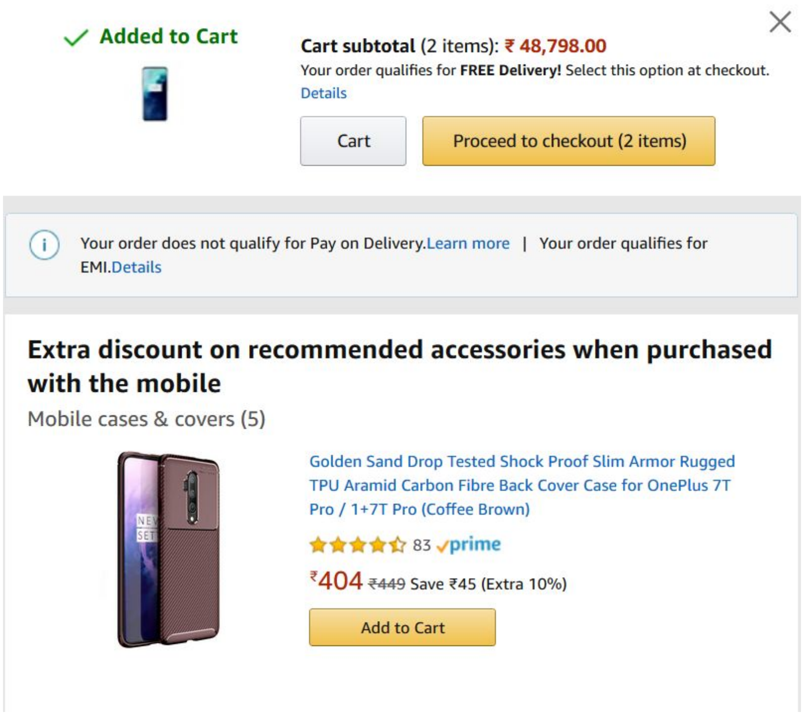
This project utilises Movie Lens dataset which comprises of 9000 movies with 3,600 tags and 100,000 ratings given by 600 users.

CHAPTER 1

INTRODUCTION

1.1 Introduction:

A recommendation model is a sort of information filtering framework that seeks to evaluate a consumer's preferences, and offer suggestions that are supported by those preferences. There are plenty of recommendation system implementations. They are becoming more popular in modern times and are being used in almost every online service that we are using today. The composition of these websites ranges among movies, videos, books and music and social media friends, stories and e-commerce products, as well as individuals on professional websites . These systems can also gather information about the user 's interests, and can use this data in the future to enhance their recommendations. For example, if Amazon witnesses that a significant number of people ordering mobile phones often order phone covers, hence they may advise the phone cover to a the customer who placed a phone in their cart.



Added to Cart

Cart subtotal (2 items): ₹ 48,798.00
Your order qualifies for FREE Delivery! Select this option at checkout.
Details

Cart | Proceed to checkout (2 items)

Your order does not qualify for Pay on Delivery.Learn more | Your order qualifies for EMI.Details

Extra discount on recommended accessories when purchased with the mobile
Mobile cases & covers (5)

Golden Sand Drop Tested Shock Proof Slim Armor Rugged TPU Aramid Carbon Fibre Back Cover Case for OnePlus 7T Pro / 1+7T Pro (Coffee Brown)
83 prime
₹404 ~~₹449~~ Save ₹45 (Extra 10%)

Add to Cart

Fig 1. Product recommendation by Amazon.

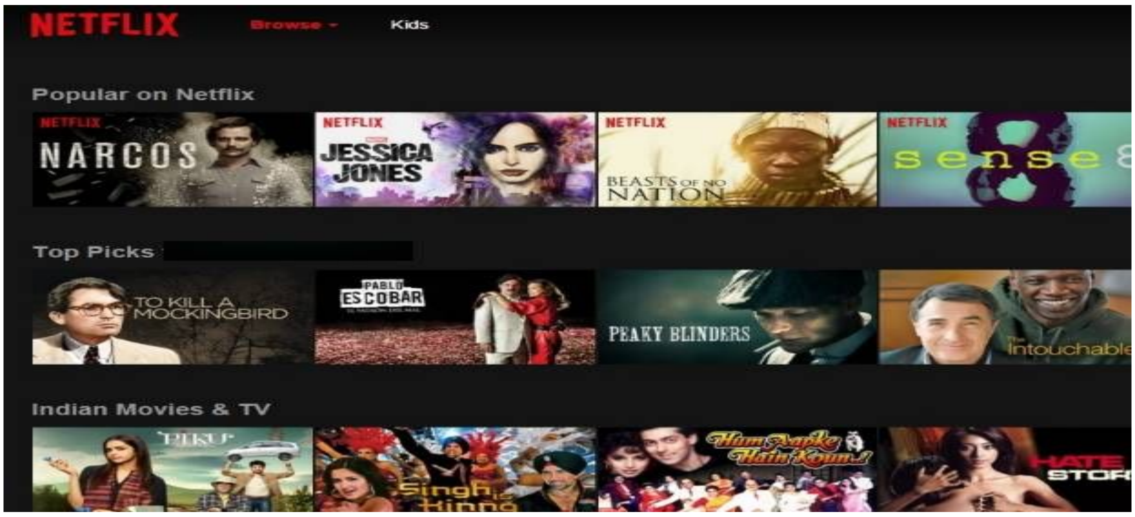
For example, There are only so many news articles in newspaper but when you go online you can see the rest of the news articles the less popular news articles that are To the right the piece of the curve which is to the portion of the curve which is to the right of this segregating area is called the long tail these are the objects which are only accessible online. all these items that could now be found in a physical store but there can be only found online but there are so many of them that it's very hard for any user to find all these items right so when you have this area of abundance and therefore you have so many items and many of them are only found online how you know how do you introduce a user to all these new items that they may not find otherwise. When you have more choice like this ie. when you have millions and millions of items that are only available online you need a better way for the user to find all these items the user doesn't even know where to start looking. It is where recommendations algorithms come in order to provide better access to desired products .

If you visit youtube, you will see the most popular videos. These are simple aggregates that keep user activity into consideration in making suggestions to several other users, however these recommendations are not dependent on individual users, they depend on the aggregate activity of many other users. Another interesting form of recommendation is one that is personalized to specific users. For instance, book recommendations suited to your liking or film recommendations based based on previously watched films or music recommendations based on your music interests. The key issue in the recommender systems is to assume these unknown values, let us assume , for example, that Alice has rated avatar and matrix but has not evaluated Lord of the Rings, so the question is whether Alice's rating for Lord of the Rings will be based on her other ratings or whether she would like pirates or not. For each user certain movies that they would have rated highly or with the system thinks they might have rated higher than we can recommend those movies to those users so you have to tackle the second problem by extrapolating unknown ratings from known ratings. But we're primarily interested in the high unknown ratings that we're interested in in those ratings where a user would have given a high rating to a film that we're not interested in on average or low ratings because they're never going to recommend those films to the user and finally the third key issue is evaluating them. Let's start with the first issue that collecting data is the first and simplest way to collect ratings, called explicit process. If users are asked to rate items now because items are rated directly by users and will get to know them, you can determine on which measure consumers will rate items. Therefore, the explicit approach does have the obvious benefit of being simple and having proper feedback from consumers. However the main concern is that it doesn't just operate on a tiny portion of users who watched a movie or listened to it who cared to review the service or leave feedback . Even though most users do not voluntarily leave ratings or reviews so that the information is not gathered explicitly. For feedback since only a tiny proportion of users leave reviews and feedback because explicit ratings don't scale, many sites use implicit approach now the concept of implicit ratings is to study user ratings from other behavioural patterns.

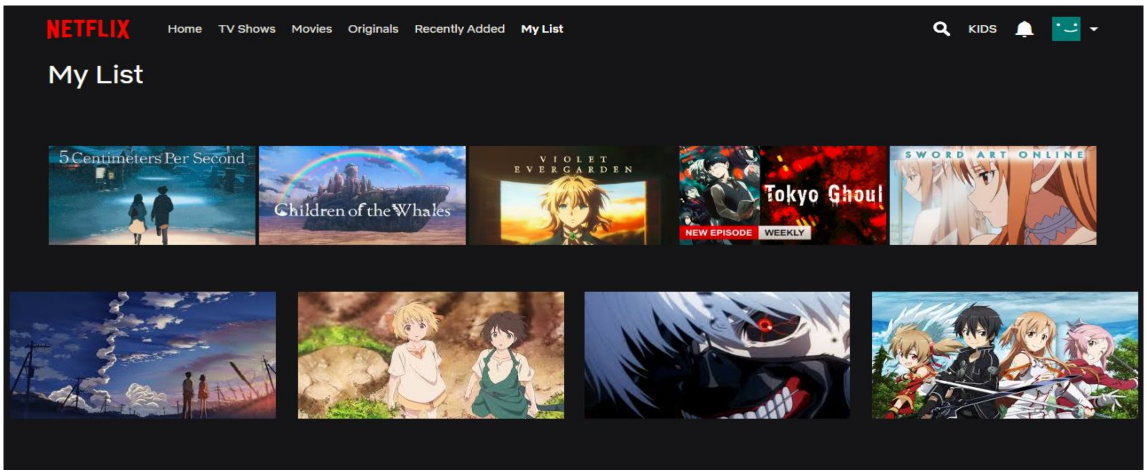
An online shopping platform may create a policy which implies a purchase to be a positive score now that the nice thing about implicit scores is how flexible they are as compared to explicit scores as the consumer doesn't need to directly rate a product as several other acts, including transactions may determine the ratings. The problem is that it's very difficult to use implicit scores to determine negative ratings. In practice, most recommended systems and most websites use a mix of explicit or implied ratings where ratings are clearly available that they use but complement them with implied ratings if appropriate, let's move on to the central problem of extrapolating utilities or extrapolating unknown utilities from known utility values, the primary or central problem that we must overcome. The second problem we've got is a cold start challenge where there is a new item or a new user, the new item won't have reviews and new users won't have a history, it's called a cold start issue and we've been struggling with it.

In 2009, Netflix held a competition in which they awarded the prize money of 1 million dollars for just 10% of improvement to their existing recommendation algorithm. Although sources suggest that they never got into production, the ideas and techniques designed and developed during this competition are still considered to be the greatest strides in the field of recommendation systems.

The need for such systems arises because of less amount of shelf space. Imagine that you are out shopping 20 years ago and you'd go to a local retailer and you'll find a certain quantity of items on the cabinets of the local retailer now even in a very large retailer like a Walmart for instance shelf space is a key it's a scarce commodity it restrict the range of items a store can hold on shelf space since it involves rental costs and thus a store can hold only a certain number of items. Similar problems arise when licensing shows/movies by platforms. Platforms will want to provide popular,highly watched movies and would like to cut cost by not providing unpopular or scarcely watched movies. Also time as a consumer is also important and they would like to spend it doing things they enjoy. The platform also wants to give the user the best experience as it would ensure user loyalty and increase brand value in the market. Also for different demographics a service needs a different catalogue. For instance Netflix provides different content based on the country.



(a)



(b)

Fig 2. Difference between catalogue and recommendations between (a)Netflix India and (b)Netflix Japan

1.2 Problem Statement:

Recommendation systems are software tools or methods for extracting information which provide a consumer with suggestions for products. The products could be music, books, films, people or groups. The goal of this system is to suggest products which a client could take an interest in, and to understand the users ' preferences and constraints. The project aims to develop recommendation engine and generate precise results for significantly large users and movie applications.

1.3 Objective:

The objective of the project was to recommend similar movies to users as per their taste and the taste of other people like them. Which is mainly used in Netflix and amazon prime videos. Also used by youtube for recommending us the best video as per user's taste.

1.4 Methodology:

Our recommender systems employ three main approaches.

First one is Demographic filtering i.e this offers generalized recommendations for each user, depending on the popularity of a film and/or genre. The program recomends the same movie for users with common demographic information. For example it will show top rated bollywood movies for people of India and for people in France it will return french cenema. This approach is termed as too basic as every user may have different preferences. The basic objective of this model is that highly popular and widely lauded movies will have a better likelihood of being enjoyed by the average audience.

Second, is filtering based on content, where we attempt to classify the preferences of consumers utilising the collected information, and suggest items on basis of the suggested profile. These profiles are usually based on a genre, similar plot, worlds, preference of directors/cast etc. They depend on the characteristics and qualities of the element itself, so if, for instance, the program has to suggest films to a customer, what the program could do is look at films that they enjoyed in the past and find similar films based on the film's director, the actors who were in the film. Film tags are the qualities that other reviewers gave it to the category, genre, etc.

The other is collaborative filtering, where identical consumers are clustered together, and using the community knowledge suggests the movies to the consumers . A collaborative filtering method is based mostly on behavior and Information about other users who encountered or purchased these products previously ie. using ratings provided by other users in a movie recommendation engine. For instance if a person likes action movies the system will retrieve most highly rated movies from others who like action movies.

1.5 Cosine Similarity:

Cosine similarity is a method of calculating similarity between documents regardless of size. Logically, the angle cosine between two data points represented in a multidimensional space is calculated. The cosine similarity is helpful as even if the multiple similar documents are distant by the measure of Euclidean distance ,they can still be quite closely oriented.

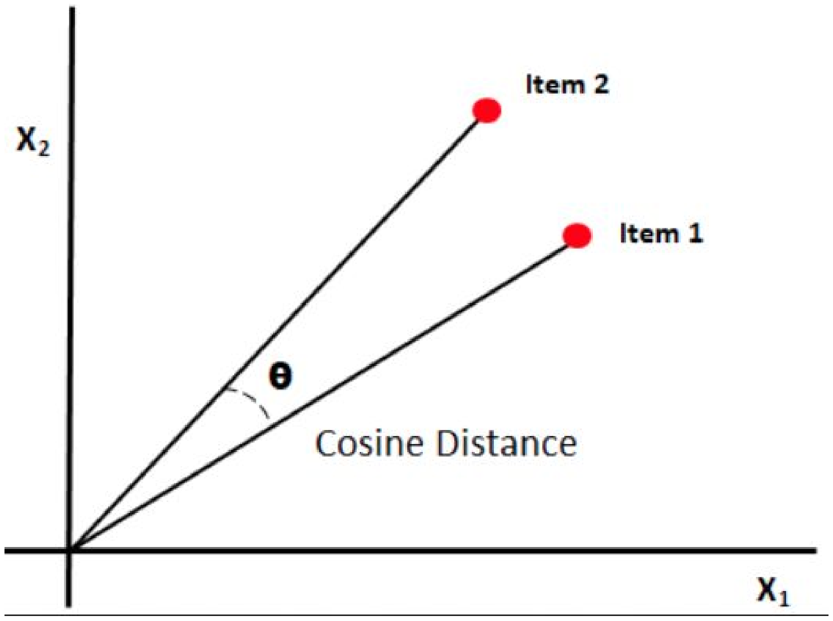


Fig 3. Graphical Representation of cosine similarity

A commonly utilized strategy for comparing similar documents is derived on counting as many common words as possible between the documents.

It is advantageous than using euclidean distance as this gives a better perspective of similarity as euclidean distance is dependent on the dimensions and hence two similar documents would be described as the same.

Cosine Similarity can be calculated using the following equation:

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

However this technique has a fundamental problem. That is, as the size of the document increases , the number of common phrases appears to rise, even though the documents refer to different topics.

CHAPTER 2

LITERATURE SURVEY

In this chapter, we have looked into different research papers that have worked on recommendation systems.

MOVIEMENDER by **Rupali Hande, Ajinkya Gutti, Kevin Shah, Jeet Gandhi, Vrushal Kamtikar:** In current technological era, where there is an wide range of content to be consumed, finding the content of one's preference seems to be an infuriating challenge. On the other hand , digital content providers aim to reach however many consumers as possible over the longest possible period of time on their platform. They proposed MovieMender, a film recommender program,. Typically, the purpose of a basic recommendation system is to consider one of the listed aspects for recommendation generation; user preference or equivalent user preference.

MOVREC is a **D.K.Yadav et al** implemented movie recommendation application bases on colbrative filtering technique. Collaborative filtering utilizes user-provided relevant data. This information is assessed and a film is suggested to consumers sorted by the rating with the highest-rated movie being first. The program also has a provision that allows users to select characteristics from which they intend to recommend the movie.

Luis M Capos et al examined two conventional recomendation systems ie. content-based filtering and collaborative filtering systems. Since both had their own drawbacks he formulated an innovative model combining Bayesian network and collaborative filtration.

Harpreet Kaur et al had enacted a composite system. A combination of content filter along with collaborative filter techniques is used on this system. This also takes into account the context of each film, while suggesting it. Relationships between user and user as well as relationships between user and element are a significant part of recommendation.

Utakarsh Gupta et al shaped a cluster, with the aid of chameleon, from user or item-specific information. This is an effective hierarchical clustering technique recommendation . A product voting mechanism is utilized to determine ranking.

The proposed solution has less errors as well as effective clustering of related objects.

Costin-Gabriel Chiru et al. devlopeded Movie Recommender system , a program that utilises the gathered user information for film recommendations. This framework tries to deal with the problem of uinque suggestions arising from data ignorance. The user's psychological profile, their watch history and the information it collected involving film ratings from other websites is based on aggregate similarity calculation. This is a composite model that uses filtering based on both content and collaborative filtering.

To estimate the degree of complexity of each case for every trainee **Hongli LIn et al.** devised a Content-Boosted Collaborative Filtering (CBCF) approach. It operates in two phases: Initially, content-based filtering which enhances the current trainee instance, following which the final predictions are provided by rating data and collaborative filters. The CBCF algorithm incorporates the strengths of both CBF and CF, and thereby addressing all of their drawbacks at the same time.

Xiu Li et al. describes an accurate framework of recommendations using opinion mining. They addressed current recommender systems which are grouped into three categories: collaborative filtering, content based filtering and filtering based on knowledge. The adjacent customers explore innitally in the collaborative filtering, and then provide a set of suggestions. Collaborative filtering does not however recognize certain product or service-related attributes. The product features have been extracted in content-based recommendation, and it generates relevant features. It then creates feature vectors. This technique suffers from the limitation of not taking into account users behavior or activity for the product. This leads to inaccuracies during the recommendations. Users propose the demand first in knowledge-based recommendation, and the entire process is strongly interactive.

Jenq-Haur et al. presented a sentiment rating scheme in order to find the exact rating of a movie. For that reason, they have facilitated sentiment lexicons adjustment, which improves the classification accuracy.

Filipa et al. presented a new recommender system which integrates movie ratings and unrated reviews on the web. Sentiment analysis was incorporated to offer user preferences analysis where the reviews were not associated with an explicit rating. A recommendation algorithm was proposed which performs based on matrix factorization with singular value decomposition (SVD).

Davide et al. presented a sentiment-based approach to recommendation of Twitter users. Sentiment-Volume-Objectivity (SVO) function is the weighting function used in this sentiment-based approach. It considers user attitude and also his/her reviews about the product or service. This weighting function aims to construct richer user profiles to apply in the recommendation application, but it leads to lack of sentiment analysis and provides relatively poor recommendation to users.

CHAPTER 3

SYSTEM DEVELOPMENT

There are different kinds of recommender systems with various strategies and some of them are defined as:

3.1. Demographic filtering- They provide generalized suggestions based On the popularity of movies and/or genres. Same movies are suggested to users with similar demographics in this system. Since every individual is unique, this approach is regaurded as too basic. The fundamental concept behind this model is the films that are more popular and/or critically praised would be more likely to be enjoyed by the average viewer.

Before moving on to this-

● A rating metric is required to score/rate films

● Compute the rating for every single film

● Order the rating to suggest users the best-suited movie.

We may use the film's average ratings as the score, but by using that it will be unfair since a film with an average rating of 8.9 with only 3 votes can not be regarded as better than the film with an avg rating of 8.1 with rating 40 votes. Therefore weighted ranking from IMDB is used (Wr).

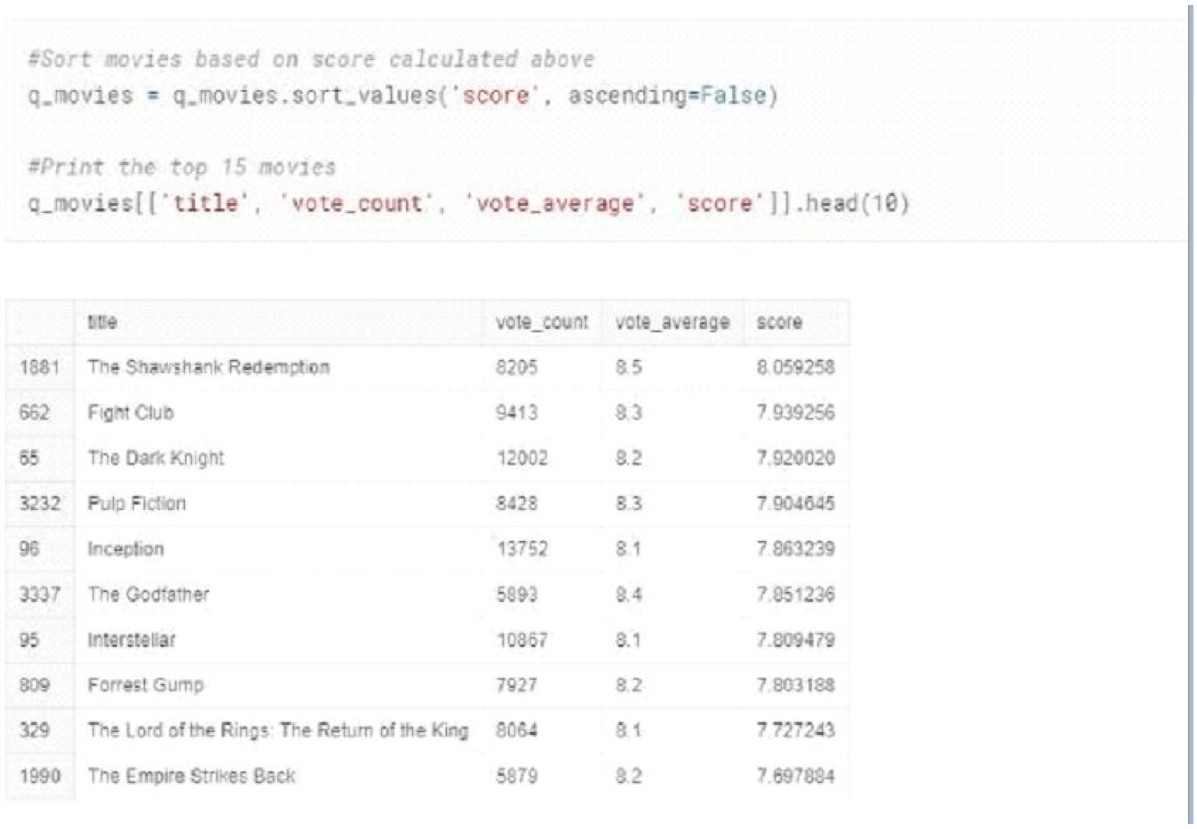
Which is calculated as:-

$$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m} \cdot R\right) + \left(\frac{m}{v+m} \cdot C\right)$$

where,

- v --> no. of ratings for the movie;
- m --> threshold required to be listed in the chart;

- R --> avg. rating of the movie;
- C --> mean vote for the entire dataset



```
#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)

#Print the top 15 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884

Fig 4. Code for Demographic Filtering

3.2. Content-based filtering systems- Items are recommended in content-based filtering on the basis of comparisons between profile of products and users. The user profile created is content Keywords (or features) were found to be important to the user. The profile could be viewed as a group of allocated key phrases Retrieved by algorithm from those in the identified items which are relevant / intriguing to the user.

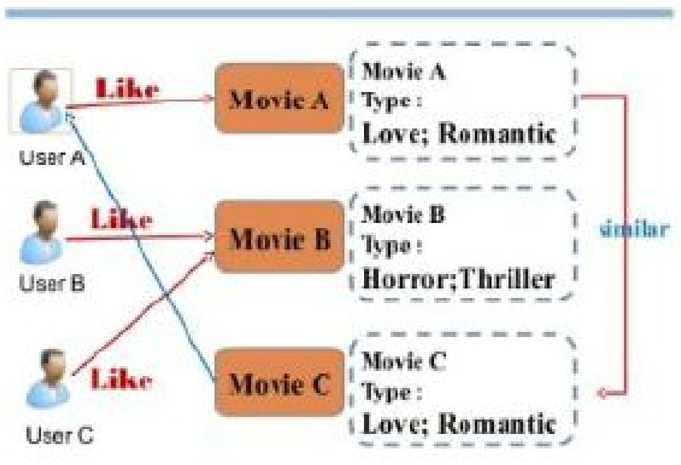


Fig 5. Content Based Filtering

The cosine similarity is used to compute a numeric value which signifies the similarity of two films. The cosine similarity score is used as it is magnitude-independent and is fairly simple and fast to measure. It is calculated as shown below:

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

Now that we know our recommendation function we will follow the following steps:

● Retrieve the movie index provided the title.

● Calculate the cosine similarity score of the desired movie and all the others in the dataset. This is now turned into a list of lists ie. list of a list of two elements first being the index of the movie and second being the score.

● The list of lists created is sorted on the basis of similarity scores.

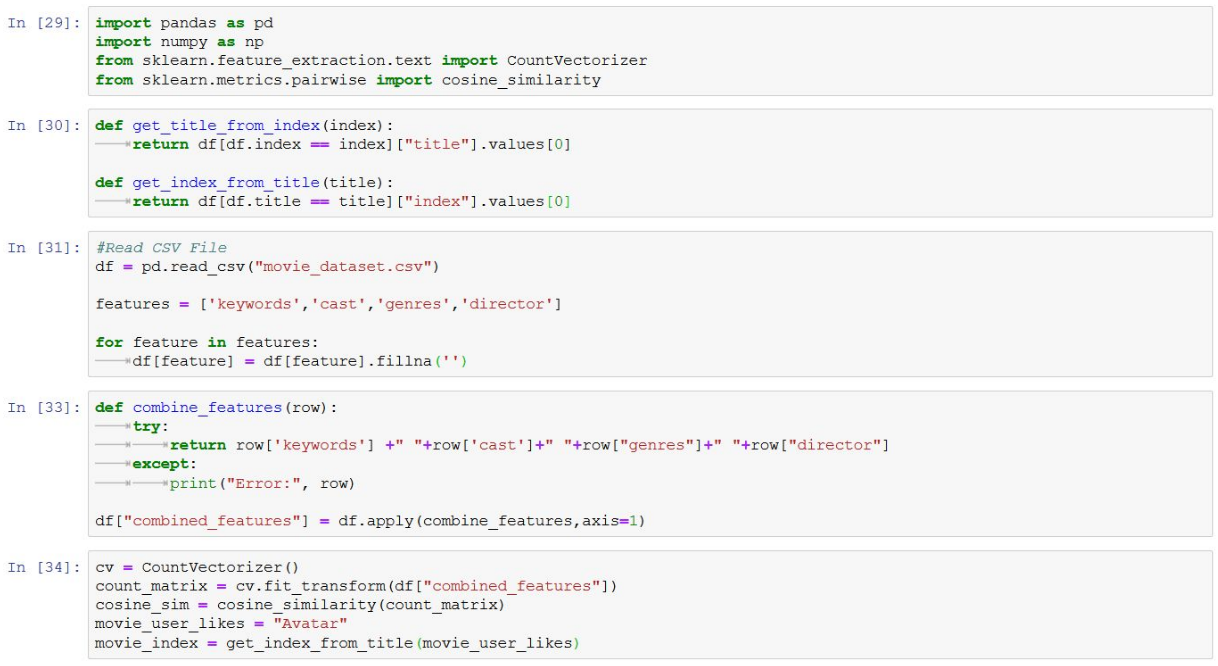
● Top 10 elements are retrieved ignoring the first element because it is the input movie itself.

● Return the corresponding titles to the indices of the top elements.

While our program did a fine job of identifying films with similar storyline , the reliability of the suggestions isn't perfect. For instance "The Dark Knight Rises" restores all the Batman movies whereas other Christopher Nolan movies are much more probable to be enjoyed by the people who enjoyed that movie. It is something which can not be handled by the current system.

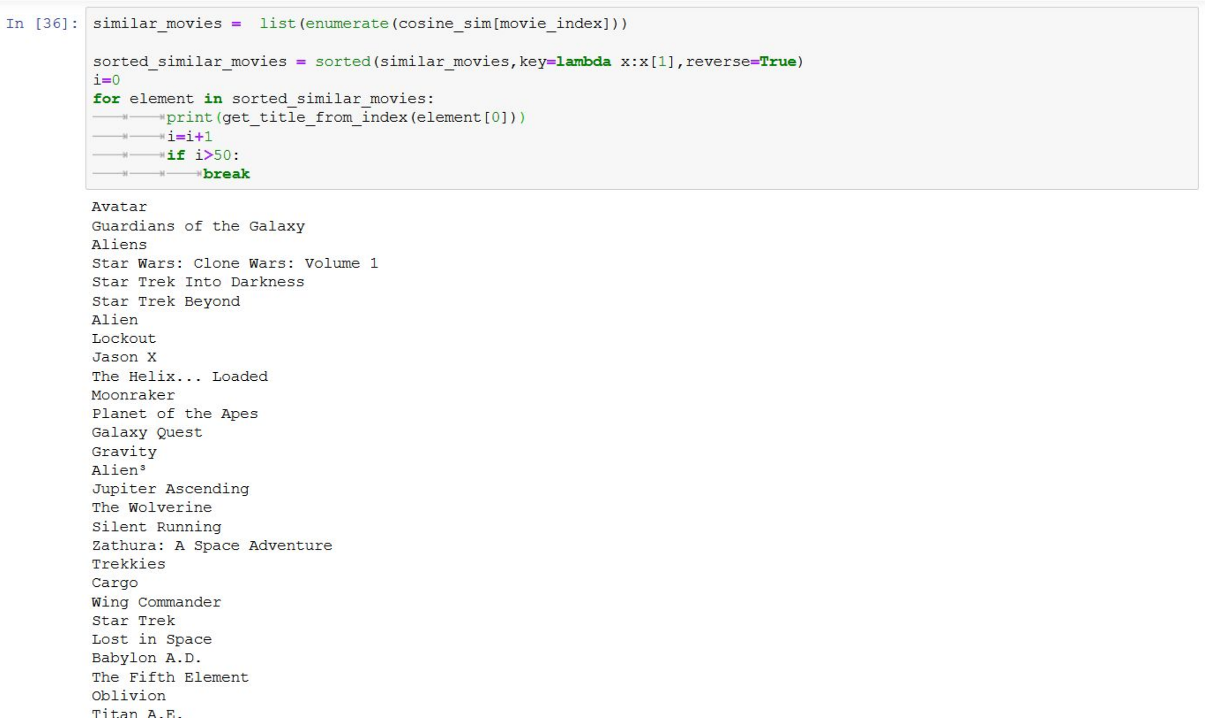
3.2.1. Genres and Keywords Based Recommender

The recommender's efficiency will improve with better use of metadata. This is precisely what we will do in this portion. We will create a recommender depending on the relevant metadata, the cast, the director, related genres, and the keywords.



```
In [29]: import pandas as pd
         import numpy as np
         from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.metrics.pairwise import cosine_similarity

In [30]: def get_title_from_index(index):
             return df[df.index == index]["title"].values[0]

         def get_index_from_title(title):
             return df[df.title == title]["index"].values[0]

In [31]: #Read CSV File
         df = pd.read_csv("movie_dataset.csv")

         features = ['keywords','cast','genres','director']

         for feature in features:
             df[feature] = df[feature].fillna('')

In [33]: def combine_features(row):
             try:
                 return row['keywords'] +" "+row['cast']+" "+row["genres"]+" "+row["director"]
             except:
                 print("Error:", row)

         df["combined_features"] = df.apply(combine_features,axis=1)

In [34]: cv = CountVectorizer()
         count_matrix = cv.fit_transform(df["combined_features"])
         cosine_sim = cosine_similarity(count_matrix)
         movie_user_likes = "Avatar"
         movie_index = get_index_from_title(movie_user_likes)
```

Fig 6.1. Credits, Genres and Keywords Based Recommender



```
In [36]: similar_movies =  list(enumerate(cosine_sim[movie_index]))

         sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)
         i=0
         for element in sorted_similar_movies:
                 print(get_title_from_index(element[0]))
                 i=i+1
                 if i>50:
                     break

         Avatar
         Guardians of the Galaxy
         Aliens
         Star Wars: Clone Wars: Volume 1
         Star Trek Into Darkness
         Star Trek Beyond
         Alien
         Lockout
         Jason X
         The Helix... Loaded
         Moonraker
         Planet of the Apes
         Galaxy Quest
         Gravity
         Alien³
         Jupiter Ascending
         The Wolverine
         Silent Running
         Zathura: A Space Adventure
         Trekkies
         Cargo
         Wing Commander
         Star Trek
         Lost in Space
         Babylon A.D.
         The Fifth Element
         Oblivion
         Titan A.E.
```

Fig 6.2. Credits, Genres and Keywords Based Recommender

The advantages of filtering based on content are:

● They are able to recommend unrated products.

● By listing the content features of an object we can easily clarify the operation of the recommender system.

● Recommenders systems based on content need only the user's information, and don't require information regarding any other user of the system.

The drawbacks of content-based filtering include:

● this will not operate for a new user who has not yet rated any item ,as appropriate ratings are needed content based recommender analyses to offer specific recommendations.

● No Serendipitous Items are recommended.

● Limited Content Analysis- The recommendation will not operate if the system is not able to distinguish between the items that a user likes and those that he does not like.

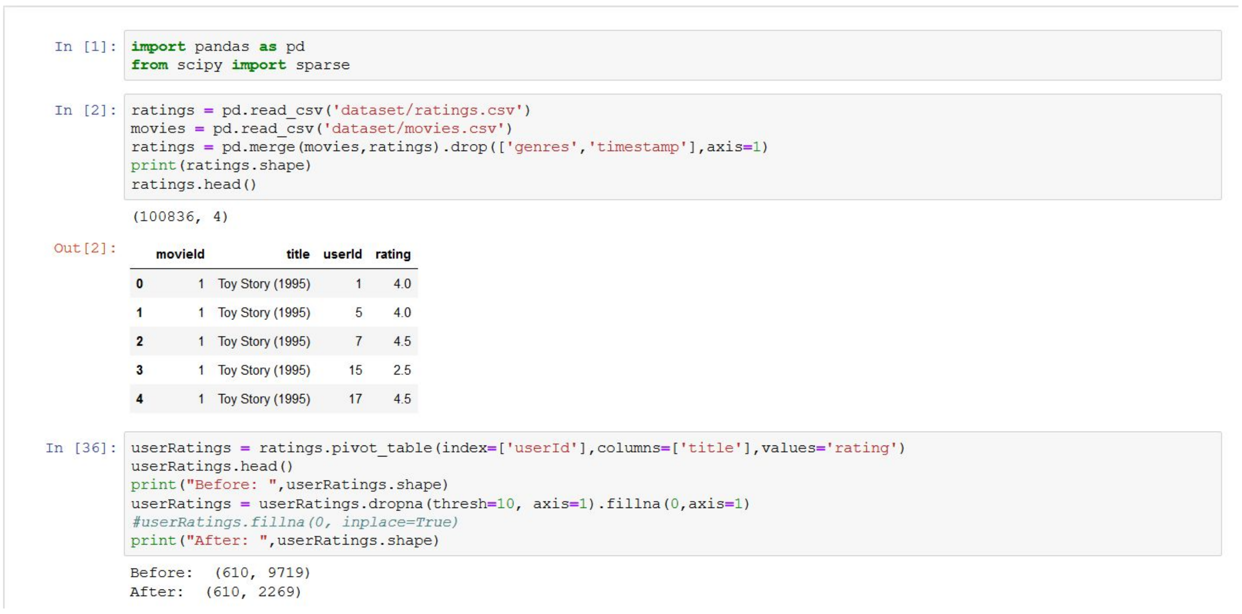
3.3. Collaborative filtering systems- Some of our content-based motors suffer strict limitations. It can only suggest films which are close to a specific film. This means, it can not classify the tastes and offer suggestions spanning different genres. The engine that has been built isn't really personal, because it wouldn't capture the tastes and preconceptions of the individual. Anybody who queries our engine for suggestions based on a movie receives the same recommendations for the film regardless of who it is. So in this section, we will use a process called Collaborative Filtering to provide the Movie Watchers with better suggestions.

Collaborative filtering framework recommends products based on Consumer and/or object similarity measurements. The system recommends those items which similar types of consumers prefer. Collaborative filtering offers several benefits

1. It is independent of content, i.e. it relies only on connections

2. Since people make explicit scores, so real evaluation of the quality of the items is done.

3. It gives serendipitous recommendations since suggestions were dependent on user similarity, rather than on object similarity.



```
In [1]: import pandas as pd
        from scipy import sparse
```

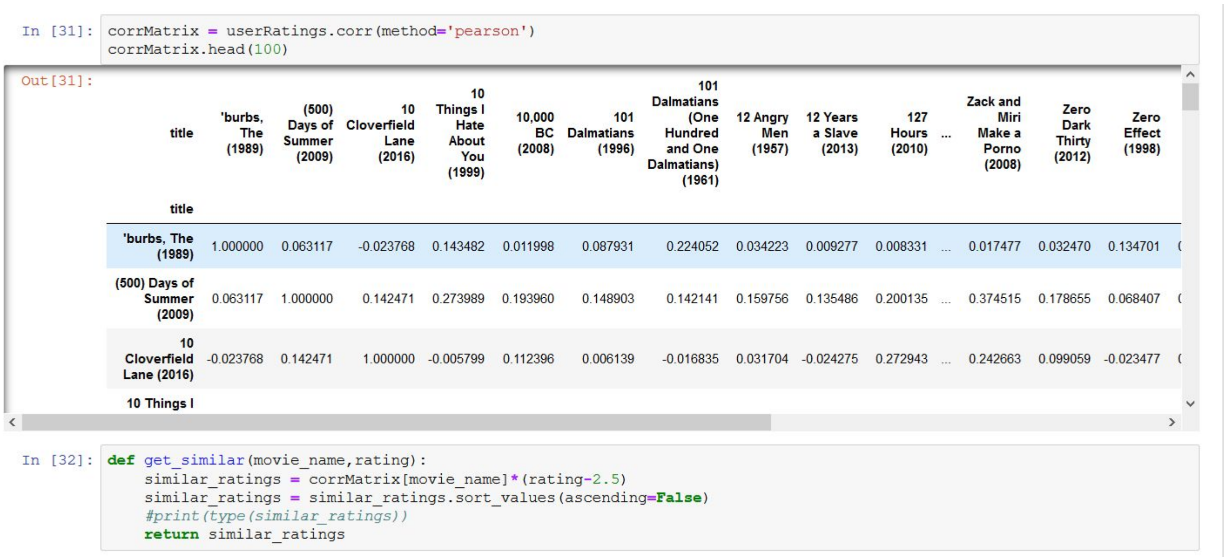
```
In [2]: ratings = pd.read_csv('dataset/ratings.csv')
        movies = pd.read_csv('dataset/movies.csv')
        ratings = pd.merge(movies,ratings).drop(['genres','timestamp'],axis=1)
        print(ratings.shape)
        ratings.head()

        (100836, 4)
```

Out[2]:

	movieId	title	userId	rating
0	1	Toy Story (1995)	1	4.0
1	1	Toy Story (1995)	5	4.0
2	1	Toy Story (1995)	7	4.5
3	1	Toy Story (1995)	15	2.5
4	1	Toy Story (1995)	17	4.5

```
In [36]: userRatings = ratings.pivot_table(index=['userId'],columns=['title'],values='rating')
         userRatings.head()
         print("Before: ",userRatings.shape)
         userRatings = userRatings.dropna(thresh=10, axis=1).fillna(0,axis=1)
         #userRatings.fillna(0, inplace=True)
         print("After: ",userRatings.shape)

         Before:  (610, 9719)
         After:  (610, 2269)
```

Fig 7.1. Collaborative filtering system



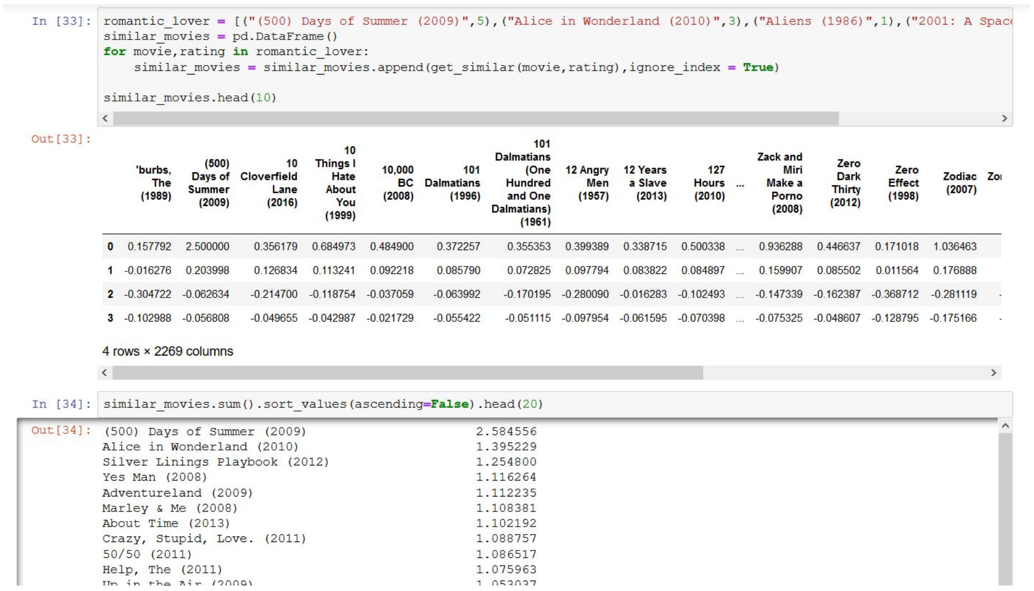
```
In [31]: corrMatrix = userRatings.corr(method='pearson')
         corrMatrix.head(100)
```

Out[31]:

title	'burbs, The (1989)	(500) Days of Summer (2009)	10 Cloverfield Lane (2016)	10 Things I Hate About You (1999)	10,000 BC (2008)	101 Dalmatians (1996)	101 Dalmatians (One Hundred and One Dalmatians) (1961)	12 Angry Men (1957)	12 Years a Slave (2013)	127 Hours (2010)	...	Zack and Miri Make a Porno (2008)	Zero Dark Thirty (2012)	Zero Effect (1998)
title														
'burbs, The (1989)	1.000000	0.063117	-0.023768	0.143482	0.011998	0.087931	0.224052	0.034223	0.009277	0.008331	...	0.017477	0.032470	0.134701
(500) Days of Summer (2009)	0.063117	1.000000	0.142471	0.273989	0.193960	0.148903	0.142141	0.159756	0.135486	0.200135	...	0.374515	0.178655	0.068407
10 Cloverfield Lane (2016)	-0.023768	0.142471	1.000000	-0.005799	0.112396	0.006139	-0.016835	0.031704	-0.024275	0.272943	...	0.242663	0.099059	-0.023477
10 Things I														

```
In [32]: 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
```

Fig 7.2. Collaborative filtering system



```
In [33]: romantic_lover = [("(500) Days of Summer (2009)",5),("Alice in Wonderland (2010)",3),("Aliens (1986)",1),("2001: A Spac
         similar_movies = pd.DataFrame()
         for movie,rating in romantic_lover:
             similar_movies = similar_movies.append(get_similar(movie,rating),ignore_index = True)

         similar_movies.head(10)
```

Out[33]:

	'burbs, The (1989)	(500) Days of Summer (2009)	10 Cloverfield Lane (2016)	10 Things I Hate About You (1999)	10,000 BC (2008)	101 Dalmatians (1996)	101 Dalmatians (One Hundred and One Dalmatians) (1961)	12 Angry Men (1957)	12 Years a Slave (2013)	127 Hours (2010)	...	Zack and Miri Make a Porno (2008)	Zero Dark Thirty (2012)	Zero Effect (1998)	Zodiac (2007)	Zo
0	0.157792	2.500000	0.356179	0.684973	0.484900	0.372257	0.355353	0.399389	0.338715	0.500338	...	0.936288	0.446637	0.171018	1.036463	
1	-0.016276	0.203998	0.126834	0.113241	0.092218	0.085790	0.072825	0.097794	0.083822	0.084897	...	0.159907	0.085502	0.011564	0.176888	
2	-0.304722	-0.062634	-0.214700	-0.118754	-0.037059	-0.063992	-0.170195	-0.280090	-0.016283	-0.102493	...	-0.147339	-0.162387	-0.368712	-0.281119	-
3	-0.102988	-0.056808	-0.049655	-0.042987	-0.021729	-0.055422	-0.051115	-0.097954	-0.061595	-0.070398	...	-0.075325	-0.048607	-0.128795	-0.175166	-

4 rows × 2269 columns

```
In [34]: similar_movies.sum().sort_values(ascending=False).head(20)
```

```
Out[34]: (500) Days of Summer (2009)                        2.584556
         Alice in Wonderland (2010)                         1.395229
         Silver Linings Playbook (2012)                     1.254800
         Yes Man (2008)                                     1.116264
         Adventureland (2009)                               1.112235
         Marley & Me (2008)                                 1.108381
         About Time (2013)                                  1.102192
         Crazy, Stupid, Love. (2011)                        1.088757
         50/50 (2011)                                       1.086517
         Help, The (2011)                                   1.075963
```

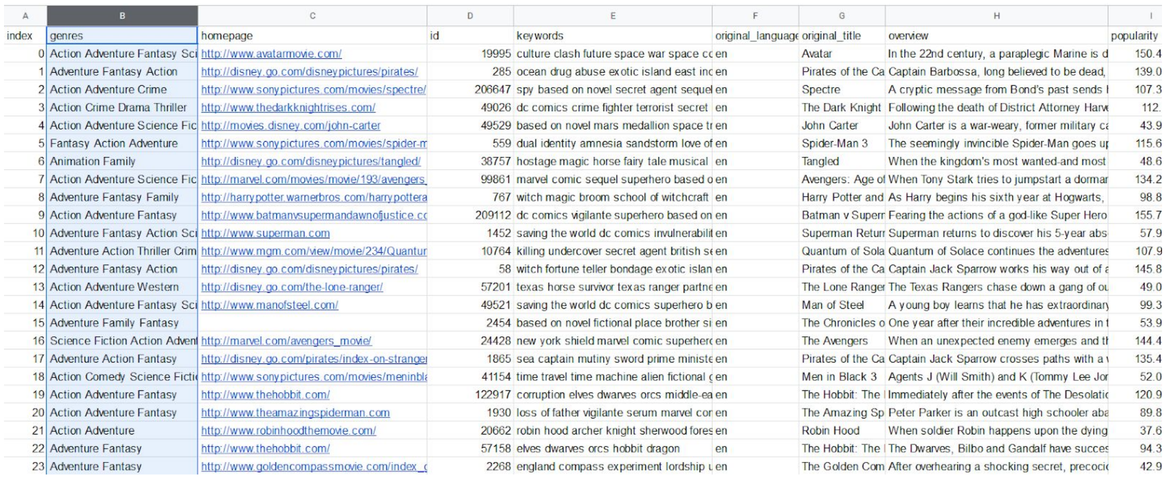
Fig 7.3. Collaborative filtering system

CHAPTER 4

RESULT

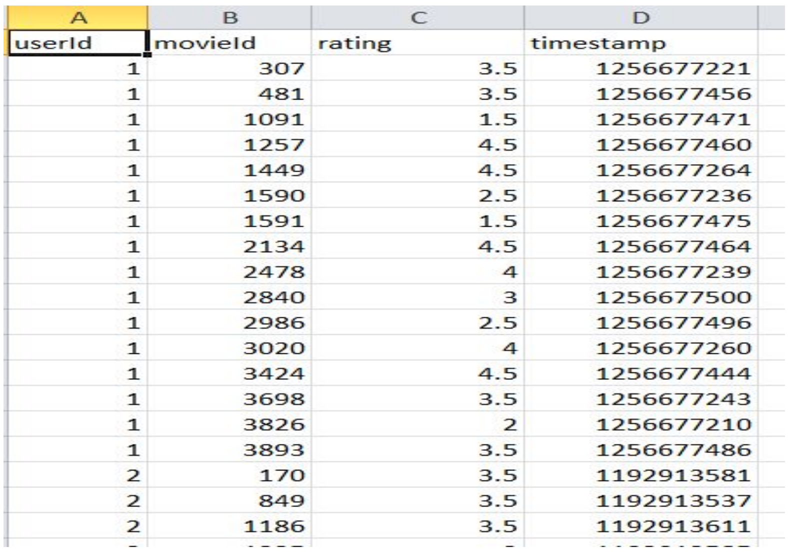
4.1 Dataset:

This project uses Movie Lens dataset which comprises of 9000 movies with 3,600 tags and 100,000 ratings given by 600 users.



A	B	C	D	E	F	G	H	I
index	genres	homepage	id	keywords	original_language	original_title	overview	popularity
0	Action Adventure Fantasy Sci	http://www.avatarmovie.com/	19995	culture clash future space war space cc	en	Avatar	In the 22nd century, a paraplegic Marine is d	150.4
1	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east inc	en	Pirates of the Ca	Captain Barbossa, long believed to be dead,	139.0
2	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel	en	Spectre	A cryptic message from Bond's past sends I	107.3
3	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc comics crime fighter terrorist secret	en	The Dark Knight	Following the death of District Attorney Harve	112.
4	Action Adventure Science Fic	http://movies.disney.com/john-carter	49529	based on novel mars medallion space tr	en	John Carter	John Carter is a war-weary, former military ca	43.9
5	Fantasy Action Adventure	http://www.sonypictures.com/movies/spider-m	559	dual identity amnesia sandstorm love of	en	Spider-Man 3	The seemingly invincible Spider-Man goes up	115.6
6	Animation Family	http://disney.go.com/disneypictures/tangled/	38757	hostage magic horse fairy tale musical	en	Tangled	When the kingdom's most wanted-and most	48.6
7	Action Adventure Science Fic	http://marvel.com/movies/movie/193/avengers_	99861	marvel comic sequel superhero based o	en	Avengers: Age of	When Tony Stark tries to jumpstart a dormar	134.2
8	Adventure Fantasy Family	http://harrypotter.warnerbros.com/harrypottera	767	witch magic broom school of witchcraft	en	Harry Potter and	As Harry begins his sixth year at Hogwarts,	98.8
9	Action Adventure Fantasy	http://www.batmanvsupermandawnofjustice.cc	209112	dc comics vigilante superhero based on	en	Batman v Superr	Fearing the actions of a god-like Super Hero	155.7
10	Adventure Fantasy Action Sci	http://www.superman.com	1452	saving the world dc comics invulnerabilit	en	Superman Returr	Superman returns to discover his 5-year abs	57.9
11	Adventure Action Thriller Crim	http://www.mgm.com/view/movie/234/Quantur	10764	killing undercover secret agent british se	en	Quantum of Sola	Quantum of Solace continues the adventures	107.9
12	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	58	witch fortune teller bondage exotic islan	en	Pirates of the Ca	Captain Jack Sparrow works his way out of a	145.8
13	Action Adventure Western	http://disney.go.com/the-lone-ranger/	57201	texas horse survivor texas ranger partne	en	The Lone Ranger	The Texas Rangers chase down a gang of ou	49.0
14	Action Adventure Fantasy Sci	http://www.manofsteel.com/	49521	saving the world dc comics superhero b	en	Man of Steel	A young boy learns that he has extraordinary	99.3
15	Adventure Family Fantasy		2454	based on novel fictional place brother si	en	The Chronicles o	One year after their incredible adventures in t	53.9
16	Science Fiction Action Advent	http://marvel.com/avengers_movie/	24428	new york shield marvel comic superherc	en	The Avengers	When an unexpected enemy emerges and th	144.4
17	Adventure Action Fantasy	http://disney.go.com/pirates/index-on-strange	1865	sea captain mutiny sword prime ministe	en	Pirates of the Ca	Captain Jack Sparrow crosses paths with a v	135.4
18	Action Comedy Science Ficti	http://www.sonypictures.com/movies/meninbla	41154	time travel time machine alien fictional (	en	Men in Black 3	Agents J (Will Smith) and K (Tommy Lee Jor	52.0
19	Action Adventure Fantasy	http://www.thehobbit.com/	122917	corruption elves dwarves orcs middle-ea	en	The Hobbit: The I	Immediately after the events of The Desolatic	120.9
20	Action Adventure Fantasy	http://www.theamazingspiderman.com	1930	loss of father vigilante serum marvel cor	en	The Amazing Sp	Peter Parker is an outcast high schooler aba	89.8
21	Action Adventure	http://www.robinhoodthemovie.com/	20662	robin hood archer knight sherwood fores	en	Robin Hood	When soldier Robin happens upon the dying	37.6
22	Adventure Fantasy	http://www.thehobbit.com/	57158	elves dwarves orcs hobbit dragon	en	The Hobbit: The I	The Dwarves, Bilbo and Gandalf have succes	94.3
23	Adventure Fantasy	http://www.goldencompassmovie.com/index_c	2268	england compass experiment lordship u	en	The Golden Com	After overhearing a shocking secret, precocic	42.9

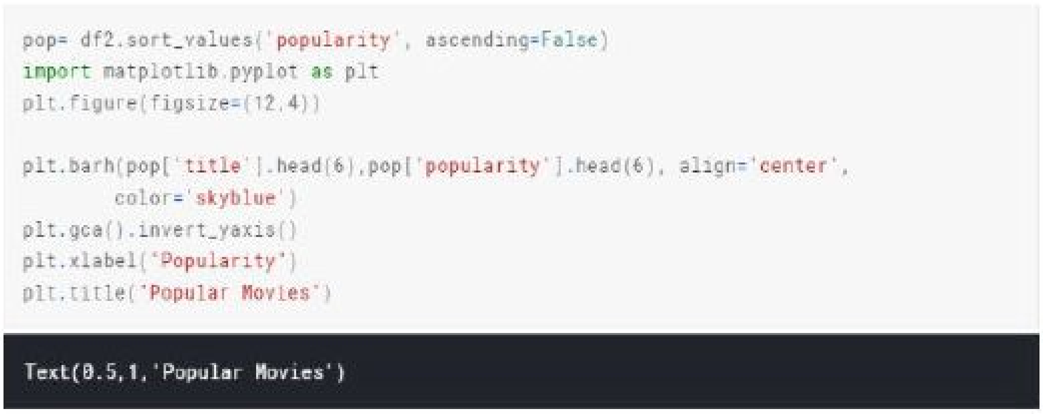
Fig 8.1. Movie Table



A	B	C	D
userId	movieId	rating	timestamp
1	307	3.5	1256677221
1	481	3.5	1256677456
1	1091	1.5	1256677471
1	1257	4.5	1256677460
1	1449	4.5	1256677264
1	1590	2.5	1256677236
1	1591	1.5	1256677475
1	2134	4.5	1256677464
1	2478	4	1256677239
1	2840	3	1256677500
1	2986	2.5	1256677496
1	3020	4	1256677260
1	3424	4.5	1256677444
1	3698	3.5	1256677243
1	3826	2	1256677210
1	3893	3.5	1256677486
2	170	3.5	1192913581
2	849	3.5	1192913537
2	1186	3.5	1192913611

Fig 8.2. Ratings Table

4.2. Demographic Filtering:



```
pop= df2.sort_values('popularity', ascending=False)
import matplotlib.pyplot as plt
plt.figure(figsize=(12,4))

plt.barh(pop['title'].head(6),pop['popularity'].head(6), align='center',
        color='skyblue')
plt.gca().invert_yaxis()
plt.xlabel('Popularity')
plt.title('Popular Movies')
```

```
Text(0.5,1,'Popular Movies')
```

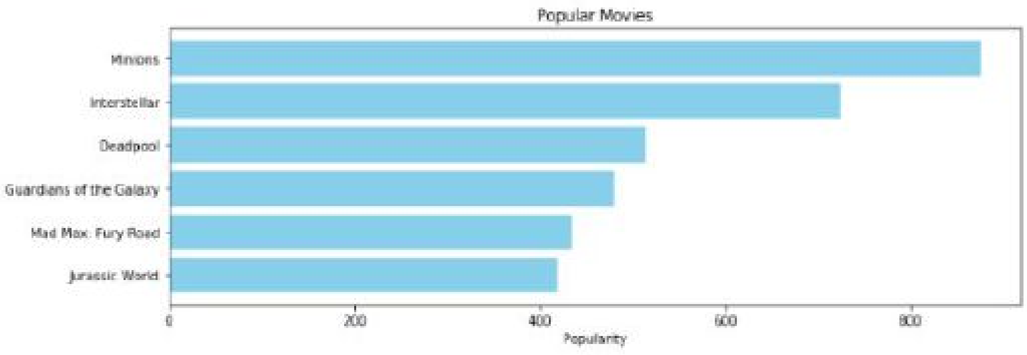
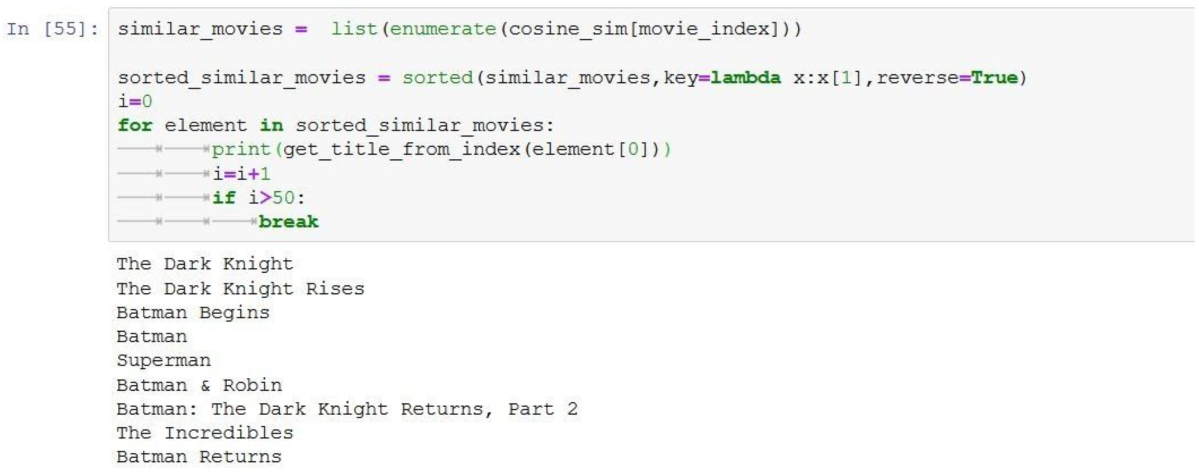


Fig 9. Output for Demographic Filtering

4.3. Content-based Filtering Systems:

When title and plot keywords are used as features for recommendation system:

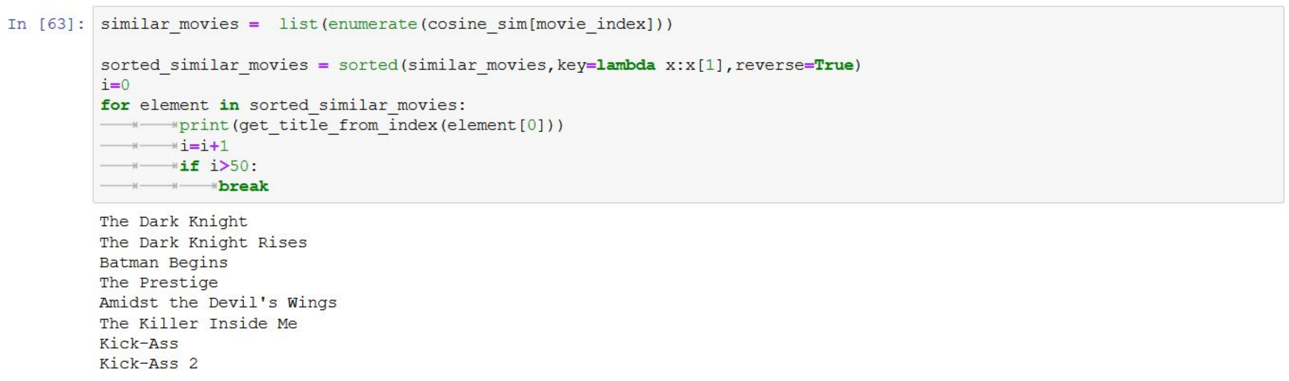


```
In [55]: similar_movies =  list(enumerate(cosine_sim[movie_index]))

         sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)
         i=0
         for element in sorted_similar_movies:
                 print(get_title_from_index(element[0]))
                 i=i+1
                 if i>50:
                         break
```

```
The Dark Knight
The Dark Knight Rises
Batman Begins
Batman
Superman
Batman & Robin
Batman: The Dark Knight Returns, Part 2
The Incredibles
Batman Returns
```

Fig. 10.1. Content-based filtering system using only title and keywords

When the features include cast, director, plot keywords, genres etc.



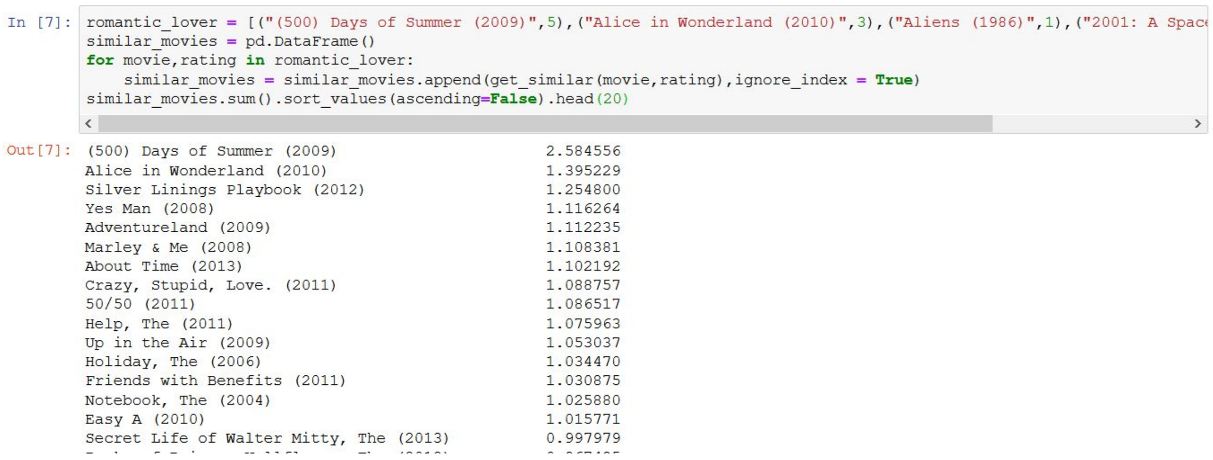
```
In [63]: similar_movies =  list(enumerate(cosine_sim[movie_index]))

         sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)
         i=0
         for element in sorted_similar_movies:
                 print(get_title_from_index(element[0]))
                 i=i+1
                 if i>50:
                         break
```

```
The Dark Knight
The Dark Knight Rises
Batman Begins
The Prestige
Amidst the Devil's Wings
The Killer Inside Me
Kick-Ass
Kick-Ass 2
```

Fig. 10.2. Content-based filtering system using title, cast, director, keywords and genres.

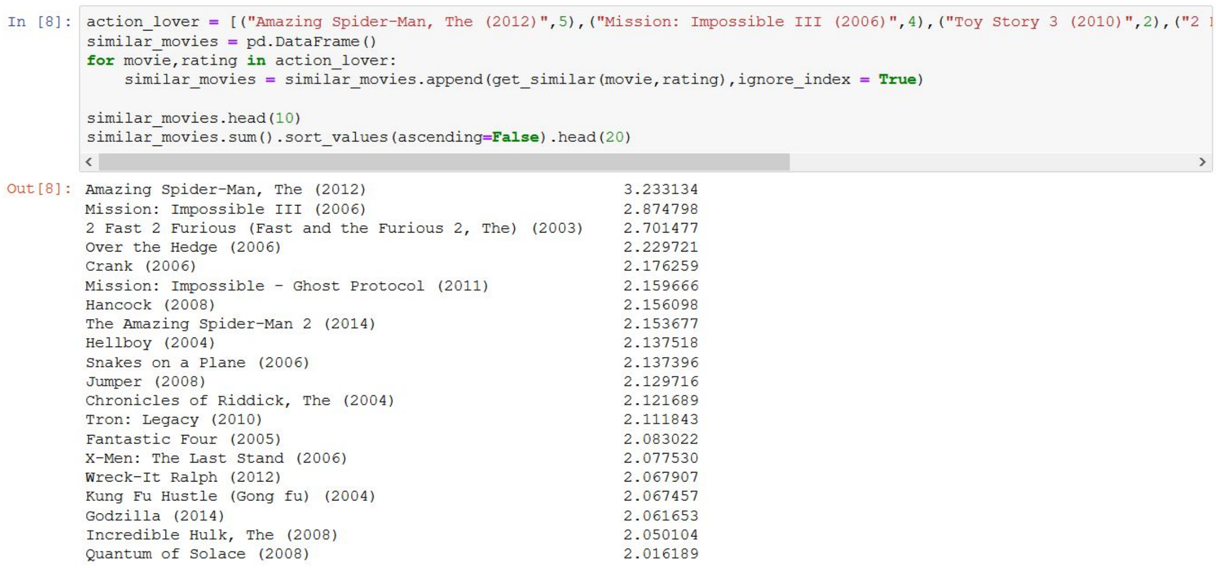
4.4. Collaborative Filtering Systems:

Using User to User Collaborative filtering System.



```
In [7]: romantic_lover = [("(500) Days of Summer (2009)",5),("Alice in Wonderland (2010)",3),("Aliens (1986)",1),("2001: A Spac
        similar_movies = pd.DataFrame()
        for movie,rating in romantic_lover:
            similar_movies = similar_movies.append(get_similar(movie,rating),ignore_index = True)
        similar_movies.sum().sort_values(ascending=False).head(20)

Out[7]: (500) Days of Summer (2009)                  2.584556
        Alice in Wonderland (2010)                   1.395229
        Silver Linings Playbook (2012)               1.254800
        Yes Man (2008)                               1.116264
        Adventureland (2009)                         1.112235
        Marley & Me (2008)                           1.108381
        About Time (2013)                            1.102192
        Crazy, Stupid, Love. (2011)                  1.088757
        50/50 (2011)                                 1.086517
        Help, The (2011)                             1.075963
        Up in the Air (2009)                         1.053037
        Holiday, The (2006)                          1.034470
        Friends with Benefits (2011)                 1.030875
        Notebook, The (2004)                         1.025880
        Easy A (2010)                                1.015771
        Secret Life of Walter Mitty, The (2013)      0.997979
```

Fig 11.1. Collaborative Filtering System (Example 1)



```
In [8]: action_lover = [("Amazing Spider-Man, The (2012)",5),("Mission: Impossible III (2006)",4),("Toy Story 3 (2010)",2),("2
        similar_movies = pd.DataFrame()
        for movie,rating in action_lover:
            similar_movies = similar_movies.append(get_similar(movie,rating),ignore_index = True)

        similar_movies.head(10)
        similar_movies.sum().sort_values(ascending=False).head(20)

Out[8]: Amazing Spider-Man, The (2012)                           3.233134
        Mission: Impossible III (2006)                           2.874798
        2 Fast 2 Furious (Fast and the Furious 2, The) (2003)    2.701477
        Over the Hedge (2006)                                    2.229721
        Crank (2006)                                             2.176259
        Mission: Impossible - Ghost Protocol (2011)              2.159666
        Hancock (2008)                                           2.156098
        The Amazing Spider-Man 2 (2014)                          2.153677
        Hellboy (2004)                                           2.137518
        Snakes on a Plane (2006)                                 2.137396
        Jumper (2008)                                            2.129716
        Chronicles of Riddick, The (2004)                        2.121689
        Tron: Legacy (2010)                                      2.111843
        Fantastic Four (2005)                                    2.083022
        X-Men: The Last Stand (2006)                             2.077530
        Wreck-It Ralph (2012)                                    2.067907
        Kung Fu Hustle (Gong fu) (2004)                          2.067457
        Godzilla (2014)                                          2.061653
        Incredible Hulk, The (2008)                              2.050104
        Quantum of Solace (2008)                                 2.016189
```

Fig 11.2. Collaborative Filtering System (Example 2)

CHAPTER 5

CONCLUSIONS

5.1 Conclusions:

A composite approach consisting of context-based filtering and collaborative filtering should be taken for implementation of the system. The approach can overcome the drawbacks of each algorithm and Enhance system efficiency. Clustering, Similarity and Classification techniques can also be implemented to obtain better recommendations and Consequently, reducing undesired recommendations and increasing accuracy, and precision.

In the future, we will use clustering and hybrid recommendations for better results. A better user interface can be designed for usage by the average consumer. Our method could be applicable for other fields in order to recommend songs, videos, locations, news, books, tourism and e-commerce sites and more.

REFERNCES

[1] Manoj Kumar, D.K Yadav, Ankur Singh, Vijay Kr. Gupta," A Movie Recommender System:

MOVREC" International Journal of Computer Applications (0975 – 8887) Volume 124 – No.3, August,2015.

[2] Prerana Khurana , Shabnam Parveen; 'Approaches of Recommender System: A Survey'; International

Journal of Computer Trends and Technology (IJCTT) – Volume 34 Number 3 - April 2016.

[3] Utkarsh Gupta and Dr Nagamma Patil2," Recommender System Based on Hierarchical Clustering

Algorithm Chameleon" 2015 IEEE International Advance Computing Conference(IACC).

[4] Harpreet Kaur Virk, Er. Maninder Singh," Analysis and Design of Hybrid Online | Movie

Recommender System "International Journal of Innovations in Engineering and Technology

(IJIET)Volume 5 Issue 2,April 2015.

[5] Christina Christakou, Leonidas Lefakis, Spyros Vrettos and Andreas Stafylopatis; "A Movie

Recommender System Based on Semi-supervised Clustering ", IEEE Computer Society Washington,

DC, USA 2015.

[6] Rupali Hande, Ajinkya Gutti, Kevin Shah, Jeet Gandhi, Vrushal Kamtikar

MOVIEMENDER- A MOVIE RECOMMENDER SYSTEM

November, 2016

[7] Beel J., Langer S., and Genzmehr M., "Mind-Map based User Modelling and Research Paper Recommendations," 2014.

[8] Choi, S.-M., Han, Y.-S.: A content recommendation system based on category correlations. In: The Fifth International Multi-Conference on Computing in Global Information Technology, pp. 1257–1260, 2010.

[9] Okkalioglu, M., Koc, M., Polat, H.: On the discovery of fake binary ratings. In: Proceedings of the 30th Annual

ACM Symposium on Applied Computing, SAC 2015,

pp. 901–907. ACM, USA, 2015.

[10] www.movielens.com