Movie Recommendation System

Introduction

In today's digital age, the volume of available movies has exploded, making it increasingly difficult for viewers to select films that match their interests. To address this issue, we can employ data science techniques to create a recommendation system that suggests movies based on user preferences and viewing history. This project leverages the TMDB 5000 Movie dataset to develop a content-based recommendation system.

Content-based recommendation systems recommend items by comparing the content of items and a profile of the user's preferences. In this project, we will use metadata such as genres, keywords, cast, and crew to determine the similarity between movies. By calculating the similarity, we can suggest movies that are closely related to a movie a user likes.

Objectives

- **Data Loading and Exploration**: Understand the structure and content of the movie and credits datasets.
- **Data Cleaning and Preprocessing**: Prepare the data for analysis by handling missing values and transforming columns.
- **Feature Engineering**: Extract and combine features from the datasets to create meaningful tags for each movie.
- **Vectorization**: Convert textual data into numerical vectors using the CountVectorizer.
- **Similarity Calculation**: Calculate the cosine similarity between movie vectors to find similar movies.
- **Recommendation Function**: Implement a function to recommend movies based on a given movie title.
- **Model Saving**: Save the processed data and similarity matrix for future use.

This system can be a valuable tool for streaming services, allowing them to enhance user experience by providing personalized movie recommendations. By understanding and implementing this system, we can gain insights into the practical applications of data science and machine learning in the entertainment industry.

Dataset

The dataset includes two CSV files:

1. `tmdb_5000_movies.csv`: Contains information about movies, including budget, genres, popularity, and more.
2. `tmdb_5000_credits.csv`: Contains information about the cast and crew of the movies.

Steps to Build the Recommendation System

1. Import Libraries

We start by importing the necessary libraries for data manipulation and analysis, including pandas for data handling, NumPy for numerical operations, and libraries from Scikit-learn for vectorization and similarity calculation.

2. Load and Explore Data

We load the `tmdb_5000_movies.csv` and `tmdb_5000_credits.csv` datasets and explore them to understand their structure and content. This involves checking the first few rows of each dataset and examining their shape to get an overview of the data.

3. Merge Datasets

To combine the movie information and the credits information, we merge the two datasets on the `title` column. This allows us to have a unified dataset that contains all relevant information about each movie in one place.

4. Select Relevant Columns

From the merged dataset, we select only the columns that are necessary for the recommendation system. These include `movie_id`, `title`, `overview`, `genres`, `keywords`, `cast`, and `crew`.

5. Data Preprocessing

Data preprocessing involves cleaning and transforming the data to make it suitable for analysis. This includes:

- Handling missing values by dropping rows with null values.
- Converting JSON-like strings in columns such as `genres`, `keywords`, `cast`, and `crew` into lists of strings using functions that parse these strings.
- Extracting only the top 3 cast members and the director from the crew.
- Removing spaces from names to create single words.

6. Create Tags

We combine all the processed information into a single column called `tags`. This column contains a mixture of genres, keywords, cast, crew, and the movie overview, all in one place. This consolidated column will be used to create a feature vector for each movie.

7. Vectorize Tags

Using the `CountVectorizer` from scikit-learn, we convert the text data in the `tags` column into numerical vectors. This process, known as vectorization, transforms the text into a matrix of token counts, which can be used to compute similarities between movies.

8. Calculate Similarity

We calculate the cosine similarity between the vectors of movies. Cosine similarity is a measure that calculates the cosine of the angle between two vectors, which in this context represents the similarity between two movies based on their tags.

9. Recommendation Function

We implement a recommendation function that takes a movie title as input and recommends the top 5 movies that are most similar to the given movie. This function uses the similarity scores calculated in the previous step to find and sort the most similar movies.

10. Save Model and Data

Finally, we save the processed data and the similarity matrix using the pickle module. This allows us to load the data and model quickly in the future without having to preprocess and calculate similarities again.

Example Usage

To use the recommendation system, simply call the recommendation function with a movie title. For example, calling the function with the title 'Gandhi' will print the top 5 recommended movies similar to 'Gandhi'.

This concludes the development of the movie recommendation system. By following these steps, we have created a robust content-based recommendation engine that can help users discover movies that align with their preferences.

11. Web Application with Streamlit

Deploy the recommendation system using Streamlit to create an interactive and user-friendly web application. The application allows users to select a movie from a dropdown list and get recommendations by clicking a button.

Streamlit Code Explanation

1. **Import Libraries**: Import necessary libraries including Streamlit, pickle, and others.
2. **Load Model and Data**: Load the preprocessed data and similarity matrix using pickle.
3. **Recommendation Function**: Define the `recommend` function to get similar movie recommendations based on the selected movie.
4. **Streamlit UI**:
    - Create a header for the web application.
    - Load the movie titles into a dropdown list using `st.selectbox`.
    - Display recommendations when the "Show Recommendation" button is clicked.