Movie Recommenation System

Objective

To develop a basic recommendation system by suggesting items that are most similar to a particular item, in this case, movies. It just tells what movies/items are most similar to the user's movie choice

Import Library

```
import pandas as pd
```

```
import numpy as np
```

Import Dataset

```
df = pd.read_csv(r'https://github.com/YBI-Foundation/Dataset/raw/refs/heads/main/Movies%20Recommendation.csv')
```

```
df.head()
```

	Movie_ID	Movie_Title	Movie_Genre	Movie_Language	Movie_Budget	Movie_Popularity	Movie_Release_Date	Movie_Revenue	Movie_Runt
0	1	Four Rooms	Crime Comedy	en	4000000	22.876230	09-12-1995	4300000	9
1	2	Star Wars	Adventure Action Science Fiction	en	11000000	126.393695	25-05-1977	775398007	12
2	3	Finding Nemo	Animation Family	en	94000000	85.688789	30-05-2003	940335536	10
3	4	Forrest Gump	Comedy Drama Romance	en	55000000	138.133331	06-07-1994	677945399	14
4	5	American Beauty	Drama	en	15000000	80.878605	15-09-1999	356296601	12

5 rows × 21 columns

```
df.info()
```

Show hidden output

```
df.shape
```

```
(4760, 21)
```

```
df.columns
```

Show hidden output

```
df_feature = df[['Movie_Genre','Movie_Keywords','Movie_Tagline','Movie_Cast','Movie_Director']].fillna('')
```

```
df_feature
```

Show hidden output

```
X = df_feature['Movie_Genre'] + '' + df_feature['Movie_Keywords'] + '' + df_feature['Movie_Tagline'] + '' + df_feature['Movie_Cast'] +
```

```
X
```

	0
0	Crime Comedyhotel new year's eve witch bet hot...
1	Adventure Action Science Fictionandroid galaxy...
2	Animation Familyfather son relationship harbor...
3	Comedy Drama Romancevietnam veteran hippie men...
4	Dramamale nudity female nudity adultery midlif...
...	...
4755	HorrorThe hot spot where Satan's waitin'.Lisa ...
4756	Comedy Family DramaIt's better to stand out th...
4757	Thriller Dramachristian film sex traffickingSh...
4758	Family
4759	Documentarymusic actors legendary perfomer cla...

4760 rows × 1 columns

```
X.shape
```

Show hidden output

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf = TfidfVectorizer()
```

```
X = tfidf.fit_transform(X)
```

```
X.shape
```

Show hidden output

```
print(X)
```

Show hidden output

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
Similarity_Score = cosine_similarity(X)
```

```
Similarity_Score
```

Show hidden output

```
Similarity_Score.shape
```

Show hidden output

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

```
Enter your favourite movie name :  avtaar
```

```
All_Movies_Title_List = df['Movie_Title']. tolist()
```

```
import difflib
```

```
Movie_Recommendation = difflib.get_close_matches(Favourite_Movie_Name, All_Movies_Title_List)
print(Movie_Recommendation)
```

```
['Avatar', 'Gattaca']
```

```
Close_Match = Movie_Recommendation[0]
print(Close_Match)
```

```
Avatar
```

```
Index_of_Close_Match_Movie = df[df.Movie_Title == Close_Match]['Movie_ID'].values[0]
```

Getting a list of similar movies

```
Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Close_Match_Movie]))
print(Recommendation_Score)
```

Show hidden output

```
len(Recommendation_Score)
```

Show hidden output

Get All Movies Sort Based on Recommendation Score wrt Fovourite Movie

```
#sorting the movies based on their similarity score
Sorted_Similar_Movies = sorted(Recommendation_Score, key = lambda x:x[1], reverse = True)
print(Sorted_Similar_Movies)
```

Show hidden output

```
# print the name 0f similar movies based on the index
print('The 30 Movies Suggested for You : \n')

i = 1

for movie in Sorted_Similar_Movies:
  index = movie[0]
  title_from_index = df[df.index==index]['Movie_Title'].values[0]
  if (i<31):
    print(i, '.',title_from_index)
    i+=1
```

```
The 30 Movies Suggested for You :

 1 . Niagara
 2 . Some Like It Hot
 3 . The Kentucky Fried Movie
 4 . The Juror
 5 . Enough
 6 . Duel in the Sun
 7 . Superman III
 8 . Eye for an Eye
 9 . The Misfits
 10 . Beyond the Black Rainbow
 11 . Brokeback Mountain
 12 . All That Jazz
 13 . Tora! Tora! Tora!
 14 . Master and Commander: The Far Side of the World
 15 . To Kill a Mockingbird
 16 . Harry Brown
 17 . The Dark Knight Rises
 18 . Running with Scissors
 19 . Edge of Darkness
 20 . Man on Wire
 21 . The Odd Life of Timothy Green
 22 . Intolerable Cruelty
 23 . The Curse of Downers Grove
 24 . The Great Gatsby
 25 . Mad Max 2: The Road Warrior
 26 . Source Code
 27 . Song One
 28 . The Longest Yard
```

```
29 . The Boy Next Door
30 . The Lazarus Effect
```

Top 10 Movie Recommendation System

```
Movie_Name = input(' Enter your Favourite movies name : ')

list_of_all_titles = df['Movie_Title'].tolist()

Find_Close_Match = difflib.get_close_matches(Movie_Name,list_of_all_titles)

Close_Match = Find_Close_Match[0]

Index_of_Movie = df[df.Movie_Title == Close_Match]['Movie_ID'].values[0]

Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Movie]))

sorted_similar_movies = sorted(Recommendation_Score, key = lambda x:x[1], reverse = True)

print('Top 10 Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = df[df.Movie_ID == index]['Movie_Title'].values
  if(i<11):
    print(i, '.', title_from_index)
    i+=1
```

```
 Enter your Favourite movies name : avtaar
Top 10 Movies suggested for you :

1 . ['Avatar']
2 . ['The Godfather']
3 . ['New Nightmare']
4 . ['Elizabethtown']
5 . ['Gerry']
6 . ['Cradle Will Rock']
7 . ['A Prairie Home Companion']
8 . ['Bright Lights, Big City']
9 . ['Rollerball']
10 . ['Walking and Talking']
```

Explaination

Recommender System is a system that seeks to predict or filter preferences according to the user's choices. Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. Recommender systems produce a list of recommendations in any of the two ways -

Collaborative filtering: Collaborative filtering approaches build a model from the user's past behavior (I.e. Items purchased or searched by the user) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that users may have an interest in.

Content-based filtering: Content-based filtering approaches uses a series of discrete characteristics of an item in order to recommend additional items with similar properties. Content-based filtering methods are totally based on a description of the item and a profile of the user's preferences. It recommends items based on the user's past preferences. Let's develop a basic recommendation system using Python and Pandas.

TfidfVectorizer (function of sklearn library)is used to convert a collection of text documents into a matrix of TF-IDF features (Term Frequency-Inverse Document Frequency), which is often used in Natural Language Processing (NLP) to quantify words' importance across documents.

cosine_similarity (function from the sklearn (scikit-learn) library)function is used to compute the cosine similarity between two sets of vectors. Cosine similarity measures the cosine of the angle between two vectors, which is often used to determine the similarity between documents or items in machine learning, especially in text mining and clustering.