DSC 630 - Week 10 Exercise - Emilio Flores

Movie Recommender Program

Objective

The objective of this project is to create a movie recommender program. A movie dataset from GroupLens was utilized. This dataset includes the name, year of release, and genre(s) of thousands of movies.

Steps Followed

- **Importing Libraries**
- **Importing the Dataset**
- **Cleaning and Formatting the Data**
- **Creating a Recommender Program**
- **Testing the Program**

Importing Libraries and Dataset

Two libraries were used: `pandas` and `cosine_similarity` from `sklearn.metrics.pairwise`. After importing the libraries, the movie dataset (a CSV file) was uploaded and converted into a DataFrame.

Cleaning and Formatting the Data

The dataset included movie names, release years, and genre(s) in a single column. The following transformations were applied:

- The **movie name** and **year of release** were split into separate columns.
- The **genre column**, which contained multiple genres in one cell, was expanded into multiple columns using one-hot encoding (`get_dummies()`).

Creating a Recommender Program

Cosine similarity was used to compute similarity scores between movies based on genre.

> Cosine similarity is a mathematical metric that measures the similarity between two vectors in a multi-dimensional space by calculating the cosine of the angle between them (Wikipedia, 2025).

- If two movies have **identical genres**, their cosine similarity score is **1.0**.
- If two movies are **completely different**, their cosine similarity score is **0**.

A **genre similarity matrix** was created using the `cosine_similarity()` method from `sklearn.metrics.pairwise`.

A function was developed to:

1. Accept a movie name as input.
2. Look up the movie in the dataset.
3. Find the **top 10 most similar movies** using cosine similarity.
4. Display an **error message** if the movie is not found.

The similarity scores were extracted and sorted. The top 10 most similar movies (excluding the user-input movie itself) were returned.

Testing the Program

The program was tested and the outcomes were acceptable:

- Displayed an **error message** when a movie was not found.
- Provided a **list of similar movies** when a valid movie was entered.

References

Wikipedia - Cosine Similarity

Import Libraries and Data

In [4]:
```
# Import libraries
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
```

In [5]:
```
# Import data set
movies_df = pd.read_csv('movies.csv')
```

Prepare Data Set

In [7]:
```
# Remove Year from title column
movies_df['Title'] = movies_df['title'].str[:-6]

# Remove white spaces
movies_df['Title'] = movies_df['Title'].str.strip()
```

In [8]:
```
# Obtain year from title column and create new column
movies_df['Year'] = movies_df['title'].str[-6:]

# Drop parentheses
movies_df['Year'] = movies_df['Year'].str.replace(r'[()]','', regex=True)
```

In [9]:
```
# Drop original 'title' column
movies_df.drop('title', axis=1, inplace=True)
```

In [10]:
```
# Ensure 'genres' is formatted as a string
movies_df['genres'] = movies_df['genres'].astype(str)

# Obtain dummy variables from 'genres' column
movies_df = movies_df.join(movies_df['genres'].str.get_dummies(sep='|'))
```

In [11]:
```
# Drop 'genres'column
movies_df.drop('genres', axis=1, inplace=True)
```

Create Recommender Program

In [13]:
```
# Extract genre columns
genre_columns = movies_df.columns[4:]

# Create matrix
genre_matrix = movies_df[genre_columns]
```

In [14]:
```
# Compute cosine similarity
similarity_matrix = cosine_similarity(genre_matrix)
```

In [15]:
```
# Create recommender function
def recommend_movies(movie_title, movies_df, similarity_matrix):
    # Find the index of the movie
    movie_index = movies_df[movies_df['Title'].str.lower() == movie_title.lower()].index

    # Add message if movie is not found
    if len(movie_index) == 0:
        return "Movie not found. Please check the title and try again."

    # Extract movie index (row number)
    movie_index = movie_index[0]

    # Get similarity scores
    similarity_scores = list(enumerate(similarity_matrix[movie_index]))

    # Sort movies by similarity score, exclude title that was typed by user
    similar_movies = sorted(similarity_scores, key=lambda x: x[1], reverse=True)[1:11]

    # Retrieve movie titles
    recommended_titles = [movies_df.iloc[i[0]]['Title'] for i in similar_movies]

    return recommended_titles
```

Test Recommender Program

In [17]:
```
# Ask for user input
print("""
Are you looking for something to watch? Type a movie you
like and we'll suggest 10 similar movies
""")
user_input = input("Name of favorite movie: ")
```

```python
# Run recommendation function
movies = recommend_movies(user_input, movies_df, similarity_matrix)

# Handle wrong input and print top 10 movies if successful
if isinstance(movies, str):
    print(movies)
else:
    print(f"Top {len(movies)} similar movies to {user_input} are:")
    for id, movie in enumerate(movies, start=1):
        print(f"{id}. {movie}")
```

```
Are you looking for something to watch? Type a movie you
like and we'll suggest 10 similar movies

Top 10 similar movies to Avatar are:
1. Spider-Man 2
2. Superman Returns
3. Star Trek
4. Transformers: Revenge of the Fallen
5. Avatar
6. Tron: Legacy
7. Avengers, The
8. John Carter
9. Amazing Spider-Man, The
10. Oblivion
```