MOVIE RECOMMENDER SYSTEM

A MINI PROJECT

Submitted by

KEVIN THOMAS KOSHY
[RA2011003010018]
RENNY SAM
[RA2011003010026]
JOEL SANTOSH GEORGE
[RA2011003010051]

Under the guidance of

MANOJ KUMAR N

(Guide Affiliation)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M.Nagar, Kattankulathur, Chengalpattu District

April 2023

MOVIE RECOMMENDER SYSTEM

A MINI PROJECT

Submitted by

KEVIN THOMAS KOSHY
[RA2011003010018]
RENNY SAM
[RA2011003010026]
JOEL SANTOSH GEORGE
[RA2011003010051]

Under the guidance of
MANOJ KUMAR N
(Guide Affiliation)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING
of
FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M.Nagar, Kattankulathur, Chengalpattu District

April 2023

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this project report **"Movie Recommender System"** is the bonafide work of **" Kevin Thomas Koshy (RA2011003010018), Renny Sam(RA2011003010026), Joel Santosh George (RA2011003010051) "** of III Year/VI Sem B.Tech(CSE) who carried out the mini project work under my supervision for the course 18CSC305J-Artificial Intelligence in SRM Institute of Science and Technology during the Academic Year 2022-2023(Even Semester).

M. Pushpalatha

Department of Computing Technology * SRM IST * Kattankulathur 603 203

Signature of HOD

Dr. M. Pushpalatha
Professor and Head
SRM Institute of Science and Technology

15/5/23

SIGNATURE

Mr. Manoj Kumar N
Assistant Professor, Department of Computing Technologies
School of Computing

ABSTRACT

This project aims to design and implement a movie recommender system using a combination of cosine similarity and content-based algorithms. The system will allow users to select a movie of their choice from a vast database and then receive similar suggestions for the same. The content-based algorithm will analyse the attributes of movies such as genre, director, actors, and plot keywords to suggest movies with similar attributes to the movie that the user has chosen. The system will use cosine similarity to compare the similarity between the attributes of the movies and generate recommendations based on the closest matches. The system will then use the similarity score to generate a list of recommended movies that are most similar to the user's preferences. The user will be able to search and view the recommendations with the help of a minimalistic UI using StreamLit which displays the posters of the movies as well for easy identification and navigation. The system will be built using Python programming language and will use data preprocessing techniques, and machine learning algorithm to generate accurate and effective recommendations. The performance of the system will be evaluated using various metrics such as accuracy, precision, recall, and F1 score. Ultimately, the goal of this project is to build a user-friendly and efficient movie recommender system that can provide personalised recommendations to users based on their individual preferences using cosine similarity and content-based algorithms.

TABLE OF CONTENTS

Chapter No.		Title	Page No.
		ABSTRACT	ii
		TABLE OF CONTENTS	iii
1		INTRODUCTION	1
	1.1	Relevance	
	1.2	Problem Statement	
	1.3	Software Requirements Specification	
2		LITERATURE SURVEY	3
3		METHODOLOGY	4
4		CODING AND TESTING	6
5		RESULTS AND DISCUSSIONS	9
	5.1	Output	
	5.2	Discussion and Result	
6		CONCLUSION AND FUTURE ENHANCEMENT	10
7		FIGURES	11
8		REFERENCES	15

INTRODUCTION

1.1 Relevance of the Project

A recommendation system or recommendation engine is a model used for information filtering where it tries to predict the preferences of a user and provide suggests based on these preferences. These systems have become increasingly popular nowadays and are widely used today in areas such as movies, music, books, videos, clothing, restaurants, food, places and other utilities. These systems collect information about a user's preferences and behaviour, and then use this information to improve their suggestions in the future.

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 way 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 for our favourable 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 same or most matched with our preferences.

A large number of companies are making use of recommendation systems to increase user interaction and enrich a user's shopping experience. Recommendation systems have several benefits, the most important being customer satisfaction and revenue. Movie Recommendation system is a very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffer from poor recommendation quality and scalability issues.

1.2 Problem Statement

There are two main types of recommender systems:

Collaborative Filtering: This approach recommends items based on the preferences of similar users. In Python, collaborative filtering can be implemented using libraries such as Surprise, which provides a variety of algorithms for collaborative filtering.

Content-based Filtering: This approach recommends items based on their attributes or features. In Python, content-based filtering can be implemented using libraries such as scikit-learn, which provides various algorithms for feature extraction and similarity calculation.

The primary goal of this project is to develop a content-based movie recommendation system that can recommend movies to users based on their preferred movie. The system uses TMDB database to fetch details of movies such as title, genre, runtime, rating, poster, etc., and uses the IMDB ID of the movie to perform web scraping on IMDB site to get the posters for the movies as well.

The report outlines the various steps involved in building the content-based movie recommendation system. We start by explaining the process of data collection using TMDB database and web scraping. Next, we describe the data preprocessing techniques used to clean and prepare the data for analysis. We then discuss the different similarity metrics used to calculate the similarity between movies. Finally, we present the results of the recommendation system and discuss future directions for improving the system.

Overall, this project provides a comprehensive approach for building a content-based movie recommendation system that can provide personalised recommendations to users based on their preferences. The system can be useful for online movie streaming platforms and other businesses that provide movie recommendations to their customers.

1.3 SOFTWARE REQUIREMENT SPECIFICATIONS

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

2 Software Specification
Text Editor (VS-code/WebStorm)
Anaconda distribution package (PyCharm Editor)
Python libraries

3 Software Requirements

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 the 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.

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,

streamlit framework, 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 NumPy library which provides objects for multi-dimensional arrays, Pandas provides an in-memory 2d table object called Data frame.

LITERATURE SURVEY

Movie recommendation systems have been the subject of extensive research in recent years, and several approaches have been proposed to address the challenge of providing accurate and personalised movie recommendations. Listed below is a brief literature survey on the existing state of movie recommendation systems:

Collaborative Filtering: Collaborative filtering is a widely used technique in movie recommendation systems. It involves analysing the past behaviour of users to identify similar users and recommend movies that have been highly rated by similar users. Collaborative filtering can be further classified into two types: user-based and item-based. User-based collaborative filtering recommends movies to a user based on the ratings of other users who have similar taste, while item-based collaborative filtering recommends movies similar to the ones a user has rated highly.

Content-based Filtering: Content-based filtering is another approach to movie recommendation that uses the features of movies such as genre, actors, and directors to recommend similar movies to the ones a user has watched or rated highly.

Hybrid Filtering: Hybrid filtering combines both collaborative and content-based filtering techniques to provide more accurate and diverse movie recommendations.

Deep Learning: Deep learning models such as neural networks have been applied to movie recommendation systems to learn from large amounts of data and provide more accurate recommendations.

Evaluation Metrics: Several evaluation metrics such as precision, recall, F1-score, and Mean Average Precision (MAP) have been proposed to measure the performance of movie recommendation systems.

Ethical and Social Issues: Research on movie recommendation systems has also highlighted ethical and social issues such as privacy, transparency, and fairness. It is important to ensure that the recommendations provided by these systems are not biased or discriminatory.

In summary, the existing state of movie recommendation systems involves a range of techniques and evaluation metrics, and there is ongoing research in areas such as deep learning and ethical considerations.

METHODOLOGY

The Methodology used for building the content-based movie recommender system is :

1. **Data collection:** Collect movie data using the TMDB database and perform web scraping of IMDB sites to obtain the posters for each movie.
2. **Data preprocessing:** Clean and preprocess the movie data. This includes removing stop words, tokenization, stemming, and lemmatization.
3. **Feature extraction:** Extract relevant features from the movie data such as genre, runtime, and rating. Use feature extraction techniques such as CountVectorizer to extract features.
4. **Similarity calculation:** Use the cosine similarity metric to calculate the similarity between movies based on their features.
5. **Recommendation generation:** Generate recommendations based on the user's preferred movie and the similarity between movies.
6. **Evaluation:** Evaluate the performance of the recommendation system using metrics such as precision, recall, and F1-score.
7. **Deployment:** Deploy the recommendation system on a web application.
8. **Maintenance:** Regularly update the movie data and user reviews to ensure the system is providing relevant recommendations. Consider adding new features and metrics to improve the performance of the recommendation system.

Key Aspects of the Project

- **Cosine Similarity:**
  Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. It is often used in information retrieval and natural language processing to compare the similarity of two documents based on their vector representations.

  In the context of text analysis, each document can be represented as a vector where the value of each dimension represents the frequency of a particular word in the document. The cosine similarity between two documents is then calculated as the cosine of the angle between their respective vector representations.

  The cosine similarity ranges from -1 to 1, where a value of 1 indicates that the two vectors are identical, 0 indicates that they are orthogonal (i.e., have no correlation), and -1 indicates that they are diametrically opposed. A higher value of cosine similarity implies a higher degree of similarity between the two documents.

- **CountVectorizer**
  CountVectorizer is a tool in the sklearn library used to convert a collection of text documents into a matrix of token counts. It essentially transforms the text data into numerical data that can be used in machine learning models.

  CountVectorizer works by counting the occurrence of each word (also known as a "token") in the entire corpus of documents and creating a matrix where each row represents a document and each column represents a unique word in the corpus. The cell values are the number of times that word appears in the corresponding document.

CODING AND TESTING

CODE:

Jupyter (Model):

```
import numpy as np
import pandas as pd
```

Python

```
movies=pd.read_csv('tmdb_5000_movies.csv')
credits=pd.read_csv('tmdb_5000_credits.csv')
```

Python

```
movies.head()
```

Python

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	prod
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{" "nam
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{" "nam
2	245000000	[{"id": 28, "name": "Action"}	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name":	en	Spectre	A cryptic message from Bond's	107.376788	[{"name": "Columbia Pictures", "id": 5}	[{"

```
import ast
def convert(obj):

    L=[]
    for i in ast.literal_eval(obj):
        L.append(i['name'])
    return L
```

Python

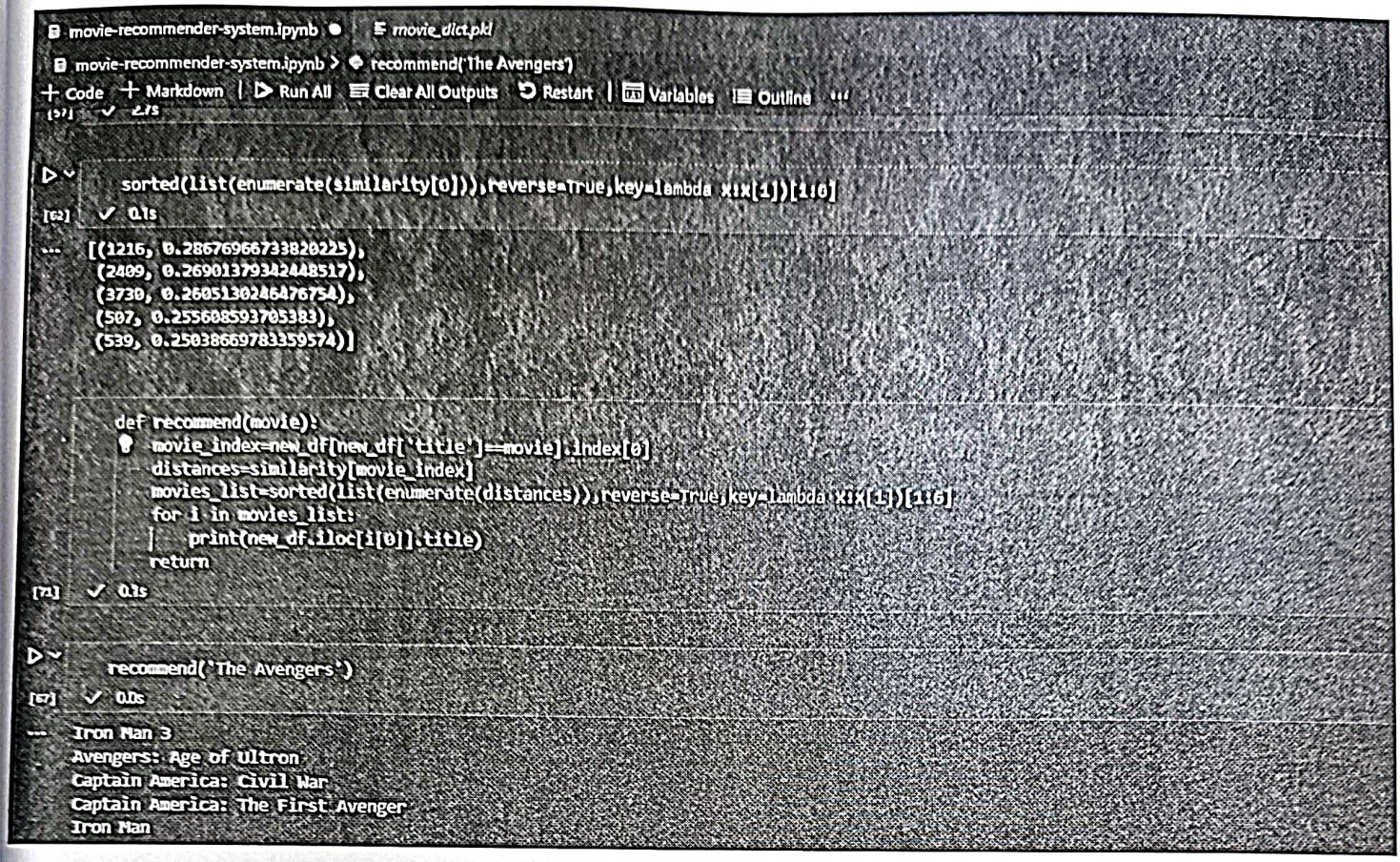
```
movies['genres']=movies['genres'].apply(convert)
```

Python

```
movies.head()
```

Python

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[{"id": 470, "name": "spy"}, {"id": 818, "name...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[{"id": 849, "name": "dc comics"}, {"id": 853,...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[{"id": 818, "name": "based on novel"}, {"id":...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...



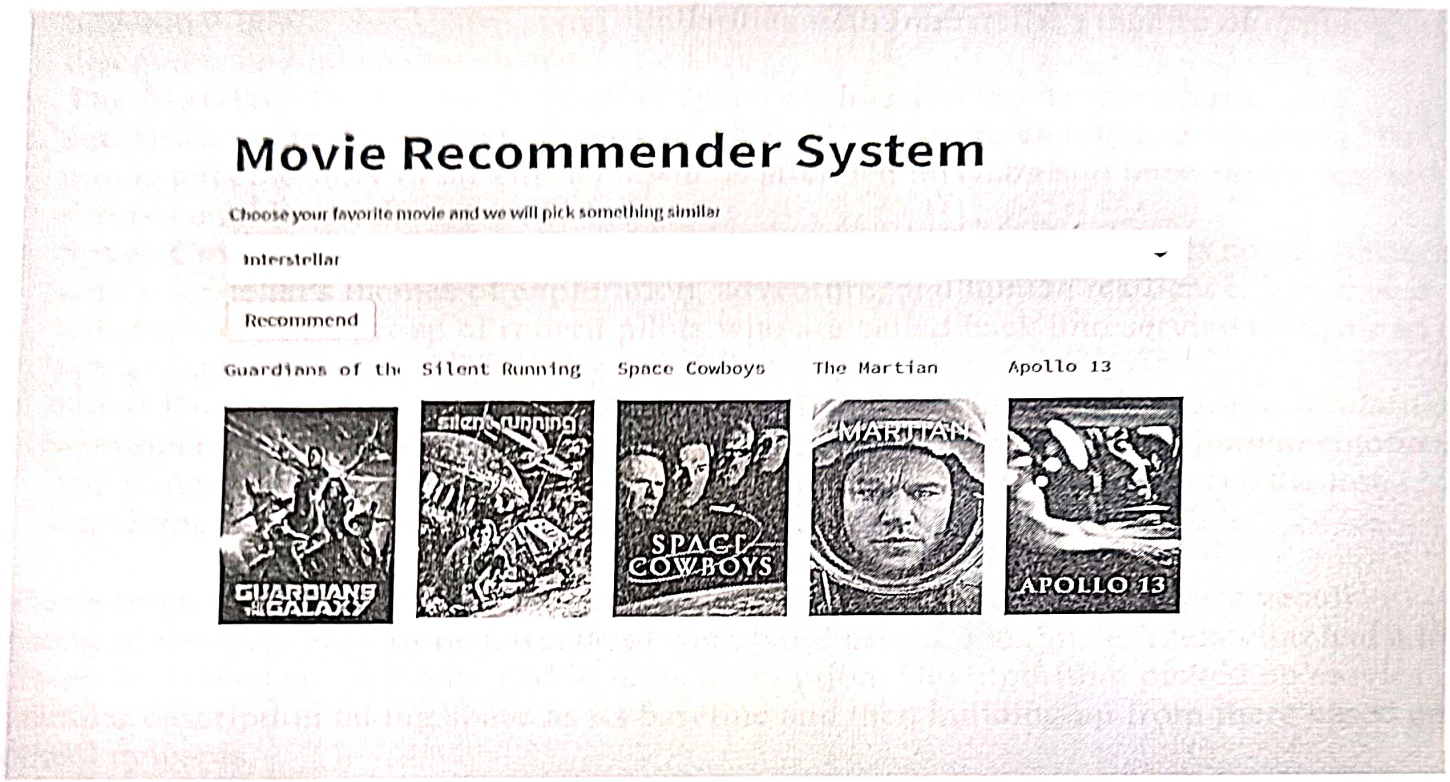
Streamlit (App.py)

```
import streamlit as st
import pickle
import pandas as pd
import requests

def fetch_poster(movie_id):
    url =
"https://api.themoviedb.org/3/movie/{}?api_key=8265bd1679663a7ea12ac168da84d2e8&la
nguage=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


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


movies_dict=pickle.load(open('movie_dict.pkl','rb'))
movies=pd.DataFrame(movies_dict)
similarity=pickle.load(open('similarity.pkl','rb'))

st.title('Movie Recommender System')
selected_movie_name=st.selectbox(
    'Choose your favorite movie and We will suggest some movies
similar',movies['title'].values
)

if st.button('Recommend'):
    recommended_movie_names, recommended_movie_posters =
recommend(selected_movie_name)
    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])
```

RESULTS AND DISCUSSION

OUTPUT:




```
movie-recommender-system.ipynb
movie-recommender-system.ipynb > recommend('Interstellar')
+ Code + Markdown | Run All Clear All Outputs Restart | Variables Outline ...
... array(['000', '007', '10', ..., 'zone', 'zoo', 'zooeydeschanel'],
      dtype=object)

from sklearn.metrics.pairwise import cosine_similarity

[54] ✓ 0.0s

similarity = cosine_similarity(vectors)

[55] ✓ 1.4s

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)

[56] ✓ 0.0s

recommend('Interstellar')

[58] ✓ 0.0s
... Guardians of the Galaxy
Silent Running
Space Cowboys
The Martian
Apollo 13
```

These recommended movies share similarities with Interstellar. Here's a short discussion on how each of these recommended movies are related:

1. **Apollo 13:** Like Interstellar, Apollo 13 is a space exploration movie based on real events. The movie tells the story of the Apollo 13 mission, which suffered a catastrophic malfunction in space. **Guardians of the Galaxy**: Guardians of the Galaxy is a space adventure movie that shares some similarities with Interstellar's themes of exploration, discovery, and human resilience.
2. **The Martian**: The Martian is another space exploration movie that shares some similarities with Interstellar's themes of survival, isolation, and human ingenuity. The movie tells the story of an astronaut who is stranded on Mars and must find a way to survive until he can be rescued.
3. **Space Cowboys:** Space Cowboys is a science-fiction movie that shares some similarities with Interstellar's themes of exploration, adventure, and human resilience. The movie tells the story of a group of retired pilots who are called back into service to repair an old Soviet satellite.
4. **Silent Running**: Silent Running is a science-fiction movie that shares some similarities with Interstellar's themes of space exploration, environmentalism, and human emotions. The movie tells the story of a botanist who is sent on a mission to preserve the last remaining forests of Earth aboard a spacecraft.

The Recommendation System which we have developed uses NLP and creates a vector consisting of the frequency of occurrence of each word in a movie. Since Interstellar had a lot of information about space mentioned in in its description. Our algorithm picked up movies with similar description taking space as its baseline and then building up from there based on the other keywords.

CONCLUSION AND FURTHER SCOPE

6.1 Conclusion

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 mono approach i.e content based filtering instead of collaborative. Both the approaches have its advantages and disadvantages in content based filtering 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
Disadvantages: 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.

Disadvantages: 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.**

The hybrid approach will resolves all these limitations by combining both content and collaborative filtering. In this project, to improve the accuracy, quality and scalability of movie recommendation system, a Hybrid approach by unifying content based filtering and collaborative filtering; using Singular Value Decomposition (SVD) as a classifier and Cosine Similarity is presented in the proposed methodology.
Existing pure approaches and proposed hybrid approaches are implemented on three different Movie datasets and the results are compared among them. Comparative results depicts that the proposed approach shows an improvement in the accuracy, quality and scalability of the movie recommendation system than the pure approaches. Also, the computing time of the proposed approach is lesser than the other two pure approaches.

6.2 Future scope:

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.

LIST OF FIGURES

Figure No.	Figure Name	Page No.
7.1	Use Case Diagram	12
7.2	System Analysis and Design	13
7.3	Cosine Similarity and Angle Approximation	14

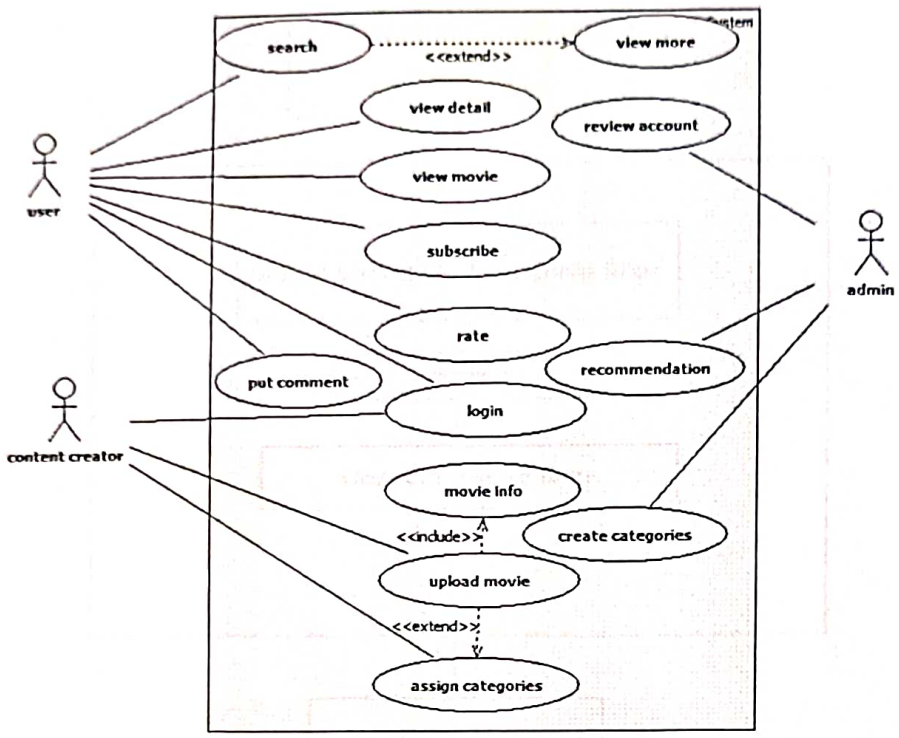
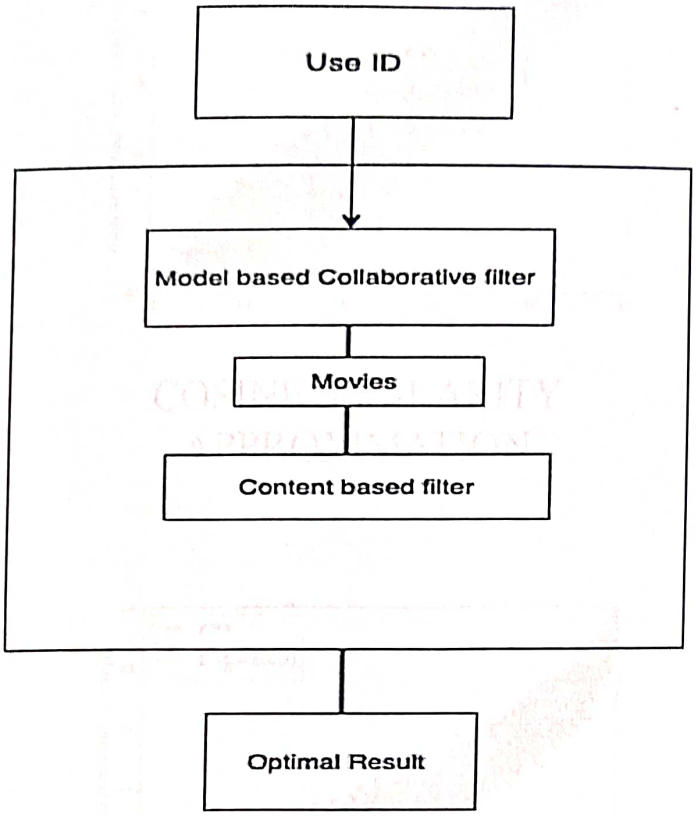
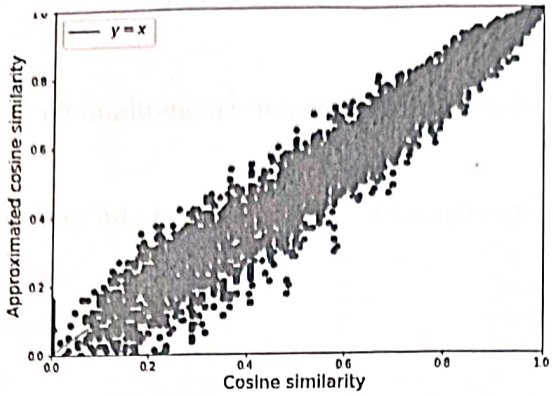


fig : Use case diagram

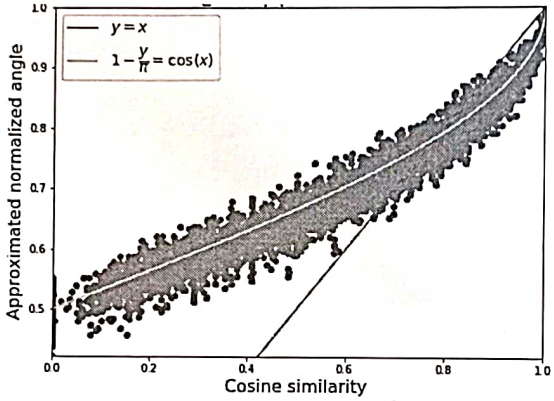
USE CASE DIAGRAM



SYSTEM ANALYSIS AND DESIGN DIAGRAM



COSINE SIMILARITY APPROXIMATION



ANGLE APPROXIMATION

REFERENCES

[1] www.kaggle.com/tmdb/tmdb-movie-metadata?select=tmdb_5000_movies.csv

[2] www.youtube.com

[3] www.geeksforgeeks.org/ml-content-based-recommender-system/

[4] https://docs.streamlit.io/

[5] https://pandas.pydata.org/

[6] https://numpy.org/