MOVIE MAGIC: DEVELOPING A PERSONALIZED RECOMMENDATION SYSTEM FOR MOVIE ENTHUSIASTS USING MACHINE LEARNING

Niharikareddy Meenigea
Data Analyst and Research Scientist Viriginia International University, USA
readdyniharikaa2@gmail.com

ABSTRACT

One of the most popular and effective uses of machine learning in business is in recommender systems. This method of information filtering is employed to forecast the user's choice. The most frequently used areas for recommender systems are books, news, articles, music, videos, and movies, among other things. In this essay, we present a collaborative filtering-based movie recommendation system that uses user-provided data to analyse it and then suggests the movies that are most appropriate for the person at hand. The recommended movie list is arranged using a variety of machine learning techniques in accordance with the ratings that previous users have given these films.

Recommender systems are one of the most well-liked and successful applications of machine learning in business. The user's choice is predicted using this information filtering technique. The most often used categories for recommender systems include, among other things, books, news, articles, music, videos, and movies. In this paper, we offer a collaborative filtering-based system for movie selection that analyses user-provided data and then recommends the movies that are best suitable for the user at hand. Using a number of machine learning approaches, the recommended movie list is organised according to the ratings that previous users have given these movies.

Keywords : Machine Learning, Collaborative filtering, Scalability, Algorithm, Used based and Item based.

INTRODUCTION

1.1 Relevance of the Project:

An information filtering paradigm known as a recommendation system, also referred to as a recommendation engine, seeks to anticipate user preferences and make recommendations in line with these preferences. Today, a wide range of sectors, including those that deal with utilities, books, music, movies, television, clothes, and restaurants, extensively use these technologies. These systems collect information about a user's preferences and behaviour, which they then utilise to improve their suggestions in the future. Movies are a fundamental aspect of life. There are many various kinds of movies, such as those meant for amusement, those meant for teaching, children's animation movies, horror movies, and action movies.

Movies can simply be distinguished by their genres, such as comedy, suspense, animation, action, etc. The release year, language, director, and other factors can also be used to differentiate between movies. There are many films to choose from when browsing our most popular movies when watching movies online. With the help of movie recommendation systems, we can easily identify our favourite movies among all of these diverse genres of films, sparing us the stress of having to spend a lot of time looking for them. It is crucial that the system that recommends movies to us is really reliable and provides suggestions for the movies that are either most comparable to or identical to our interests. Many companies utilise recommendation algorithms to enhance consumer engagement and the shopping experience.

Client satisfaction and revenue are two of recommendation systems' most important benefits. The movie suggestion system is a very useful and important tool. However, because of the limitations with a pure

collaborative method, scalability concerns and low recommendation quality are also problems with movie recommendation systems.

1.2 Problem Statement:

The goal of the project is to recommend a movie to the user. Providing customers of online service providers with related content culled from relevant and irrelevant collections of objects.

1.3 Objective of the Projects

· Improving the Accuracy of the recommendation system
· Improve the Quality of the movie Recommendation system
· Improving the Scalability.
· Enhancing the user experience.

1.4 Scope of the Project:

This project seeks to provide trustworthy movie recommendations to the public. The project's goal is to develop movie recommendation systems that are more precise, high-quality, and scalable than existing pure algorithms. A hybrid technique is employed to achieve this by fusing collaborative filtering and content-based filtering. Recommendation algorithms are used in social networking sites as tools for information filtering to lessen data overload. The quality, accuracy, and scalability of movie recommendation systems can therefore be improved through more research in this field. The movie suggestion system is a very useful and important tool. Movie recommendation systems are, however, also impacted by scalability issues and subpar recommendation quality due to the drawbacks of a pure collaborative approach.

METHODOLOGY

Movie Recommendation System Using Collaborative Filtering:
Collaborative filtering systems analyse the user's behaviour and preferences and predict what they would like based on similarity with other users. There are two kinds of collaborative filtering systems; user-based recommender and item-based recommender.

1. Use-based filtering: Users' preferences are frequently taken into account while creating customised solutions. This strategy is based on consumer preferences. Users first rate some movies (1–5) before the procedure begins. Both implicit and explicit ratings are possible. When a person expressly ranks an item on a scale or gives it a thumbs-up or thumbs-down, the rating is known as an explicit rating. Often explicit ratings are hard to gather as not every user is much interested in providing feedbacks. In these scenarios, we gather implicit ratings based on their behaviour. For instance, if a user buys a product more than once, it indicates a positive preference. In context to movie systems, we can imply that if a user watches the entire movie, he/she has some likeability to it. Note that there are no clear rules in determining implicit ratings. Next, for each user, we first find some defined number of nearest neighbours. We calculate correlation between users' ratings using Pearson Correlation algorithm. The assumption that if two users' ratings are highly correlated, then these two users must enjoy similar items and products is used to recommend items to users.

2. Item-based filtering: Unlike the user-based filtering method, item-based focuses on the similarity between the item’s users like instead of the users themselves. The most similar items are computed ahead of time. Then for recommendation, the items that are most similar to the target item are recommended to the user.

OUTPUT

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

Data Collection and Pre-Processing

```
[4] # loading the data from the csv file to apandas dataframe
    movies_data = pd.read_csv('/content/movies.csv')
```

```
[5] # printing the first 5 rows of the dataframe
    movies_data.head()
```

	index	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity
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
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be	139.082615

Movie Recommendation Sytem

```
movie_name = input(' Enter your favourite movie name : ')

list_of_all_titles = movies_data['title'].tolist()

find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)

close_match = find_close_match[0]

index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]

similarity_score = list(enumerate(similarity[index_of_the_movie]))

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

```
Enter your favourite movie name : Superman
```

```
 Enter your favourite movie name : Superman
Movies suggested for you :

1 . Superman
2 . Superman II
3 . Superman IV: The Quest for Peace
4 . Man of Steel
5 . Superman III
6 . Crimson Tide
7 . Superman Returns
8 . Batman Returns
9 . Suicide Squad
10 . The Killer Inside Me
11 . The Dark Knight Rises
12 . Nanny McPhee and the Big Bang
13 . Batman Begins
14 . The Dark Knight
15 . The Godfather
16 . The Helix... Loaded
17 . Batman
18 . Batman
19 . Batman & Robin
20 . The Island of Dr. Moreau
21 . The Hunting Party
22 . The Abyss
23 . Steel
24 . Lethal Weapon 4
25 . Dick Tracy
26 . On the Waterfront
27 . 1941
28 . Star Trek IV: The Voyage Home
29 . Don Juan DeMarco
```

APPENDICES

```
import numpy as np
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# loading the data from the csv file to apandas dataframe
movies_data = pd.read_csv('/content/movies.csv')

# printing the first 5 rows of the dataframe
movies_data.head()


# number of rows and columns in the data frame
movies_data.shape
```

```
# selecting the relevant features for recommendation
selected_features = ['genres','keywords','tagline','cast','director']
print(selected_features)


for feature in selected_features:
  movies_data[feature] = movies_data[feature].fillna('')
combined_features = movies_data['genres']+' '+movies_data['keywords']+' '+movies_data['tagline']+' '+movies_
data['cast']+' '+movies_data['director']
print(combined_features)

feature_vectors = vectorizer.fit_transform(combined_features)
similarity = cosine_similarity(feature_vectors)
find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index_of_the_movie)
sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)
print(sorted_similar_movies)
print('Movies suggested for you : \n')

i = 1
for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
movie_name = input(' Enter your favourite movie name : ')

list_of_all_titles = movies_data['title'].tolist()
find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
close_match = find_close_match[0]
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
similarity_score = list(enumerate(similarity[index_of_the_movie]))
sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

REFERENCES

[1] Hirdesh Shivhare, Anshul Gupta and Shalki Sharma (2015), "Recommender system using fuzzy c-means clustering and genetic algorithm based weighted similarity measure", IEEE International Conference on Computer, Communication and Control.

[2] Manoj Kumar, D.K. Yadav, Ankur Singh and Vijay Kr. Gupta (2015), "A Movie Recommender System: MOVREC", International Journal of Computer Applications (0975 – 8887) Volume 124 – No.3.

[3] Mohapatra, H., Panda, S., Rath, A., Edalatpanah, S., & Kumar, R. (2020). A tutorial on powershell pipeline and its loopholes. International journal of emerging trends in engineering research, 8(4), 975-982.