Movie Recommender Systems

```
This project implements a movie recommendation system using two
primary approaches: content-based filtering and collaborative
filtering.

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from scipy.sparse.linalg import svds
```

pandas: For data manipulation and working with DataFrames.

numpy: For numerical operations.

TfidfVectorizer: From scikit-learn, used to convert movie overviews into numerical feature vectors.

cosine_similarity: From scikit-learn, used to calculate the cosine similarity between movie feature vectors.

svds: From scipy.sparse.linalg, used for Singular Value Decomposition (SVD), a matrix factorization technique used in collaborative filtering.

```
# Load datasets
movies = pd.read_csv("movies_metadata.csv", low_memory=False)
ratings = pd.read_csv("ratings_small.csv")
```

Loads the movie metadata and ratings datasets into pandas DataFrames. The low_memory=False argument is used to avoid data type inference issues when reading the movies_metadata.csv file.

```
# Data Preprocessing
movies = movies[['id', 'title', 'overview']].dropna()
movies['id'] = pd.to_numeric(movies['id'], errors='coerce')
movies = movies.dropna().reset_index(drop=True)
```

Selects the id, title, and overview columns from the movies DataFrame and removes rows with any missing values (dropna()).

Converts the id column to a numeric type, handling any conversion errors by setting invalid values to NaN (errors='coerce').

Removes any rows with NaN values resulting from the id conversion and resets the index of the DataFrame.

```
# Content-Based Filtering
tfidf = TfidfVectorizer(stop_words="english")
```

```
movies["overview"] = movies["overview"].fillna("")
tfidf_matrix = tfidf.fit_transform(movies["overview"])
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

TfidfVectorizer: Creates a TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer to convert the movie overviews into numerical representations. stop_words='english' removes common English words that don't carry much meaning.

Fills any remaining missing values in the overview column with empty strings ("").

tfidf.fit_transform: Fits the TF-IDF vectorizer to the movie overviews and transforms them into a TF-IDF matrix.

cosine_similarity: Calculates the cosine similarity between each pair of movies based on their TF-IDF vectors. This results in a matrix where each element (i, j) represents the cosine similarity between movie i and movie j.

```
# Function to get recommendations based on content similarity
def recommend_content(title, num_recommendations=5):
    idx = movies.index[movies["title"] == title].tolist()
    if not idx:
        return "Movie not found!"
    idx = idx[0]

    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
[1:num_recommendations+1]

    movie_indices = [i[0] for i in sim_scores]
    return movies["title"].iloc[movie_indices].tolist()
```

Defines a function recommend_content that takes a movie title and the number of recommendations as input.

Finds the index of the movie with the given title in the movies DataFrame. If the movie is not found, it returns "Movie not found!".

Gets the cosine similarity scores for the movie with all other movies using the cosine_sim matrix.

Sorts the similarity scores in descending order and selects the top num_recommendations movies.

Returns a list of the titles of the recommended movies.

```
# Collaborative Filtering using Matrix Factorization
ratings_matrix = ratings.pivot(index="userId", columns="movieId",
values="rating").fillna(0)
```

Creates a user-item rating matrix from the ratings DataFrame using the pivot function.

The rows represent users (userId), the columns represent movies (movieId), and the values represent the ratings.

Missing values (i.e., movies not rated by a user) are filled with 0.

```
# Convert the matrix to a NumPy array
ratings_array = ratings_matrix.to_numpy()
```

Converts the ratings matrix from a pandas DataFrame to a NumPy array, which is required for the SVD calculation.

```
# Perform Singular Value Decomposition
U, sigma, Vt = svds(ratings_array, k=50)
sigma = np.diag(sigma)
```

Performs Singular Value Decomposition (SVD) on the ratings matrix using the svds function from scipy.sparse.linalg.

k=50 specifies the number of singular values to keep. This reduces the dimensionality of the matrix and helps to capture the most important patterns in the data.

Converts the singular values (sigma) into a diagonal matrix.

```
# Predict ratings
predicted_ratings = np.dot(np.dot(U, sigma), Vt)
predicted_ratings_df = pd.DataFrame(predicted_ratings,
index=ratings_matrix.index, columns=ratings_matrix.columns)
```

Reconstructs the predicted ratings matrix by multiplying the U, sigma, and Vt matrices.

Converts the predicted ratings matrix back into a pandas DataFrame with the original user and movie IDs as indices and column names.

```
# Function to get movie recommendations for a user
def recommend_collaborative(user_id, num_recommendations=5):
    if user_id not in predicted_ratings_df.index:
        return "User not found!"
    sorted_movies =
predicted_ratings_df.loc[user_id].sort_values(ascending=False)
    return sorted_movies.head(num_recommendations).index.tolist()
```

Defines a function recommend_collaborative that takes a user ID and the number of recommendations as input.

Checks if the user ID exists in the predicted ratings DataFrame. If not, it returns "User not found!".

Sorts the predicted ratings for the user in descending order and selects the top num_recommendations movies.

Returns a list of the movie IDs of the recommended movies.

```
# Example usage:
print(recommend_content("The Godfather"))
print(recommend_collaborative(1))
```

```
['The Godfather: Part II', 'The Godfather Trilogy: 1972-1990', 'The Godfather: Part III', 'Blood Ties', 'Household Saints']
[1374, 2968, 2105, 1954, 2193]
```

Demonstrates how to use the recommend_content and recommend_collaborative functions.

Prints the top 5 content-based recommendations for the movie "The Godfather".

Prints the top 5 collaborative filtering recommendations for user ID 1.

This project implements a movie recommendation system using both content-based and collaborative filtering techniques. Content-based filtering relies on movie overviews to find similar movies, while collaborative filtering uses user ratings to predict movies a user might like. The notebook provides functions to generate recommendations based on these methods and demonstrates their usage with example inputs. The SVD is implemented using matrix factorization.