Assignment No-2

Title-Produce and evaluate content based movie recommendation system.

Objectives-

● To understand the principles of content-based filtering in recommendation systems. ● To implement a content-based movie recommendation system using movie features (e.g., genres, descriptions).
● To evaluate the effectiveness of the recommendation system using appropriate metrics.

Outcomes-Students will be able

● Understand and apply the principles of content-based filtering in recommendation systems.
● Demonstrate proficiency in data preprocessing and feature extraction techniques. ● Implement a functional content-based movie recommendation system in Python. ● Evaluate the recommendation system using relevant metrics and analyze its performance.
● Document the entire process, including methodology, results, and conclusions, in a clear and professional manner.

Theory

A content-based movie recommendation system suggests movies to users based on the characteristics of the movies they have liked in the past. This approach analyzes the attributes of movies, such as genre, description, and other relevant features, to find similar movies that match the user's preferences.

Key Components

1. Dataset:
    ○ A dataset containing movie details is essential. Common attributes include: ■ Title: The name of the movie.
        ■ Genres: The categories or genres the movie falls under (e.g., Action, Drama).
        ■ Description: A brief summary of the movie's plot.
2. Preprocessing:
    ○ Combine relevant features to create a content profile for each movie. This often involves concatenating the genres and descriptions into a single string, which represents the movie's content.
3. Feature Extraction:
    ○ Use techniques like TF-IDF (Term Frequency-Inverse Document Frequency) to convert text data into numerical format. TF-IDF helps in understanding the importance of words in the context of the entire dataset.
    ○ It assigns weights to words based on their frequency in a particular document compared to their frequency across all documents.
4. Similarity Calculation:
    ○ Cosine Similarity is commonly used to measure the similarity between movies. It calculates the cosine of the angle between two vectors (the TF-IDF representations of two movies). A smaller angle indicates more similarity. 5. Recommendation Generation:
    ○ For a given movie, the system finds other movies with high similarity scores. The top N movies with the highest similarity scores are recommended to the user.

Implementation Steps

1. Load the Dataset

You can use a structured dataset, such as one from MovieLens or IMDB, or create a sample dataset.

2. Preprocess the Data

Combine relevant attributes (like genres and descriptions) into a single string for each movie. 3. Create the

TF-IDF Matrix

Use a TF-IDF vectorizer to transform the text data into a matrix form, where each row represents a movie, and each column represents a word from the dataset.

4. Compute Cosine Similarity

Calculate the cosine similarity matrix from the TF-IDF matrix to find how similar each movie is to the others.

5. Build the Recommendation Function

Create a function that:

- Takes a movie title as input.
- Finds the index of that movie in the dataset.
- Retrieves the similarity scores for that movie.
- Sorts the scores and selects the top N movies to recommend.

```python
# Import required libraries
import pandas as pd
import ast
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

# Step 1: Load dataset
movies = pd.read_csv('/content/tmdb_5000_movies.csv')   # adjust path
if needed
movies = movies[['title', 'genres', 'overview']]
movies.dropna(inplace=True)

# Step 2: Preprocess data
movies['genres'] = movies['genres'].apply(lambda x: '
'.join([i['name'] for i in ast.literal_eval(x)]))
movies['content'] = movies['genres'] + ' ' + movies['overview']

# Step 3: TF-IDF vectorization
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(movies['content'])

# Step 4: Compute cosine similarity
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

# Step 5: Recommendation function
def get_recommendations(title, cosine_sim=cosine_sim):
    if title not in movies['title'].values:
        return "Movie not found in dataset."
    idx = movies.index[movies['title'] == title][0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
[1:6]
    movie_indices = [i[0] for i in sim_scores]
    return movies['title'].iloc[movie_indices]

# Step 6: Test the recommendation system
movie_name = "The Godfather"
print(f"Recommendations for '{movie_name}':")
print(get_recommendations(movie_name))
```

```
Recommendations for 'The Godfather':
1614             Take the Lead
3814             Desert Dancer
2395    Step Up 2: The Streets
1483        Step Up Revolution
2820                   Step Up
Name: title, dtype: object
```