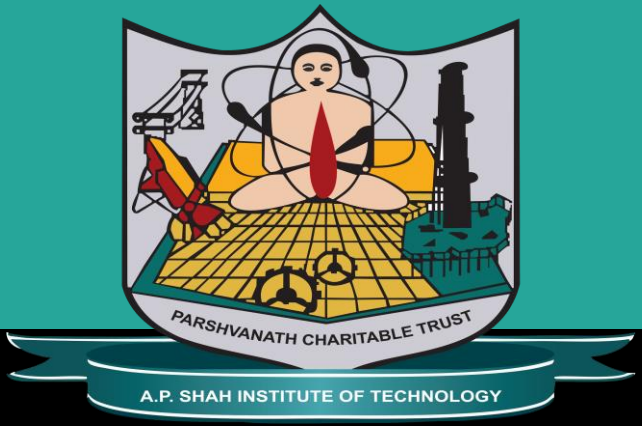


Computer Engineering Department

A.P. Shah Institute of Technology

G.B.Road,Kasarvadavali, Thane(W), Mumbai-400615

UNIVERSITY OF MUMBAI

Academic Year 2020-2021

A Project Report on

Movie Recommendation System

Submitted in partial fulfillment of the degree of
Bachelor of Engineering(Sem-8)

in

Computer Engineering

By

Pritamkumar Jain (16102048)
Sayyam Shah (17202009)
Prathamesh Sherkar (16102029)
Ranjeet Singh (17102074)

Under the Guidance of
Prof. Amol Kalugade

1.Project Conception and Initiation

1.1 Abstract

- With the development of mobile Internet, the TV industry is facing threats and challenges. This is because Big Data is changing the industry. The primary task of TV industry like Netflix is how to take the advantage of Big Data technology.
- For Netflix programs, audience rating is the metrics whether the program is good or not. The more time the audience is watching the particular show,the more popular the show is for the Audience.
- This paper proposes a movie recommendation system. The system is based on Big Data technology and content based recommendation technique which can automatically push programs to audience according to their interest.

1.2 Objectives

- The primary objective is to build an algorithm that can predict similar movies according to user’s interest.
- After building the algorithm we will be making an website to deploy the algorithm on the web and to make the algorithm user friendly.

1.3 Literature Review

1.TV program recommendation system based on big data: DOI: 10.1109/ICIS.2016.7550923 : There are errors of program ratings recommendation system, and the program list is affected by human emotion as well. Our Program Recommended system based on Big Data reasonably gives solution to those drawbacks.

To apply Big Data technology into TV programs recommendation, the core work is to use data mining analysis algorithms on the massive database. One of the important things is that the diversity of television programs makes recommendation algorithms different. For example, for news, current affairs and drama series we need to analyze the audience's watching characteristic respectively. Hence, we can analyze the program's features as follows:

1)Program ratings.
2)Television ratings.
3)Program type.
4)Program broadcast time.

2.Verma J P, Patel B, Patel A. Big Data Analysis: Recommendation System with Hadoop Framework[C]//Computational Intelligence Communication Technology (CICT), 2015 IEEE International Conference on. IEEE, 2015: 92-97.

The growth of the technology and the big usage of recommendation system in many systems like in learning system, tourism system, and e-commerce system gives focus on the techniques used in those system development. Recommendation systems are defined as a software tool and techniques which providing advice for item to a user. The suggestions are like what music to listen, what online news to read etc. Recommendation system is used for finding the needed information from wider information available on the internet. Recommendation system mainly uses three approaches content based recommendation system, collaborative filtering recommendation system and hybrid recommendation system.

1.4 Problem Definition

To Build a recommendation system website in which if a user watch a particular movie then the system must recommend next top 10 movies which is similar to the movie user has watched.

1.5 Scope

This paper proposes a recommendation system, which can improve audience rating. In this system we have used data set of 5000 movies. This system uses two type of recommendation system 1: Demographic 2: Content Based. Demographic system uses IMDB formula to find top popular movies next in this system we use CountVectorizer and cosine similarity to find movies similar to users likes.

1.6 Technology stack

- Google Colab is been used to perform the machine Learning Algorithm in which are doing unsupervised Learning and by Content based filtering technique we will be recommending the next five to ten movies based on the movie which was watched by user.
- Python Flask will be used to make the web framework which will fetch the data from the colab and display that data to the website and will help the site look interactive and easy to communicate and html, css will be used to make the site attractive to the end user.

1.7 Benefits for environment & Society

1. <u>Benefits for environment</u> - Recommender systems help the users to get personalized recommendations, helps users to take correct decisions and redefine the users web browsing experience, retain the customers, enhance their experience.Recommendation engines provide personalization and helps to reduce the stress by finding similar movies easily.

2. <u>Benefits for society</u> - By using this recommendation system the cable operators having there default channel which are free to the users can start showing them the popular movies on that channel .By this way the cable operators will get the money which will be generated by the advertisement and the end users will not have to pay extra money for getting the channels which are showing similar movies.

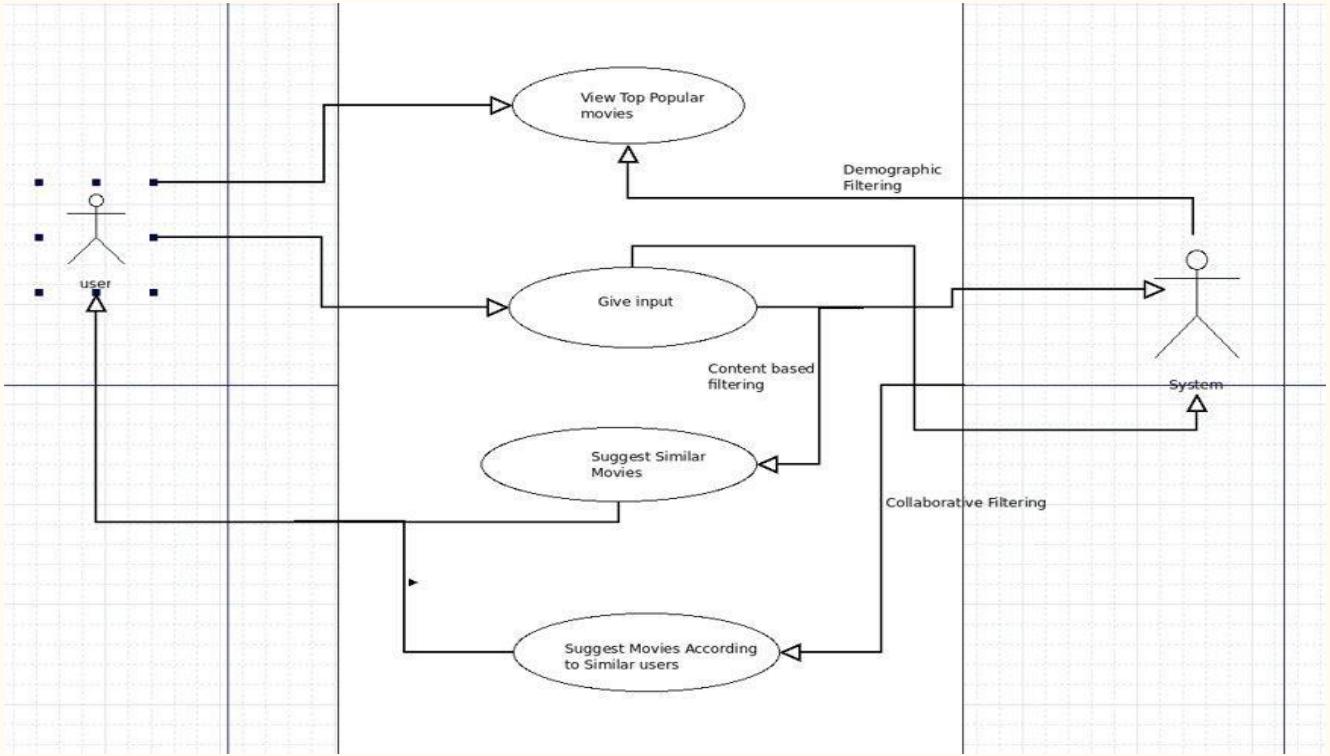
2. Project Design

2.1 Proposed System

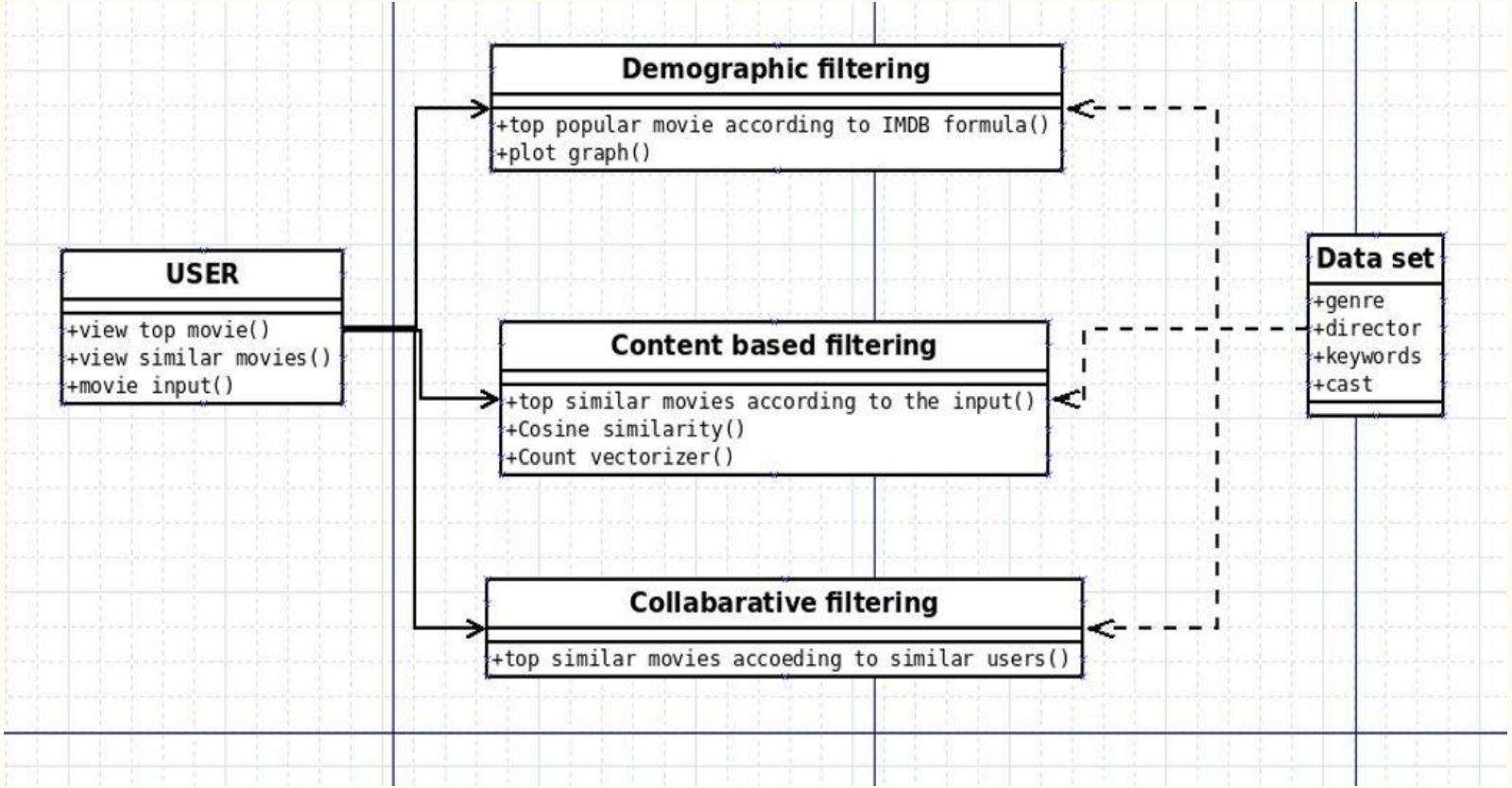
We propose a recommendation system in which at first we will build an system using content based filtering to recommend movies. we will also make use of uipath tool to take run time data into excel sheet to get latest movie dataset and after that we will make a website to make the recommendation system user friendly.

2.2 Design(Flow Of Modules)

2.2.1 Use Case Diagram :



2.2.2 Class Diagram :



3.Implementation

3.1 Proposed system

Our model is a movie recommendation system in which at first we will build a system using content based filtering to recommend movies. We will also make use of uipath tool to take run time data into excel sheet to get latest movie data set and followed by that we will make a website to make the recommendation system user friendly.

3.1.1 Platforms for execution

Google Colab

4. Results :

\+ Code + Text

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

```
[ ] from google.colab import files
    uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
[2] url = 'https://raw.githubusercontent.com/pritam123-jain/Final-year/master/movie_dataset.csv' #fetching dataset from github
    df = pd.read_csv(url) #Reading the file
```

```
[3] df.head(1) #displaying Head
```

	index	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	productic
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_ "name":

\+ Code + Text RAM Disk Editing

[3] Fiction colony so... di...

```
[ ] df.shape  #listing no of row and coloumn
```

```
(4803, 24)
```

```
[4] lst = [x.upper() for x in df.original_title] #Creating list of all the title in upper case
    print(lst)
```

```
['AVATAR', "PIRATES OF THE CARIBBEAN: AT WORLD'S END", 'SPECTRE', 'THE DARK KNIGHT RISES', 'JOHN CARTER', 'SPIDER-MAN 3', 'TANGLED', 'AVENGERS: AGE OF ULTRON', 'H
```

```
[5] df.insert(2,"real_title",lst)  #inserting new coloum with names in upper case from the list
    #df.drop(columns="real_title").head(1)
```

```
[6] df.head(1) # displaying top 1 dataset values
```

	index	budget	real_title	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companie
0	0	237000000	AVATAR	Action Adventure Fantasy Science	http://www.avatarmovie.com/	19995	culture clash future space war	en	Avatar	In the 22nd century, a paraplegic	150.437577	[{"name": "Ingeniou Film Partners", "ic 289

+ Code + Text

RAM Disk Editing

```
[5] df.insert(2,"real_title",lst)  #inserting new coloum with names in upper case from the list
    #df.drop(columns="real_title").head(1)
```

```
[6] df.head(1) # displaying top 1 dataset values
```

	index	budget	real_title	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companie
0	0	237000000	AVATAR	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingeniou Film Partners", "ic 289

```
df.columns # show coloum
```

```
Index(['index', 'budget', 'real_title', 'genres', 'homepage', 'id', 'keywords',
       'original_language', 'original_title', 'overview', 'popularity',
       'production_companies', 'production_countries', 'release_date',
       'revenue', 'runtime', 'spoken_languages', 'status', 'tagline', 'title',
       'vote_average', 'vote_count', 'cast', 'crew', 'director'],
      dtype='object')
```

```
[8] coloums = ['real_title', 'cast', 'director', 'genres','vote_average','vote_count'] # filtering by requirement
```

+ Code + Text

RAM Disk Editing

```
[8] coloums = ['real_title', 'cast', 'director', 'genres','vote_average','vote_count'] # filtering by requirement
    df[coloums].head(2)
```

	real_title	cast	director	genres	vote_average	vote_count
0	AVATAR	Sam Worthington Zoe Saldana Sigourney Weaver S...	James Cameron	Action Adventure Fantasy Science Fiction	7.2	11800
1	PIRATES OF THE CARIBBEAN: AT WORLD'S END	Johnny Depp Orlando Bloom Keira Knightley Stel...	Gore Verbinski	Adventure Fantasy Action	6.9	4500

```
[9] df.shape
```

```
(4803, 25)
```

```
[10] df[coloums].isnull().values.any() # checking null value if any = true else false
```

```
True
```

```
df1 = df.dropna() #droping n/a values
```

```
[12] df1.head(1)
```

	index	budget	real_title	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companie

+ Code + Text

RAM Disk

Editing

```
[11] df1 = df.dropna() #droping n/a values
```

```
[12] df1.head(1)
```

	index	budget	real_title	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companie
0	0	237000000	AVATAR	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingeniou Film Partners", "ic 289

```
[13] df1.shape
```

```
(1432, 25)
```

```
[14] C = df1['vote_average'].mean() # total votes average
     print(C)
```

```
6.318156424581003
```

```
[15] m = df1['vote_count'].quantile(0.8)
```

+ Code + Text

RAM Disk

Editing

```
[15] m = df1['vote_count'].quantile(0.8)
     print(m)
```

```
2231.3999999999996
```

```
[16] filter = df1.copy().loc[df1['vote_count'] >= m]
     filter.shape
```

```
(287, 25)
```

```
[17] def weighted_rating(x, m=m, C=C):
         v = x['vote_count']
         R = x['vote_average']
         return (v/(v+m) * R) + (m/(m+v) * C)
     #true Bayesian estimate = weighted rating
     #R = average for the movie (mean) = (Rating)
     #v = number of votes for the movie = (votes)
     #m = minimum votes required to be listed in the Top
     #C = the mean vote across the whole report
```

```
filter['rating'] = filter.apply(weighted_rating, axis="columns")
filter['rating'].head()
```

```
0    7.059761
1    6.707124
2    6.306049
```

+ Code + Text

RAM Disk Editing

```
[18] filter['rating'] = filter.apply(weighted_rating, axis="columns")
     filter['rating'].head()
```

```
0    7.059761
1    6.707124
2    6.306049
3    7.347711
5    6.060670
Name: rating, dtype: float64
```

```
filter = filter.sort_values('rating', ascending=False)
#Print the movies
filter[['original_title', 'vote_count', 'vote_average', 'popularity', 'rating']].head(10)
```

	original_title	vote_count	vote_average	popularity	rating
662	Fight Club	9413	8.3	146.757391	7.920222
65	The Dark Knight	12002	8.2	187.322927	7.904979
96	Inception	13752	8.1	167.583710	7.851242
3337	The Godfather	5893	8.4	143.659698	7.828213
95	Interstellar	10867	8.1	724.247784	7.796451
329	The Lord of the Rings: The Return of the King	8064	8.1	123.630332	7.713808
1990	The Empire Strikes Back	5879	8.2	78.517830	7.682252

+ Code + Text

RAM Disk Editing

```
[19] filter = filter.sort_values('rating', ascending=False)
     #Print the movies
     filter[['original_title', 'vote_count', 'vote_average', 'popularity', 'rating']].head(10)
```

	original_title	vote_count	vote_average	popularity	rating
662	Fight Club	9413	8.3	146.757391	7.920222
65	The Dark Knight	12002	8.2	187.322927	7.904979
96	Inception	13752	8.1	167.583710	7.851242
3337	The Godfather	5893	8.4	143.659698	7.828213
95	Interstellar	10867	8.1	724.247784	7.796451
329	The Lord of the Rings: The Return of the King	8064	8.1	123.630332	7.713808
1990	The Empire Strikes Back	5879	8.2	78.517830	7.682252
262	The Lord of the Rings: The Fellowship of the Ring	8705	8.0	138.049577	7.656846
2912	Star Wars	6624	8.1	126.393695	7.651008
1818	Schindler's List	4329	8.3	104.469351	7.625912

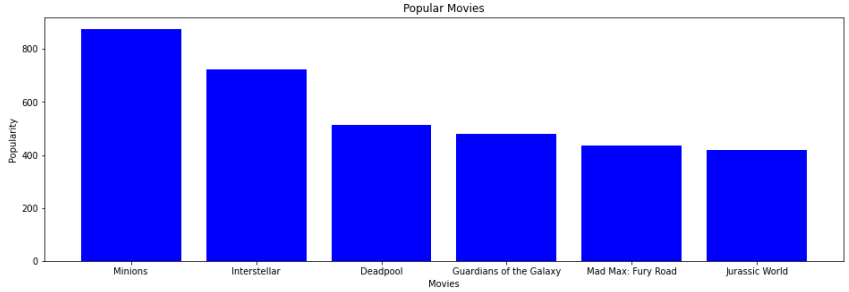
```
pop = df1.sort_values('popularity', ascending=False)
import matplotlib.pyplot as plt
plt.figure(figsize=(16,5))
plt.barh(pop['title'].head(6),pop['popularity'].head(6), align='center', color='blue')
```

+ Code + Text

RAM Disk Editing

```
[20] pop = df1.sort_values('popularity', ascending=False)
     import matplotlib.pyplot as plt
     plt.figure(figsize=(16,5))
     plt.bar(pop['title'].head(6),pop['popularity'].head(6), align='center',color='blue')
     plt.xlabel('Movies')
     plt.ylabel("Popularity")
     plt.title("Popular Movies")
```

```
Text(0.5, 1.0, 'Popular Movies')
```



+ Code + Text

RAM Disk Editing

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

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

```
[23] def combine_features(row):
         return row['keywords']+" "+row['cast']+" "+row['genres']+" "+row['director']
```

```
[24] df["combined_features"] = df.apply(combine_features,axis=1)
     print(df['combined_features'].head(10))
```

```
0    culture clash future space war space colony so...
1    ocean drug abuse exotic island east india trad...
2    spy based on novel secret agent sequel mi6 Dan...
3    dc comics crime fighter terrorist secret ident...
4    based on novel mars medallion space travel pri...
5    dual identity amnesia sandstorm love of one's ...
6    hostage magic horse fairy tale musical Zachary...
7    marvel comic sequel superhero based on comic b...
8    witch magic broom school of witchcraft wizardr...
9    dc comics vigilante superhero based on comic b...
Name: combined_features, dtype: object
```

```
[25] cv = CountVectorizer() #count the no of text
     count = cv.fit_transform(df["combined_features"]) # combined strings(movie contents) to CountVectorizer() object
```

+ Code + Text

RAM Disk Editing

```
[25] cv = CountVectorizer() #count the no of text
     count = cv.fit_transform(df["combined_features"]) # combined strings(movie contents) to CountVectorizer() object
```

```
[26] cosine_sim = cosine_similarity(count)
     print (cosine_sim)
```

```
[[1.         0.10540926 0.12038585 ... 0.         0.         0.        ]
 [0.10540926 1.         0.0761387  ... 0.03651484 0.         0.        ]
 [0.12038585 0.0761387  1.         ... 0.         0.11145564 0.        ]
 ...
 [0.         0.03651484 0.         ... 1.         0.         0.04264014]
 [0.         0.         0.11145564 ... 0.         1.         0.        ]
 [0.         0.         0.         ... 0.04264014 0.         1.        ]]
```

```
[27] cosine_sim.shape
```

```
(4803, 4803)
```

```
[28] movie_user_likes = input("Enter movie name-").upper()
     print('You Selected-'+movie_user_likes)
```

```
Enter movie name-deadpool
You Selected-DEADPOOL
```

```
[31] def get_title_from_index(index):
```

+ Code + Text

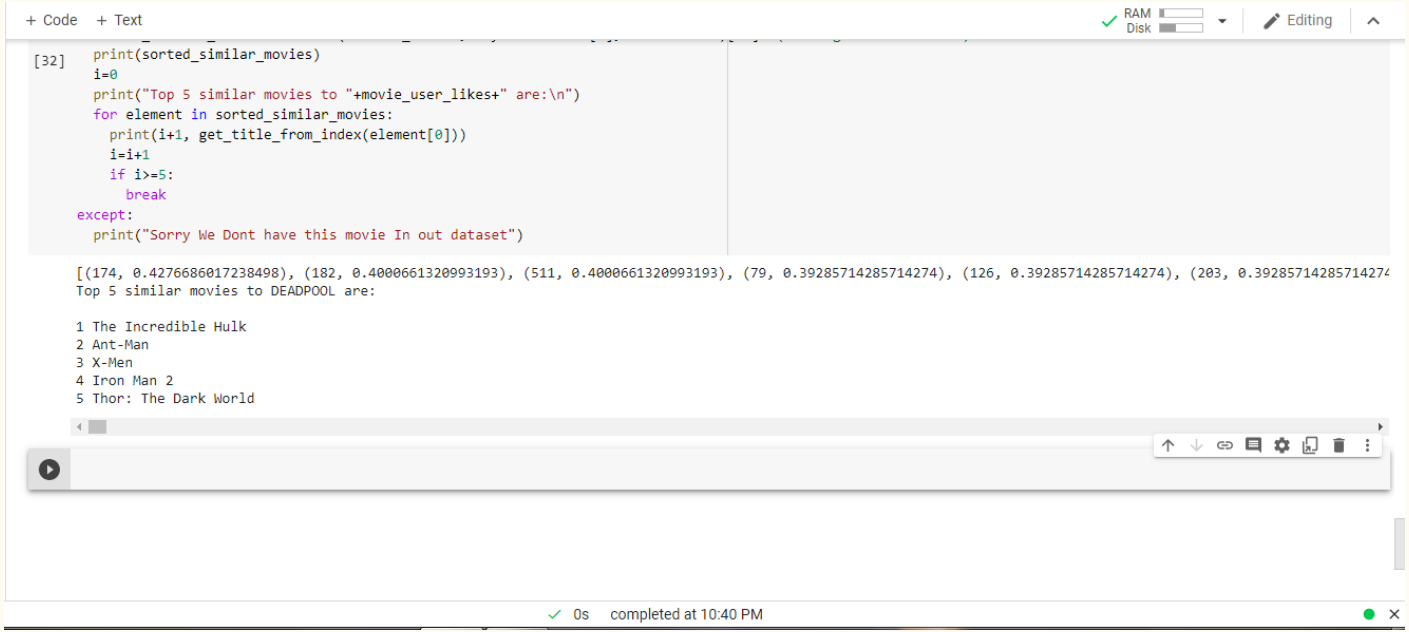
RAM Disk Editing

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

```
try:
  df_index = get_index_from_title(movie_user_likes)
  similar_movies = list(enumerate(cosine_sim[df_index])) #(movie_id,similarity)
  sorted_similar_movies = sorted(similar_movies, key=lambda x:x[1],reverse=True)[1:] #(sorting similar movies)
  print(sorted_similar_movies)
  i=0
  print("Top 5 similar movies to "+movie_user_likes+" are:\n")
  for element in sorted_similar_movies:
    print(i+1, get_title_from_index(element[0]))
    i=i+1
    if i>=5:
      break
except:
  print("Sorry We Dont have this movie In out dataset")
```

```
[(174, 0.4276686017238498), (182, 0.4000661320993193), (511, 0.4000661320993193), (79, 0.3928571428571274), (126, 0.3928571428571274), (203, 0.3928571428571274
Top 5 similar movies to DEADPOOL are:

1 The Incredible Hulk
2 Ant-Man
3 X-Men
4 Iron Man 2
```


+ Code
+ Text
RAM
Disk
Editing
[32]
print(sorted_similar_movies)
i=0
print("Top 5 similar movies to "+movie_user_likes+" are:\n")
for element in sorted_similar_movies:
print(i+1, get_title_from_index(element[0]))
i=i+1
if i>=5:
break
except:
print("Sorry We Dont have this movie In out dataset")
[(174, 0.4276686017238498), (182, 0.4000661320993193), (511, 0.4000661320993193), (79, 0.3928571428571427
Top 5 similar movies to DEADPOOL are:
1 The Incredible Hulk
2 Ant-Man
3 X-Men
4 Iron Man 2
5 Thor: The Dark World
0s
completed at 10:40 PM

5.Conclusion

Conclusion and future scope

We will be making use of the uipath to take real time data into the excel sheet for recommendation of recently released movies and after that we will be making an website by using python flask.

6. References

- Oh J, Sung Y, Kim J, et al. Time-Dependent User Profiling for TV Recommendation[C]//Cloud and Green Computing (CGC), 2012 Second International Conference on. IEEE, 2012: 783-787.
- Verma J P, Patel B, Patel A. Big Data Analysis: Recommendation System with Hadoop Framework[C]//Computational Intelligence & Communication Technology (CICT), 2015 IEEE International Conference on. IEEE, 2015: 92-97.

7. Bibliography

[1] Oh J, Sung Y, Kim J, et al. Time-Dependent User Profiling for TV Recommendation[C]//Cloud and Green Computing (CGC), 2012 Second International Conference on. IEEE, 2012: 783-787.

[2] Verma J P, Patel B, Patel A. Big Data Analysis: Recommendation System with Hadoop Framework[C]//Computational Intelligence Communication Technology (CICT), 2015 IEEE International Conference on. IEEE, 2015: 92- 97.

Thank You