



International Centre for Education and Research (ICER), VIT-Bangalore

Second and Third Review

Project Title: MOVIE RECOMMENDATION SYSTEM
USING MACHINE LEARNING
Domain: Entertainment

Course Code: CS7510
Course Title: Project 2

Register Number: 23MSP3128
Name: DEVAKINANDAN PALLA
Date: 11-03-24

ABSTRACT

- To build an effective content recommendation system that minimizes the downsides of content based filtering and creating a more intelligent recommender system extracting valuable information from data.

Methodology used in base paper

- The methodology used in the paper on content-based recommendation systems involves a novel approach for calculating feature weights to improve movie representation.
- It includes material collection and quantitative analysis, with a focus on text information analysis to offer recommendations.
- The paper also discusses the unique characteristics of movies that set them apart in recommender systems, such as diversity and uniqueness, which are utilized to build a movie prototype and determine similarity.

Introduction

Types of Movie Recommender Systems: The three major types of movie recommender systems are content based, collaborative filtering and a hybrid model that uses best of both the features.

What is Content Based Filtering: Imagine a movie recommender that pays attention to what you love - it looks at the genre, cast, director, period of release, and other details of movies you enjoy to suggest similar ones, making your movie-watching experience much to your taste. This can be realized by the process called content based filtering.

What is Collaborative Filtering Method: Picture a friend who knows your movie preferences inside out - collaborative filtering works a bit like that, analyzing what others with similar tastes enjoyed to recommend movies you might also love, creating a sense of shared movie taste or pattern of association watching among users.

Disadvantage of Collaborative Filtering: One disadvantage of collaborative filtering is the "cold start" problem, where new users or items have limited or no interaction history, making it challenging to provide accurate recommendations. Another disadvantage of collaborative filtering is the "popularity bias" or "herding effect," where popular items tend to receive more recommendations, leading to limited diversity in recommendations and potentially overlooking niche or lesser-known items.

How does Content Based Filtering aim to solve this issue: Content-based filtering overcomes this cold start problem by relying on the intrinsic attributes of items, such as genre or keywords, to make initial recommendations for users. Also, recommend less popular but highly relevant items based on the user's preferences and tastes. This feature ensures that users receive recommendations relevant to their individual interests, irrespective of the popularity of the items, thereby enhancing the diversity and quality of recommendations, and avoiding this herding problem.

Creating a movie recommender website: An intelligent movie recommended system is thereby created and incorporated into a website using the necessary features and thus acting as a prototype for future enhancement by noting down the positive features of each model.

Need of study

- 1. Making Content More Interesting:** When we study ways to improve a content-based system, we're figuring out how to make the stuff you see online or on TV even cooler. This means finding ways to suggest movies, shows, or videos that you'll love based on what you already like.
- 2. Helping You Discover New Things:** Sometimes, it's hard to find new things to watch because there are so many options out there. By improving content-based systems, we can help you find new movies or shows that match your interests, making it easier for you to discover cool stuff you might not have found otherwise.
- **3. Additional ideas to improve a hybrid based system:** A hybrid based system works on the positives of both the content and the collaborative filtering system. An improved performance of content based system can provide massive insights.

Problem Statement

- In this modern days' digital age, streaming platforms like Netflix, Amazon Prime, and Hotstar are offering a vast collection of movies to their users for them to choose. However, with such a large set of choices, users often struggle to find and choose movies that match their preferences and interests and are thus often confused.
- To mitigate this challenge, the streaming platforms require a proper recommendation system to suggest relevant movies to users based on their past viewing history, preferences, and behavior.

Objectives

- 1.Enhancing User Experience:** The primary objective of a movie recommender system is to improve user satisfaction and engagement by providing personalized and relevant movie recommendations. By suggesting movies that align with users' preferences and interests, the recommender system enhances the overall viewing experience.
- 2.Increasing User Engagement:** A well-designed recommender system can increase user engagement by helping users discover new movies they may enjoy. By surfacing content that matches users' tastes, the system encourages users to spend more time on the platform and consume a greater variety of content.
- 3.Driving User Retention:** Recommender systems play a crucial role in retaining users on streaming platforms. By consistently delivering high-quality recommendations tailored to individual preferences, the system can reduce user churn and increase long-term user retention.

Objectives

- The ultimate objective is to develop an intelligent movie recommender system that can surpass any sort of minor abruptions or create confusion.
- The data is extracted cleverly to create a robust system which recommends user the movies that are thematically strong, have the actor or actress/ directed by the same director, and keeping in mind that the year of release is not too old or out of the zone of the movie that has been shown interest by the user.

Related Work

- The work being presented by me has already been attempted partially but never fully by various papers.
- I have observed some of the best features in various papers and incorporated every feature that I thought was interesting and merged them.
- I have also applied novelty on my own through extracting intelligent features from dataset from what I believed would make a good recommender system.
- The use of various libraries such as Count Vectorizer and Porter Stemmer, cosine similarity, Bag of Words technique have all been individually picked from papers.
- There was a lot of self learning and thought put into making things better and I am looking forward to expand my scope.

Proposed Methodology [Max. 3 Slides]



Task Pipeline:

- The initial phase involves rigorous data pre-processing, including the handling of null values, filling, duplicates, and outliers. The data is extracted from 2 files: one being the movies dataset and the other being the credits dataset.
- On the basis of title, the 2 dataset are merged and thus we have a highly informative dataset containing information ranging from cast, crew, year, release dates and so on.
- This merged dataset is unclean and firstly has to undergo some basic eda such as removal of null and duplicate values.
- The feature selection is done next and the features deemed to be informative personally were chosen. Genres, cast, crew, release date and the overview of the movie were selected.

Contd.

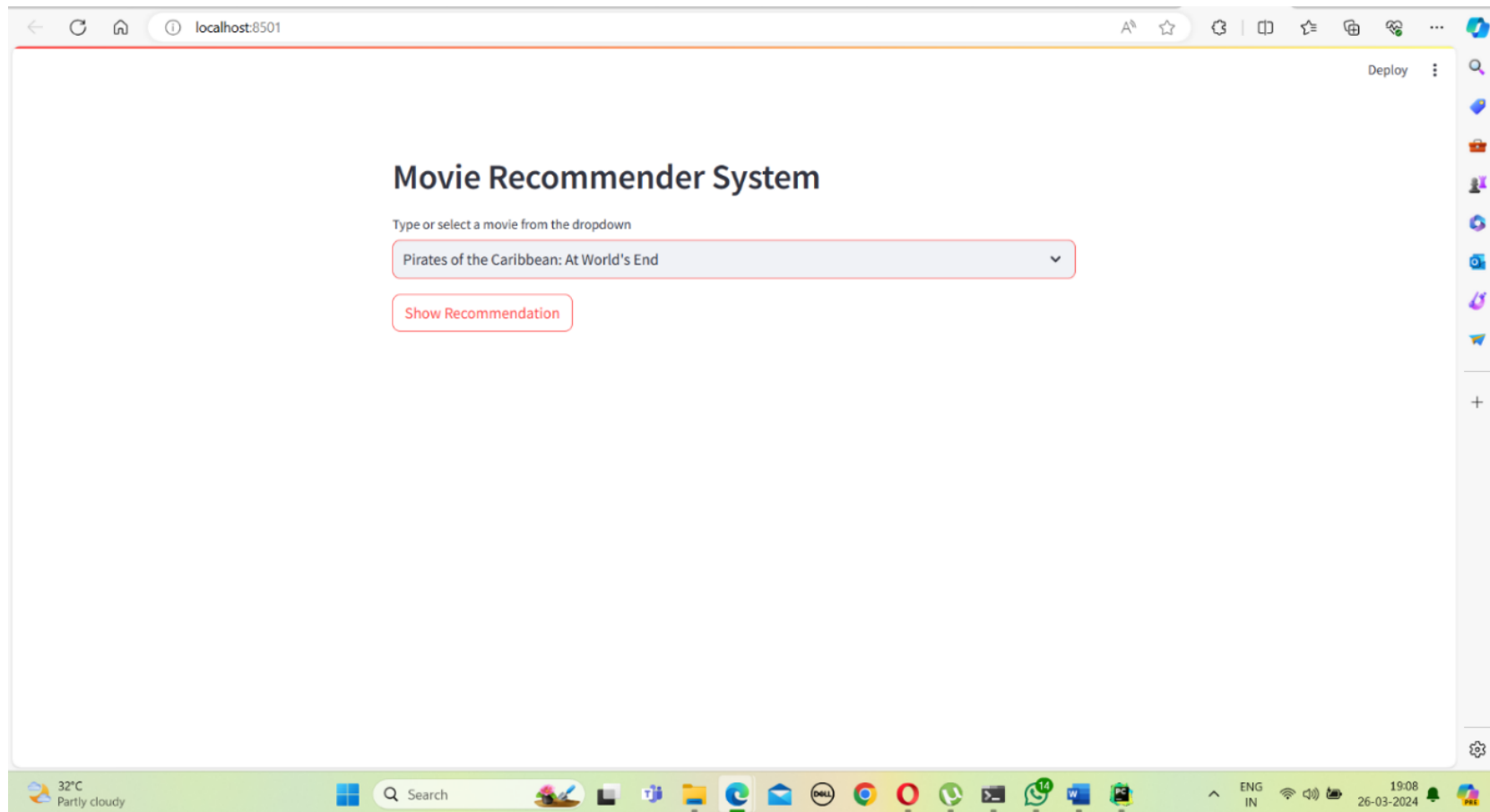
- Every column underwent a series of transformation using appropriate function being applied to it. The top 3 cast members were chosen from the name section of the cast, and the director alone was chosen from the crew.
- Release date was spilt into four sections: golden age, classic, semi classic and modern films that can provide users with recommendations according to their taste in movies released at one particular point in time or age of cinema.
- All the column specific cleaning was performed and a tag column was created that mixed the attributes of each of these features.
- Necessary visualization techniques were employed which displayed various necessary insights and provide a good idea about the importance and significance of various elements.
- After all that, vectorization is implemented using the method "Bag of words" which arranges the combined tag words in a ranking of most frequent 5000 words, and the tags of every movie now acts as a matrix where it can now match itself with the 5000 tags and arrange itself in the form of a vector with unique location in the matrix space.
- The similarity is then calculated and the top 5 most similar movies are recommended for every movie. This whole process is later incorporated on a website using PyCharm

Dataset

- Source of the Dataset- [TMDB 5000 Movie Dataset \(kaggle.com\)](https://www.kaggle.com/tmdb/tmdb-movie-metadata)
- No. of Observations:
- Column/Feature Details
- Details about the columns – Discrete, categorical, continuous etc...
- Screenshot of the dataset

Results and Discussion

- The toolbox has all the movie list that were extracted from the movie_list variable and are shown in dragdown.



- The below website shows top 5 similar movies to the POC series. The first 3 are franchise films and the other 2 are probably recommended on the basis of the keywords: water and ships, etc

Deploy






Movie Recommender System

Type or select a movie from the dropdown

Pirates of the Caribbean: At World's End


Show Recommendation

Pirates of the | Pirates of the | Pirates of the | Silver Medalist | Waterworld




32°C
Partly cloudy

Search

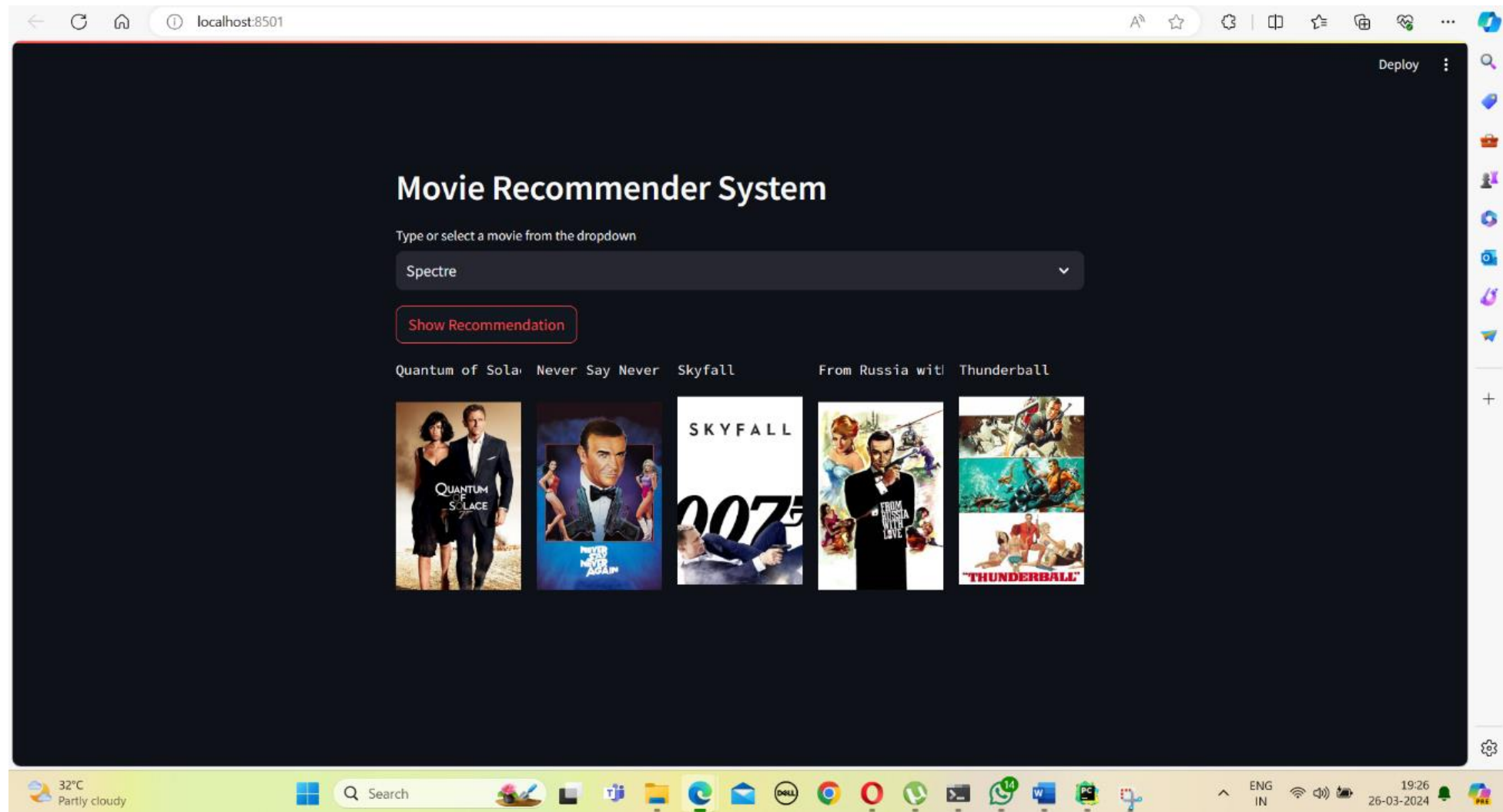


ENG
IN

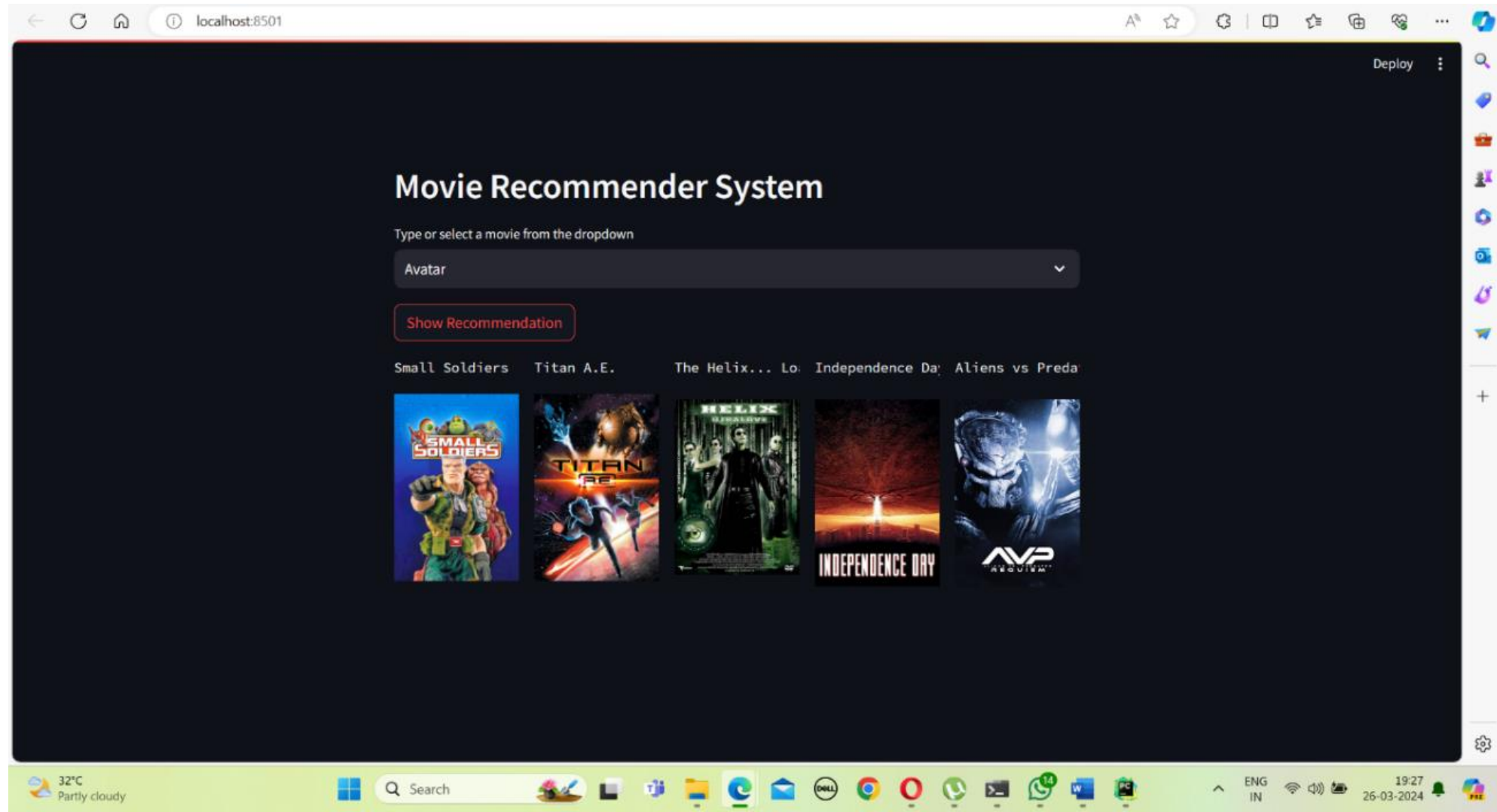
19:09
26-03-2024



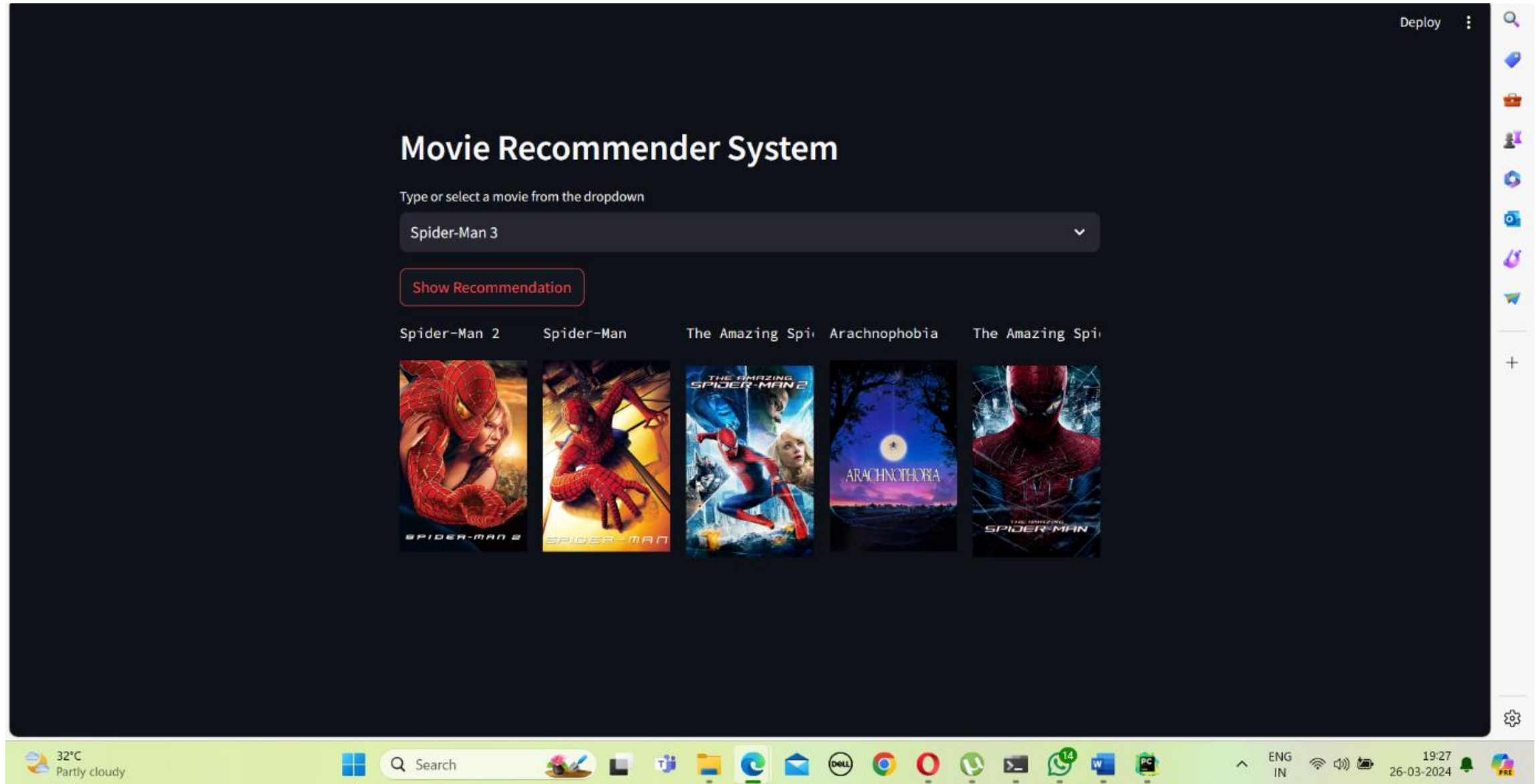
- After some theme adjustments, we got an attractive background to work on with example of the movie “Spectre”, which has movies recommended based on the character James Bond, detective movies done by the actor Daniel Craig and a similar genre movie set around the same period



- The dataset is a bit old which is probably the reason avatar 2 wasn't released into the database. The movie matrix was recommended due to keywords such as dream/simulation, etc. The movie independence day was probably suggested due to the presence of alien characters and the primary movie small soldiers is recommended because the story theme is about a colonel destroying the natives for his selfish needs which is thematically very similar to Avatar. I am extremely impressed at the fact that my recommender system drew such parallels that I couldn't observe earlier.



- Similarly spiderman recommendation is pretty much expected, and the recommendation of arachnophobia is probably because it is based on spiders involved in mysterious situations in a particular city.



Performance Analysis

- The Jaccardian similarity is 0.2(1/5).(One element match)

```
recommend('Avatar')
```

```
Small Soldiers  
Titan A.E.  
The Helix... Loaded  
Independence Day  
Aliens vs Predator: Requiem
```

```
from sklearn.metrics import jaccard_score  
  
def recommend_jaccard(movie):  
    index = new_df[new_df['title'] == movie].index[0]  
    movie_tags = vector[index] # Tags vector for the given movie  
    distances = [] # List to store distances  
    for i in range(len(vector)):  
        if i != index:  
            tag_similarity = jaccard_score(movie_tags, vector[i], average='micro')  
            distances.append((i, tag_similarity))  
    distances.sort(reverse=True, key=lambda x: x[1])  
    for i in distances[:5]:  
        print(new_df.iloc[i[0]].title)
```

```
recommend('Avatar')
```

```
The Helix... Loaded  
Amidst the Devil's Wings  
Khiladi 786  
The Ten  
Harrison Montgomery
```

The Jaccardian distance can go up to 0.4 for macro averaging

```
In [96]: recommend_similarity('Frozen')
```

```
The Book of Life  
Aladdin  
Jungle Shuffle  
Delgo  
Brave
```

```
In [97]: from sklearn.metrics import jaccard_score
```

```
def recommend(movie):  
    index = new_df[new_df['title'] == movie].index[0]  
    movie_tags = vector[index] # Tags vector for the given movie  
    distances = [] # List to store distances  
    for i in range(len(vector)):  
        if i != index:  
            tag_similarity = jaccard_score(movie_tags, vector[i], average='macro')  
            distances.append((i, tag_similarity))  
    distances.sort(reverse=True, key=lambda x: x[1])  
    for i in distances[:5]:  
        print(new_df.iloc[i[0]].title)
```

```
# Choose a movie to recommend  
movie = 'Frozen'
```

```
print("Recommendations based on Jaccard similarity (macro-averaging):")  
recommend(movie)
```

```
Recommendations based on Jaccard similarity (macro-averaging):  
Aladdin  
The Book of Life  
The Adventurer: The Curse of the Midas Box  
The Simpsons Movie  
Pocahontas
```

```
recommend_similarity('Inception')
```

```
The Helix... Loaded  
The Count of Monte Cristo  
Flatliners  
Duplex  
Transformers: Revenge of the Fallen
```

```
from sklearn.metrics import jaccard_score
```

```
def recommend(movie):  
    index = new_df[new_df['title'] == movie].index[0]  
    movie_tags = vector[index] # Tags vector for the given movie  
    distances = [] # List to store distances  
    for i in range(len(vector)):  
        if i != index:  
            tag_similarity = jaccard_score(movie_tags, vector[i], average='macro')  
            distances.append((i, tag_similarity))  
    distances.sort(reverse=True, key=lambda x: x[1])  
    for i in distances[:5]:  
        print(new_df.iloc[i[0]].title)
```

```
# Choose a movie to recommend  
movie = 'Inception'
```

```
print("Recommendations based on Jaccard similarity (macro-averaging):")  
recommend(movie)
```

```
Recommendations based on Jaccard similarity (macro-averaging):  
Transformers: Revenge of the Fallen  
Duplex  
The Helix... Loaded  
A Sound of Thunder  
Chicago Overcoat
```

Conclusion

- In text document similarity, a Jaccard similarity of 0.2-0.4 is considered decent since the volume is large and diverse, and the similarity measure is being used to identify broad thematic similarities rather than exact matches, which is quite acceptable for our case.
- [1] has given a lot of insights in this regard.
- A website has also been designed to satisfy the recommender system.

Future Enhancement

- Deploying it onto a permanent website
- Incorporating more features like timestamp, duration and box office collection and eventually incorporating in a hybrid recommender system.

References : APA format

- [1]-Narang, S., Malhotra, H., Chaudhary, R., Gandhi, V., & Pokhriyal, S. Movie Recommendation by Using Jaccard Distance.
- Keshava, M. C., Srinivasulu, S., Reddy, P. N., & Naik, B. D. (2020). Machine learning model for movie recommendation system. *Int. J. Eng. Res. Tech*, 9, 800-801.
- Yogesh, S., & Kumar, S. G. MACHINE LEARNING BASED MOVIE RECOMMENDER APPLICATION WITH DEPLOYMENT.
- Eur. Chem. Bull. 2023, 12(Special Issue 4), 10264-10272
- Furtado, F., & Singh, A. (2020). Movie recommendation system using machine learning. *International journal of research in industrial engineering*, 9(1), 84-98.
- Mohammad, J. F., & Urolagin, S. (2022, May). Movie Recommender System Using Content-based and Collaborative Filtering. In *2022 IEEE IAS Global Conference on Emerging Technologies (GlobConET)* (pp. 963-968). IEEE.