MARS : Movie Analysis and Recommendation System

A Project report submitted in fulfilment of the requirements for the degree of Bachelor of Technology

by

ANKIT GHOSH (2015UCP1467)

DEEPAK CHOUDHARY (2015UCP1437)

YATIN KUMAR (2013UCP1643)

Under Guidance of

Dr. Pilli Emmanuel Shubhakar

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

May 2019

Certificate

We,

ANKIT GHOSH (2015UCP1467)

DEEPAK CHOUDHARY (2015UCP1437)

YATIN KUMAR (2013UCP1643),

Declare that this thesis titled, "MARS : Movie Analysis and Recommendation System" and the work presented in it are our own. I confirm that:

- This project work was done wholly or mainly while in candidature for a B.Tech. degree in the department of computer science and engineering at Malaviya National Institute of Technology Jaipur (MNIT).
- Where any part of this thesis has previously been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed.Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this Dissertation is entirely our own work.
- We have acknowledged all main sources of help.

Signed:

____

Date:

____

Dr. Pilli Emmanuel Shubhakar

Associate Professor

May 2019

Department of Computer Science and Engineering

Malaviya National Institute of Technology Jaipur

Abstract

Name of the student: **ANKIT GHOSH (2015UCP1467)**

DEEPAK CHOUDHARY (2015UCP1437)

YATIN KUMAR (2013UCP1643)

Degree for which submitted: **B.Tech.** Department: **Computer Science and Engineering**

Thesis title: **MARS : Movie Analysis and Recommendation System**

Thesis supervisor: **Dr. Pilli Emmanuel Shubhakar**

Month and year of thesis submission: **May 2019**

Recommendation system is an assistive model for users with the intent of suggesting a set of new items to view (e.g., movie, news, research articles etc.) or buy (e.g., book, product etc.). Now a days it has altered the way of seeking out the things of our interest by using information filtering approach. Websites like Netflix and Prime Video are extensively using Recommendation systems in order to suggest movies and shows to users. A movie recommendation system based on collaborative filtering handles the information provided by users, analyzes them, and suggests the best suited film to users according to their processed information. In our proposed system, we have implemented a simple popularity based recommendation system first and then used content-based method and then systematically proceeded to using the collaborative filtering approach. Finally we combined the results of both content-based and collaborative filtering approaches to make an efficient movie recommendation system on which we have done some qualitative analysis. A combination of outputs of these two techniques reveals more precise recommendations concerning movies and the recommendations also reflect on the historical personal choices of the given user as well. The MovieLens dataset was used to explore the proposed system and get results and various other metadata was also created for the given movies in order to increase the features used for deciding which movies should be recommended to a specific user. The major advantages of our proposed system is that we have eliminated the limitations of content based method and collaborative method by combining their best algorithmic parts which produce goog qualitative results. We have used Python 3 for coding and used Google Colab as our development platform.

Acknowledgements

It is a matter of great pleasure and privilege for us to present our B. Tech Project report on “MARS : Movie Analysis and Recommendation System”. We would like to express our gratitude towards our supervisor, **Dr.Pilli Emmanuel Shubhakar**, Associate Professor, Department of Computer Science and Engineering, Malaviya National Institute of Technology Jaipur, for his time, invaluable guidance and cooperation throughout the period. He provided us with continuous encouragement and support. He has been the principle encouraging force behind this and gave constructive direction to our work.

We are grateful to respected **Dr.Girdhari Singh**, Head, Department of Computer Science and Engineering for permitting us to utilize all the necessary facilities of the institution.

We express our deep gratitude to Mr. Suraj and the staff of EICT Lab for providing us with a healthy working environment.

We also express our heartfelt gratitude towards all the teachers of Department of Computer Science and Engineering and technical staff for all their guidance and help throughout our stay at MNIT Jaipur.

Ankit Ghosh 2015UCP1467
Deepak Choudhary 2015UCP1437
Yatin Kumar 2013UCP1643

Contents

List of Figures

Chapter 1

Introduction and Objectives

This chapter is an introduction to what recommendation systems are and what were the key motivating factors for us in developing a combined approach to making movie recommender and also clarifies the objective of our project.

1.1 Introduction

We live in a data society in which individuals are frequently gone up against with substantial measures of information, for example through the web. We are approached to settle on decisions that are practically difficult to make without extra data or direction. Recommender frameworks can give such direction by helping the client in the basic leadership process or by settling on the choice for the client. These frameworks utilize the colossal measure of accessible information such that clients never can. Suggestion frameworks are as of now being utilized in many areas. GroupLens (Resnick, Iacovou, Suchak, Bergstorm, and Riedl, 1994), for example, is a framework that suggests news stories that could be of intrigue to the client. AbeBooks suggests books, Last.fm encourages the client to nd new music and MovieLens, IMDb and Netix suggest motion pictures. Amazon.com is a recommender that is had some expertise in every one of the three recently referenced media.

1.2 Motivation

There are a lot of recommendation systems existing currently which generally use the collaborative filtering approach. Examples include Netflix, Primevideo, Movielens, themoviedatabase, etc. We studied a few algorithms that are generally used for recommendation systems and content-based recommendation system and collaborative filtering were the most used methods. But both these methods have their own drawbacks (which will be explained in the later sections). So, we identified the drawbacks and thought of mitigating those limitations by combining the best of both types of recommendation systems.

1.3 Objective

Our main objective is to develop a quality movie recommendation system which combines two general approaches (i.e. content based and collaborative filtering) and produces recommendations based on the input provided.

Chapter 2

Existing Methods

Some of the existing algorithms for movie recommendation are given below.

2.1 Collaborative Filtering

Shifting methodologies for programmed suggestion have been proposed. The most seasoned and most created technique is community ltering. This technique utilizes between client correlations to produce new suggestions. A cooperative framework comprises of a database which contains the clients' appraisals and is improved as the client associates with the framework after some time. Clients are thought about dependent on their appraisals and the acquired similitudes and contrasts are used to make a proposal. This strategy experiences the sparsity issue. All the clients don't abuse the choice to rate things they have seen or utilized. The accessible rating dataset is consequently ordinarily extremely meager, particularly when a client is new or when the framework is new and individuals are simply beginning to utilize it. Another issue is the rst-rater issue, for example before a thing has been prescribed for the first time, the framework won't suggest it. This issue applies to new things and darken things which makes it less appealing for individuals with non-standard tastes. An incredible component of Collaborative Filtering is that it can astonish the client with significant things that are not fundamentally the same as things in the clients' prole. This supposed 'fresh' suggestion capacity is conceivable in light of the fact that it employments individuals to-individuals relationships.

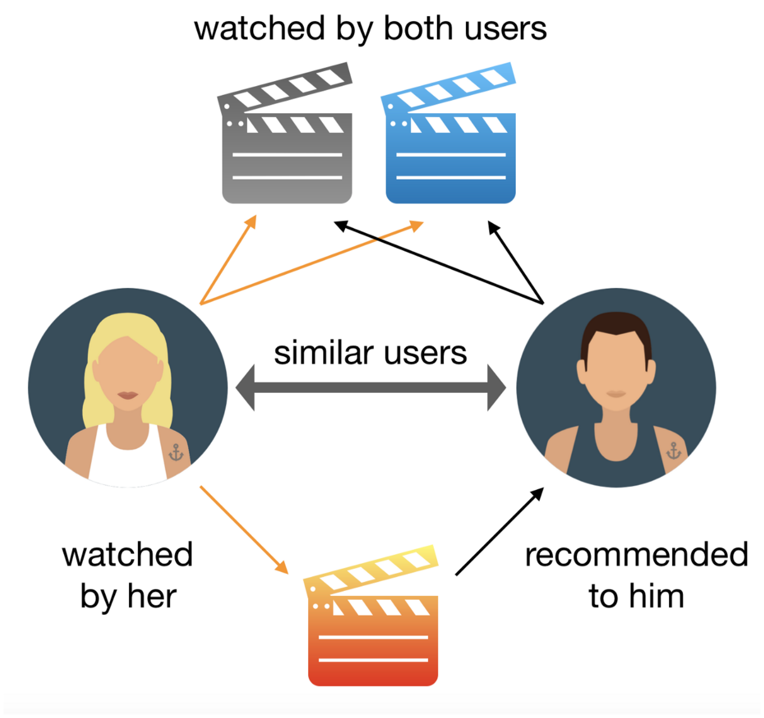


Figure 2.1: **User Based Collaborative Filtering**

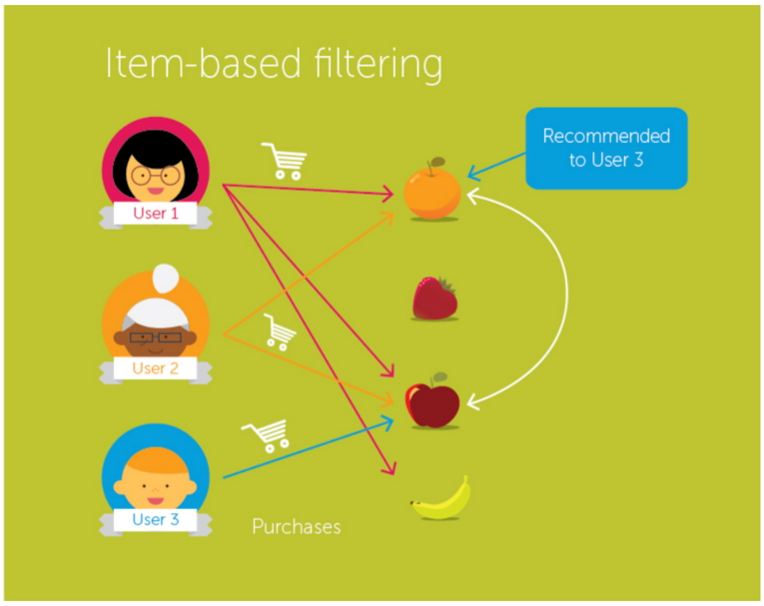


Figure 2.2: **Item Based Collaborative Filtering**

2.2 Content Based Filtering

Another strategy utilizes content based data. Content based ltering utilizes thing to-thing connection to look at portrayals of substance in a thing to portrayals of substance in things the client has appraised. The likenesses among things and the rating data are utilized to foresee how much the client will abhorrence or like another thing. An inconvenience of this strategy is that it is totally reliant upon machine clear portrayals of things, which might be difcult to get. Content based techniques are not ready to astonish the client, since it utilizes the component estimations of the things that the client has just appraised and won't prescribe a thing that does not share any of the predefined values. Similarly as collaborative ltering, content based techniques additionally experience the ill effects of the sparsity issue.

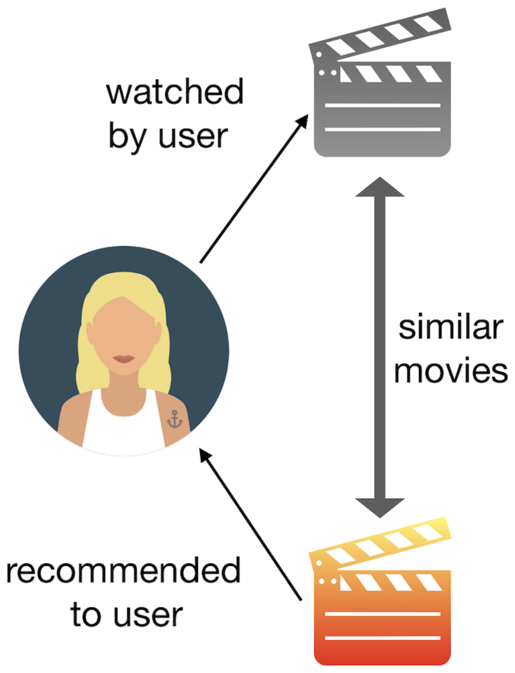


FIGURE 2.3: **Content Based Filtering**

Chapter 3

Implementation

We will now explain the whole implementation of how we made specific improvements on certain algorithms and will perform a qualitative analysis on the result as well.

3.1 Dataset Information

We have used the MovieLens dataset from grouplens.org.
Full Dataset: It consists of 26 million ratings and seven hundred and fifty thousand tag applications applied to forty-five thousand movies by two hundred and seventy thousand users. Last updated on September 2017
Small Dataset: It comprises of one hundred thousand ratings and one thousand three hundred tag applications applied to nine thousand movies by seven hundred users. Last updated on September 2017

3.1.1 Available Dataset

1. Ratings Data File Structure (**ratings.csv**)

    - All ratings of all movies are contained in the document ratings.csv. Each line of this record after the header represents one rating of one film by one user, and has the accompanying structure: userId,movieId,rating,timestamp
    - The lines within this file are sorted by userId, and, within user, by movieId.
    - Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).

- Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

2. Tags Data File Structure (**tags.csv**)

   - All labels/tags are contained in the document tags.csv. Each line of this record after the header represents one label connected to one film by one user, and has the accompanying structure: userId,movieId,tag,timestamp
   - The lines within this file are ordered first by userId, then, within user, by movieId.
   - Tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag is determined by each user.
   - Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

3. Movies Data File Structure (**movies.csv**)

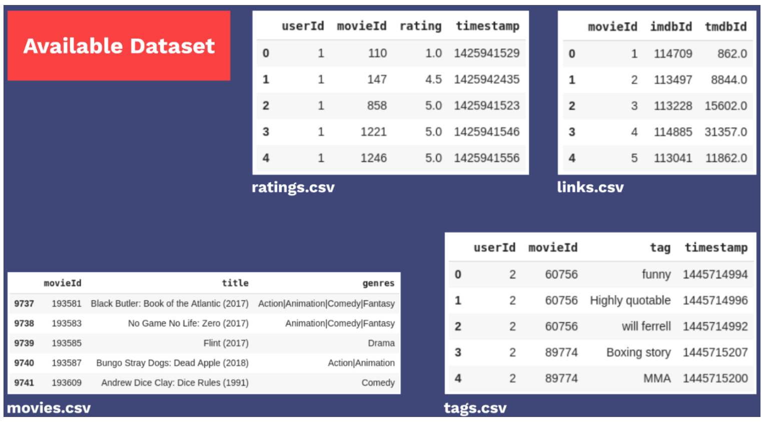
   - Movie information is contained in the file movies.csv. Each line of this file after the header row represents one movie, and has the following format: movieId,title,genres
   - Movie titles are entered manually or imported from www.themoviedb.org/, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.
   - Genres are a pipe-separated list, and are selected from the following:

     (a) Action
     (b) Adventure
     (c) Animation
     (d) Children's
     (e) Comedy
     (f) Crime
     (g) Documentary
     (h) Drama
     (i) Fantasy
     (j) Film-Noir
     (k) Horror
     (l) Musical

(m) Mystery
(n) Romance
(o) Sci-Fi
(p) Thriller
(q) War
(r) Western
(s) (no genres listed)

4. Links Data File Structure (**links.csv**)

- Identifiers that can be used to link to other sources of movie data are contained in the file links.csv. Each line of this file after the header row represents one movie, and has the following format: movieId,imdbId,tmdbId
- movieId is an identifier for movies used by https://movielens.org. E.g., the movie Toy Story has the link https://movielens.org/movies/1.
- imdbId is an identifier for movies used by http://www.imdb.com. E.g., the movie Toy Story has the link http://www.imdb.com/title/tt0114709/.
- tmdbId is an identifier for movies used by https://www.themoviedb.org. E.g., the movie Toy Story has the link https://www.themoviedb.org/movie/862.


Available Dataset

	userId	movieId	rating	timestamp
0	1	110	1.0	1425941529
1	1	147	4.5	1425942435
2	1	858	5.0	1425941523
3	1	1221	5.0	1425941546
4	1	1246	5.0	1425941556

ratings.csv

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

links.csv

	movieId	title	genres
9737	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy
9738	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy
9739	193585	Flint (2017)	Drama
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy

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

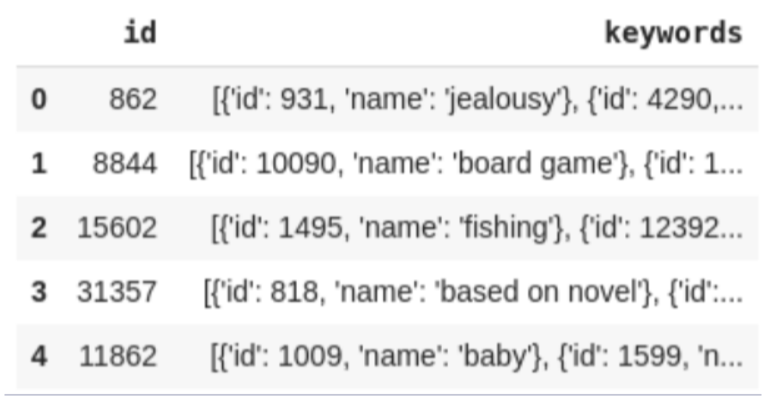
tags.csv

FIGURE 3.1: **Available Dataset**

3.1.2 Created Dataset

1. Keywords Data File Structure (**keywords.csv**)

   - Format: id,keywords
   - Description
     - id: It's movie ID given by TMDb
     - Keywords: Tags/keywords for the movie. It list of tags/keywords



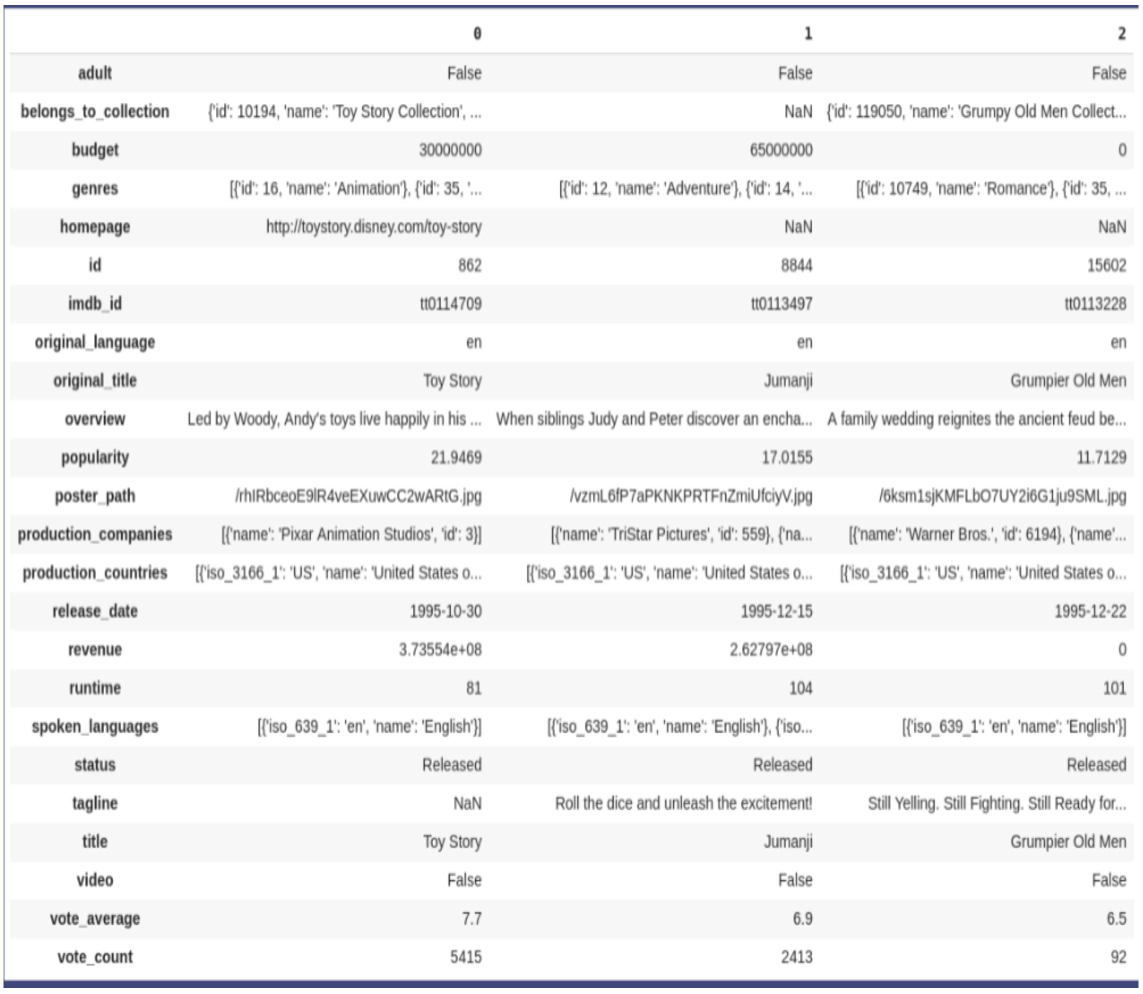
	id	keywords
0	862	[{'id': 931, 'name': 'jealousy'}, {'id': 4290,...
1	8844	[{'id': 10090, 'name': 'board game'}, {'id': 1...
2	15602	[{'id': 1495, 'name': 'fishing'}, {'id': 12392...
3	31357	[{'id': 818, 'name': 'based on novel'}, {'id':...
4	11862	[{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n...

Figure 3.2: **Keywords Dataset**

2. Movies Metadata Data File Structure (**movies_metadata.csv**)

   - Format: adult,belongs_to_collection,budget,genres,homepage,id,imdb_id, original_language,original_title,overview,popularity,poster_path, production_companies,production_countries,release_date,revenue,runtime, spoken_languages,status,tagline,title,video,vote_average,vote_count
   - Description
     - adult: Indicates if the movie is X-Rated or Adult.
     - belongs_to_collection: A stringified dictionary that gives information on the movie series the particular film belongs to.
     - budget: The budget of the movie in dollars.
     - genres: A stringified list of dictionaries that list out all the genres associated with the movie.
     - homepage: The Official Homepage of the move.
     - id: The ID of the movie.
     - imdb_id: The IMDB ID of the movie.
     - original_language: The language in which the movie was originally shot in.

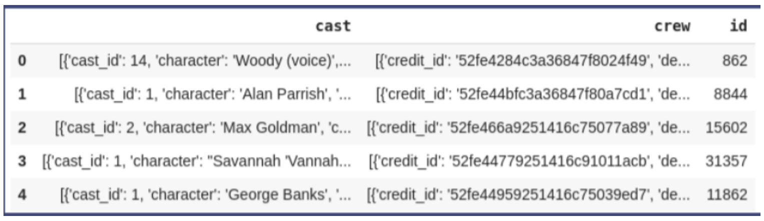
- original_title: The original title of the movie.
- overview: A brief blurb of the movie.
- popularity: The Popularity Score assigned by TMDB.
- poster_path: The URL of the poster image.
- production_companies: A stringified list of production companies involved with the making of the movie.
- production_countries: A stringified list of countries where the movie was shot/produced in.
- release_date: Theatrical Release Date of the movie.
- revenue: The total revenue of the movie in dollars.
- runtime: The runtime of the movie in minutes.
- spoken_languages: A stringified list of spoken languages in the film.
- status: The status of the movie (Released, To Be Released, Announced, etc.)
- tagline: The tagline of the movie.
- title: The Official Title of the movie.
- video: Indicates if there is a video present of the movie with TMDB.
- vote_average: The average rating of the movie.
- vote_count: The number of votes by users, as counted by TMDB.



	0	1	2
adult	False	False	False
belongs_to_collection	{'id': 10194, 'name': 'Toy Story Collection', ...	NaN	{'id': 119050, 'name': 'Grumpy Old Men Collect...
budget	30000000	65000000	0
genres	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...
homepage	http://toystory.disney.com/toy-story	NaN	NaN
id	862	8844	15602
imdb_id	tt0114709	tt0113497	tt0113228
original_language	en	en	en
original_title	Toy Story	Jumanji	Grumpier Old Men
overview	Led by Woody, Andy's toys live happily in his ...	When siblings Judy and Peter discover an encha...	A family wedding reignites the ancient feud be...
popularity	21.9469	17.0155	11.7129
poster_path	/rhIRbceoE9lR4veEXuwCC2wARtG.jpg	/vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg	/6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg
production_companies	[{'name': 'Pixar Animation Studios', 'id': 3}]	[{'name': 'TriStar Pictures', 'id': 559}, {'na...	[{'name': 'Warner Bros.', 'id': 6194}, {'name'...
production_countries	[{'iso_3166_1': 'US', 'name': 'United States o...	[{'iso_3166_1': 'US', 'name': 'United States o...	[{'iso_3166_1': 'US', 'name': 'United States o...
release_date	1995-10-30	1995-12-15	1995-12-22
revenue	3.73554e+08	2.62797e+08	0
runtime	81	104	101
spoken_languages	[{'iso_639_1': 'en', 'name': 'English'}]	[{'iso_639_1': 'en', 'name': 'English'}, {'iso...	[{'iso_639_1': 'en', 'name': 'English'}]
status	Released	Released	Released
tagline	NaN	Roll the dice and unleash the excitement!	Still Yelling. Still Fighting. Still Ready for...
title	Toy Story	Jumanji	Grumpier Old Men
video	False	False	False
vote_average	7.7	6.9	6.5
vote_count	5415	2413	92

FIGURE 3.3: **Movies Metadata Dataset**

3. Credits Data File Structure (**credits.csv**)
    (a) Format: cast,crew,id
    (b) Description
        i. cast: Information about casting. Name of actor, gender and it's character name in movie
        ii. crew: Information about crew members. Like who directed the movie, editor of the movie and so on.
        iii. id: It's movie ID given by TMDb



	cast	crew	id
0	[{'cast_id': 14, 'character': 'Woody (voice)',...	[{'credit_id': '52fe4284c3a36847f8024f49', 'de...	862
1	[{'cast_id': 1, 'character': 'Alan Parrish', '...	[{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de...	8844
2	[{'cast_id': 2, 'character': 'Max Goldman', 'c...	[{'credit_id': '52fe466a9251416c75077a89', 'de...	15602
3	[{'cast_id': 1, 'character': "Savannah 'Vannah...	[{'credit_id': '52fe44779251416c91011acb', 'de...	31357
4	[{'cast_id': 1, 'character': 'George Banks', '...	[{'credit_id': '52fe44959251416c75039ed7', 'de...	11862

FIGURE 3.4: **Credits Dataset**

3.2 Popularity Based Recommendation System

- This Recommender offers generalized recommendations to every user based on popularity of movies and genre, if provided.
- The basic idea behind this recommender is that movies which are more popular and more appreciated will have a higher probability of being liked by the average crowd.
- This model does not provide personalized recommendations based on the preferences of the user.

3.2.1 Design

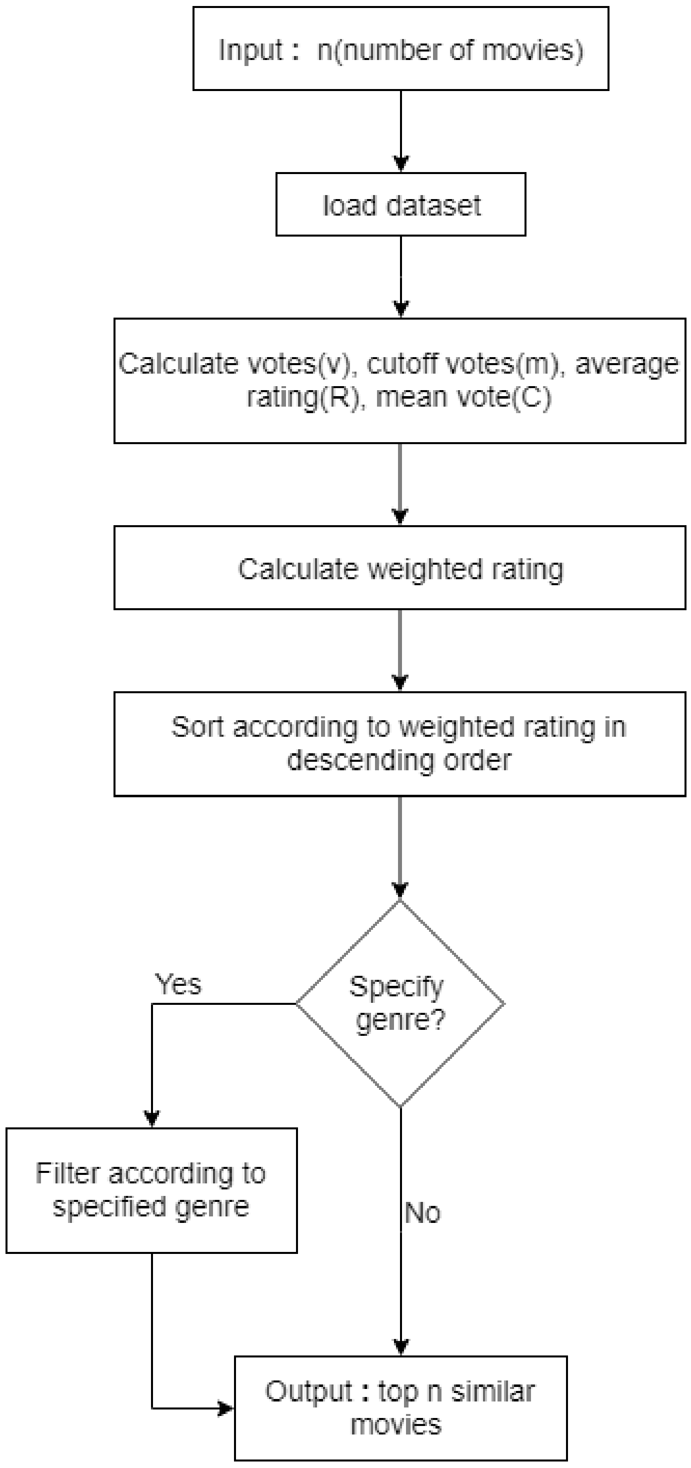

Input : n(number of movies)
load dataset
Calculate votes(v), cutoff votes(m), average rating(R), mean vote(C)
Calculate weighted rating
Sort according to weighted rating in descending order
Specify genre?
Yes
No
Filter according to specified genre
Output : top n similar movies

FIGURE 3.5: **Popularity Based Recommendation System**

3.2.2 Implementation

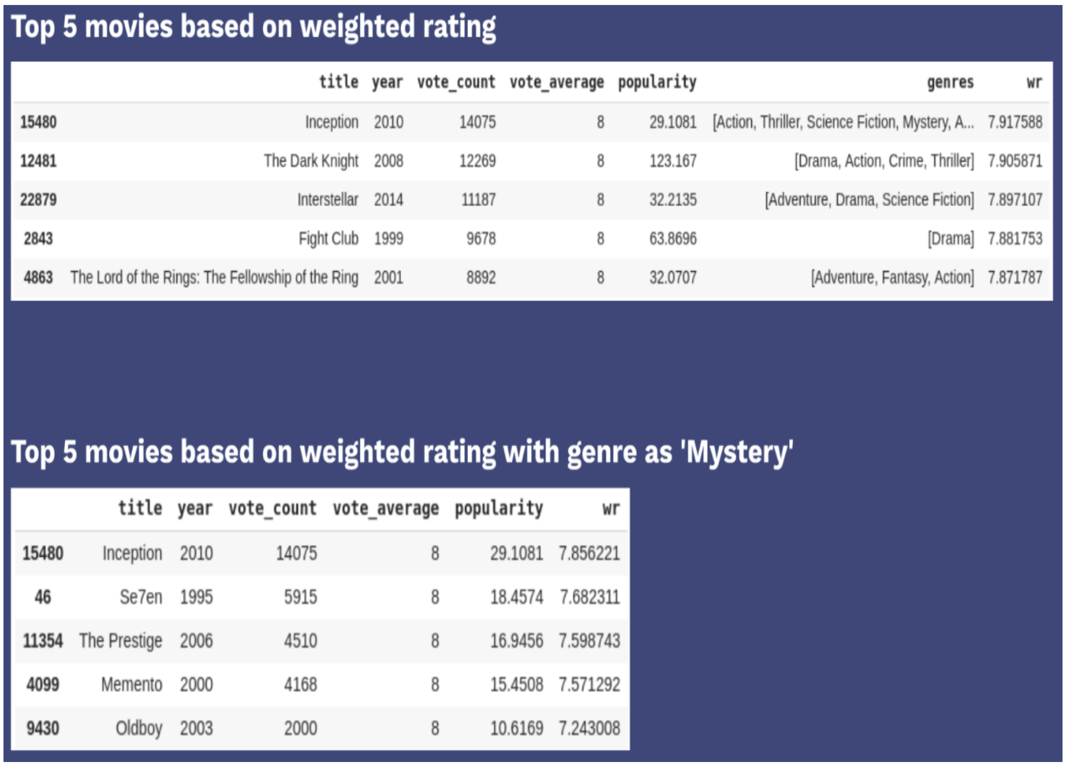
- We have sorted our movies based on ratings and popularity and displayed the top movies in our list.
- For refining purpose, to get the top movies of a particular genre, we can pass a genre parameter as well.
- We are using the ratings.csv file to come up with our Top Popular Movies Chart.
- We are using IMDB's weighted rating formula to construct our chart.
- Mathematical representation of the same is as follows:

$$WeightedRating(WR) = ((\frac{v}{v+m}) * R) + ((\frac{m}{v+m}) * C)$$

v - number of votes for the movie
m - minimum votes required to be listed in the chart
R - average rating of the movie
C - mean vote across the whole report

- Then, we determined an appropriate value for m, the minimum number of votes required to be listed in the chart.
- We used 95th percentile as our cutoff, i.e. for a movie to feature in the charts, it must have more votes than at least 95% of the movies in the list.
- Therefore, to qualify to be considered for the chart, a movie has to have at least 'm' votes on TMDB as given in ratings.csv file.
- Here, only few movies are qualified to be on our chart.

3.2.3 Result



Top 5 movies based on weighted rating

	title	year	vote_count	vote_average	popularity	genres	wr
15480	Inception	2010	14075	8	29.1081	[Action, Thriller, Science Fiction, Mystery, A...	7.917588
12481	The Dark Knight	2008	12269	8	123.167	[Drama, Action, Crime, Thriller]	7.905871
22879	Interstellar	2014	11187	8	32.2135	[Adventure, Drama, Science Fiction]	7.897107
2843	Fight Club	1999	9678	8	63.8696	[Drama]	7.881753
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.0707	[Adventure, Fantasy, Action]	7.871787

Top 5 movies based on weighted rating with genre as 'Mystery'

	title	year	vote_count	vote_average	popularity	wr
15480	Inception	2010	14075	8	29.1081	7.856221
46	Se7en	1995	5915	8	18.4574	7.682311
11354	The Prestige	2006	4510	8	16.9456	7.598743
4099	Memento	2000	4168	8	15.4508	7.571292
9430	Oldboy	2003	2000	8	10.6169	7.243008

FIGURE 3.6: **Popularity Based Recommender Output**

3.2.4 Limitations

It gives similar recommendations to everyone, regardless of the user's personal taste (like genre, director, cast). If a person who loves romantic movies and maybe hates action movies were to look at our Top 5 Chart, he/she wouldn't probably like most of the movies. If he/she were to look at our charts by genre, he/she still wouldn't be getting the best recommendations.

3.3 Content Based Recommendation System

In the next step we built an engine that computes similarity between movies based on certain metrics/features and suggests movies that are most similar to a particular movie that a user likes. We have used movie metadata to build this recommendation system, thereby applying the methodology of Content Based Filtering.
We have built two Content Based Recommenders which are based on:

- Movie Overviews and Taglines from movies_metadata.csv file
- Using movie Cast, Crew, Keywords and Genre from credits.csv and keywords.csv file

3.3.1 Design

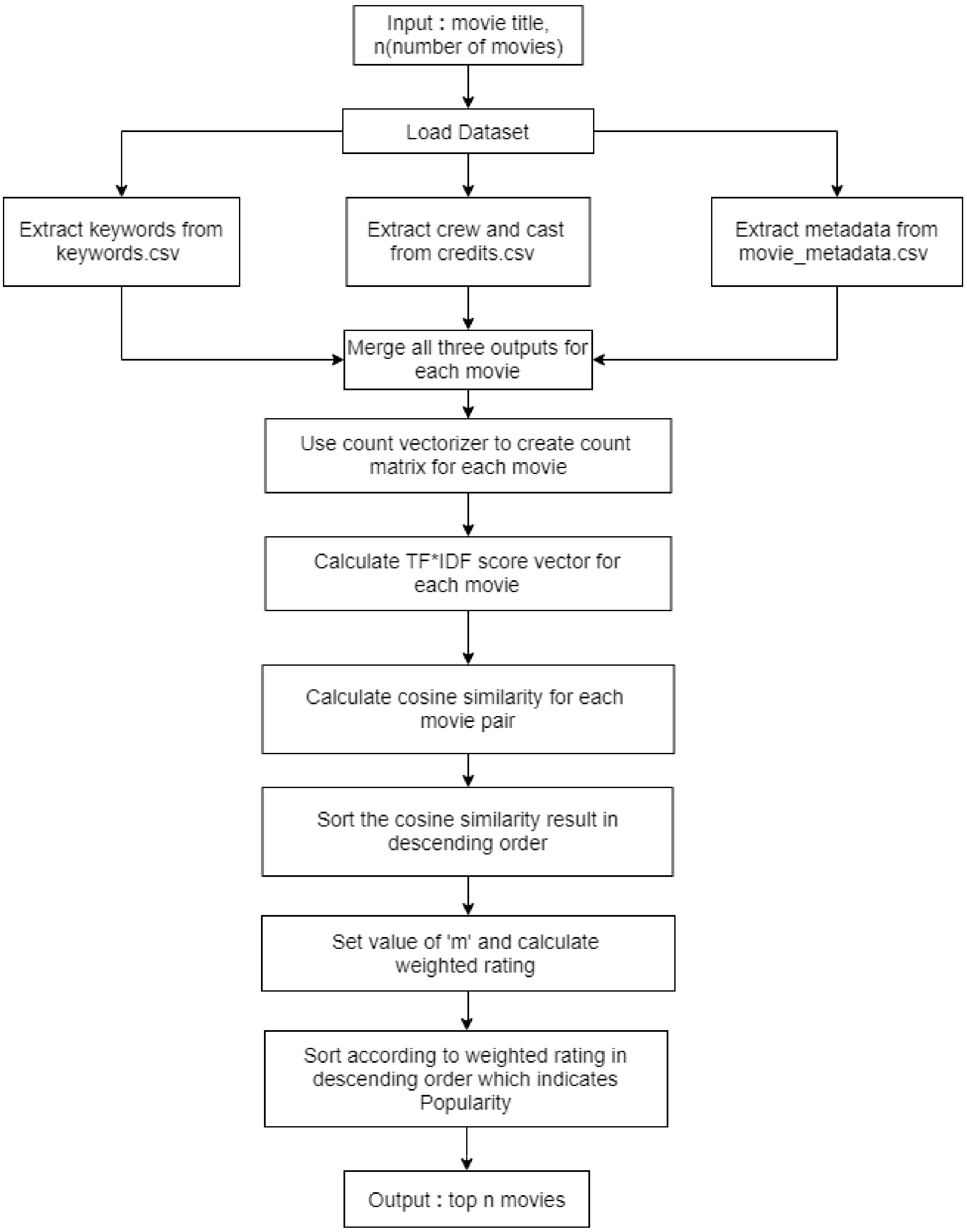
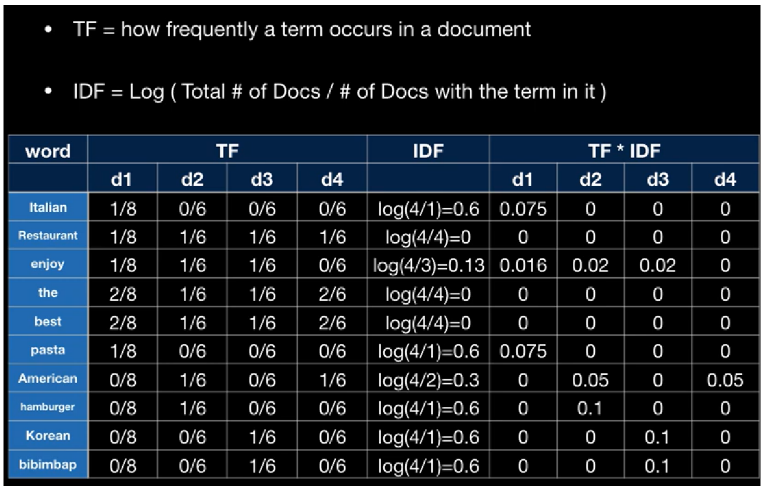

Input : movie title, n(number of movies)
Load Dataset
Extract keywords from keywords.csv
Extract crew and cast from credits.csv
Extract metadata from movie_metadata.csv
Merge all three outputs for each movie
Use count vectorizer to create count matrix for each movie
Calculate TF*IDF score vector for each movie
Calculate cosine similarity for each movie pair
Sort the cosine similarity result in descending order
Set value of 'm' and calculate weighted rating
Sort according to weighted rating in descending order which indicates Popularity
Output : top n movies

FIGURE 3.7: **Content Based Recommendation System**

3.3.2 Implementation

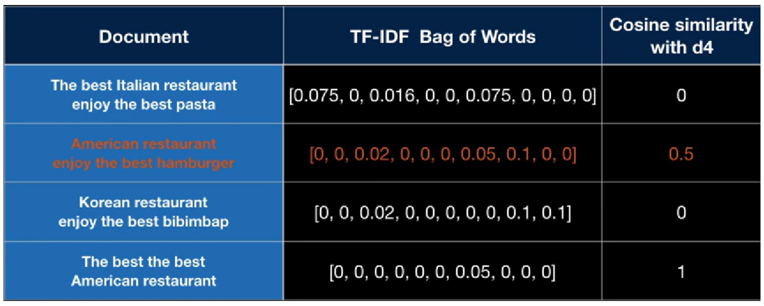
- We do not have a quantitative metric to judge our rcommender's performance so we have done this qualitatively.
- We have used the TF-IDF Vectorizer and calculated the Dot Product which directly gave us the Cosine Similarity Score.
- Therefore, we have used sklearn's linear_kernel instead of cosine_similarities since it is faster as mentioned in its documentation.
  We thereby generated a pairwise cosine similarity matrix for all the movies in our dataset.
- Based on the cosine similarity score, we can now get the most similar movies. Higher the cosine similarity score of the given movie with a movie in the dataset, higher will be its similarity and thus movies with higher cosine similarity score will be recommended. An illustration is given as follows:



- TF = how frequently a term occurs in a document
- IDF = Log (Total # of Docs / # of Docs with the term in it)

word	TF				IDF	TF * IDF			
	d1	d2	d3	d4		d1	d2	d3	d4
Italian	1/8	0/6	0/6	0/6	log(4/1)=0.6	0.075	0	0	0
Restaurant	1/8	1/6	1/6	1/6	log(4/4)=0	0	0	0	0
enjoy	1/8	1/6	1/6	0/6	log(4/3)=0.13	0.016	0.02	0.02	0
the	2/8	1/6	1/6	2/6	log(4/4)=0	0	0	0	0
best	2/8	1/6	1/6	2/6	log(4/4)=0	0	0	0	0
pasta	1/8	0/6	0/6	0/6	log(4/1)=0.6	0.075	0	0	0
American	0/8	1/6	0/6	1/6	log(4/2)=0.3	0	0.05	0	0.05
hamburger	0/8	1/6	0/6	0/6	log(4/1)=0.6	0	0.1	0	0
Korean	0/8	0/6	1/6	0/6	log(4/1)=0.6	0	0	0.1	0
bibimbap	0/8	0/6	1/6	0/6	log(4/1)=0.6	0	0	0.1	0

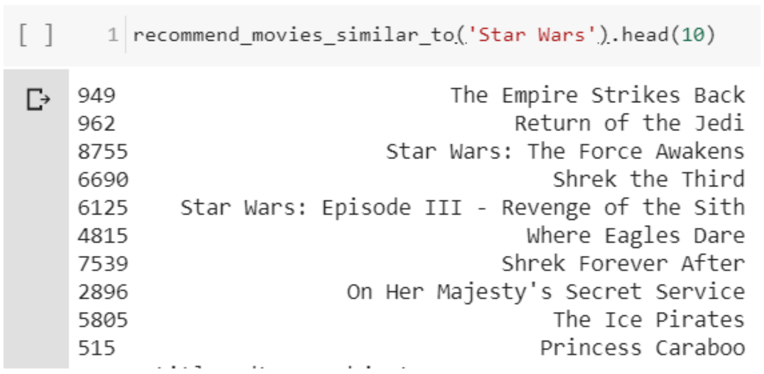
FIGURE 3.8: **TF-IDF calculation example**



Document	TF-IDF Bag of Words	Cosine similarity with d4
The best Italian restaurant enjoy the best pasta	[0.075, 0, 0.016, 0, 0, 0.075, 0, 0, 0, 0]	0
American restaurant enjoy the best hamburger	[0, 0, 0.02, 0, 0, 0, 0.05, 0.1, 0, 0]	0.5
Korean restaurant enjoy the best bibimbap	[0, 0, 0.02, 0, 0, 0, 0, 0, 0.1, 0.1]	0
The best the best American restaurant	[0, 0, 0, 0, 0, 0, 0.05, 0, 0, 0]	1

FIGURE 3.9: **Cosine Similarity calculation example**

- As we can see, the recommendations for a user who likes 'Star Wars' is listed using movie overview and taglines as the features..

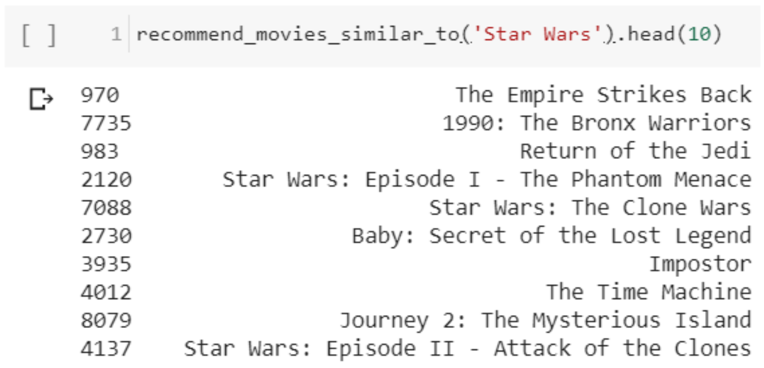


```
[ ]  recommend_movies_similar_to('Star Wars').head(10)
```

```
949                          The Empire Strikes Back
962                              Return of the Jedi
8755                    Star Wars: The Force Awakens
6690                                 Shrek the Third
6125    Star Wars: Episode III - Revenge of the Sith
4815                               Where Eagles Dare
7539                              Shrek Forever After
2896                 On Her Majesty's Secret Service
5805                                 The Ice Pirates
515                                 Princess Caraboo
```

FIGURE 3.10: **Content Based Recommender example 1**

- The above example shows that it doesn't take into considerations very important features such as cast, crew, director and genre, which determine the rating and the popularity of a movie.
- To enhance our metadata based content recommender, we have merged our current dataset with the crew from credits.csv and the keywords from keywords.csv datasets.
- Crew: From crew, we picked the director as our feature since the other features don't contribute much to the vibe of the movie.
- Cast: We only selected the major characters and their respective actors. Arbitrarily we have chosen the top 3 actors that appear in the credits list.
- A metadata dump for every movie has been created which consists of genres, director, keywords, and main actors.
- We have then used Count Vectorizer to create our count matrix
- Then we calculate the cosine similarities and return movies that are most similar as done before.
- Since our cosine similarity scores have changed, we expect it to give us different and better results.



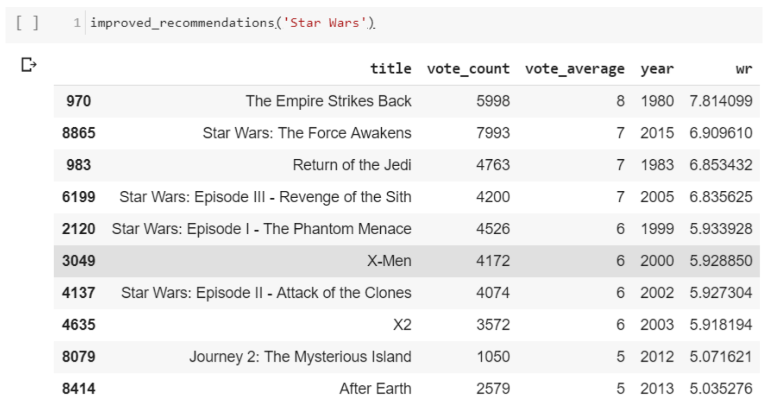
```
[ ]  recommend_movies_similar_to('Star Wars').head(10)
```

```
970                            The Empire Strikes Back
7735                         1990: The Bronx Warriors
983                                Return of the Jedi
2120          Star Wars: Episode I - The Phantom Menace
7088                          Star Wars: The Clone Wars
2730                     Baby: Secret of the Lost Legend
3935                                          Impostor
4012                                   The Time Machine
8079                     Journey 2: The Mysterious Island
4137     Star Wars: Episode II - Attack of the Clones
```

FIGURE 3.11: **Content Based Recommender example 2**

- Finally, we have added a mechanism to remove bad movies and return movies which are popular and have had a good critical response.
- We have taken the top 25 movies based on similarity scores and calculated the vote of the 60th percentile movie. Then, using this as the value of m, we calculated the weighted rating of each movie using IMDB's formula like we did in the Popularity Based Recommender (section 3.2).

3.3.3 Result



```
[ ]  improved_recommendations('Star Wars')
```

	title	vote_count	vote_average	year	wr
970	The Empire Strikes Back	5998	8	1980	7.814099
8865	Star Wars: The Force Awakens	7993	7	2015	6.909610
983	Return of the Jedi	4763	7	1983	6.853432
6199	Star Wars: Episode III - Revenge of the Sith	4200	7	2005	6.835625
2120	Star Wars: Episode I - The Phantom Menace	4526	6	1999	5.933928
3049	X-Men	4172	6	2000	5.928850
4137	Star Wars: Episode II - Attack of the Clones	4074	6	2002	5.927304
4635	X2	3572	6	2003	5.918194
8079	Journey 2: The Mysterious Island	1050	5	2012	5.071621
8414	After Earth	2579	5	2013	5.035276

FIGURE 3.12: **Content Based Recommender with improved output**

We can try out different weights for our features (directors, actors, genres), limiting the number of keywords that can be used, weighing genres based on their frequency, etc.

Thus our, Content Based Recommendation system concludes here.

3.3.4 Limitations

- It is only capable of suggesting movies which are similar to a certain movie.
- It is not capable of capturing tastes and providing recommendations across genres.
- Also, the recommendation system that we built is not really personal as it doesn't capture the personal biases and tastes of the user.
- Anyone querying our system for recommendations based on a movie will receive the same recommendations for that movie, regardless of who he/she is.

3.4 Collaborative Filtering Based Recommendation System

The Collaborative Filtering Recommendation System is entirely based on the past behavior and not on the context. More accurately, it is based on the similarity in preferences, choices,and tastes of two users. It analyses how alike the choices of one user is to another and makes recommendations on the basis of that concept. For example, if user 1 likes movies A, B, C and user 2 likes movies B, C, D, then they have similar interests and 1 should like movie D and 2 should like movie A. This makes it one of the most commonly used algorithm as it is independent of any additional information. In general, collaborative filtering is the powerhouse of recommendation systems. The algorithm has a very interesting property of being able to do feature learning on its own, which means that it can start to learn for itself what features to use which are latent in nature.

3.4.1 Design

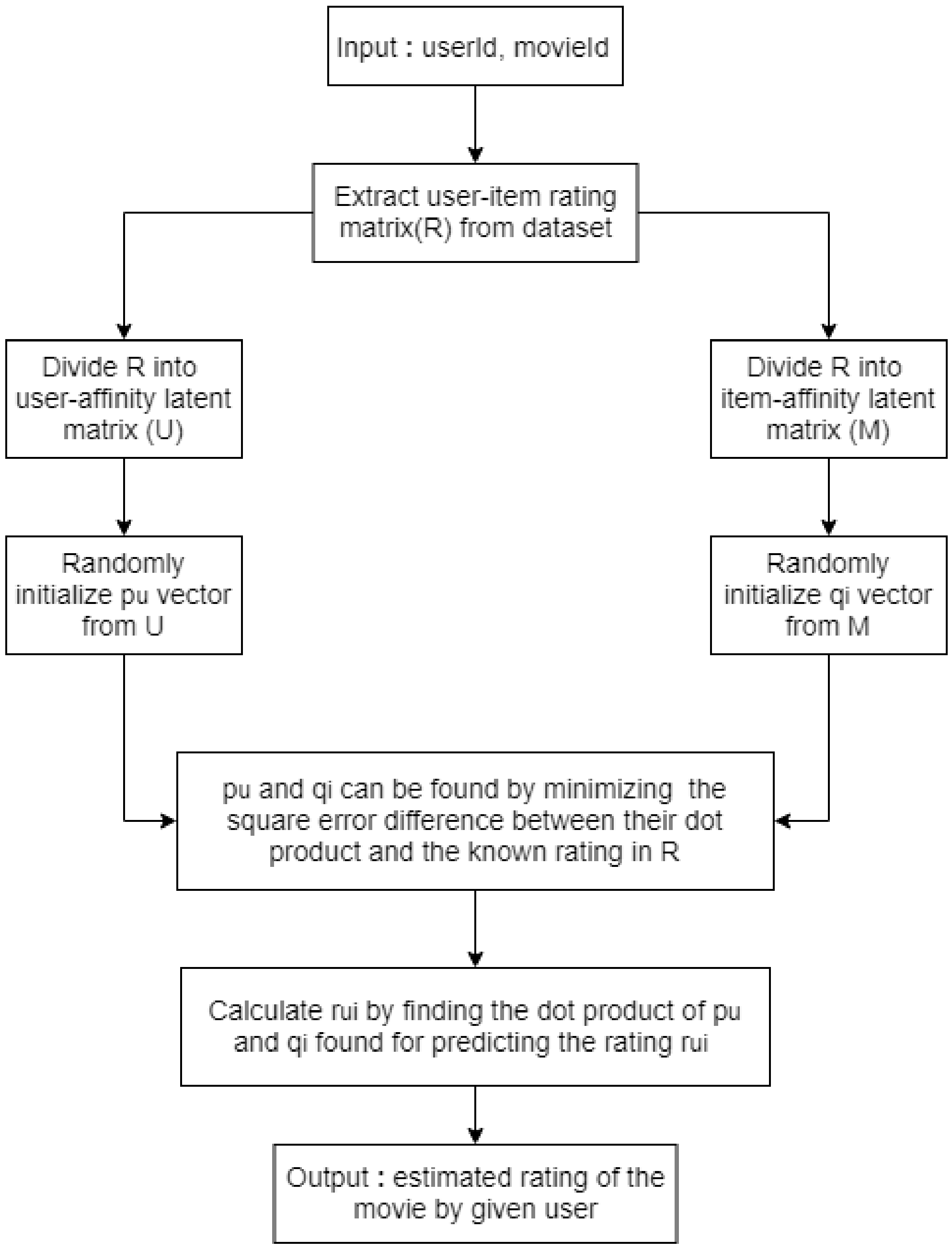

Input : userId, movieId
Extract user-item rating matrix(R) from dataset
Divide R into user-affinity latent matrix (U)
Divide R into item-affinity latent matrix (M)
Randomly initialize pu vector from U
Randomly initialize qi vector from M
pu and qi can be found by minimizing the square error difference between their dot product and the known rating in R
Calculate rui by finding the dot product of pu and qi found for predicting the rating rui
Output : estimated rating of the movie by given user

FIGURE 3.13: **Collaborative Filtering Based Recommendation System**

3.4.2 Implementation

There are two methods of implementing Collaborative Filtering :

3.4.2.1 Memory-Based Algorithm

1. In this strategy, we discover resemble the other alike clients dependent on comparability and prescribe motion pictures which first client's carbon copy has chosen before. This calculation is compelling yet expends a great deal of assets and time. It requires to process each client pair data which takes a ton of time. Consequently, for huge base stages, this calculation is hard to execute without an extremely solid parallelizable framework.

2. It is fundamentally the same as past calculation, yet as opposed to discovering client's clone, we have a go at discovering motion picture's copy. When we have film's clone grid, we can without much of a stretch prescribe comparative motion pictures to client who have appraised any motion picture from the dataset. This calculation is far less asset expending than client synergistic sifting. Thus, for another client, the calculation takes far lesser time than client team up as we needn't bother with all comparability scores between the clients. What's more, with a fixed number of motion pictures, film motion picture resemble the other alike lattice is fixed after some time.

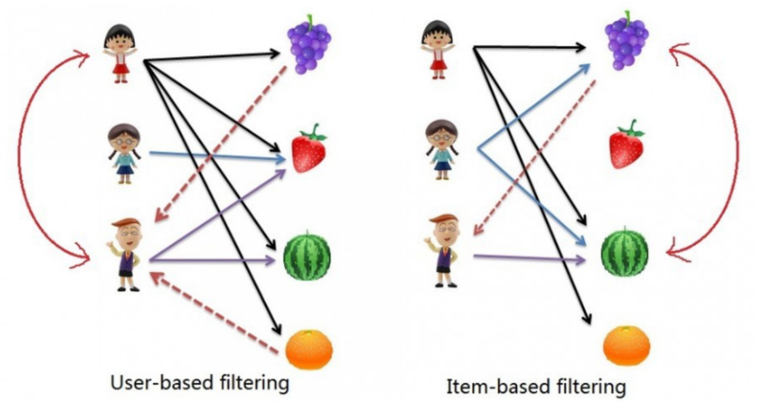


FIGURE 3.14: **User-based and Item-based Collaborative Filtering**

Memory-Based approach has the following drawbacks :

- It doesn't address the well-known cold-start problem, that is when new user or new item enters the system.
- It can't deal with sparse data, meaning it's hard to find users that have rated the same items.
- It suffers when new users or items that don't have any ratings enter the system.
- It tends to recommend popular items only.

3.4.2.2 Model-Based Algorithm

Model-based Collaborative Filtering is based on matrix factorization which has received greater exposure, mainly as an unsupervised learning method for dimensionality reduction and latent variable decomposition. Matrix factorization is widely used for recommendation systems where it can deal better with scalability and sparsity than Memory-based Collaborative Filtering:

- The goal of Matrix Factorization is to learn the latent preferences of users and the latent attributes of items from known ratings (learn the features that describe the characteristics of ratings) to then predict the unknown ratings through the dot product of the latent features of users and items.
- When we have a very sparse matrix, with a lot of dimensions, by applying matrix factorization, we can restructure the user-item matrix into low-rank structure, and we can represent the matrix by the multiplication of two low-rank matrices, where the rows contain the latent vector.
- We fit this matrix to approximate your original matrix, as closely as possible, by multiplying the low-rank matrices together, which fills in the entries missing in the original matrix.

A well-known matrix factorization method is Singular value decomposition (SVD). At a high level, SVD is an algorithm that decomposes a matrix A into the best lower rank (i.e. smaller/simpler) approximation of the original matrix A.
We have therefore used Matrix Factorization with Singular Value Decomposition in our implementation of the Collaborative Filtering Based Recommendation System.

3.4.2.3 Procedure

- R = M σ U
- If R is an m*n matrix then
  - M is an m*r orthogonal matrix
  - σ is an r*r diagonal matrix with non-negative real numbers on the diagonal
  - U is an r*n orthogonal matrix
- Now, $r_{ui} = p_u.q_i$
  - r_{ui} represent the rating of the movie
  - p_u represents the affinity of user u for each of the latent factors
  - q_i represents the affinity of the item i for the latent factors
- Randomly initialize all vectors p_u and q_i
- For given number of times, repeat: (no of iterations)
  - For all known rating r_{ui} , repeat :

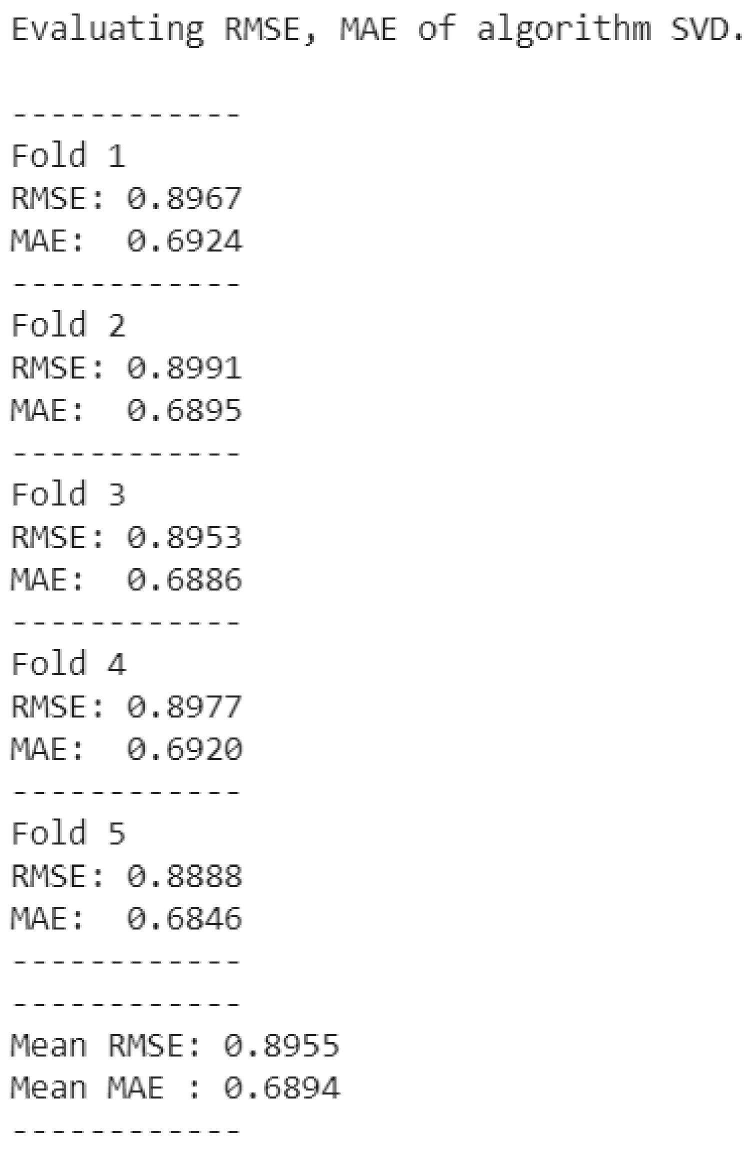
$$p_u = p_u + c * q_i(r_{ui} - p_u * q_i)$$

$$q_i = q_i + c * p_u(r_{ui} - p_u * q_i)$$

- Once all the vectors p_u and q_i have been computed, we can estimate all the ratings we want using the formula as given above.

3.4.3 Result

As shown below, the mean Root Mean Square Error for predicted ratings for a given user comes to around 0.8955 which is a good result of estimated ratings.

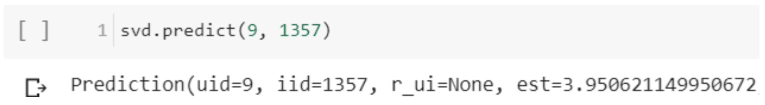


```
Evaluating RMSE, MAE of algorithm SVD.

------------
Fold 1
RMSE: 0.8967
MAE:  0.6924
------------
Fold 2
RMSE: 0.8991
MAE:  0.6895
------------
Fold 3
RMSE: 0.8953
MAE:  0.6886
------------
Fold 4
RMSE: 0.8977
MAE:  0.6920
------------
Fold 5
RMSE: 0.8888
MAE:  0.6846
------------
------------
Mean RMSE: 0.8955
Mean MAE : 0.6894
------------
```

FIGURE 3.15: **RMSE and MAE calculation using SVD**

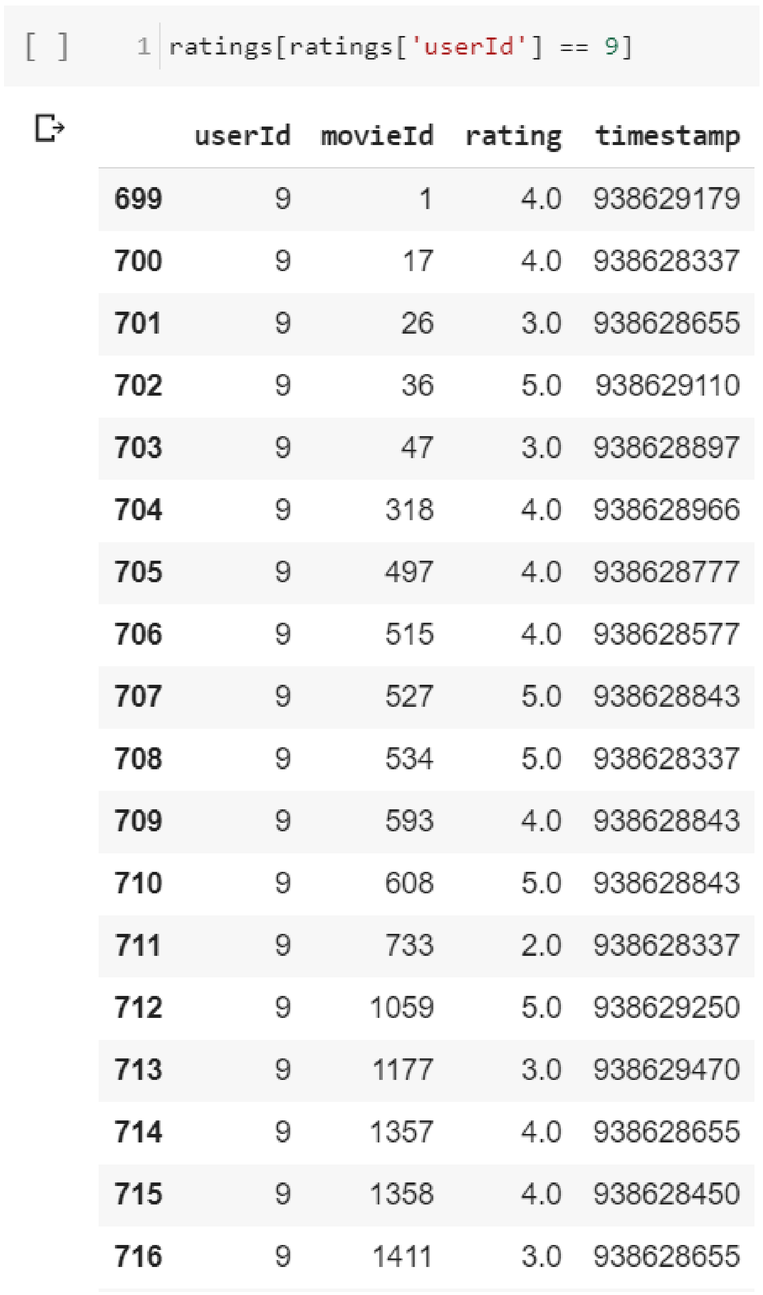
Estimated prediction of rating that user with userId = 9 will give to movie with movieId = 1357 is shown as an example to show the closeness in estimated ratings and actual ratings.



```
[ ]  svd.predict(9, 1357)

Prediction(uid=9, iid=1357, r_ui=None, est=3.950621149950672
```

FIGURE 3.16: **Estimated rating output using SVD for Collaborative Filtering**



```
[ ]  ratings[ratings['userId'] == 9]
```

```
     userId  movieId  rating  timestamp
699       9        1     4.0  938629179
700       9       17     4.0  938628337
701       9       26     3.0  938628655
702       9       36     5.0  938629110
703       9       47     3.0  938628897
704       9      318     4.0  938628966
705       9      497     4.0  938628777
706       9      515     4.0  938628577
707       9      527     5.0  938628843
708       9      534     5.0  938628337
709       9      593     4.0  938628843
710       9      608     5.0  938628843
711       9      733     2.0  938628337
712       9     1059     5.0  938629250
713       9     1177     3.0  938629470
714       9     1357     4.0  938628655
715       9     1358     4.0  938628450
716       9     1411     3.0  938628655
```

FIGURE 3.17: **Actual ratings given by user 9 to a set of movies for cross-referencing**

One important feature of this recommendation system is that it doesn't care what the movie is (or what it contains). It works purely on the basis of an assigned movie ID and tries to predict ratings based on how the other users have perceived the movie.

3.4.4 Limitations

This recommendation system suffers from the ‘cold start’ problem, where if a new user is introduced to the system, then similarity with other users can’t be calculated and hence the result will not be accurate.

3.5 Combined Recommendation System

In this recommendation system, we have combined the above three approaches, namely, Popularity Based, Content Based and Collaborative Filtering using Singular Value Decomposition.
The Hybrid Recommendation System thus overcomes the drawbacks of both Content Based and Collaborative Filtering methods.
There will not be any 'cold start' problem as it occurs in Collaborative Filtering approach and it will also be able to give personalised recommendations to user which the Content Based Recommendation System cannot do.

3.5.1 Design

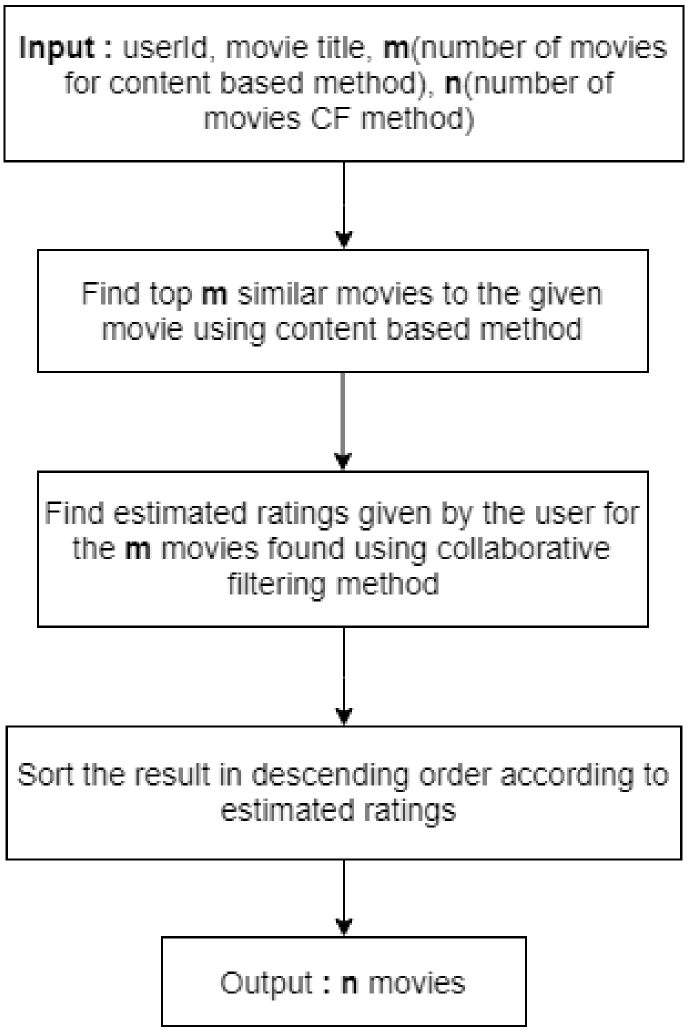

Input : userId, movie title, m(number of movies for content based method), n(number of movies CF method)
Find top m similar movies to the given movie using content based method
Find estimated ratings given by the user for the m movies found using collaborative filtering method
Sort the result in descending order according to estimated ratings
Output : n movies

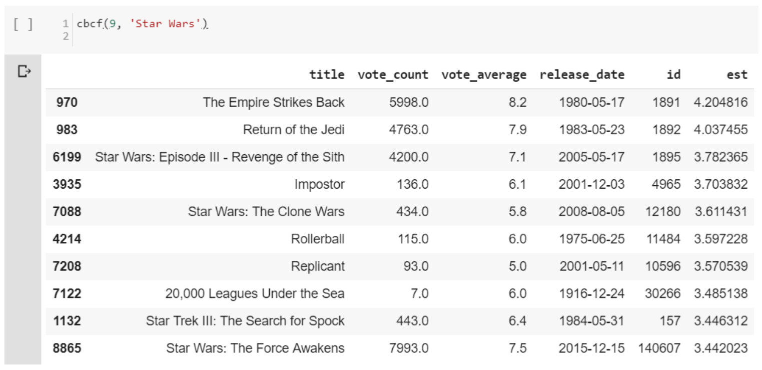
Figure 3.18: **Combined Recommendation System**

3.5.2 Implementation

- Using the Movie Title, we have applied content-based recommendation and stored top 25 (or more) similar movies
- Then we have calculated the estimated ratings the given user will give to the 25 (or more) most similar movies that is stored in the previous step.
- Finally we sort the stored similar movies on basis of estimated ratings in descending order and display result.

3.5.3 Result

We have to input the userId of the user for whom we need to recommend movies. The user also specifies the title of a single movie similar to which he/she would like to get recommendations.



```
cbcf(9, 'Star Wars')
```

```
                                           title  vote_count  vote_average  release_date      id       est
 970                     The Empire Strikes Back      5998.0           8.2    1980-05-17    1891  4.204816
 983                          Return of the Jedi      4763.0           7.9    1983-05-23    1892  4.037455
6199  Star Wars: Episode III - Revenge of the Sith    4200.0           7.1    2005-05-17    1895  3.782365
3935                                    Impostor       136.0           6.1    2001-12-03    4965  3.703832
7088                     Star Wars: The Clone Wars     434.0           5.8    2008-08-05   12180  3.611431
4214                                  Rollerball       115.0           6.0    1975-06-25   11484  3.597228
7208                                   Replicant        93.0           5.0    2001-05-11   10596  3.570539
7122                 20,000 Leagues Under the Sea        7.0           6.0    1916-12-24   30266  3.485138
1132             Star Trek III: The Search for Spock   443.0           6.4    1984-05-31     157  3.446312
8865                  Star Wars: The Force Awakens    7993.0           7.5    2015-12-15  140607  3.442023
```

FIGURE 3.19: **Combined Recommender output**

On carrying out a qualitative analysis on the result above we can see that out of the ten movies that are recommended, five movies are from the 'Star Wars' series. Also all the movies that are recommended are rated above 3.44/5.0 which is a fair rating.
Thus, we conclude our implementation of Movie Analysis and Recommendation System here.

Chapter 4

Tools and Technologies

4.1 Python

Python is a broadly utilized abnormal state, universally useful, dynamic, and translated programming language. Its structure logic underlines code coherence, and its grammar enables developers to express ideas in less lines of code than would be conceivable in dialects, for example, C++ or Java. The language gives builds expected to empower clear projects on both a little and extensive scale. Python bolsters various programming ideal models, including object-situated, basic and practical programming or procedural styles. It includes a dynamic sort framework and programmed memory the board and has a substantial and far reaching standard library. Python translators are accessible for establishment on many working frameworks, permitting Python code execution on a wide assortment of frameworks. We have utilized python 3 for structure our venture.

4.1.1 Scripting Language

A scripting or content language is a programming language that underpins contents, programs composed for an extraordinary run-time condition that computerize the execution of assignments that could on the other hand be executed one-by-one by a human administrator. Scripting dialects are regularly deciphered (as opposed to ordered). Natives are typically the rudimentary assignments or API calls, and the language enables them to be joined into increasingly complex projects. Conditions that can be robotized through scripting incorporate programming applications, website pages inside an internet browser, the shells of working frameworks (OS), installed frameworks, just as various amusements.

A scripting language can be seen as a space explicit language for a specific domain; on account of scripting an application, this is otherwise called an augmentation language. Scripting dialects are likewise now and then alluded to as abnormal state programming dialects, as they work at an abnormal state of reflection, or as control dialects.

4.1.2 Object Oriented Programming Language

Item situated writing computer programs is a programming worldview dependent on the idea of ”objects”, which may contain information, as fields, frequently known as properties; and code, as methodology, regularly known as techniques. A distinctive component of articles is that an item’s systems can get to and regularly alter the information fields of the item with which they are related (objects have an idea of ”this” or ”self”). In OO programming, PC programs are planned by making them out of articles that collaborate with each other. There is huge decent variety in item situated programming, yet most well known dialects are class-based, implying that objects are occurrences of classes, which commonly likewise decides their sort.

4.2 Google Colaboratory

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

4.2.1 Why Google Colab

As the name suggests, Google Colab comes with collaboration backed in the product. In fact, it is a Jupyter notebook that leverages Google Docs collaboration features. It also runs on Google servers and we don’t need to install anything. Moreover, the notebooks are saved to our Google Drive account. We have mainly used Google Colab because of it provides a free GPU as well.

4.2.2 Jupyter Notebooks

The Jupyter Notebook is an open source web application that we can use to make and share records that contain live code, conditions, representations, and content. Jupyter

Notebooks are a turn off task from the IPython venture, which used to have an IPython Notebook venture itself. The name, Jupyter, originates from the center bolstered programming dialects that it underpins: Julia, Python, and R. Jupyter ships with the IPython part, which enables us to compose our projects in Python, however there are as of now more than 100 different bits that we can likewise utilize.

4.2.3 Google Colab v/s Jupyter Notebooks

Infrastructure - Google Colab runs on Google Cloud Platform (GCP). Hence it is robust and flexible.
Hardware - Google Colab recently added support for Tensor Processing Unit(TPU) apart from its existing GPU and CPU instances. So, it's a big deal for all deep learning people.
Pricing - Despite being so good at hardware, the services provided by Google Colab are completely free. This makes it even more awesome.
Integration with Google Drive - Yes, we can use our google drive as an interactive file system with Google Colab. This makes it easy to deal with larger files while computing our stuff.
Boon for Research and Startup Community - Perhaps this is the only tool available in the market which provides such a good PaaS (Platform as a Service) for free to users. This is overwhelmingly helpful for startups, the research community and students in deep learning space.

It is much eaiser to start running code on Google Colab where as Jupyter Notebook has to be installed and set up first.

Chapter 5

Conclusion and Future Scope

We have explained in detail in chapter 3, how we built our Movie Recommendation system and why we applied certain tweaks in each algorithm.
The final result obtained as shown in section 3.5.3 is a qualitative proof of how our Recommendation System combines the best results and advantages of both Content Based and Collaborative Filtering and mitigates the drawbacks and limitations faced by each algorithm applied separately.
Our implementation requires a lot of memory for the large data set provided mainly due to calculation of cosine similarities.
Our Recommendation system can handle data up to twenty thousand movies after which the system crashes.
This problem can be handled if the code can be converted so that it can run on distributed systems using either Hadoop or Spark.

Bibliography

1. https://www.themoviedb.org/?language=en-US
2. https://www.themoviedb.org/documentation/api
3. https://grouplens.org/datasets/movielens/latest/
4. https://www.kaggle.com
5. https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV#
6. https://medium.com/@cfpinela/content-based-recommender-systems-a68c2aee2235
7. https://en.wikipedia.org/wiki/Collaborative_filtering
8. http://nicolas-hug.com/tags#matrix-factorization