Project Report: Movie Recommendation System

Project by
BSCS22128 Hoor Khalid

Problem Definition

Introduction

In today's digital era, the abundance of movies makes it challenging for users to choose what to watch next. A personalized movie recommendation system can assist users by suggesting films based on their preferences and past behavior, enhancing their viewing experience.

Objective

The objective of this project is to develop a movie recommendation system using an SQL database and Python as the backend, with a front end built using Bootstrap and HTML. The system will provide users with personalized movie recommendations based on their viewing history and ratings.

Methodology

Data Collection

Data for the movie recommendation system is collected from publicly available movie databases such as TMDb. This data includes movie details, genres, user ratings, actors, directors.

Data Preprocessing

Data preprocessing involves cleaning the collected data, handling missing values, and normalizing the data to ensure it is in a suitable format for analysis. This step is crucial for ensuring the accuracy and efficiency of the recommendation algorithm.

Database Design

An SQL database is designed to store the movie data, user profiles, and ratings. The database schema includes tables for users, movies, and user-movie interactions (ratings).

Recommendation Algorithm

The recommendation system uses collaborative filtering, content-based filtering, or a hybrid approach to suggest movies. Collaborative filtering relies on user similarity and item similarity, while content-based filtering uses movie metadata.

Backend Development

The backend is developed using Python, utilizing libraries such as Pandas, NumPy, and Scikit-learn for data processing and machine learning. Flask, a lightweight web framework, is used to handle server-side operations and API requests.

Frontend Development

The frontend is developed using HTML and Bootstrap to create a responsive and user-friendly interface. Users can search for movies, view recommendations, and provide ratings through the web interface.

Features

User Management:

- Registration and login for users.
- Different roles: Admin, regular User.new user, old user
- Profile management for users to update their information.
- Signup
- Login
- Search
- Filter

Movie Management:

- CRUD operations for movies (Create, Read, Update, Delete).
- Ability for admin to add/edit/delete movies.
- Detailed information about each movie including title, description, release year, language, length, and average rating.

Recommendation System:

- Algorithm to recommend movies based on user preferences, ratings, and viewing history. Personalized recommendations for each user.
- Integration with external APIs for enhanced recommendation features.

Rating and Reviews:

- Users can rate movies on a numerical scale and provide written reviews.
- Average rating calculation for each movie.
- Option to view and filter reviews.
- Community Chat

Genre Management:

- CRUD operations for genres.
- Association of genres with movies for categorization.
- Filter movies by genre.

Actor Management:

- CRUD operations for actors.
- Association of actors with movies for cast information.
- View movies by actors.

Recommendation

- Recommendation based on user preferences

Toolkit

Languages and Frameworks

1. **Python:** Used for backend development and implementing recommendation algorithms.
2. **SQL:** Used for designing and querying the database.
3. **Flask:** Python web framework for handling server-side operations.
4. **HTML and Bootstrap:** For building a responsive and interactive user interface.

Libraries and Tools

Pandas and NumPy: For data manipulation and analysis.
MySQL: For database management.
Jinja2: Templating engine for rendering HTML in Flask.

Implementation

Database Schema

The database schema consists of the following tables:

Users:Stores user information.

- userID
- username
- age
- email
- -hashed_password

Movies: Stores movie details.

- movieID
- title
- description
- release_date
- language
- length
- avg_rating

Ratings: Stores user ratings.

- ratingID
- date
- numeric_rating
- movieID

- userID

Comments: Stores user comments on movies.
- commentID
- text
- userID
- movieID

Producers: Stores producer information.
- producerID
- name

Directors: Stores director information.
- directorID
- name

Recommendations: Stores user recommendations.
- recommendationID
- userID
- movieID

Backend Implementation
1. Data Preprocessing: Clean and prepare data using Pandas.
2. Recommendation Algorithm: Implement collaborative filtering using Scikit-learn.
3. API Development: Create API endpoints using Flask for data retrieval and recommendation.

Frontend Implementation User Interface: Design using HTML and Bootstrap.
Integration: Connect frontend with backend APIs to display recommendations.

Conclusion

The movie recommendation system provides personalized suggestions to users, enhancing their viewing experience. The project successfully integrates an SQL database with a Python backend and a Bootstrap-based frontend, demonstrating the effective use of modern web development and machine learning techniques.

Future Work

Future enhancements could include incorporating more advanced recommendation algorithms, real-time data updates, and expanding the system to include TV shows and other multimedia content.