MOVIE RECOMMENDER SYSTEM

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

by

ADITYA SHARMA (19BCD7165)
ABHISHEK BASU (19BCE7326)
SWAPNIL SAH (19BCE7458)

Under the Guidance of

DR. HARI SEETHA

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

JANUARY 2022

CERTIFICATE

This is to certify that the Capstone Project work titled “**MOVIE RECOMMENDER SYSTEM**” that is being submitted by SWAPNIL SAH(19BCE7458), ABHISHEK BASU(19BCE7326) and ADITYA SHARMA(19BCD7165) is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. Hari Seetha
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR
B. Tech. CSE

DEAN
School of Computer Science and Engineering

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to my project guide, Dr. Hari Seetha, Department of Computer Science and Engineering, Vellore Institute of Technology, Amaravati, for her expert guidance and constant supervision in structuring our project work. Without her consistent mentorship it wouldn't have been possible to successfully complete this project. We are highly thankful to Vellore Institute of Technology, Amaravati, for providing us with all the necessary information and updates regarding the project throughout the course. We would also like to thank our friends and family for their support and for making all the requirements available to complete this project on time.

ABSTRACT

Recommender System is a tool helping users locate content material and overcome information overload. It predicts interests of users and makes recommendations consistent with the interest model of users. The unique content material-based recommender system is the continuation and improvement of collaborative filtering, which doesn’t need the user’s assessment for objects. Instead, the similarity is calculated primarily based totally on the facts of objects which are selected by users, after which the recommendation is made accordingly. With the development of machine learning, the contemporary content material-based recommender system can construct profiles for customers and products respectively. Building or updating the profile consistent with the evaluation of objects which are sold or visited by customers. The device can evaluate the person and the profile of objects after which advise the maximum comparable merchandise. So this recommender technique that compares person and product immediately can not be added right into a collaborative filtering model. The basis of a content-based algorithm is acquisition and quantitative evaluation of the content. As the studies of acquisition and filtering of textual content facts are mature, many contemporary content-based recommender systems make recommendations consistent with the evaluation of textual content. There are loads of functions extracted from the movie, they are various and unique, which is likewise the distinction from different recommender structures. We use those functions to assemble film fashions and calculate similarity. Finally we compare the method to demonstrate the development.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION AND LITERATURE SURVEY

1. Introduction

In recent years, improving the effectiveness of the commercial web services especially to use the personalized recommendation technology to realize the electronic commerce personalized service has gradually become a hot topic that can cause widespread interest. But at present, most e-commerce recommendations are usually: recommended best-selling products. Recommend related product according to user's browsing history is recommended so to speak, the first two recommended due to fundamental not considering the personality traits of the different users, therefore recommend simply does not have the characteristics of individuation, the third recommend a personalized composition, but most of the site also only stay in only the users against a person's purchase history, just for each user set up a personal purchase records, no transverse to the comprehensive information, so there is no collaboration recommended value which also is unable to realizes real-time comprehensive recommended goods.

Movies are a part and parcel of life. There are different types of movies like some for entertainment, some for educational purposes, some are animated movies for children, and some are horror movies or action films. Movies can be easily differentiated through their genres like comedy, thriller, animation, action etc. Other ways to distinguish among movies can be either by releasing year, language, director etc. Watching movies online, there are a number of movies to search for in our most liked movies . Movie Recommendation Systems helps us to search our preferred movies among all of these different types of movies and hence reduce the trouble of spending a lot of time searching our favorable movies. So, it requires that the movie recommendation system should be very reliable and should provide us with the recommendation of movies which are exactly the same or most matched with our preferences.

1.1 Objectives

Problem Statement

For building a recommender system from scratch, we come across numerous problems. There already exist a number of recommender systems based on the user information, so what should we do in case the website hasn't achieved the required number of users. We will then go on to solve the representation of a movie, which is how a system can understand a movie. That is the precondition for comparing similarity between two movies. Movie features such as genre, actor and director is a way that can categorize movies. But for each feature of the movie, there should be different weight for them and each of them plays a different role for recommendation. So we get these questions:

- How to recommend movies when there is no user information.
- What kind of movie features can be used for the recommender system.
- How to calculate the similarity between two movies.
- Is it possible to set weight for each feature?

1.2 Background and Literature Survey

In the age of information overload, it is very difficult for users to get information that really interests them. And it is also very difficult for the content providers to differentiate their content from the crowd. Therefore, many researchers and companies are developing recommendation systems to resolve the contradiction. The mission of the recommender system is to connect users and information, which on the one hand helps

users to find information valuable to them and on the other hand directs information to specific users. This is a win-win situation for customers and content providers.
This final report introduces a more practical recommendation method that can be used on a movie website that doesn't have enough users.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology, hardware and software details.
- Chapter 3 contains the software and hardware specifications.
- Chapter 4 discusses the results obtained after the project was implemented.
- Chapter 5 concludes the report and mentions future work.
- Chapter 6 consists of codes.

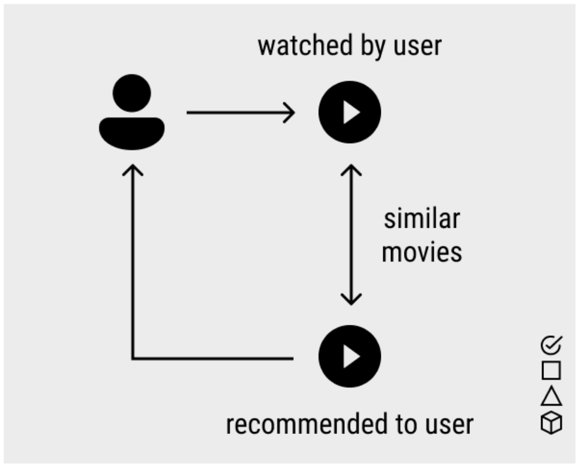
CHAPTER 2

MOVIE RECOMMENDER SYSTEM

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 Proposed System

The following block diagram shows the system architecture of this project.



2.2 Working Methodology

1.**Collection of Data**: Collecting all the required data sets from Kaggle and TMDB. In this project we also collected and fetched data from web sources.

2.**Data Analysis**: Make sure that the collected data sets are correct and analyze the data in the csv files. i.e. checking whether all the column fields are present in the data sets.

3.**Algorithms**: In our project we have used an algorithm called cosine similarity to build the machine learning recommendation model.

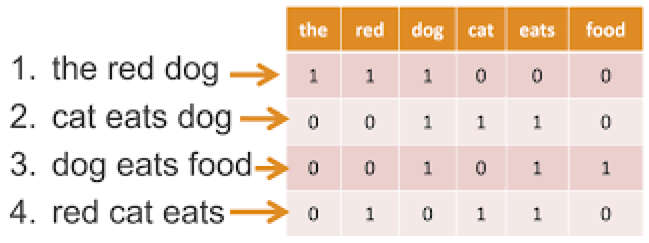
4.**Improvements in the project**: In the later stage we can implement different algorithms and methods for better recommendation.

2.3 Standards

Cosine Similarity: Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

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

CountVectorizer: CountVectorizer means breaking down a sentence or any text into words by performing preprocessing tasks like converting all words to lowercase, thus removing special characters. In NLP models can't understand textual data they only accept numbers, so this textual data needs to be vectorized.



	the	red	dog	cat	eats	food
1. the red dog	1	1	1	0	0	0
2. cat eats dog	0	0	1	1	1	0
3. dog eats food	0	0	1	0	1	1
4. red cat eats	0	1	0	1	1	0

TF-IDF: TF-IDF (Term Frequency - Inverse Document Frequency) is a handy algorithm that uses the frequency of words to determine how relevant those words are to a given document. It's a relatively simple but intuitive approach to weighting words, allowing it to act as a great jumping off point for a variety of tasks. This includes building search engines, summarizing documents, or other tasks in the information retrieval and machine learning domains.

CHAPTER 3
SYSTEM REQUIREMENTS SPECIFICATION

This chapter involves both the hardware and software requirements needed for the project and detailed explanation of the specifications.

3.1 Hardware Requirements

- A PC with Windows/Linux OS
- Processor with 1.7-2.4gHz speed
- Minimum of 8gb RAM
- 2gb Graphic card

3.2 Software Specification

- Text Editor (jupyter notebook/google colab)
- Anaconda distribution package (PyCharm Editor)
- Python libraries

3.3 Software Requirements

3.3.1 Anaconda distribution:

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system conda. The anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.3

3.3.2 Python libraries:

For the computation and analysis we need certain python libraries which are used to perform analytics. Packages such as SKlearn, Numpy, pandas, etc are needed.

SKlearn: It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

NumPy: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

Pandas: Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike the NumPy library which provides objects for multi-dimensional arrays, Pandas provides an in-memory 2d table object called Data frame.

Streamlit: It is an open-source app framework for Machine Learning and Data Science teams to create beautiful web apps in minutes.

Nltk: NLTK is a leading platform for building Python programs to work with human language data.

CHAPTER 4

RESULTS AND DISCUSSIONS

We successfully recommended movies based on their respective titles.

Since our project is a movie recommendation system. One can develop a movie recommendation system by using either content based or collaborative filtering or combining both. In our project we have developed a content-based approach. In content based filtering it is based on the user ratings or user likes only such kind of movie will be recommended to the user.

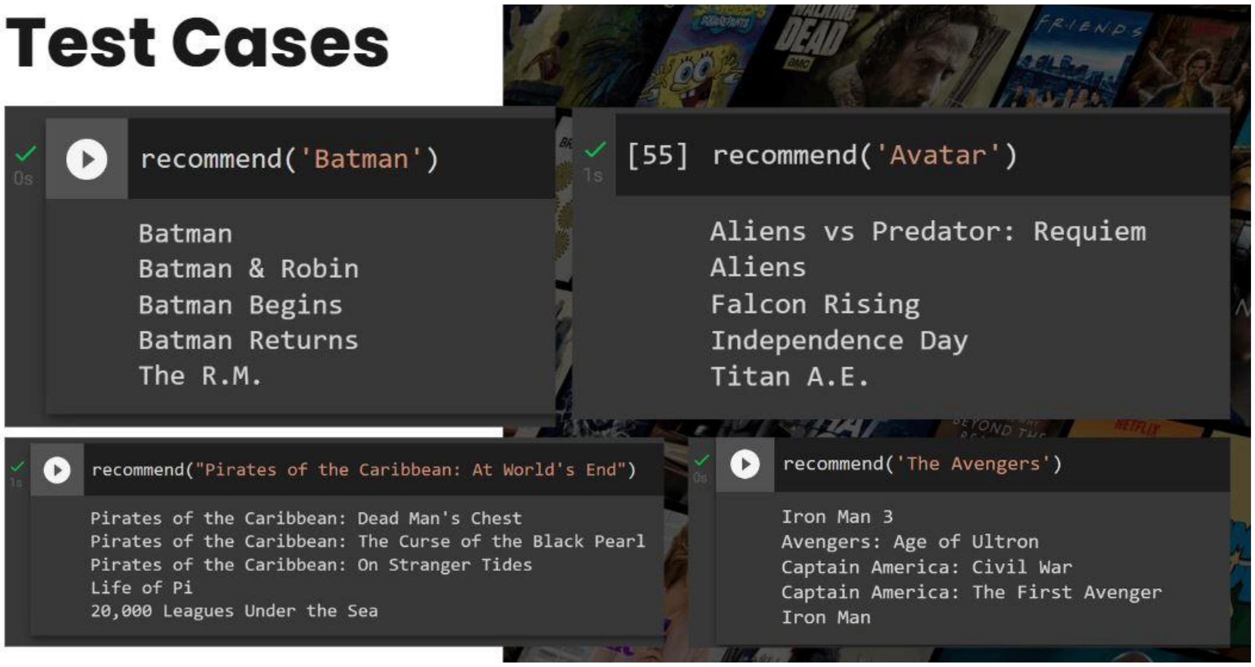
Advantages: It is easy to design and it takes less time to compute

Dis-advantages: The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

In Collaborative filtering the recommendation is comparison of similar users.

Advantages: No need for domain knowledge because the embeddings are automatically learned. The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.

Dis-advantages: The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item. This issue is often called the cold-start problem.



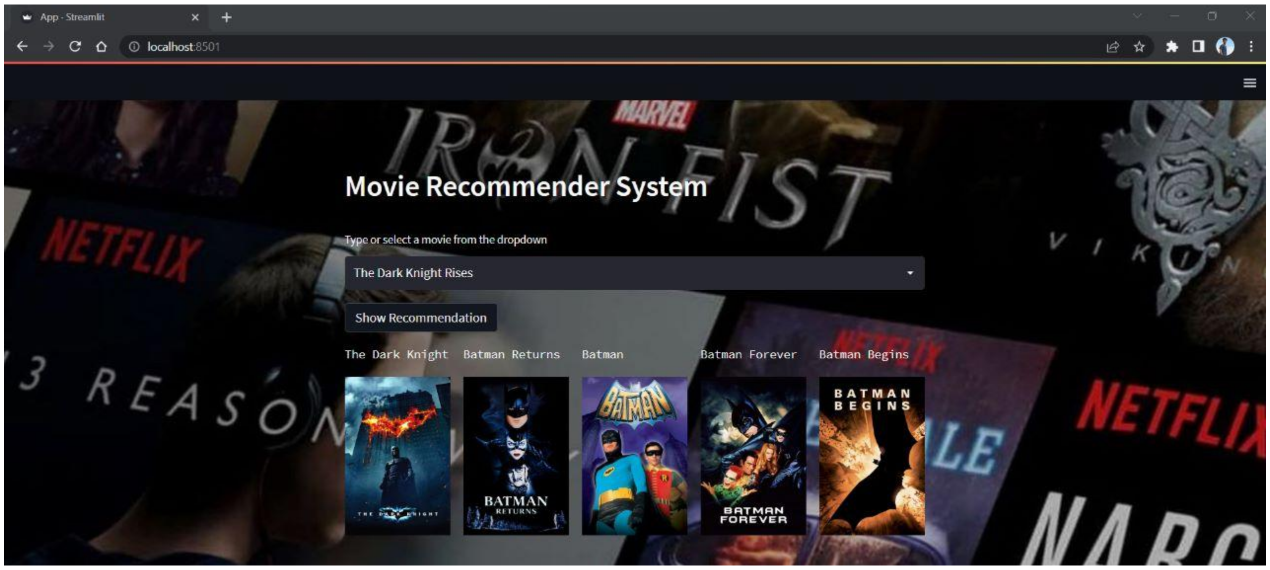
Test Cases

```
recommend('Batman')

Batman
Batman & Robin
Batman Begins
Batman Returns
The R.M.
```

```
[55] recommend('Avatar')

Aliens vs Predator: Requiem
Aliens
Falcon Rising
Independence Day
Titan A.E.
```

```
recommend("Pirates of the Caribbean: At World's End")

Pirates of the Caribbean: Dead Man's Chest
Pirates of the Caribbean: The Curse of the Black Pearl
Pirates of the Caribbean: On Stranger Tides
Life of Pi
20,000 Leagues Under the Sea
```

```
recommend('The Avengers')

Iron Man 3
Avengers: Age of Ultron
Captain America: Civil War
Captain America: The First Avenger
Iron Man
```

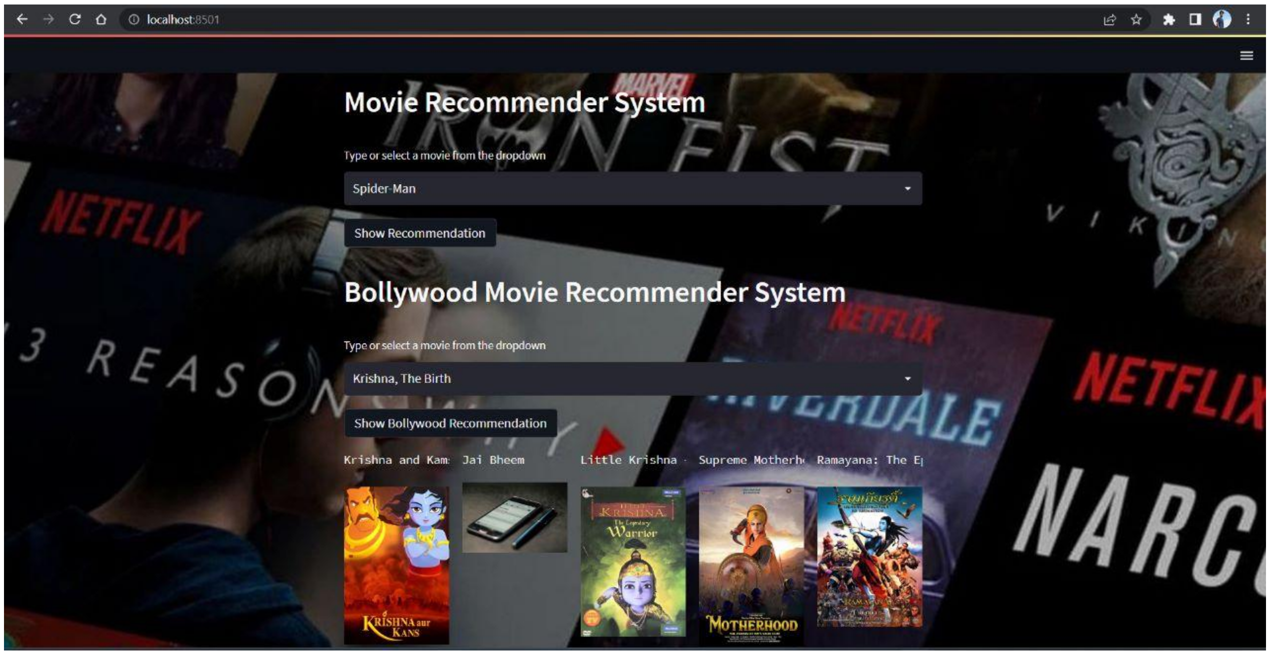


Movie Recommender System

Type or select a movie from the dropdown

The Dark Knight Rises

The Dark Knight   Batman Returns   Batman   Batman Forever   Batman Begins


localhost:8501
Movie Recommender System
Type or select a movie from the dropdown
Spider-Man
Show Recommendation
Bollywood Movie Recommender System
Type or select a movie from the dropdown
Krishna, The Birth
Show Bollywood Recommendation
Krishna and Kam
Jai Bheem
Little Krishna
Supreme Motherh
Ramayana: The E

CHAPTER 5

CONCLUSION AND FUTURE WORK

Recommender system has become more and more important because of the information overload. For a content-based recommender system specifically, we attempt to find a new way to improve the accuracy of the representative of the movie. For the problems we mentioned at the beginning, firstly, we use a content-based recommender algorithm which means there is no cold start problem. We list all the features in our recommender system. We have extracted these features from the genres, directors, cast of the company and are accurate.Then we introduced the cosine similarity which is commonly used in industry. For the weight of features, we introduced TF-IDF which improves the representation of the movie. This thesis introduces a content-based recommender system for the movie
website.The features used in the system are extracted from various aspects of the movie, which are diverse and unique. We introduce a new approach for setting weight for these features, the movie can be represented more accurately by TF-IDF which is the key point of our research.

In the proposed approach, It has considered Genres of movies but, in future we can also consider age of user as according to the age movie preferences also changes, like for example, during our childhood we like animated movies more as compared to other movies. There is a need to work on the memory requirements of the proposed approach in the future. The proposed approach has been implemented here on different movie datasets only. It can also be implemented on the Film Affinity and Netflix datasets and the performance can be computed in the future.

CHAPTER 6

APPENDIX

Code

```
import numpy as np
import pandas as pd
movies = pd.read_csv('E:\\practice data\\tmdb_5000_movies.csv')
credits = pd.read_csv('E:\\practice data\\tmdb_5000_credits.csv')
movies.head(2)
movies.shape
credits.head()
movies = movies.merge(credits,on='title')
movies.head()
movies                                                    =
movies[['movie_id','title','overview','genres','keywords','cast','crew']]
movies.head()
import ast
def convert(text):
L = []
for i in ast.literal_eval(text):
L.append(i['name'])
return L
```

```
movies.dropna(inplace=True)
movies['genres'] = movies['genres'].apply(convert)
movies.head()
movies['keywords'] = movies['keywords'].apply(convert)
movies.head()
import ast
ast.literal_eval('[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]')
def convert3(text):
L = []
counter = 0
for i in ast.literal_eval(text):
if counter < 3:
L.append(i['name'])
counter+=1
return L
movies['cast'] = movies['cast'].apply(convert)
movies.head()
movies['cast'] = movies['cast'].apply(lambda x:x[0:3])
def fetch_director(text):
L = []
for i in ast.literal_eval(text):
if i['job'] == 'Director':
L.append(i['name'])
return L
movies['crew'] = movies['crew'].apply(fetch_director)
```

```
movies.head(5)
def collapse(L):
L1 = []
for i in L:
L1.append(i.replace(" ",""))
return L1
movies['cast'] = movies['cast'].apply(collapse)
movies['crew'] = movies['crew'].apply(collapse)
movies['genres'] = movies['genres'].apply(collapse)
movies['keywords'] = movies['keywords'].apply(collapse)
movies.head()
movies['overview'] = movies['overview'].apply(lambda x:x.split())
movies['overview'][0]
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords']
+ movies['cast'] + movies['crew']
new = movies.drop(columns=['overview','genres','keywords','cast','crew'])
new['tags'] = new['tags'].apply(lambda x: " ".join(x))
new.head()
new.info()
new['tags'][0]
import nltk
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
def stem(text):
y = []
for i in text.split():
y.append(ps.stem(i))
```

```
return " ".join(y)
stem(new['tags'][0])
new['tags'] = new['tags'].apply(stem)
new.head()
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')
vector = cv.fit_transform(new['tags']).toarray()
vector
vector[0]
cv.get_feature_names()
vector.shape
from sklearn.metrics.pairwise import cosine_similarity
similarity = cosine_similarity(vector)
cosine_similarity(vector).shape
similarity
new[new['title'] == 'The Lego Movie'].index[0]
def recommend(movie):
index = new[new['title'] == movie].index[0]
distances = sorted(list(enumerate(similarity[index])),reverse=True,key =
lambda x: x[1])
for i in distances[1:6]:
print(new.iloc[i[0]].title)
```

Front End

```
import pickle
import streamlit as st
import requests
import base64

def add_bg_from_local(image_file):
    with open(image_file, "rb") as image_file:
        encoded_string = base64.b64encode(image_file.read())
    st.markdown(
    f"""
    <style>
```

```
   .stApp {{
                                                         background-image:
url(data:image/{"png"};base64,{encoded_string.decode()});
       background-size: cover
   }}
   </style>
   """,
   unsafe_allow_html=True
   )
add_bg_from_local('cover.jpg')
def fetch_poster(movie_id):
   try:
                                                                url          =
"https://api.themoviedb.org/3/movie/{}?api_key=8265bd1679663a7ea12ac1
68da84d2e8&language=en-US".format(movie_id)
       data = requests.get(url)
       data = data.json()
       poster_path = data['poster_path']
       full_path = "https://image.tmdb.org/t/p/w500/" + poster_path
       return full_path
   except:
                                                          full_path          =
"https://images.unsplash.com/photo-1517404215738-15263e9f9178?ixlib=r
b-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8Nnx8dXJsfGVufDB8fDB
8fA%3D%3D&w=1000&q=80"
       return full_path
```

```
def recommend(movie):
    index = movies[movies['title'] == movie].index[0]
        distances = sorted(list(enumerate(similarity[index])), reverse=True, 
key=lambda x: x[1])
    recommended_movie_names = []
    recommended_movie_posters = []
    for i in distances[1:6]:
        # fetch the movie poster
        movie_id = movies.iloc[i[0]].movie_id
        recommended_movie_posters.append(fetch_poster(movie_id))
        recommended_movie_names.append(movies.iloc[i[0]].title)

    return recommended_movie_names,recommended_movie_posters

st.header('Movie Recommender System')
movies= pickle.load(open('movie_list.pkl','rb'))
similarity = pickle.load(open('similarity.pkl','rb'))

movie_list = movies['title'].values
selected_movie = st.selectbox(
    "Type or select a movie from the dropdown",
    movie_list
)
if st.button('Show Recommendation'):
        recommended_movie_names , recommended_movie_posters = 
recommend(selected_movie)
```

```
    col1, col2, col3, col4, col5 = st.columns(5)
    with col1:
        st.text(recommended_movie_names[0])
        st.image(recommended_movie_posters[0])
    with col2:
        st.text(recommended_movie_names[1])
        st.image(recommended_movie_posters[1])

    with col3:
        st.text(recommended_movie_names[2])
        st.image(recommended_movie_posters[2])
    with col4:
        st.text(recommended_movie_names[3])
        st.image(recommended_movie_posters[3])
    with col5:
        st.text(recommended_movie_names[4])
        st.image(recommended_movie_posters[4])
def fetch_poster(movie_id):
    try:
                                                                url          =
"https://api.themoviedb.org/3/movie/{}?api_key=8265bd1679663a7ea12ac1
68da84d2e8&language=en-US".format(movie_id)
        data = requests.get(url)
        data = data.json()
        poster_path = data['poster_path']
        full_path = "https://image.tmdb.org/t/p/w500/" + poster_path
        return full_path
    except:
        full_path = "https://images.unsplash.com/photo-1517404215738-15263e9f9178?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8Nnx8dXJsfGVufDB8fDB8fA%3D%3D&w=1000&q=80"
        return full_path



def recommend(movie):
    index = movies[movies['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])
    recommended_movie_names = []
    recommended_movie_posters = []
    for i in distances[1:6]:
        # fetch the movie poster
        movie_id = movies.iloc[i[0]].movie_id
        recommended_movie_posters.append(fetch_poster(movie_id))
        recommended_movie_names.append(movies.iloc[i[0]].title)

    return recommended_movie_names, recommended_movie_posters



st.header('Bollywood Movie Recommender System')
movies = pickle.load(open('movie_listonlyB.pkl', 'rb'))
similarity = pickle.load(open('similarityonlyB.pkl', 'rb'))
```

```
movie_list = movies['title'].values
selected_movie = st.selectbox(
    "Type or select a movie from the dropdown",
    movie_list
)
if st.button('Show Bollywood Recommendation'):
        recommended_movie_names,    recommended_movie_posters    =
recommend(selected_movie)
    col1, col2, col3, col4, col5 = st.columns(5)
    with col1:
        st.text(recommended_movie_names[0])
        st.image(recommended_movie_posters[0])
    with col2:
        st.text(recommended_movie_names[1])
        st.image(recommended_movie_posters[1])

    with col3:
        st.text(recommended_movie_names[2])
        st.image(recommended_movie_posters[2])
    with col4:
        st.text(recommended_movie_names[3])
        st.image(recommended_movie_posters[3])
    with col5:
        st.text(recommended_movie_names[4])
        st.image(recommended_movie_posters[4])
```

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] RyuRi Kim, Ye Jeong Kwak, HyeonJeong Mo, Mucheol Kim, Seungmin Rho,Ka Lok Man, Woon Kian Chong (2015),“Trustworthy Movie Recommender System with Correct Assessment and Emotion Evaluation”, Proceedings of the International MultiConference of Engineers and Computer Scientists Vol II.

[4] Zan Wang, Xue Yu*, Nan Feng, Zhenhua Wang (2014), “An Improved Collaborative Movie Recommendation System using Computational Intelligence”,Journal of Visual Languages & Computing,Volume 25, Issue 6.

[5] Debadrita Roy, Arnab Kundu, (2013), “Design of Movie Recommendation System by Means of Collaborative Filtering”, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 4.

BIODATA

Name	: ABHISHEK BASU
Mobile Number	: 8900472197
E-mail	: abhishek.19bce7326@vitap.ac.in
Permanent Address	: Inda in front of guru pir baba kharagpur , west bengal , 721301

Name	: ADITYA SHARMA
Mobile Number	: 9711478330
E-mail	: aditya.19bcd7165@vitap.ac.in
Permanent Address	: J-10, Third Floor, Saket,New Delhi-110017

Name	: SWAPNIL SAH
Mobile Number	: 9389988495
E-mail	: swapnil.19bce7458@vitap.ac.in
Permanent Address	: Jeewan Amar Niwas, Ranikhet, UK-263645