MOVIE RECOMMANDATION SYSTEM

by

18BLC1156 Prerit Agrawal

18BLC1159 Abhyuday Mandloi Singh

A project report submitted to

Dr. SYED IBRAHIM

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfilment of the requirements for the course of

ECE3001-DATA ANALYTICS & VISUALISATION

in

B. Tech. ELECTRONICS AND COMMUNICATION ENGINEERING

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

NOVEMBER 2020

BONAFIDE CERTIFICATE

Certified that this project report entitled "**MOVIE RECOMMANDATION SYSTEM"** is a bonafide work of **Prerit Agrawal (18BLC1156) and Abhyuday Singh(18BLC1159) who** carried out the Project work under my supervision and guidance for **ECM3001-DATA ANALYTICS & VISUALISATION.**

Dr. SYED IBRAHIM

Assistance Professor

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

Our Movie recommendation system adds a whole new dimension to the movie watching experience by providing real-time personalized movie recommendations to users. It takes a collaborative social-networking approach where a user's own tastes are mixed with that of the entire community to generate meaningful results. Most existing movie services like IMDB (www.imdb.com) do not personalize their recommendations but simply provide an overall rating for a movie. This significantly decreases the value of each recommendation as it does not cater to the individual movie preferences of the user. Unlike these systems, Our Recommendation Engine will continually analyze individual user's movie preferences and recommend custom movie recommendations. The overall goal is to ease the movie discovery process. A movie recommendation is important in our social life due to its strength in providing enhanced entertainment. Such a system can suggest a set of movies to users based on their interest, or the popularities of the movies. Although, a set of movie recommendation systems have been proposed, most of these either cannot recommend a movie to the existing users efficiently or to a new user by any means. In our project we propose a movie recommendation system that has the ability to recommend movies to a new user as well as the others. It mines movie databases to collect all the important information, such as, popularity and attractiveness, required for recommendation

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Syed Ibrahim**, Assistant Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work. We are extremely grateful to Dr.Tiripurasundari Dean of Academics, VIT Chennai, for extending the facilities of the School towards our project and for his continuous support.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution

TABLE OF CONTENTS

SERIAL NO.		TITLE	PAGE NO.
		ABSTRACT	3
		ACKNOWLEDGEMENT	4
1		INTRODUCTION	6
	1.1	OBJECTIVES AND GOALS	6
	1.2	BENEFITS	6
	1.3	FEATURES	7
2		DESIGN	7
	2.1	BLOCK DIAGRAM	7
3	3.1	SOFTWARE –CODING AND ANALYSIS (SNAPSHOTS OF CODING AND RESULTS)	14
4		CONCLUSION AND FUTURE WORK	20
	4.1	RESULT, CONCLUSION AND INFERENCE	20
	4.2	FUTURE WORK	21
		COST	
5		REFERENCES	22
6		PHOTO GRAPH OF THE PROJECT ALONG WITH THE TEAM MEMBERS	23

1. INTRODUCTION

1.1 OBJECTIVES AND GOALS

First of all let me explain what do we mean by recommendation system before stating for objective -

A recommendation system generates a compiled list of items in which a user might be interested, in the reciprocity of their current selection of item(s). It expands users' suggestions without any disturbance or monotony, and it does not recommend items that the user already knows.

Objective

To build a recommendation system to predict a list of movies for users based on other movies likes or dislikes. This recommendation will be for every user and similar movie based on his/her unique interest.
We would be using collaborative filtering

The idea behind collaborative filtering is to consider users' opinions on different movies and recommend the best movies to each user based on the user's previous rankings and the opinion of other similar types of users.

We are using here two types of CF :
1.) User-based nearest neighbor.
2.)Item-based nearest neighbor.

Goals

First we will make training algorithm based on item to item based CF using cosine similarity and collaborative filtering then according to our dataset we will predict top movies based on item to item CF.Secondly, we will be using item to item based CF. On the basis of same movies liked by user ,we will built training model. Using our dataset we will find top recommended movies by item based filtering. We will than take top 9

movies from each method and merge them adjacently from consecutive filtering techniques.

1.2 BENEFITS

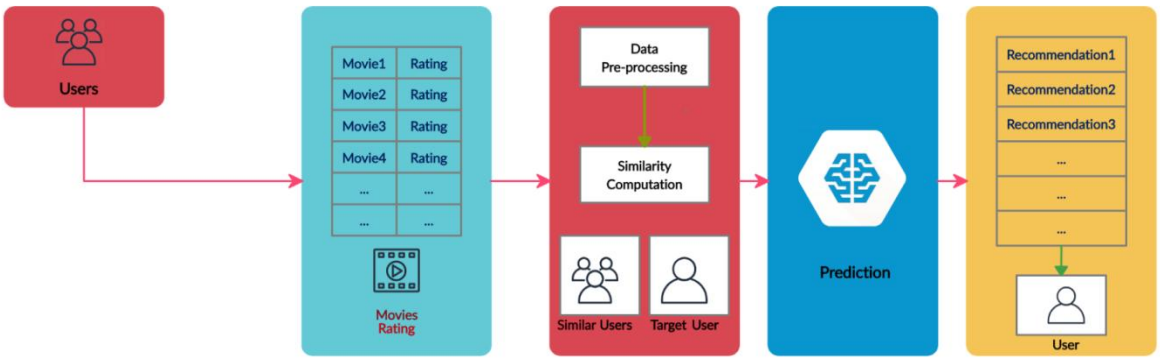
Capable of recommending unrated items.We can easily explain the working of recommender system by listing the Content features of an item. Also it is dependent on the relation between users which implies that it is content-independent. CF recommender systems can suggest serendipitous items by observing similar-minded people's behavior. They can make real quality assessment of items by considering other peoples experience.

1.3 FEATURES

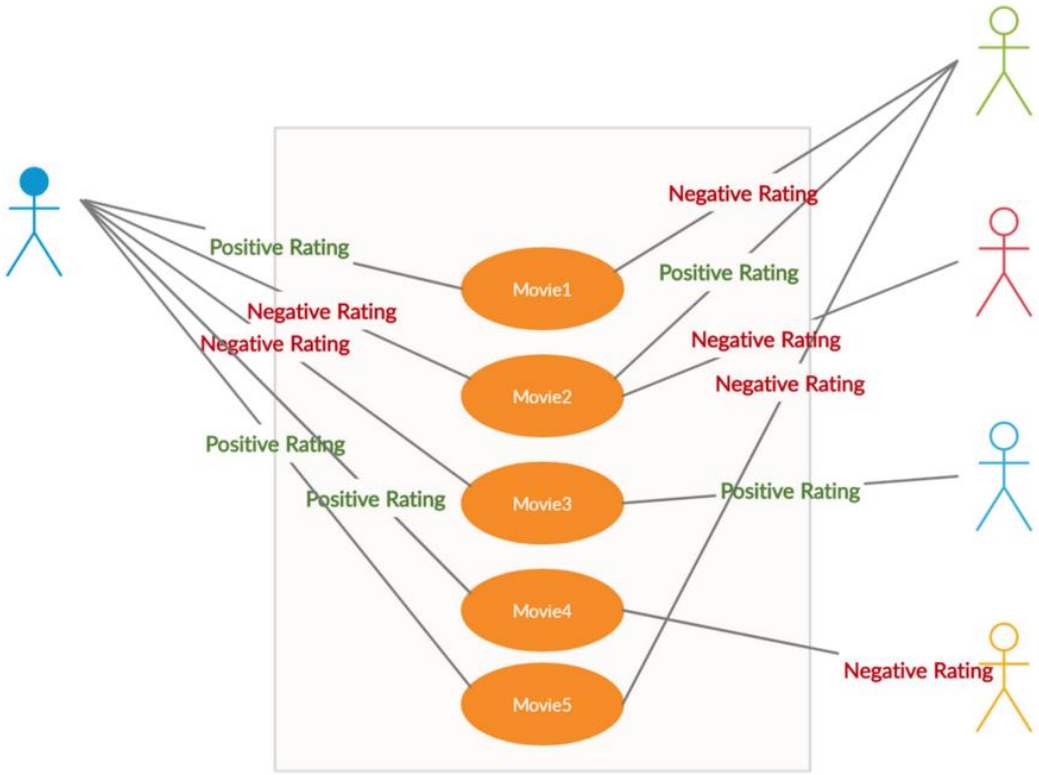
Quality as well as Quantity of data is important consider . Lesser time consumption (Compared to SVD or Random Algorithm). System has high efficiency. That's why Amazon with huge amounts of data uses this method

2. DESIGN

2.1 BLOCK DIAGRAM



For Collaborative Filtering



3. ANALYSIS

3.1 SOFTWARE CODING AND ANALYSIS

IMPORTING LIBRARIES

In [31]:
```
#  Data Manupulation
import numpy as np
import pandas as pd
# Plotting graphs
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import sparse
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import pairwise_distances
```

LOADING DATASET

In [32]:
```
ratings=pd.read_csv("ratings.csv")
```

In [33]:
```
ratings.head(50)
```

Out[33]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
5	1	70	3.0	964982400
6	1	101	5.0	964980868
7	1	110	4.0	964982176
8	1	151	5.0	964984041
9	1	157	5.0	964984100
10	1	163	5.0	964983650
11	1	216	5.0	964981208
12	1	223	3.0	964980985
13	1	231	5.0	964981179

MERGING MOVIE ID WITH ITS TITLE

In [47]:
```
ratings=pd.read_csv('ratings.csv')
movies=pd.read_csv('movies.csv')
ratings=pd.merge(movies,ratings)
ratings.head(50)
```

Out[47]:

	movieId	title	genres	userId	rating	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1	4.0	964982703
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	5	4.0	847434962
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	7	4.5	1106635946
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	15	2.5	1510577970
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	17	4.5	1305696483
5	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	18	3.5	1455209816
6	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	19	4.0	965705637
7	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	21	3.5	1407618878
8	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	27	3.0	962685262
9	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	31	5.0	850466616
10	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	32	3.0	856736119
11	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	33	3.0	939647444
12	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	40	5.0	832058959
13	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	43	5.0	848993983
14	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	44	3.0	869251860
15	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	45	4.0	951170182
16	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	46	5.0	834787906

DESCRIPTION ABOUT DATASET AND AVG RATING OF EACH MOVIE

In [48]:
```
ratings.describe()
```

Out[48]:

	movieId	userId	rating	timestamp
count	100783.000000	100783.000000	100783.000000	1.007830e+05
mean	19440.695802	326.104492	3.501483	1.205971e+09
std	35538.528370	182.663731	1.042667	2.163152e+08
min	1.000000	1.000000	0.500000	8.281246e+08
25%	1198.500000	177.000000	3.000000	1.018184e+09
50%	2991.000000	325.000000	3.500000	1.186087e+09
75%	8127.000000	477.000000	4.000000	1.435994e+09
max	193609.000000	610.000000	5.000000	1.537799e+09

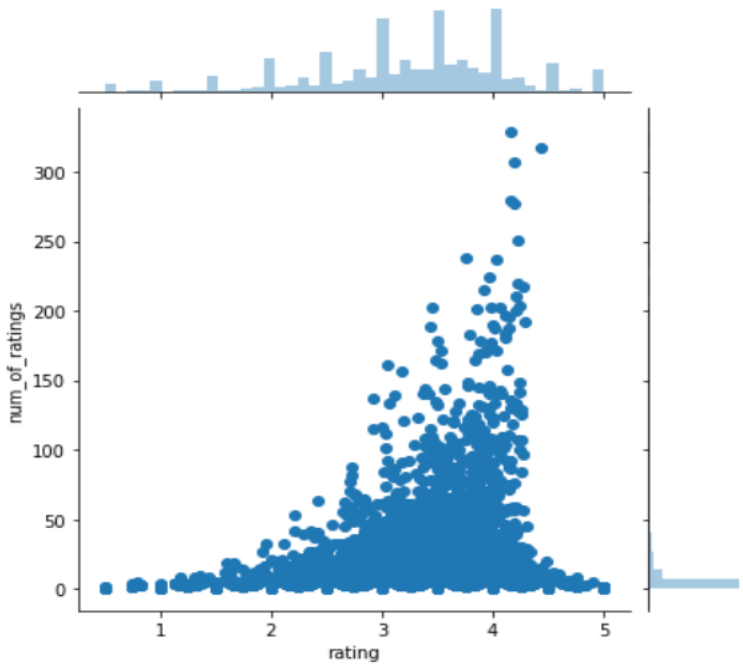
```
In [49]: rating = pd.DataFrame(ratings.groupby('movieId')['rating'].mean())
         rating.head(50)
```

Out[49]:

movieId	rating
1	3.920930
2	3.431818
3	3.259615
4	2.357143
5	3.071429
6	3.946078
7	3.185185
8	2.875000
9	3.125000
10	3.496212
11	3.671429
12	2.421053
13	3.125000

GRAPHICAL REPRESENTATION OF EACH RATING

```
In [51]: sns.jointplot(x='rating',y='num_of_ratings',data=rating)
Out[51]: <seaborn.axisgrid.JointGrid at 0x1df582711c8>
```



PIVOTING MOVIE'S AS ROW & USER'S AS COLUMN IN A MATRIX WITH RATING AS VALUE BY USER TO EACH MOVIE

```
In [52]: movie_matrix=ratings.pivot_table(index='userId',columns='title',values='rating')
         movie_matrix.head()
```

Out[52]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]² (2009)	[REC]³ 3 Génesis (2012)	anohana: The Flower We Saw That Day - The Movie (2013)	eXistenZ (1999)	x (20(
userId																		
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na

5 rows × 9719 columns

REPLACING RATING BY 0 WHERE USER HADN'T REACTED

```
In [52]: movie_matrix=ratings.pivot_table(index='userId',columns='title',values='rating')
         movie_matrix.head()
```

Out[52]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]² (2009)	[REC]³ 3 Génesis (2012)	anohana: The Flower We Saw That Day - The Movie (2013)	eXistenZ (1999)	x (20(
userId																		
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	Na

5 rows × 9719 columns

CONVERTING IT INTO ITEM BASED FILTERING , MAKING ROW AND COLUMN AS MOVIES

```
In [54]: item_similarity_df=movie_matrix.corr(method='pearson')
         item_similarity_df.head(50)
```

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)	Zodiac (2007)
'burbs, The (1989)	1.000000	0.063117	-0.023768	0.145444	0.011998	0.087931	0.224052	0.034223	0.009277	0.008331	...	0.017477	0.032470	0.134701	0.153158
(500) Days of Summer (2009)	0.063117	1.000000	0.142471	0.277510	0.193960	0.148903	0.142141	0.159756	0.135486	0.200135	...	0.374515	0.178655	0.068407	0.414585
10 Cloverfield Lane (2016)	-0.023768	0.142471	1.000000	-0.005014	0.112396	0.006139	-0.016835	0.031704	-0.024275	0.272943	...	0.242663	0.099059	-0.023477	0.272347
10 Things I Hate About You (1999)	0.145444	0.277510	-0.005014	1.000000	0.247414	0.226706	0.214580	0.013589	0.093553	0.044650	...	0.245977	0.106490	0.134328	0.093932

DEFINING FUNCTION TO GET SCORE ACCORDING TO COSINE SIMILARITY AND WE USED USER no.370 , AND PUTTED ITS DATA OF RATING IN TRAINING MODEL TO GET RECOMMENDATIONS

```
In [56]: def get_recom_movies(movie_name,user_rating):
             similar_score=item_similarity_df[movie_name]*(user_rating-2.5)
             similar_score=similar_score.sort_values(ascending=False)

             return similar_score
```

```
In [124]: movie_test=[("Clueless (1995)",3.5),
          ("Seven (a.k.a. Se7en) (1995)",3.5),
          ("Rumble in the Bronx (Hont faan kui) (1995)",3.5),
          ("Clerks (1994)",4.5),
          ("Interview with the Vampire: The Vampire Chronicles (1994)",3.5),
          ("Star Wars: Episode IV - A New Hope (1977)",2),
          ("Léon: The Professional (a.k.a. The Professional) (Léon) (1994)",3.5),
```

WE GET TOP RECOMMENDATION USING ITEM BASED FILTERING

```
recom_movies=pd.DataFrame()

for movies,rating in movie_test:
    recom_movies = recom_movies.append(get_recom_movies(movies,rating))
print("                        TOP RECOMMANDATION FOR YOU        ")
recom_movies.head()
recom_movies.sum().sort_values(ascending=False).head(60)
```

TOP RECOMMANDATION FOR YOU

```
)ut[124]: Terminator, The (1984)                                    10.626796
          Indiana Jones and the Temple of Doom (1984)               10.585880
          Indiana Jones and the Last Crusade (1989)                 10.495208
          Ferris Bueller's Day Off (1986)                           10.437058
          Breakfast Club, The (1985)                                10.118314
          Pulp Fiction (1994)                                       10.056241
          Monty Python and the Holy Grail (1975)                    10.046632
          Terminator 2: Judgment Day (1991)                         10.027793
          Clerks (1994)                                              9.910716
          Batman (1989)                                              9.886649
          Trainspotting (1996)                                       9.786275
          RoboCop (1987)                                             9.675467
```

NOW IMPLEMENTING USER BASED FILTERING

NOW BY FINDING SIMILARITY COEFICIENT BETWEEN DIFFERENT USERS , WE HAVE FOLLOWING MATRIX

In [68]:
```
final_user.head()
```

Out[68]:

movieId	1	2	3	4	5	6	7	8	9	10	...
userId											
1	-3.663793e-01	1.837611e-16	-3.663793e-01	1.837611e-16	1.837611e-16	-3.663793e-01	1.837611e-16	1.837611e-16	1.837611e-16	1.837611e-16	...
2	2.143879e-16	2.143879e-16	2.143879e-16	2.143879e-16	2.143879e-16	2.143879e-16	2.143879e-16	2.143879e-16	2.143879e-16	2.143879e-16	...
3	3.643809e-16	3.643809e-16	3.643809e-16	3.643809e-16	3.643809e-16	3.643809e-16	3.643809e-16	3.643809e-16	3.643809e-16	3.643809e-16	...
4	1.973730e-16	1.973730e-16	1.973730e-16	1.973730e-16	1.973730e-16	1.973730e-16	1.973730e-16	1.973730e-16	1.973730e-16	1.973730e-16	...
5	3.636364e-01	1.009294e-16	1.009294e-16	1.009294e-16	1.009294e-16	1.009294e-16	1.009294e-16	1.009294e-16	1.009294e-16	1.009294e-16	...

5 rows × 9724 columns

BY TRANSPOSING ABOVE MATRIX AND MULTIPLYING WITH SAME , WE WILL GET SIMILARITY COEFICIENT BETWEEN DIFFERENT USERS

In [69]:
```
# user similarity on replacing NAN by user avg
b = cosine_similarity(final_user)
np.fill_diagonal(b, 0 )
similarity_with_user = pd.DataFrame(b,index=final_user.index)
similarity_with_user.columns=final_user.index
similarity_with_user.head()
```

Out[69]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602
userId													
1	0.000000	1.264516e-03	5.525772e-04	0.048419	0.021847	-0.045497	-6.199672e-03	0.047013	1.950985e-02	-8.754088e-03	...	0.018127	-0.017172
2	0.001265	0.000000e+00	1.290714e-29	-0.017164	0.021796	-0.021051	-1.111357e-02	-0.048085	1.371140e-29	3.011629e-03	...	-0.050551	-0.031581
3	0.000553	1.290714e-29	0.000000e+00	-0.011260	-0.031539	0.004800	8.818877e-30	-0.032471	8.338880e-30	1.764256e-30	...	-0.004904	-0.016117
4	0.048419	-1.716402e-02	-1.125978e-02	0.000000	-0.029620	0.013956	5.809139e-02	0.002065	-5.873603e-03	5.159032e-02	...	-0.037687	0.063122
5	0.021847	2.179571e-02	-3.153892e-02	-0.029620	0.000000	0.009111	1.011715e-02	-0.012284	4.245419e-30	-3.316512e-02	...	0.015964	0.012427

5 rows × 610 columns

TOP 30 SIMILAR USERS TO A PARTICULAR USER

In [73]:
```
# top 30 neighbours for each user
sim_user_30_u = find_n_neighbours(similarity_with_user,30)
sim_user_30_u.head()
```

Out[73]:

userId	top1	top2	top3	top4	top5	top6	top7	top8	top9	top10	...	top21	top22	top23	top24	top25	top26	top27	top28	top29	top30
1	301	597	414	477	57	369	206	535	590	418	...	484	469	72	593	44	297	434	483	449	552
2	189	246	378	209	227	326	393	332	196	528	...	114	153	596	495	407	567	93	308	496	222
3	441	496	549	231	527	537	313	518	244	246	...	309	586	230	303	34	161	39	47	127	138
4	75	137	590	391	43	128	462	250	290	85	...	472	593	299	32	349	268	159	199	173	426
5	145	35	565	134	58	444	446	347	530	142	...	94	569	411	588	584	404	498	323	536	437

5 rows × 30 columns

TOP MOVIES RECOMMENDED BY USER BASED FILTERING OF USER:370

In [113]:
```
user = int(input("Enter the user id to whom you want to recommend : "))

predicted_movies = User_item_score1(user)
print(" ")
print("The Recommendations for User Id ")
for i in predicted_movies:
    print(i)
```

```
Enter the user id to whom you want to recommend : 370

The Recommendations for User Id
Three Billboards Outside Ebbing, Missouri (2017)
Return of Martin Guerre, The (Retour de Martin Guerre, Le) (1982)
Hoop Dreams (1994)
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
Crumb (1994)
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)
Planet Earth (2006)
Brazil (1985)
Apocalypse Now (1979)
Seven Samurai (Shichinin no samurai) (1954)
Godfather, The (1972)
Casablanca (1942)
Glory (1989)
Wallace & Gromit: The Wrong Trousers (1993)
Godfather: Part II, The (1974)
Spirited Away (Sen to Chihiro no kamikakushi) (2001)
Central Station (Central do Brasil) (1998)
Goodfellas (1990)
```

taking top 9 movies from user and item based CF , we combined them adjacently to get recommendation which involve both kind of filtering for more perfect recommendation

In [161]:
```
# taking top 9 movies from user and item based CF , we combined them adjacently to get recomm
# both kind of filtering for more perfect recommendation
item_based = ['Terminator, The (1984)',
'Indiana Jones and the Temple of Doom (1984)',
'Indiana Jones and the Last Crusade (1989)',
'Ferris Buellers Day Off (1986)',
'Breakfast Club, The (1985)',
'Pulp Fiction (1994)',
'Monty Python and the Holy Grail (1975)',
'Terminator 2: Judgment Day (1991)',
'Clerks (1994)'
 ]
user_based=['Three Billboards Outside Ebbing, Missouri (2017)',
'Return of Martin Guerre, The (Retour de Martin Guerre, Le) (1982)',
'Hoop Dreams (1994)',
'Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)',
'Crumb (1994)',
'Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)',
'Planet Earth (2006)',
'Brazil (1985)',
'Apocalypse Now (1979)']
print('.....................TOP RECOMMENDED MOVIE VIA USER AS WELL AS MOVIE BASED FILTERING.
print(' ')
for i in range(9):
  print(item_based[i])
  print(user_based[i])
```

FINAL RECOMMENDATIOM TO USER 370

```
.....................TOP RECOMMENDED MOVIE VIA USER AS WELL AS MOVIE BASED FILTERING.....................

Terminator, The (1984)
Three Billboards Outside Ebbing, Missouri (2017)
Indiana Jones and the Temple of Doom (1984)
Return of Martin Guerre, The (Retour de Martin Guerre, Le) (1982)
Indiana Jones and the Last Crusade (1989)
Hoop Dreams (1994)
Ferris Buellers Day Off (1986)
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
Breakfast Club, The (1985)
Crumb (1994)
Pulp Fiction (1994)
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)
Monty Python and the Holy Grail (1975)
Planet Earth (2006)
Terminator 2: Judgment Day (1991)
Brazil (1985)
Clerks (1994)
Apocalypse Now (1979)
```

4. CONCLUSION AND FUTURE WORK

4.1 RESULT AND CONCLUSION

RESULT

Collaborative filtering (CF) is a very popular recommendation system algorithm for the prediction and recommendation based on other users' ratings and collaboration. User-based collaborative filtering was the first automated collaborative filtering mechanism. It is also called k-NN collaborative filtering. The problem of collaborative filtering is to predict how well a user will like an item that he has not rated given a set of existing choice judgments for a population of users .

ITEM –BASED CF has given different results from user based , so for more accuracy we should consider each angle for getting recommendation , and different algorithm to predict movies , so at last we can merge output of each training model and get more precise results.

CONCLUSION

Over the years, Machine learning has solved several challenges for companies like Netflix, Amazon, Google, Facebook, and others. The recommender system for Netflix helps the user filter through information in a massive list of movies and shows based on his/her choice. A recommender system must interact with the users to learn their preferences to provide recommendations.

4.2 FUTURE WORK AND COST

FUTURE WORK:

We used two types of CF model to train our model. But here are many more types of recommendation algorithms.Making training model from all possible algorithms, like Content filtering recommender systems, Reducing dimensionality algo ,Probabilistic algorithm like Bayesian-network model,EM algorithm etc. So using this we can get several recommendation , and merge top movies from each algorithm .At last we will have best recommendation considering each angle , make our model more accurate.

5. REFERENCES

REFERENCE JOURNALS:

[1] How retailers can keep up with consumers, McKinsey & Company, https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

[2] How Netflix's Recommendation System Works, Netflix Research, https://help.netflix.com/en/node/100639

[3] Recommendations, Figuring out how to bring unique joy to each member, Netflix Research, https://research.netflix.com/research-area/recommendations

[4] Collaborative Filtering, University of Pittsburgh, Peter Brusilovsky, Sue Yeon and Danielle Lee, https://pitt.edu/~peterb/2480-122/CollaborativeFiltering.pdf

BIODATA

Name :PRERIT AGRAWAL

Mobile Number :7000185123

E-mail :
3@vitstudent.ac.in

Permanent Address: RAIPUR

Name : ABHYUDAY SINGH

Mobile Number :9826990507

E-mail :abhyudaysingh.mandloi2018@vitstudent.ac.in

Permanent Address: Indore