MOVIE RECOMMENDATION SYSTEM

A **Movie Recommendation System** in Machine Learning suggests movies based on user preferences using techniques such as **content-based filtering**, **collaborative filtering**, or **hybrid models.**

This code is a **Movie Recommendation System** that suggests movies similar to a user-inputted movie using **TF-IDF Vectorization** and **Cosine Similarity**. Below is a detailed breakdown of each section:

---

1. Importing Required Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

- **NumPy & Pandas**: Used for data manipulation.
- **Matplotlib & Seaborn**: (Not used in the code but typically used for data visualization).
- **difflib**: Helps find the closest matching movie name.
- **TfidfVectorizer**: Converts text data (movie descriptions) into numerical vectors.
- **cosine_similarity**: Measures similarity between movies based on text features.

---

2. Loading the Dataset

```
data=pd.read_csv('/content/movies.csv')
```

- Reads the movies.csv file into a Pandas DataFrame named data.

---

3. Selecting Relevant Features

```
selected_features=['genres','keywords','tagline','cast','director']
print(selected_features)
```

- The model considers **genres, keywords, tagline, cast, and director** as important features for movie recommendations.

---

4. Handling Missing Data

```
for i in selected_features:
  data[i]=data[i].fillna('')
```

- If any selected feature has **missing values (NaN)**, they are replaced with an empty string ('') to avoid errors.

---

5. Combining Features into a Single Text Column

```
combined_features = data['genres']+' '+data['keywords']+' '+data['tagline']+' '+data['cast']+' '+data['director']
print(combined_features)
```

- All selected features are combined into a **single string per movie**.
- This string serves as the **"movie description"** used to compute similarity.

Example of combined_features for a movie:

'Action Adventure Superhero Fight Iron Man Robert Downey Jr. Jon Favreau'

---

6. Converting Text Data into Feature Vectors (TF-IDF)

```
vectorizer=TfidfVectorizer()
feature_vectors=vectorizer.fit_transform(combined_features)
print(feature_vectors)
```

- TfidfVectorizer():
  - Converts combined_features into **numerical vectors**.
  - Assigns weights to words using **TF-IDF (Term Frequency-Inverse Document Frequency)**.
- fit_transform(combined_features):
  - Learns the vocabulary and transforms text into a **sparse matrix**.
- feature_vectors is a **numerical representation of movies**, making them comparable.

---

7. Calculating Cosine Similarity

```
similarity = cosine_similarity(feature_vectors)
print(similarity)
```

- **cosine_similarity(feature_vectors)** computes similarity scores between all movies.
- The result is a **square matrix**, where:
  - **Rows = Movies**
  - **Columns = Movies**
  - Each entry [i, j] contains the **cosine similarity score** between movie i and movie j.

---

8. Taking User Input for Movie Recommendation

```
movie_name = input('Enter your favourite movie name: ')
```

- The user is prompted to enter their favorite movie.

---

9. Fetching a Close Match for the Input

```
list_of_movie_names = data['title'].tolist()
print(list_of_movie_names)
```

- Converts the column **title** into a list of movie names.

```
find_close_match = difflib.get_close_matches(movie_name, list_of_movie_names)
print(find_close_match)
```

- **difflib.get_close_matches()** finds the closest matching movie name to the user input.

```
close_match = find_close_match[0]
print(close_match)
```

- Picks the **top closest match**.

---

10. Finding the Index of the Matched Movie

```
index_of_movie = data[data.title == close_match]['index'].values[0]
print(index_of_movie)
```

- Retrieves the **index** of the movie that best matches the user's input.
- This index will be used to fetch similar movies.

---

11. Fetching Similar Movies

```
similarity_score = list(enumerate(similarity[index_of_movie]))
print(similarity_score)
```

- similarity[index_of_movie] extracts the similarity scores of the selected movie with all other movies.
- Converts the scores into a **list of tuples**:
  - **(index, similarity score)**

```
sorted_similarity_scores = sorted(similarity_score, key=lambda x: x[1], reverse=True)
print(sorted_similarity_scores)
```

- Sorts the similarity scores in **descending order** (most similar movies first).

---

12. Displaying Recommended Movies

```
print("Movies suggested for you: \n")

i = 1
for movie in sorted_similarity_scores:
  index = movie[0]
  title_from_index = data[data.index == index]['title'].values[0]
  if i < 25:
    print(i, '.', title_from_index)
    i += 1
```

- Loops through the **sorted similarity scores**.
- Extracts the **title of similar movies**.
- Displays the **top 25 recommended movies**.

- **Example Run**

Input:

Enter your favourite movie name: Iron Man

Output:

Movies suggested for you:

1. Iron Man
2. Iron Man 2
3. Iron Man 3
4. The Avengers
5. Captain America: Civil War
6. Avengers: Age of Ultron
7. Thor
8. Spider-Man: Homecoming

...

- **Iron Man 2 & 3** are suggested first because they share the most similar features.

- **Summary of Key Concepts**

1. **TF-IDF Vectorization**: Converts text data into numerical vectors.
2. **Cosine Similarity**: Measures similarity between movie descriptions.
3. **difflib.get_close_matches()**: Finds the closest matching movie from user input.
4. **Sorting Similarity Scores**: Finds the top similar movies.
5. **Recommendation**: Suggests movies similar to the input.