Web-Based Movie Recommendation System

By

Adithya.C

RRN:201421601003

Arul Kumaran P

RRN:201421601006

BCA Data Science

Movie Recommendation System

**Introduction:**

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 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 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.

**Problem Statement:**

Problem statement for this project is to create a web application that recommends a movie of various film industry around the world by the movie genre or movie name or with simple one line and accurate rating from various platform with their OTT platforms details.

**Overview**

Existing System:

* Collaborative Filtering: Recommendations are based on the similarity of user’s preferences.
* Content based Filtering: Recommendation are based on attributes of the movie, such as genre, cast, director, etc.
* Deep Learning Based Approaches: Use deep neural network to model the users and items to generate recommendations.

**Limitations of existing system:**

* Cold start Problem: New users or items might not have sufficient data for the recommendation system to work effectively.
* Scalability: Most users only watch a limited number of movies, making it difficult to find similarities between users.
* Personalization: Recommendation system can struggle to capture the unique tastes and preferences of individual users.

**SYSTEM REQUIREMENTS SPECIFICATION:**

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

**Hardware Requirements:**

* A PC/Laptop with Window
* Processor with 1.7-2.4gHz speed
* Minimum of 8gb RAM
* 2gb Graphic card

**Software Requirements:**

* Python 3.11
* MySql Workbench
* Visual Studio Code User version
* Any web browser

**Modules:**

* Module I : Creating database.
* Module II : Developing Web Application.
* Module III : Creating self-updating algorithm with API.
* Module IV : Creating Recommendation Algorithm.

**Module I** : Creating Database:

* + - The first module involves in creating the database for the web application.
    - Mysql query language is used to create and manipulate the database.
    - The database is named as moviesDB , and the table is named as movie\_table.
    - The table contains columns like title,year,diretorname, actornames, keyword, genres etc

**Module II: Developing Web Application**:

* This module involves in building a web application page , this module include both frontend and backend.
* The front end of the application is created using HTML language and CSS.
* Two front web pages are created, they are

Home page

Result page

* The post method is used to push the searched keyword to backend and the searched result is displayed.
* The Home page contains a search bar, and a button to get recommendation and a about page that describe about the web application.
* The backend is create using python language and the packages used are flask and mysql and the libraries are Flask, mysql-connector, render\_template, requests.
* The flask package is used to create the backend as server that contains mysql query, it accept the post method from front end and performs query in backend.
* The requests is used to handle the request between frontend and backend.
* The render template is used to push the result to landing page.
* The jsonify is used to send the data structured to the front end.

**Module III:** Creating a self updating algorithim with API

* This module involves in creating algorithim that scraps a web page and fetch content, structure it and store it in database.
* This algorithm is fed with the IMDb web page URL and a time interval is given to the algorithm to perform the webscraping.
* For webscraping python language is used with package Beautiful soup for webscraping and pandas to handle csv data.
* And tweepy is used to fetch the comments and likes of the movie using the twitter developer account.

**Module IV**: Creating the recommendation system:

* The recommendation system is build using python language.
* The libraries used are
  + - * numpy
      * pandas
      * sklearn

The content based filtering method is used to produce recommendation, because it can handle large scale of data and users.

**Code:**

from flask import Flask,render\_template, request

import mysql.connector

app = Flask(\_\_name\_\_)

# Define the route for the home page

@app.route('/')

def home():

    return render\_template('index.html')

# Define the route for the search page

@app.route('/search', methods=['POST'])

def search():

    # Get the search keyword from the form data

    keyword = request.form['search']

    # Connect to the MySQL database

    connection = mysql.connector.connect(host='localhost',

                                         database='movies',

                                         user='root',

                                         password='12345')

    # Define a SQL query to search for the keyword in the database

    query = "SELECT \* FROM movie WHERE actorname LIKE '%{}%'".format(keyword)

    # Execute the query

    cursor = connection.cursor()

    cursor.execute(query)

    # Get the search results

    result = cursor.fetchall()

    # Close the database connection

    cursor.close()

    connection.close()

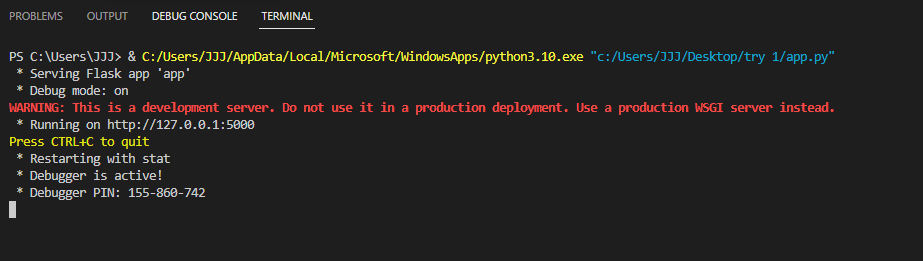
    # Render the search results template with the search results

    return render\_template('search.html', results=result)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**output:**

****

**Architectural Diagram:**

\ Home

Search (genres, actor name, director name, year, keyword)

Recommend Movie

Query in database with the searched keyword

Login with to fetch watch history

Content based filtering with watch history data.

Optimal result

**Explanation of the Activity Diagram:**

* The User searches for movies with keywords.
* The System returns a list of movies matching the keywords.
* To get recommendation the user has to login with id and the watch history is fetched and movies are recommended by content based filtering.

**Requirements Specification (SRS) document for a movie recommendation system:**

1. Introduction

The purpose of this document is to outline the requirements for a movie recommendation system. The system is designed to recommend movies to users based on their preferences and viewing history.

2. Scope

The movie recommendation system will be a web-based application that allows users to rate movies, and receive personalized movie recommendations. The system will incorporate machine learning algorithms to analyze user data and recommend movies based on their preferences.

**3. Functional Requirements**

3.1 Movie Ratings

The system shall allow users to view movies rating on a scale of 1 to 5 stars.

3.2 Movie Recommendations

The system shall provide personalized movie recommendations based on user ratings and viewing history.

3.3 Recommendation Settings

The system shall allow users to adjust their recommendation settings, the types of movies recommended.

3.4 Search

The system shall allow users to search for movies by title, genre, or actor.

**4. Non-functional Requirements**

4.1 Performance

The system shall be able to handle a large number of users and movie ratings without significant delay.

4.2 Usability

The system shall be intuitive and easy to use for users of all levels of technical expertise.

4.3 Security

The system shall protect user data by using encryption and secure login procedures.

4.4 Reliability

The system shall be available for use 24/7 with minimal downtime.

Recommendation Engine:

The recommendation engine component of the system generates personalized movie recommendations for users. The engine uses collaborative filtering, content-based filtering,techniques to generate recommendations. The following steps are involved in the recommendation engine:

1. Collaborative Filtering: Collaborative filtering is a technique that recommends movies based on the similarity between users. The engine identifies users who have similar viewing habits and recommends movies that those users have rated highly.
2. Content-Based Filtering: Content-based filtering is a technique that recommends movies based on their attributes like genre, cast, and director. The engine identifies movies that are similar to the user's previously watched movies and recommends those movies.

User Interface:

The user interface component of the system allows users to rate movies, view recommended movies, and provide feedback. The interface is designed to be intuitive and user-friendly. Users can rate movies by selecting a rating from 1 to 5 stars. The interface also displays recommended movies based on the user's viewing history and preferences.

Reference:

<https://www.tutorialspoint.com/flask/index.html>

<https://www.w3schools.com/python/pandas/default.asp>

[MySQL Tutorial (w3schools.com)](https://www.w3schools.com/MySQL/default.asp)

<https://www.w3schools.com/html/>

A. Gomez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms Business Value and Innovation", ACM Trans. Manag. Inf. Syst., vol. 6, no. 4, pp. 1-19, Jan. 2016.

B. Samih, A. Ghadi and A. Fennan, "ExMrec2vec: Explainable Movie Recommender System based on Word2vec", Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 8, 2021.

Conclusion:

The movie recommendation system is a project that aims to provide personalized movie recommendations to users. The system uses collaborative filtering, content