Project Report

On

MOVIE RECOMMENDER SYSTEM

Submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

In

Electronics and Communication Engineering

SUMMER TRAINING

FACULTY SUPERVISER

Dr. Jasmine Saini

SUBMITTED BY

Sagar Makhija-17102228

May-June, 2020

TABLE OF CONTENTS

CERTIFICATE

It is certified that the work contained in the project report titled **"Movie Recommender System"** by **"Sagar (17102228)"** has been carried out under my supervision and this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Dr. Jasmine Saini

ECE

JIIT NOIDA

May-June, 2020

ACKNOWLEDGEMENT

I would like to acknowledge my parents and all the teachers, who supported me both morally and technically, especially my supervisor Dr. Jasmine Saini who helped me at every step in the making of the project, helped me by clarifying my queries related to the project and technical problems. Also, my special thanks to all class fellows who helped me in clarification of any issue as well as implementation and in documentation.

Signature of Student

Sagar (17102228)

Chapter 1 – INTRODUCTION

A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behaviour 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 user's 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.

1.1 Motivation

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.

Chapter 2 – 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 10 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.

Chapter 3- PROJECT IDEA, METHODOLOGY AND IMPLEMENTATION

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

1. **Demographic Filtering (DF)**: It aims to classify the user based on personal attributes and make recommendations based on demographic classes. The users are divided into demographic classes in terms of their personal attributes. These classes serve as the input data to the recommendation process The objective of this process is to find the classes of people who like a certain product. If people from class C like product s and there is person c (this user belongs to class C), who has not seen yet product s, then this product can be recommended to person c. The customers provide the personal data via surveys that they fill in during the registration process or can be extracted from the purchasing history of the users. This technique may not require collecting the complex data such as history of users‟ purchases and ratings. However, the weaknesses of DF are that the classification can be too general and this leads to lose the individuality of the users, this method uses data that are provided by users. This data can be either incomplete or untrue and the classification is created according to the customer's interest, which tend to vary over time. DF does not support the adoption of the user profile to changes.

   RS based on Demographic filtering (DF) classify users according to their demographic information and recommend services accordingly. In DF the user profiles are created by classifying users in stereotypical descriptions, representing the features of classes of users [5]. Demographic information identifies those users that like related services. Semi-trusted third parties use DF to recommend services by using data on individual users. DF creates categories of users which have similar demographic characteristics and then the cumulative buying behavior or preferences of users within these categories are being tracked. For a new user, recommendations are made by first finding which category he falls in and then the cumulative buying preferences of previous users is applied to that category which he belongs. Like collaborative techniques, demographic techniques also form "people-to-people"

correlations but use dissimilar data. A collaborative and content-based technique requires a history of user ratings which is not of the kind required by Demographic approach.

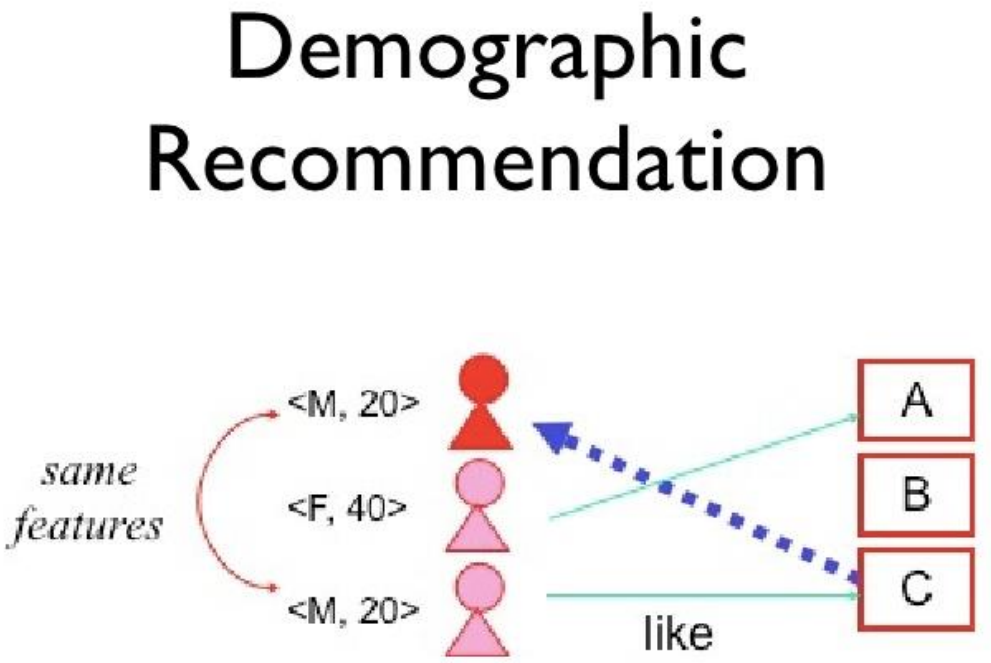


Fig. 3.1 : How Demographic recommendation works(www.researchgate.net)

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. 3.2 : Demographic Filtering(slideshare.net)

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.

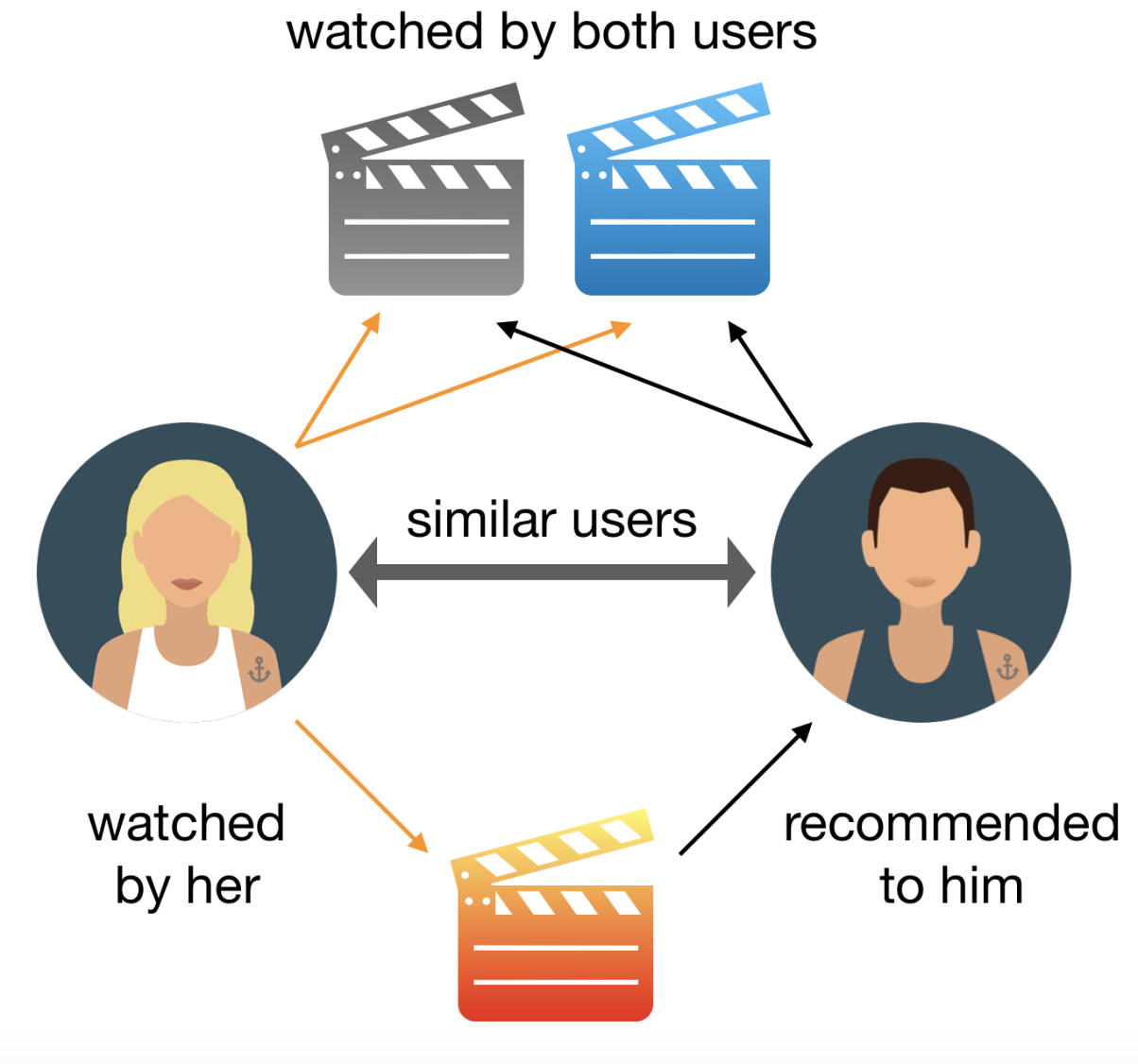


Fig. 3.3 : Content Based filtering(www.towardsdatascience.com)

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 will 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.

Cast, Director, 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.

```
features = ['keywords','cast','genres','director']
##Step 3: Create a column in DF which combines all selected features
for feature in features:
    df[feature] = df[feature].fillna('')

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

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

#print "Combined Features:", df["combined_features"].head()
```

Fig. 3.4: Director, 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 needs 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 content based recommender evaluates the user preferences and provides accurate recommendations.

● No recommendation of serendipitous items.

● Limited Content Analysis

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 16 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. 3.5: User 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 17 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 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. 3.6: 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(m*n) 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.

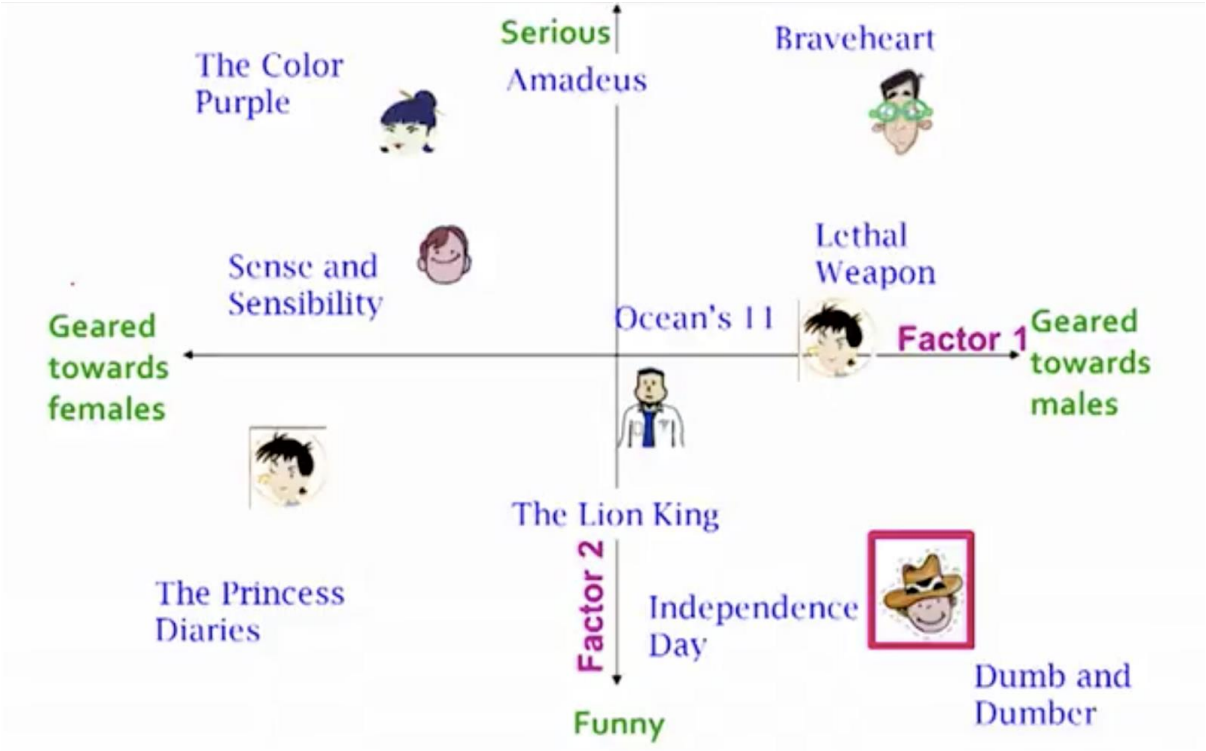


Fig 3.7 : Single Value Decomposition

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.

Chapter 4- 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 structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0- released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

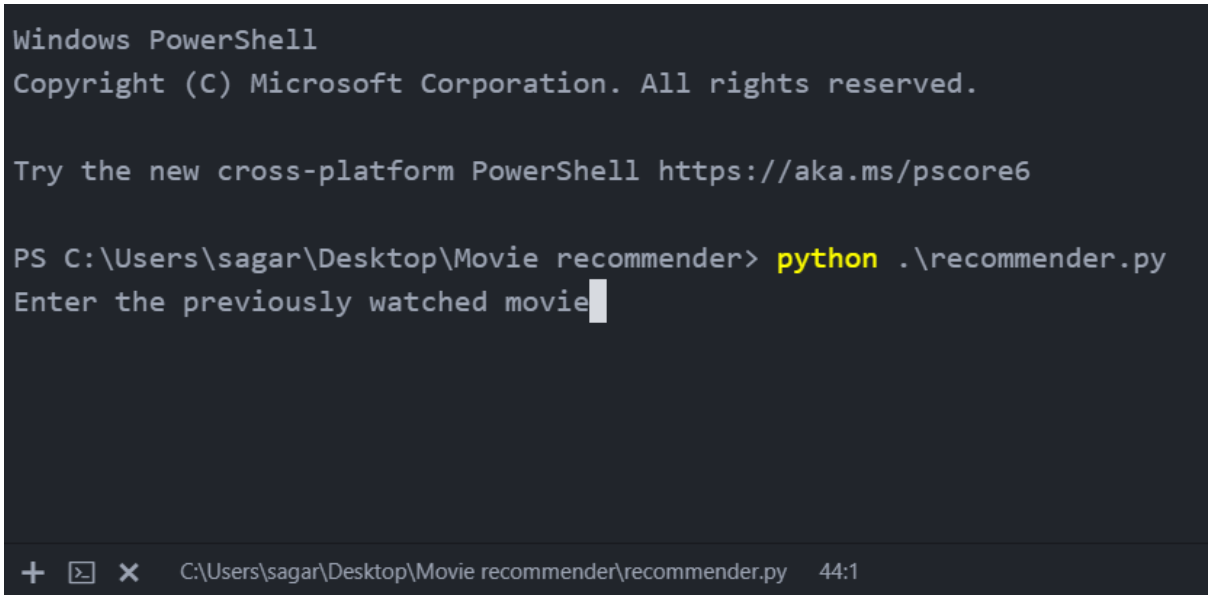
Python interpreters are available for many operating systems. A global community of programmers develops and maintains C-Python, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and C-Python development.

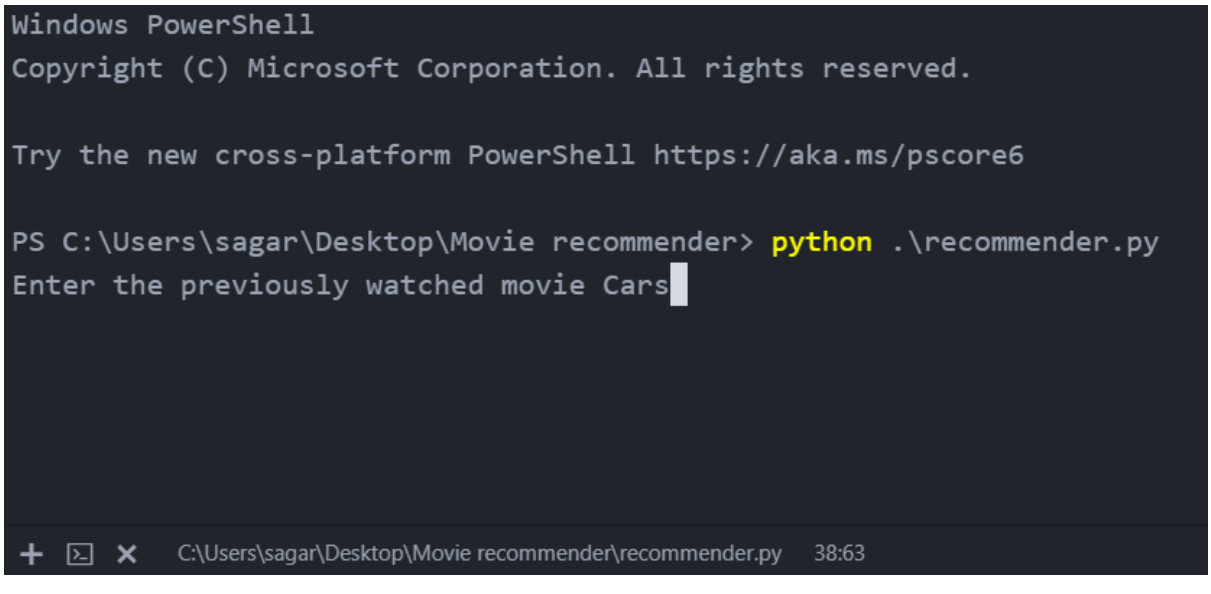
2. PyCharm

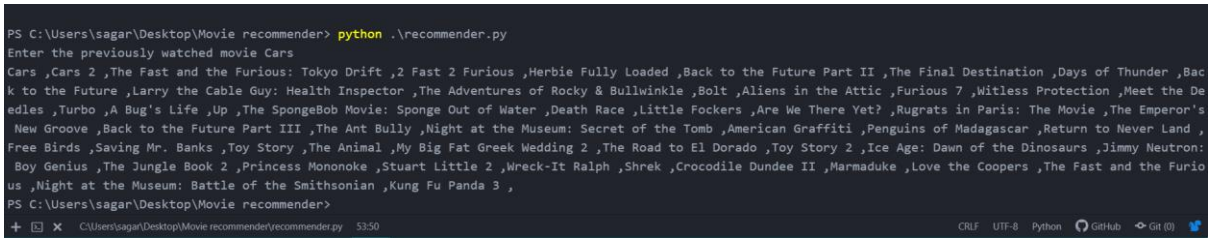
PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

Chapter 5- OUTPUT

(i)

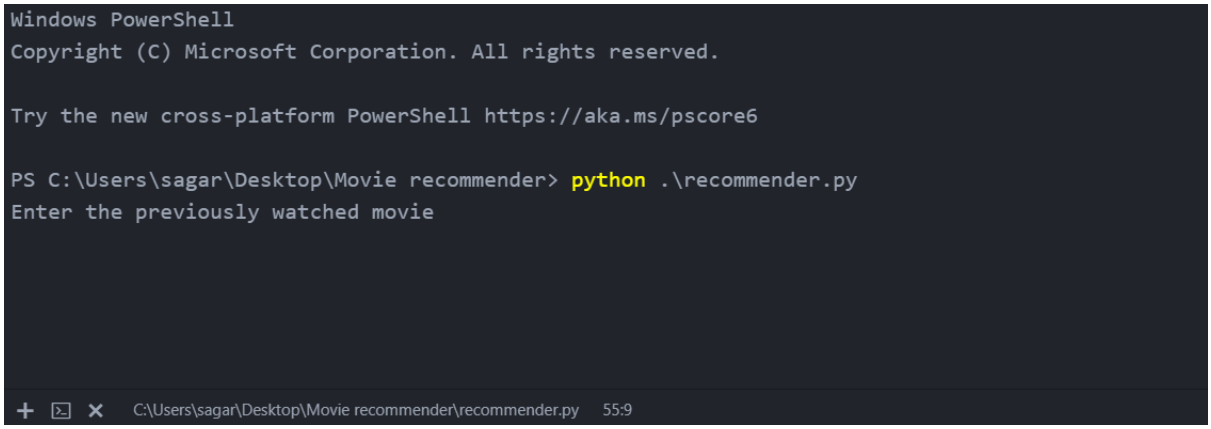


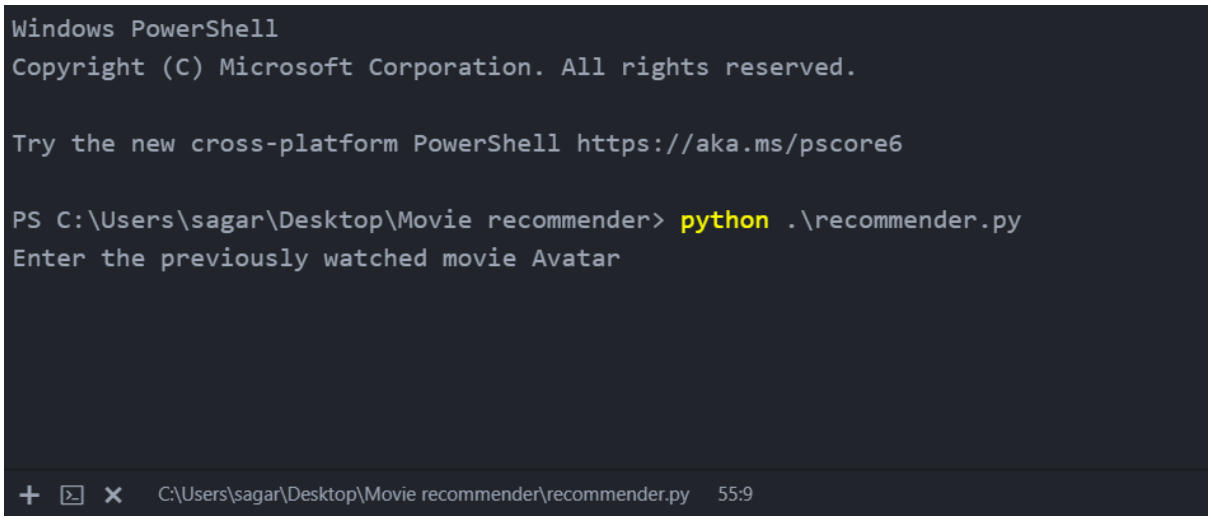


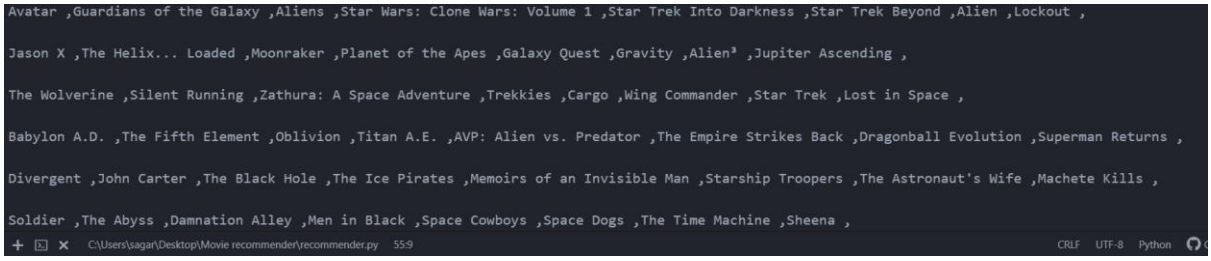


We can see a long list of recommended movies in the above picture based on our previously watched movie.

(ii)







These movies are the top 50 movies recommended on the movie previously watched, we can choose top 5, top 10 as per the requirement needed.

Conclusion\LEARNING OUTCOME

We have created a movie recommender program that can easily recommend a number of top movies that are related and/or have the same type of story or hero or director and other factors all in context to the movie that was previously seen by the user. 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.

FUTURE SCOPE

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 user's 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-to USB 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. Two main approaches are widely used for recommender systems. One is content-based filtering, where we try to profile the user's 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.

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 SpaceTime 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. www.Research.ijcaonline.org
8. www.towardsdatascience.com
9. www.slideshare.com
10. www.wikipedia.com
11. www.ijert.org\research