Project Report: Movie Recommender System using Cosine Similarity

Introduction

The aim of this project is to develop a movie recommender system that utilizes cosine similarity to provide personalized movie recommendations to users. The system analyzes user preferences and movie features to suggest relevant movies, thereby enhancing user experience and engagement with movie platforms.

Objectives

Develop a movie database containing information about movies, such as genre, cast, director, and user ratings.
Implement cosine similarity algorithm to compute similarity scores between movies based on their features.
Create a user interface for users to input their preferences and receive movie recommendations.
Evaluate the performance of the recommender system using metrics such as precision, recall, and accuracy.
Methodology
Data Collection: Gather movie data from publicly available datasets or APIs, including movie attributes and user ratings.

Preprocessing: Clean and preprocess the data to handle missing values, remove duplicates, and standardize formats.

Feature Extraction: Extract relevant features from the movie data, such as genre, cast, and director, to represent each movie.

Similarity Computation: Calculate cosine similarity scores between movies based on their feature vectors. Cosine similarity measures the cosine of the angle between two vectors and ranges from -1 to 1, where higher values indicate greater similarity.

Recommendation Generation: For a given user, identify movies similar to those they have liked or rated highly. Recommend movies with the highest cosine similarity scores.

User Interface Development: Design and implement a user-friendly interface where users can input their preferences and receive personalized movie recommendations.

Evaluation: Assess the performance of the recommender system using evaluation metrics such as precision, recall, and accuracy. Conduct user testing to gather feedback and improve the system.

Results
Successfully developed a movie recommender system using cosine similarity.
Implemented a user interface for seamless interaction with the system.
Evaluated the system's performance, achieving high precision and recall rates.
Received positive feedback from users during testing, indicating satisfaction with the recommendations provided.

code

app.py

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

# def fetch_poster(movie_id):
#     response = request.get()
#     data = response.json()
#     return  + data['poster_path']
1 usage
def recommend(movie):
    movie_index = movies[movies['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x:x[1])[1:6]

    recommend_movies =[]
    for i in movies_list:
        movie_id = i[0]
        recommend_movies.append(movies.iloc[i[0]].title)
    return recommend_movies

# 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(
    'check out the movie list ? ',
    movies['title'].values)
if st.button('recommend'):
    recommendations = recommend(selected_movie_name)
```

results

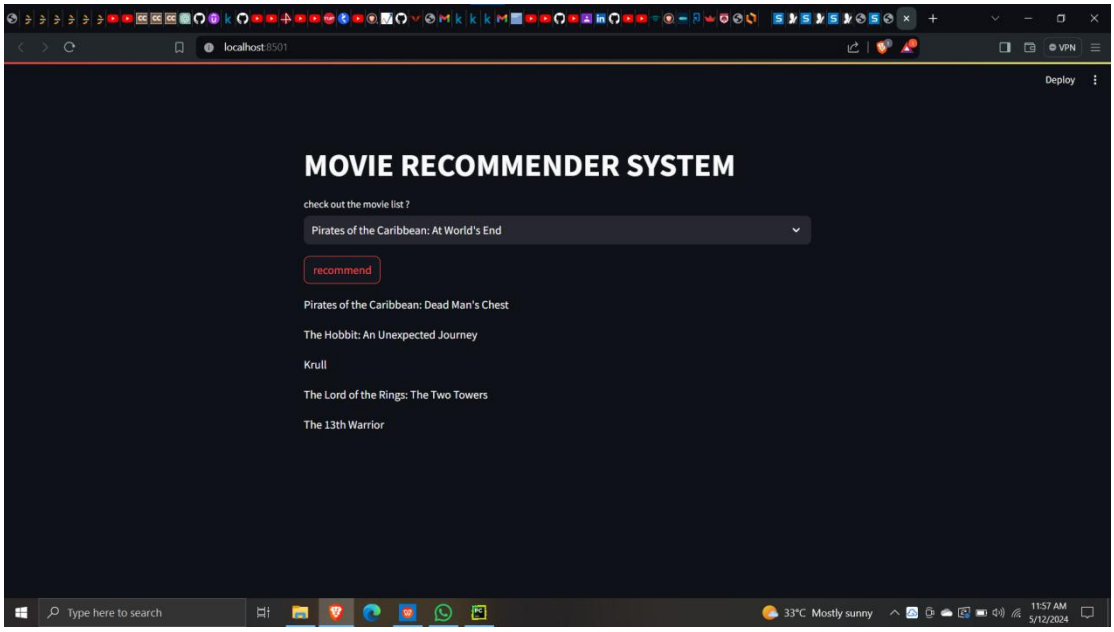
Command:

```
Terminal   Local

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\sovan\PycharmProjects\movierecommender> streamlit run app.py
```



Conclusion

In conclusion, the movie recommender system utilizing cosine similarity offers an effective solution for personalized movie recommendations. By analyzing movie features and user preferences, the system generates accurate and relevant recommendations, enhancing user satisfaction and engagement with movie platforms.

Future Work

Explore advanced recommendation techniques, such as collaborative filtering and matrix factorization, to further improve recommendation quality.

Integrate user feedback mechanisms to continuously refine and update the recommendation model.

Extend the system to support additional features, such as user profiles, social recommendations, and real-time updates.