Brian_Reppeto_DSC630_Week_10

November 2, 2024

0.0.1 DSC 630 Week :

Activity 10.2

Author: Brian Reppeto 11/2/2024

0.0.2 Import libraries

[61]:
```
# import libraries

import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
import re
```

0.0.3 Load and clean data

[62]:
```
# load datasets

movies_df = pd.read_csv('ml-latest-small/movies.csv')
ratings_df = pd.read_csv('ml-latest-small/ratings.csv')
```

0.0.4 Create user-movie matrix

[63]:
```
# Create a user-movie matrix with users as rows, movies as columns, and ratings␣
 ↪as values

user_movie_matrix = ratings_df.pivot(index='userId', columns='movieId',␣
 ↪values='rating')
user_movie_matrix = user_movie_matrix.fillna(0)  # Fill missing values with 0
```

0.0.5 Calculate cosine similarity between movies

[70]:
```
# Calculate cosine similarity on the complete user-movie matrix.. this creates␣
 ↪a # bewteen -1 and 1 with 1 being the most similar

movie_similarity = cosine_similarity(user_movie_matrix.T)
movie_similarity_df = pd.DataFrame(movie_similarity, index=user_movie_matrix.
 ↪columns, columns=user_movie_matrix.columns)
```

0.0.6 Clean the titles to make searching more flexible (remove year)

[73]:
```
# Function to extract movie title without year using the regular expression␣
 ↪library.  When testing I did not always know the year
# so I removed as I would assume most would not

def clean_movie_title(title):
    # Remove the year in parentheses, if present
    return re.sub(r'\s*\(\d{4}\)$', '', title).strip().lower()
```

0.0.7 Function to get top 10 similar movies for a given movie title

[72]:
```
# Function cleans the movie title, finds matching movies for the top 10 movies

def recommend_movies(movie_title, movies_df, movie_similarity_df, top_n=10):
    # Clean the input movie title to ignore the year
    cleaned_title = clean_movie_title(movie_title)

    # Find movieId for movies matching the cleaned title
    movie_ids = movies_df[movies_df['title'].str.lower().
 ↪apply(clean_movie_title) == cleaned_title]['movieId'].values
    if len(movie_ids) == 0:
        return f"Movie '{movie_title}' not found in the dataset."

    # Use the first matching movieId for recommendations
    movie_id = movie_ids[0]

    # Get similarity scores for the given movie
    similar_movies = movie_similarity_df[movie_id].
 ↪sort_values(ascending=False)[1:top_n+1]

    # Get movie titles for the recommended movies
    recommended_titles = movies_df[movies_df['movieId'].isin(similar_movies.
 ↪index)]['title'].values
    return recommended_titles
```

0.0.8 Recommendation without using the year in the title

[69]:
```
# Enter the name of the movie in quotes below (note that this dataset is small␣
 ↪and all movies might not be in the dataset)

recommend_movies("superman", movies_df, movie_similarity_df)
```

[69]:
```
array(['E.T. the Extra-Terrestrial (1982)', 'Aliens (1986)',
       'Terminator, The (1984)', 'Highlander (1986)',
       'Star Trek II: The Wrath of Khan (1982)',
       'Indiana Jones and the Temple of Doom (1984)',
```

```
'Planet of the Apes (1968)', 'Superman II (1980)',
'RoboCop (1987)', 'Predator (1987)'], dtype=object)
```

0.1 Recommender System Overview

The recommender system suggests ten movies similar to a user specified movie. The recommender provides recommendations even if the input movie has only a few ratings.

Steps:

Data Loading: Load movie and rating data, focusing on movies (with movieId and title) and user ratings.

User-Movie Matrix: A user-movie matrix is created, where each row represents a user, each column a movie, and each cell the user's rating for that movie. Missing ratings are set to zero.

Movie Similarity Calculation: Cosine similarity is computed across movies in the matrix, generating a similarity matrix that quantifies how closely movies are rated alike by users.

Title Cleaning: A function strips years from titles to allow users to input movie names without needing the release year (Toy Story instead of Toy Story (1995).

Recommendation Function: The main function finds similar movies based on the similarity matrix, sorts them, and returns the top ten.

[ ]: