EasyChair Preprint

№ 1950

Movie Recommendation System

Mohit Soni and Shivam Bansal

November 13, 2019

Movie Recommendation System

A

MINOR PROJECT REPORT

Submitted by

Mohit Soni(41914802716)

Shivam Bansal(42214802716)

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

Dr. Pooja Mam

Assistant Professor, CSE

Department of Computer Science and Engineering

Maharaja Agrasen Institute of Technology,

PSP area, Sector 22, Rohini, New Delhi - 110085

(Affilated to Guru Gobind Singh Indraprastha, New Delhi)

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

CERTIFICATE

This is to Certified that this MINOR project report "Movie Recommendation System "is submitted by "Mohit Soni(41914802716) and Shivam Bansal(42214802716)" who carried out the project work under my supervision.

I approve this MINOR project for submission.

Prof. Namita Gupta
(HoD, CSE)

Dr. Pooja Gupta,
Assistant Professor
(Project Guide)

ABSTRACT

A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behavior of a customer and based on that, recommends products which the users might be likely to buy. If a completely new user visits an e-commerce site, that site will not have any past history of that user. So how does the site go about recommending products to the user in such a scenario? One possible solution could be to recommend the best selling products, i.e. the products which are high in demand. Another possible solution could be to recommend the products which would bring the maximum profit to the business. Three main approaches are used for our recommender systems. One is Demographic Filtering i.e They offer generalized recommendations to every user, based on movie popularity and/or genre. The System recommends the same movies to users with similar demographic features. Since each user is different , this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience. Second is content-based filtering, where we try to profile the users interests using information collected, and recommend items based on that profile. The other is collaborative filtering, where we try to group similar users together and use information about the group to make recommendations to the user.

ACKNOWLEDGEMENT

It gives me immense pleasure to express my deepest sense of gratitude and sincere thanks to my respected guide Dr. Pooja Gupta(Assistant Professor), CSE, MAIT Delhi, for their valuable guidance, encouragement and help for completing this work. Their useful suggestions for this whole work and co-operative behavior are sincerely acknowledged.

I am also grateful to my teachers Dr. Pooja Gupta, Prof. Namita Gupta for their constant support and guidance.

I also wish to express my indebtedness to my parents as well as my family member whose blessings and support always helped me to face the challenges ahead.

Place: Delhi

Mohit Soni(41914802716)

Shivam Bansal(42214802716)

Date: 14 Nov 2019

TABLE OF CONTENTS

Introduction.....7
Technology Used.....9
Literature Survey.....10
Approach.....12
Result.....21
Conclusion.....25
References.....26

LIST OF FIGURES

1.1 Demographic Filtering.....13

2.1 Content Based Filtering.....14

2.2 Credits, Genres and Keywords Based Recommender.....15

3.1 User Based Filtering.....17

3.2 Item Based Filtering.....18

3.3 Single Value Decomposition.....19

4.1 Demographic Output.....21

4.2 Content Based Output_1.....22

4.3 Content Based Output_2.....22

4.4 Collaborative Based Output_1.....23

4.5 Collaborative Based Output_2.....23

INTRODUCTION

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google. Often, these systems are able to collect information about a users choices, and can use this information to improve their suggestions in the future. For example, Facebook can monitor your interaction with various stories on your feed in order to learn what types of stories appeal to you. Sometimes, the recommender systems can make improvements based on the activities of a large number of people. For example, if Amazon observes that a large number of customers who buy the latest Apple Macbook also buy a USB-C-toUSB Adapter, they can recommend the Adapter to a new user who has just added a Macbook to his cart. Due to the advances in recommender systems, users constantly expect good recommendations. They have a low threshold for services that are not able to make appropriate suggestions. If a music streaming app is not able to predict and play music that the user likes, then the user will simply stop using it. This has led to a high emphasis by tech companies on improving their recommendation systems. However, the problem is more complex than it seems. Every user has different preferences and likes. In addition, even the taste of a single user can vary depending on a large number of factors, such as mood, season, or type of activity the user is doing. For example, the type of music one would like to hear while exercising differs greatly from the type of music he'd listen to when cooking dinner. Another issue that recommendation systems have to solve is the exploration vs exploitation problem. They must explore new domains to discover more about the user, while still making the most of what is already known about of the user. Three main approaches are used for our recommender systems. One is Demographic Filtering i.e They offer generalized recommendations to every user, based on movie popularity and/or

genre. The System recommends the same movies to users with similar demographic features. Since each user is different , this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience. Second is content-based filtering, where we try to profile the users interests using information collected, and recommend items based on that profile. The other is collaborative filtering, where we try to group similar users together and use information about the group to make recommendations to the user.

Technology Used

1. Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

2. Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

LITERATURE SURVEY

MOVREC is a movie recommendation system presented by D.K. Yadav et al. based on collaborative filtering approach. Collaborative filtering makes use of information provided by user. That information is analyzed and a movie is recommended to the users which are arranged with the movie with highest rating first.

Luis M Capos et al has analyzed two traditional recommender systems i.e. content based filtering and collaborative filtering. As both of them have their own drawbacks he proposed a new system which is a combination of Bayesian network and collaborative filtering.

A hybrid system has been presented by Harpreet Kaur et al. The system uses a mix of content as well as collaborative filtering algorithm. The context of the movies is also considered while recommending. The user - user relationship as well as user - item relationship plays a role in the recommendation.

The user specific information or item specific information is clubbed to form a cluster by Utkarsh Gupta et al. using chameleon. This is an efficient technique based on Hierarchical clustering for recommender system. To predict the rating of an item voting system is used. The proposed system has lower error and has better clustering of similar items.

Urszula Kużelewska et al. proposed clustering as a way to deal with recommender systems. Two methods of computing cluster representatives were presented and evaluated. Centroid-based solution and memory-based collaborative filtering methods were used as a basis for comparing effectiveness of the proposed two methods. The result was a significant increase in the accuracy of the generated recommendations when compared to just centroid-based method.

Costin-Gabriel Chiru et al. proposed Movie Recommender, a system which uses the information known about the user to provide movie recommendations. This system attempts to solve the problem of unique recommendations which results from ignoring the data specific to the user. The psychological profile of the user, their watching history and the data involving movie scores from other websites is collected. They are based on aggregate similarity calculation. The system is a hybrid model which uses both content based filtering

and collaborative filtering.

To predict the difficulty level of each case for each trainee Hongli LIn et al. proposed a method called contentboosted collaborative filtering (CBCF).The algorithm is divided into two stages, First being the content-based filtering that improves the existing trainee case ratings data and the second being collaborative filtering that provides the final predictions. The CBCF algorithm involves the advantages of both CBF and CF, while at the same time, overcoming both their disadvantages.

Approach

There are various types of recommender systems with different approaches and some of them are classified as below:

1. Demographic Filtering- They offer generalized recommendations to every user, based on movie popularity and/or genre. The System recommends the same movies to users with similar demographic features. Since each user is different , this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.

Before getting started with this -

- We need a metric to score or rate movie
- Calculate the score for every movie
- Sort the scores and recommend the best rated movie to the users.

We can use the average ratings of the movie as the score but using this won't be fair enough since a movie with 8.9 average rating and only 3 votes cannot be considered better than the movie with 7.8 as as average rating but 40 votes. So, I'll be using IMDB's weighted rating (wr) which is given as :-

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

where,

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

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

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

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

Fig. 1.1 Demographic Filtering

2. Content-based Filtering Systems: In content-based filtering, items are recommended based on comparisons between item profile and user profile. A user profile is content that is found to be relevant to the user in form of keywords(or features). A user profile might be seen as a set of assigned keywords (terms, features) collected by algorithm from items found relevant (or interesting) by the user. A set of keywords (or features) of an item is the Item profile. For example, consider a scenario in which a person goes to buy his favorite cake ‘X’ to a pastry. Unfortunately, cake ‘X’ has been sold out and as a result of this the shopkeeper recommends the person to buy cake ‘Y’ which is made up of ingredients similar to cake ‘X’. This is an instance of content-based filtering.

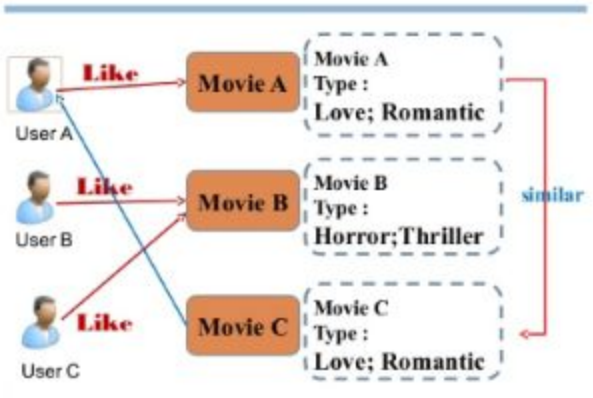

Fig. 2.1 Content Based Filtering

We will be using the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies. We use the cosine similarity score since it is independent of magnitude and is relatively easy and fast to calculate. Mathematically, it is defined as follows:

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

We are now in a good position to define our recommendation function. These are the following steps we'll follow :-

- Get the index of the movie given its title.
- Get the list of cosine similarity scores for that particular movie with all movies. Convert it into a list of tuples where the first element is its position and the second is the similarity score.
- Sort the aforementioned list of tuples based on the similarity scores; that is, the second element.
- Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).

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

While our system has done a decent job of finding movies with similar plot descriptions, the quality of recommendations is not that great. "The Dark Knight Rises" returns all Batman movies while it is more likely that the people who liked that movie are more inclined to enjoy other Christopher Nolan movies. This is something that cannot be captured by the present system.

Credits, Genres and Keywords Based Recommender

It goes without saying that the quality of our recommender would be increased with the usage of better metadata. That is exactly what we are going to do in this section. We are going to build a recommender based on the following metadata: the 3 top actors, the director, related genres and the movie plot keywords.

From the cast, crew and keywords features, we need to extract the three most important actors, the director and the keywords associated with that movie. Right now, our data is present in the form of "stringified" lists , we need to convert it into a safe and usable structure

In [18]:
```
# Parse the stringified features into their corresponding python objects
from ast import literal_eval

features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(literal_eval)
```

Next, we'll write functions that will help us to extract the required information from each feature.

In [19]:
```
# Get the director's name from the crew feature. If director is not listed, return NaN
def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
```

In [20]:
```
# Returns the list top 3 elements or entire list; whichever is more.
def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        #Check if more than 3 elements exist. If yes, return only first three. If no, return entire list.
        if len(names) > 3:
            names = names[:3]
        return names

    #Return empty list in case of missing/malformed data
    return []
```

Fig. 2.2 Credits, Genres and Keywords Based Recommender

Advantages of content-based filtering are:

- They capable of recommending unrated items.
- We can easily explain the working of recommender system by listing the Content features of an item.
- Content-based recommender systems use need only the rating of the concerned user,and not any other user of the system.

Disadvantages of content-based filtering are:

- It does not work for a new user who has not rated any item yet as enough ratings are required contentbased recommender evaluates the user preferences and provides accurate recommendations.
- No recommendation of serendipitous items.
- Limited Content Analysis- The recommend does not work if the system fails to distinguish the items hat a user likes from the items that he does not like.

3. Collaborative filtering based systems: Our content based engine suffers from some severe limitations. It is only capable of suggesting movies which are close to a certain movie. That is, it is not capable of capturing tastes and providing recommendations across genres.

Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who she/he is.

Therefore, in this section, we will use a technique called Collaborative Filtering to make recommendations to Movie Watchers. It is basically of two types:-

a) User based filtering- These systems recommend products to a user that similar users have liked. For measuring the similarity between two users we can either use

pearson correlation or cosine similarity. This filtering technique can be illustrated with an example. In the following matrix's, each row represents a user, while the columns correspond to different movies except the last one which records the similarity between that user and the target user. Each cell represents the rating that the user gives to that movie. Assume user E is the target.

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	
C			5		2		
D		1		5		4	
E			4			2	1
F	4	5		1			NA

Since user A and F do not share any movie ratings in common with user E, their similarities with user E are not defined in Pearson Correlation. Therefore, we only need to consider user B, C, and D. Based on Pearson Correlation, we can compute the following similarity.

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	0.87
C			5		2		1
D		1		5		4	-1
E			4			2	1
F	4	5		1			NA

Fig. 3.1User Based Filtering

Although computing user-based CF is very simple, it suffers from several problems. One main issue is that users' preference can change over time. It indicates that precomputing the matrix based on their neighboring users may lead to bad performance. To tackle this problem, we can apply item-based CF.

b) Item Based Collaborative Filtering - Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with

Pearson Correlation or Cosine Similarity. The major difference is that, with item-based collaborative filtering, we fill in the blank vertically, as oppose to the horizontal manner that user-based CF does. The following table shows how to do so for the movie Me Before

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You
A	2		2	4	5	2.94*
B	5		4			1
C			5		2	2.48*
D		1		5		4
E			4			2
F	4	5		1		1.12*
Similarity	-1	-1	0.86	1	1	

Fig. 3.2 Item Based Filtering

It successfully avoids the problem posed by dynamic user preference as item-based CF is more static. However, several problems remain for this method. First, the main issue is *scalability*. The computation grows with both the customer and the product. The worst case complexity is O(mn) with m users and n items. In addition, *sparsity* is another concern. Take a look at the above table again. Although there is only one user that rated both Matrix and Titanic rated, the similarity between them is 1. In extreme cases, we can have millions of users and the similarity between two fairly different movies could be very high simply because they have similar rank for the only user who ranked them both.

Single Value Decomposition

One way to handle the scalability and sparsity issue created by CF is to leverage a latent factor model to capture the similarity between users and items. Essentially, we want to turn the recommendation problem into an optimization problem. We can view it as how good we are in predicting the rating for items given a user. One common metric is Root Mean Square Error (RMSE). **The lower the RMSE, the better the performance**.

Now talking about latent factor you might be wondering what is it ?It is a broad idea which describes a property or concept that a user or an item have. For instance, for music, latent factor can refer to the genre that the music belongs to. SVD decreases the dimension of the utility matrix by extracting its latent factors. Essentially, we map each user and each item into a latent space with dimension r. Therefore, it helps us better understand the relationship between users and items as they become directly comparable. The below figure illustrates this idea.

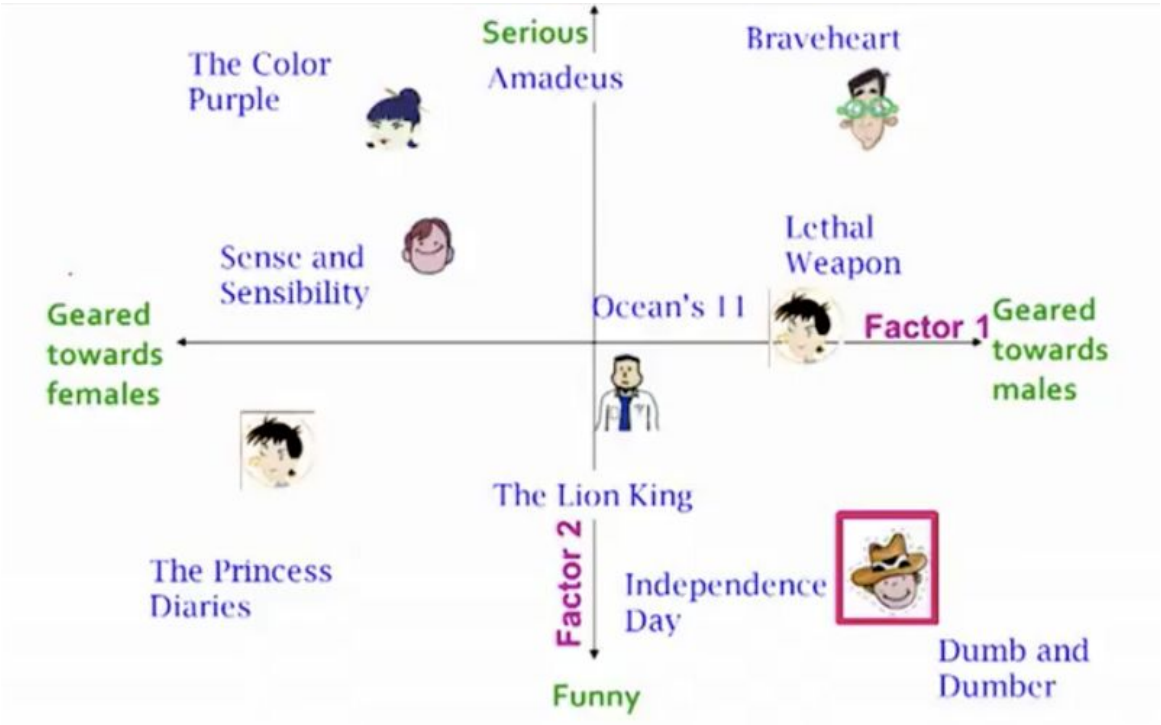

Fig 3.3 Single Value Decomposition

Now enough said , let's see how to implement this. Since the dataset we used before did not have userId(which is necessary for collaborative filtering) let's load another dataset. We'll be using the Surprise library to implement SVD.

Advantages of collaborative filtering based systems:

- 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

Disadvantages of collaborative filtering are:

- Early rater problem: Collaborative filtering systems cannot provide recommendations for new items since there are no user ratings on which to base a prediction.
- Gray sheep: In order for CF based system to work, group with similar characteristics are needed. Even if such groups exist, it will be very difficult to recommend users who do not consistently agree or disagree to these groups.
- Sparsity problem: In most cases, the amount of items exceed the number of users by a great margin which makes it difficult to find items that are rated by enough people.

RESULTS

1. Demographic Filtering

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

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

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

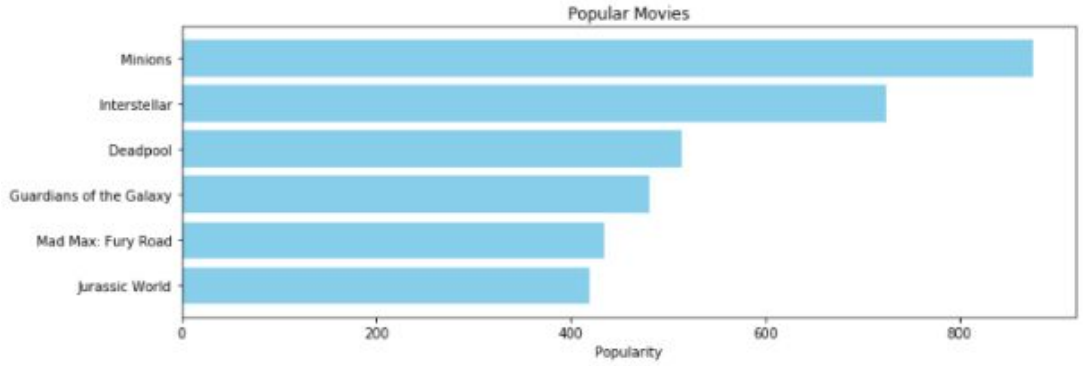

Fig. 4.1 Demographic Output

2. Content-based Filtering Systems

```
In [16]: get_recommendations('The Dark Knight Rises')

Out[16]: 65                              The Dark Knight
         299                              Batman Forever
         428                              Batman Returns
         1359                                    Batman
         3854    Batman: The Dark Knight Returns, Part 2
         119                               Batman Begins
         2507                                  Slow Burn
         9           Batman v Superman: Dawn of Justice
         1181                                       JFK
         210                               Batman & Robin
         Name: title, dtype: object

In [17]: get_recommendations('The Avengers')

Out[17]: 7              Avengers: Age of Ultron
         3144                            Plastic
         1715                            Timecop
         4124                This Thing of Ours
         3311              Thank You for Smoking
         3033                     The Corruptor
         588     Wall Street: Money Never Sleeps
         2136         Team America: World Police
         1468                       The Fountain
         1286                        Snowpiercer
         Name: title, dtype: object
```

Fig. 4.2 Content Based Output_1

get_recommendations() function by passing in the new cosine_sim2 matrix is

```
In [29]: get_recommendations('The Dark Knight Rises', cosine_sim2)

Out[29]: 65              The Dark Knight
         119               Batman Begins
         4638    Amidst the Devil's Wings
         1196                The Prestige
         3073          Romeo Is Bleeding
         3326             Black November
         1503                      Takers
         1986                      Faster
         303                     Catwoman
         747               Gangster Squad
         Name: title, dtype: object

In [30]: get_recommendations('The Godfather', cosine_sim2)

Out[30]: 867      The Godfather: Part III
         2731      The Godfather: Part II
         4638    Amidst the Devil's Wings
         2649          The Son of No One
         1525             Apocalypse Now
         1018           The Cotton Club
         1170     The Talented Mr. Ripley
         1209              The Rainmaker
         1394              Donnie Brasco
         1850                    Scarface
         Name: title, dtype: object
```

Fig. 4.3 Content Based Output_2

3. Collaborative filtering based systems

```
reader = Reader()
ratings = pd.read_csv('the-movies-dataset/ratings_small.csv')
ratings.head()
```

Out[32]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

```
Note that in this dataset movies are rated on a scale of 5 unlike the earlier one.
```

In [34]:
```
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
#data.split(n_folds=5)
```

In [36]:
```
svd = SVD()
#evaluate(svd, data, measures=['RMSE', 'MAE'])
cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8995  0.8951  0.8919  0.8934  0.9028  0.8966  0.0040
MAE (testset)     0.6926  0.6907  0.6876  0.6879  0.6922  0.6902  0.0021
Fit time          10.85   15.22   12.99   12.19   15.55   13.36   1.79
Test time         0.26    0.46    0.14    0.48    0.40    0.35    0.13
```

Fig. 4.4 Collaborative Based Output_1

In [38]:
```
ratings[ratings['userId'] == 1]
```

Out[38]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205
5	1	1263	2.0	1260759151
6	1	1287	2.0	1260759187
7	1	1293	2.0	1260759148
8	1	1339	3.5	1260759125
9	1	1343	2.0	1260759131
10	1	1371	2.5	1260759135
11	1	1405	1.0	1260759203
12	1	1953	4.0	1260759191
13	1	2105	4.0	1260759139
14	1	2150	3.0	1260759194
15	1	2193	2.0	1260759198
16	1	2294	2.0	1260759108
17	1	2455	2.5	1260759113
18	1	2968	1.0	1260759200
19	1	3671	3.0	1260759117

In [39]:
```
svd.predict(1, 302, 3)
```

Out[39]:
```
Prediction(uid=1, iid=302, r_ui=3, est=2.8519641281630195, details={'was_impossible': False})
```

Fig. 4.5 Collaborative Based Output_2

For movie with ID 302, we get an estimated prediction of **2.851**. One startling feature of this recommender 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 predicted the movie.

CONCLUSION

A hybrid approach is taken between context based filtering and collaborative filtering to implement the system. This approach overcomes drawbacks of each individual algorithm and improves the performance of the system. Techniques like Clustering, Similarity and Classification are used to get better recommendations thus reducing MAE and increasing precision and accuracy. In future we can work on hybrid recommender using clustering and similarity for better performance. Our approach can be further extended to other domains to recommend songs, video, venue, news, books, tourism and e-commerce sites, etc.

References

1. Peng, Xiao, Shao Liangshan, and Li Xiuran. "Improved Collaborative Filtering Algorithm in the Research and Application of Personalized Movie Recommendations", 2013 Fourth International Conference on Intelligent Systems Design and Engineering Applications, 2013.

2. Munoz-Organero, Mario, Gustavo A. Ramíez-González, Pedro J. Munoz-Merino, and Carlos Delgado Kloos. "A Collaborative Recommender System Based on Space-Time Similarities", IEEE Pervasive Computing, 2010.

3. Al-Shamri, M.Y.H.. "Fuzzy-genetic approach to recommender systems based on a novel hybrid user model", Expert Systems With Applications, 200810

4. Hu Jinming. "Application and research of collaborative filtering in e-commerce recommendation system", 2010 3rd International Conference on Computer Science and Information Technology, 07/2010

5. Xu, Qingzhen Wu, Jiayong Chen, Qiang. "A novel mobile personalized recommended method based on money flow model for stock exchange.(Researc", Mathematical Problems in Engineering, Annual 2014 Issue

6. Yan, Bo, and Guanling Chen. "AppJoy : personalized mobile application discovery", Proceedings of the 9th international conference on Mobile systems applications and services - MobiSys 11 MobiSys 11, 2011.

7. Davidsson C, Moritz S. Utilizing implicit feedback and context to recommend mobile applications from first use.In: Proc. of the Ca RR 2011. New York: ACM Press, 2011. 19-22.http://dl.acm.org/citation.cfm?id=1961639[doi:10.1145/1961634.1961639]

8. Bilge, A., Kaleli, C., Yakut, I., Gunes, I., Polat, H.: A survey of privacy-preserving collaborative filtering schemes. Int. J. Softw. Eng. Knowl. Eng. 23(08), 1085–1108 (2013)CrossRefGoogle Scholar

9. Calandrino, J.A., Kilzer, A., Narayanan, A., Felten, E.W., Shmatikov, V.: You might also like: privacy risks of collaborative filtering. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 231–246, Oakland, CA, USA (2011)

10. Research.ijcaonline.org

11. Hu Jinming. "Application and research of collaborative filtering in e-commerce recommendation system", 2010 3rd International Conference on Computer Science and Information Technology, 07/2010
12. www.pacis2014.org