SOFT COMPUTING - SECTION 10 P2

MOVIE RECOMMENDATION SYSTEM

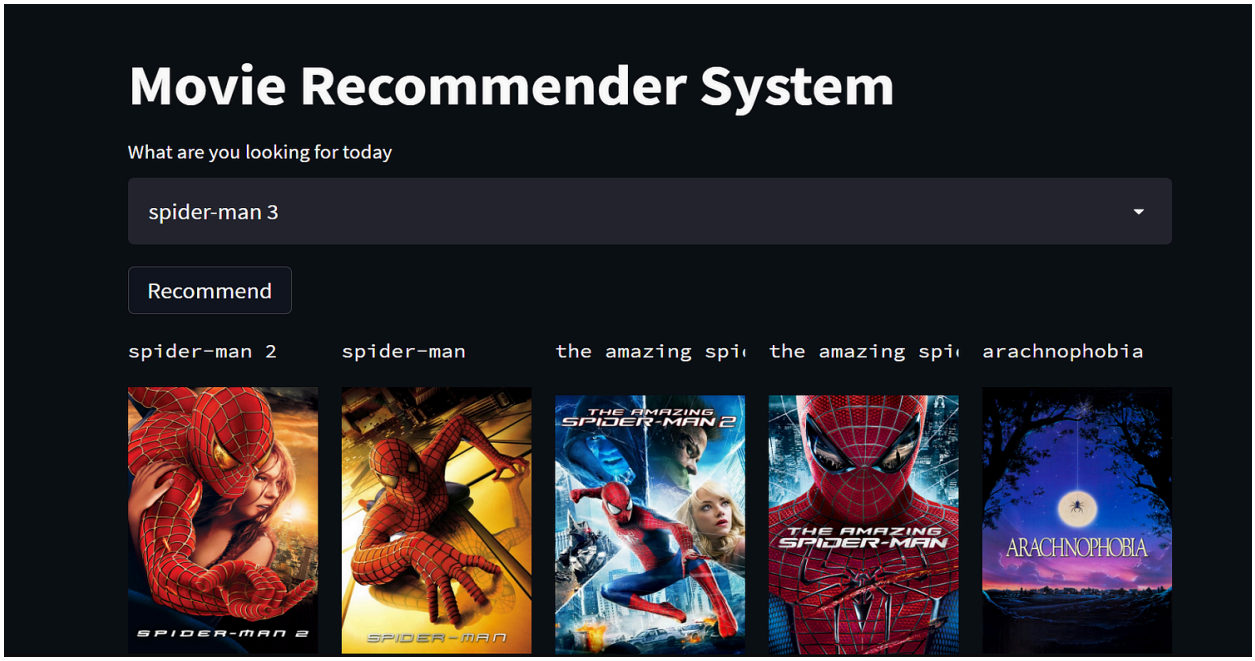
Using machine learning

Github link: https://github.com/Ayushcode10/movie-recommender-system-tmdb-dataset

Ayush Saxena - 22SCSE420090

Deep Sagar Chaudhary - 22SCSE420097

Harshit Harlalka - 22SCSE420092



Introduction

This project aims to develop a content-based movie recommendation system using the TMDB dataset. By leveraging movie metadata such as genres, cast, crew, and keywords, the system generates personalized recommendations for users. Implemented using Python

---

and Streamlit, the application provides an interactive user interface for exploring movie suggestions.

Objectives

- Develop a content-based filtering algorithm to recommend movies.
- Utilize the TMDB dataset for high-quality movie metadata.
- Build a user-friendly web application for real-time recommendations.

The objective of this project is to develop a content-based movie recommendation system that leverages the rich metadata available in the TMDB dataset. By analyzing features such as genres, cast, crew, and keywords, the system aims to provide personalized and accurate movie suggestions for users. Additionally, the project seeks to create an intuitive, user-friendly interface using Streamlit, enabling users to interact with the system effortlessly. The overarching goal is to combine data processing, machine learning techniques, and an engaging frontend to deliver an efficient and enjoyable movie recommendation experience.

Dataset

Dataset Used: TMDB 5000 Movies and Credits dataset.

Source: Kaggle - TMDB Dataset.

Key Features:

- **Movies Data**: Title, genres, overview, popularity, etc.
- **Credits Data**: Cast, crew, and director information.

The project uses the TMDB 5000 Movies and Credits dataset from Kaggle, containing metadata like genres, cast, crew, and movie overviews. These datasets were cleaned,

merged, and processed to extract key features, forming the foundation for building an effective movie recommendation system.

Features

Content-Based Filtering:

- Recommendations based on movie metadata like genres, cast, and keywords.
- Uses cosine similarity to calculate the similarity between movies.

Interactive User Interface:

- Developed with Streamlit for easy deployment and user interaction.
- Users can select a movie and view the top recommended movies.

Search Functionality:

- Allows users to search for specific movies in the database.

The system uses content-based filtering to recommend movies. Key features include movie metadata analysis, a search function, and a user-friendly interface built with Streamlit.

Technologies Used

Programming Language: Python

Libraries:

- Data Processing: Pandas, Numpy
- Machine Learning: Scikit-learn
- Web Framework: Streamlit

- Visualization: Matplotlib, Seaborn (if applicable)

Deployment: Streamlit Community Cloud (or alternatives like Render)

The project utilizes Python for programming, with libraries such as Pandas, Numpy, and Scikit-learn for data processing and machine learning. The app is built using Streamlit, and deployed on platforms like Streamlit Cloud.

System Architecture

Data Preprocessing:

- Cleaned and merged “tmdb_5000_movies.csv” and “tmdb_5000_credits.csv”.
- Extracted features such as genres, keywords, cast, and crew.
- Transformed text data into feature vectors using TF-IDF and CountVectorizer.

Recommendation Algorithm:

- Calculated similarity scores using cosine similarity.
- Sorted similarity scores to recommend top movies.

Frontend:

- Streamlit-based user interface for movie selection and displaying recommendations.

The system processes and cleans movie metadata, calculates similarity scores using cosine similarity, and provides real-time recommendations through a Streamlit-based web application.

Implementation

8.1 Data Cleaning

- Removed missing and duplicate values.
- Merged datasets to create a unified structure.

8.2 Feature Engineering

- Extracted and processed important columns:
    - **Genres**: Transformed into a list of keywords.
    - **Keywords**: Cleaned and tokenized for better recommendations.
    - **Cast**: Limited to top-billed actors.
    - **Crew**: Focused on directors.

8.3 Recommendation Logic

- Combined metadata columns into a single feature vector.
- Applied cosine similarity to find movies with the highest similarity to the selected movie.

8.4 User Interface

- Built an interactive app using Streamlit.
- Included dropdowns, search bars, and display elements for recommendations.

The system cleans and processes the dataset, extracts key features like genres and cast, calculates similarity using cosine similarity, and implements the recommendation logic within a Streamlit app.

9. Results

- The system effectively recommends movies based on content similarity.
- Offers recommendations within milliseconds for the selected movie.

Example:

For the movie "Inception," the system recommends:

- The Matrix
- Interstellar
- Memento
- Shutter Island

The system successfully recommends movies based on content similarity, with fast response times and relevant suggestions for users, such as recommending movies similar to "Inception."

10. Challenges Faced

- Handling missing or inconsistent data in the dataset.

- Balancing computational efficiency with recommendation quality.
- Designing an intuitive user interface.

Challenges included handling missing or inconsistent data, optimizing the recommendation algorithm for efficiency, and designing an intuitive user interface for ease of use.

Future Scope

- **Hybrid Recommendations**: Combine content-based and collaborative filtering for improved results.
- **Advanced NLP**: Use deep learning techniques like word embeddings to understand movie overviews better.
- **Personalization**: Integrate user preferences and ratings for personalized suggestions.
- **Scalability**: Handle larger datasets and deploy the app to serve multiple users simultaneously.

Future improvements could include integrating collaborative filtering, personalizing recommendations, enhancing the system with NLP techniques, and making it scalable for larger datasets.

How to Run This Project on Your PC

Follow these steps to set up and run the Movie Recommender System on your local machine:

1. Prerequisites

Ensure you have the following installed on your system:

- **Python** (Version 3.8 or later)
- **pip** (Python package manager)
- **Git** (to clone the repository)

---

2. Clone the Repository

Clone the project repository from GitHub:

bash

Copy code

```
git clone https://github.com/Ayushcode10/movie-recommender-system-tmdb-dataset.git

cd movie-recommender-system-tmdb-dataset
```

---

3. Install Required Dependencies

Install the necessary Python libraries listed in the `requirements.txt` file:

bash

Copy code

```
pip install -r requirements.txt
```

---

4. Download the Dataset

1. Download the TMDB 5000 Movies and Credits dataset from Kaggle.
2. Place the downloaded CSV files (`tmdb_5000_movies.csv` and `tmdb_5000_credits.csv`) in the `dataset` folder within the project directory.

---

5. Run the Application

Start the Streamlit application using the following command:

bash

Copy code

```
streamlit run app.py
```

---

6. Access the Application

Once the app starts, you will see a URL in the terminal (e.g., `http://localhost:8501`). Open this URL in your web browser to use the Movie Recommender System.

---

7. Troubleshooting

- **File Not Found Errors**: Ensure all datasets are placed in the correct folder (`dataset`).
- **Dependency Issues**: Run `pip install -r requirements.txt` again to resolve missing dependencies.

Port Conflicts: If the app fails to start, check if port `8501` is in use or specify a different port:

bash

Copy code

```
streamlit run app.py --server.port 8502
```

Conclusion

The Movie Recommender System effectively provides content-based recommendations using the TMDB dataset. The interactive application serves as a proof of concept for personalized movie suggestions and can be enhanced with collaborative filtering, advanced NLP, and hybrid recommendation techniques.

The project successfully implements a content-based movie recommendation system. With potential for future improvements, it demonstrates the effectiveness of using metadata for personalized movie suggestions in an interactive and user-friendly format.