Recommendation System using Textual Features

> The "Movie Recommendation System using Textual Features" project is designed to provide personalized movie recommendations to users based on their favorite movies. The system employs natural language processing techniques to analyze and compare textual features such as genres, keywords, taglines, cast, and directors from a collection of movies. By calculating the similarity between movies using the cosine similarity metric, the system suggests a list of movies that share similar textual characteristics with the user's input.

Dependencies:

In [1]:
```
import numpy as np
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Data Collection and Pre-Processing:

In [2]:
```
movies_data = pd.read_csv("content\movies.csv")
```

In [3]:
```
movies_data.head()
```

Out[3]:

	index	budget	genres	homepage	id	keywords	origi
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india trad...	
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel mi6	
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc comics crime fighter terrorist secret ident...	
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	based on novel mars medallion space travel pri...	

5 rows × 24 columns

In [4]:
```
movies_data.shape
```

Out[4]:
```
(4803, 24)
```

Selecting the Relevant Features only:

In [5]:
```
selected_features = ['genres','keywords','tagline','cast','director']
print(selected_features)
```

```
['genres', 'keywords', 'tagline', 'cast', 'director']
```

*We need to fill the **null** values with **null** string:*

In [6]:
```
for feature in selected_features:
    movies_data[feature] = movies_data[feature].fillna('')
```

Combining all 5 selected features:

In [7]:
```
combined_features = movies_data['genres']+' '+movies_data['keywords']+' '+movie
```

In [8]:
```
print(combined_features)
```

```
0       Action Adventure Fantasy Science Fiction cultu...
1       Adventure Fantasy Action ocean drug abuse exot...
2       Action Adventure Crime spy based on novel secr...
3       Action Crime Drama Thriller dc comics crime fi...
4       Action Adventure Science Fiction based on nove...
                              ...
4798    Action Crime Thriller united states\u2013mexic...
4799    Comedy Romance  A newlywed couple's honeymoon ...
4800    Comedy Drama Romance TV Movie date love at fir...
4801      A New Yorker in Shanghai Daniel Henney Eliza...
4802    Documentary obsession camcorder crush dream gi...
Length: 4803, dtype: object
```

Now we need to convert the text data to feature vectors:

In [9]:
```
vectorizer = TfidfVectorizer()
```

In [10]:
```
feature_vectors = vectorizer.fit_transform(combined_features)
```

In [11]:
```
print(feature_vectors)
```

```
  (0, 2432)     0.17272411194153
  (0, 7755)     0.1128035714854756
  (0, 13024)    0.1942362060108871
  (0, 10229)    0.16058685400095302
  (0, 8756)     0.22709015857011816
  (0, 14608)    0.15150672398763912
  (0, 16668)    0.19843263965100372
  (0, 14064)    0.20596090415084142
  (0, 13319)    0.2177470539412484
  (0, 17290)    0.20197912553916567
  (0, 17007)    0.23643326319898797
  (0, 13349)    0.15021264094167086
  (0, 11503)    0.27211310056983656
  (0, 11192)    0.09049319826481456
  (0, 16998)    0.1282126322850579
  (0, 15261)    0.07095833561276566
  (0, 4945)     0.24025852494110758
  (0, 14271)    0.21392179219912877
  (0, 3225)     0.24960162956997736
  (0, 16587)    0.12549432354918996
  (0, 14378)    0.33962752210959823
  (0, 5836)     0.1646750903586285
  (0, 3065)     0.22208377802661425
  (0, 3678)     0.21392179219912877
  (0, 5437)     0.1036413987316636
  :     :
  (4801, 17266) 0.2886098184932947
  (4801, 4835)  0.24713765026963996
  (4801, 403)   0.17727585190343226
  (4801, 6935)  0.2886098184932947
  (4801, 11663) 0.21557500762727902
  (4801, 1672)  0.1564793427630879
  (4801, 10929) 0.13504166990041588
  (4801, 7474)  0.11307961713172225
  (4801, 3796)  0.3342808988877418
  (4802, 6996)  0.5700048226105303
  (4802, 5367)  0.22969114490410403
  (4802, 3654)  0.262512960498006
  (4802, 2425)  0.24002350969074696
  (4802, 4608)  0.24002350969074696
  (4802, 6417)  0.21753405888348784
  (4802, 4371)  0.1538239182675544
  (4802, 12989) 0.1696476532191718
  (4802, 1316)  0.1960747079005741
  (4802, 4528)  0.19504460807622875
  (4802, 3436)  0.21753405888348784
  (4802, 6155)  0.18056463596934083
  (4802, 4980)  0.16078053641367315
  (4802, 2129)  0.3099656128577656
  (4802, 4518)  0.16784466610624255
  (4802, 11161) 0.17867407682173203
```

Cosine Similarity

We need to find the 'Similarity Score' using the 'Cosine Similarity'

> Cosine similarity is a metric used to measure the similarity between two vectors in a multi-dimensional space. It is commonly used in various fields, including natural language processing, information retrieval, and recommendation systems. Cosine similarity calculates the cosine of the angle between two vectors, representing how closely the vectors are aligned with each other.

In [22]:
```
similarity = cosine_similarity(feature_vectors)
```

In [23]:
```
print(similarity)
```

```
[[1.         0.07219487 0.037733   ... 0.         0.         0.        ]
 [0.07219487 1.         0.03281499 ... 0.03575545 0.         0.        ]
 [0.037733   0.03281499 1.         ... 0.         0.05389661 0.        ]
 ...
 [0.         0.03575545 0.         ... 1.         0.         0.02651502]
 [0.         0.         0.05389661 ... 0.         1.         0.        ]
 [0.         0.         0.         ... 0.02651502 0.         1.        ]]
```

In [24]:
```
print(similarity.shape)
```

```
(4803, 4803)
```

User Input

Now we need to take a movie name as input from the user.

In [25]:
```
movie_name = input('Enter your favourite movie name : ')
```

```
Enter your favourite movie name : batman
```

In [26]:
```
list_of_all_titles = movies_data['title'].tolist()
print(list_of_all_titles)
```

```
h', 'Star Wars: Episode II - Attack of the Clones', 'Monsters, Inc.', 'The
Wolverine', 'Star Wars: Episode I - The Phantom Menace', 'The Croods', 'Ast
erix at the Olympic Games', 'Windtalkers', "The Huntsman: Winter's War", 'T
eenage Mutant Ninja Turtles', 'Gravity', "Dante's Peak", 'Teenage Mutant Ni
nja Turtles: Out of the Shadows', 'Fantastic Four', 'Night at the Museum',
'San Andreas', 'Tomorrow Never Dies', 'The Patriot', "Ocean's Twelve", 'Mr.
& Mrs. Smith', 'Insurgent', 'The Aviator', "Gulliver's Travels", 'The Green
Hornet', '300: Rise of an Empire', 'The Smurfs', 'Home on the Range', 'Alle
giant', 'Real Steel', 'The Smurfs 2', 'Speed 2: Cruise Control', "Ender's G
ame", 'Live Free or Die Hard', 'The Lord of the Rings: The Fellowship of th
e Ring', 'Around the World in 80 Days', 'Ali', 'The Cat in the Hat', 'I, Ro
bot', 'Kingdom of Heaven', 'Stuart Little', 'The Princess and the Frog', 'T
he Martian', 'The Island', 'Town & Country', 'Gone in Sixty Seconds', 'Glad
iator', 'Minority Report', 'Harry Potter and the Chamber of Secrets', 'Casi
no Royale', 'Planet of the Apes', 'Terminator 2: Judgment Day', 'Public Ene
mies', 'American Gangster', 'True Lies', 'The Taking of Pelham 1 2 3', 'Lit
tle Fockers', 'The Other Guys', 'Eraser', 'Django Unchained', 'The Hunchbac
k of Notre Dame', "The Emperor's New Groove", 'The Expendables 2', 'Nationa
l Treasure', 'Eragon', 'Where the Wild Things Are', 'Epic', 'The Tourist',
'End of Days', 'Blood Diamond', 'The Wolf of Wall Street', 'Batman Foreve
```

Now we need to create a list with all the movie names given in the dataset:

In [27]:
```
find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)
```

```
['Batman', 'Batman', 'Catwoman']
```

In [28]:
```
close_match = find_close_match[0]
print(close_match)
```

```
Batman
```

In [29]:
```
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].val
print(index_of_the_movie)
```

```
1359
```

Now we need to get a list of similar movies:

In [30]:
```
similarity_score = list(enumerate(similarity[index_of_the_movie]))
print(similarity_score)
```

```
588616309692061), (60, 0.0), (61, 0.01635015002611376), (62, 0.005416634534
4265655), (63, 0.01833889030024455), (64, 0.028404894035231252), (65, 0.177
5581506611392), (66, 0.01794874160703813), (67, 0.0), (68, 0.00513012306968
4274), (69, 0.01618329334393077), (70, 0.005371331822946504), (71, 0.021555
882140030593), (72, 0.12266571244563478), (73, 0.04415940718548016), (74,
0.005495141965586583), (75, 0.022057119691207824), (76, 0.00551618927330124
3), (77, 0.008716823552559781), (78, 0.012046810653425794), (79, 0.02911392
8626152034), (80, 0.02148467123933299), (81, 0.02794528136793173), (82, 0.0
0576827305044206), (83, 0.019994216200751452), (84, 0.013308886706222196),
(85, 0.017148934252787445), (86, 0.009327381057907095), (87, 0.0), (88, 0.0
14079698000279712), (89, 0.025715410181997155), (90, 0.02910820842351401),
(91, 0.005291140630302694), (92, 0.011341251045199981), (93, 0.022412156654
16583), (94, 0.005318310294658278), (95, 0.02564889421647799), (96, 0.03097
291388389124), (97, 0.004252359556894674), (98, 0.032193726743405454), (99,
0.026147609016141292), (100, 0.04263848763117278), (101, 0.035984048368772
1), (102, 0.015251059855876558), (103, 0.054354666030584614), (104, 0.00558
9170793477882), (105, 0.02119994427869696), (106, 0.011481954496263711), (1
07, 0.01737021995321565), (108, 0.023241981578083354), (109, 0.021776146296
002157), (110, 0.01663593844713563), (111, 0.025868460781820198), (112, 0.0
1422619779516028), (113, 0.039521340648464964), (114, 0.027429761464728431),
```

In [31]:
```
len(similarity_score)
```

Out[31]:
```
4803
```

Let's sort the movies based on the 'Similarity Score':

In [32]:
```
sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse =
print(sorted_similar_movies)
```

```
[(1359, 1.0), (428, 0.4311643836232694), (210, 0.25737999820859625), (3, 0.
20438773732168222), (119, 0.19262528757150407), (65, 0.1775581506611392),
(1512, 0.14705162654306442), (813, 0.14414128303962467), (2530, 0.137373224
73729185), (1017, 0.13713281929528845), (473, 0.1321707514794208), (753,
0.13216136203404205), (278, 0.1299626071515124025), (14, 0.1281502653576938
6), (72, 0.12266571244563478), (2313, 0.1183433298876972), (2381, 0.1175238
0293327759), (5, 0.11533013884014888), (2858, 0.11413652180839844), (4183,
0.11353876531321921), (30, 0.11336219425554919), (2655, 0.1127095391661329
7), (10, 0.11240850198526249), (1035, 0.11197582745207195), (2685, 0.109804
21440315517), (870, 0.10722005407914785), (41, 0.10650154604688718), (1296,
0.10584478077207024), (438, 0.10277392559139559), (1477, 0.1019970955832286
3), (299, 0.10154632458465614), (1474, 0.10078843818576026), (1002, 0.10052
62424012352), (1085, 0.10023066759543668), (2753, 0.09951168410001496), (30
3, 0.09883937789831629), (2805, 0.09653055839236144), (163, 0.0949081555508
9474), (1803, 0.09486132363970294), (584, 0.09484559891817565), (4267, 0.09
468285761112293), (1076, 0.094040199293739), (3630, 0.09220755061184922),
(32, 0.0899394677673743), (9, 0.0897616996163815), (1141, 0.089753411996293
7), (2740, 0.08940625662471981), (1740, 0.08909361212904546), (3326, 0.0886
4740283572532), (3466, 0.08668912224324149), (3350, 0.08655054623169336),
(1241, 0.08609237902838108), (681, 0.08543137246751808), (2365, 0.085343844
```

Now let's print the movies based on the index:

In [33]:
```
print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
      index = movie[0]
      title_from_index = movies_data[movies_data.index==index]['title'].values[
      if (i<30):
            print(i, '.',title_from_index)
            i+=1
```

```
Movies suggested for you :

1 . Batman
2 . Batman Returns
3 . Batman & Robin
4 . The Dark Knight Rises
5 . Batman Begins
6 . The Dark Knight
7 . A History of Violence
8 . Superman
9 . Beetlejuice
10 . Bedazzled
11 . Mars Attacks!
12 . The Sentinel
13 . Planet of the Apes
14 . Man of Steel
15 . Suicide Squad
16 . The Mask
17 . Salton Sea
18 . Spider-Man 3
19 . The Postman Always Rings Twice
20 . Hang 'em High
21 . Spider-Man 2
22 . Dungeons & Dragons: Wrath of the Dragon God
23 . Superman Returns
24 . Jonah Hex
25 . Exorcist II: The Heretic
26 . Superman II
27 . Green Lantern
28 . Superman III
29 . Something's Gotta Give
```

Movie Recommendation System

In [34]:
```
movie_name = input('>> Enter your favourite movie name : ')

list_of_all_titles = movies_data['title'].tolist()

find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)

close_match = find_close_match[0]

index_of_the_movie = movies_data[movies_data.title == close_match]['index'].val

similarity_score = list(enumerate(similarity[index_of_the_movie]))

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse =

print('\nSome movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
      index = movie[0]
      title_from_index = movies_data[movies_data.index==index]['title'].values[
      if (i<30):
            print(i, '.',title_from_index)
            i+=1
```

```
>> Enter your favourite movie name : spiderman

Some movies suggested for you :

1 . Spider-Man
2 . Spider-Man 3
3 . Spider-Man 2
4 . The Notebook
5 . Seabiscuit
6 . Clerks II
7 . The Ice Storm
8 . Oz: The Great and Powerful
9 . Horrible Bosses
10 . The Count of Monte Cristo
11 . In Good Company
12 . Finding Nemo
13 . Clear and Present Danger
14 . Brothers
15 . The Good German
16 . Drag Me to Hell
17 . Bambi
18 . The Queen
19 . Charly
20 . Escape from L.A.
21 . Daybreakers
22 . The Life Aquatic with Steve Zissou
23 . Labor Day
24 . Wimbledon
25 . Cold Mountain
26 . Hearts in Atlantis
27 . Out of the Furnace
28 . Bullets Over Broadway
29 . The Purge: Election Year
```

So here we are provided with movies similar to the one entered by our user. This project harnesses the power of machine learning and natural language processing to offer movie enthusiasts tailored suggestions that go beyond conventional genre-based recommendations. By considering a variety of textual attributes, the system can identify subtle connections between movies, enabling it to provide diverse and intriguing movie choices to users. Through user interaction, the system continuously refines its recommendations, ensuring an engaging and user-centric experience.

Whether users are seeking hidden gems that align with their cinematic preferences or are looking to explore new horizons in the world of movies, this Movie Recommendation System offers a novel approach to movie suggestions, enhancing their viewing experience and broadening their cinematic horizons.

In [ ]: