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 [35]:
```
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 [37]:
```
movies_data = pd.read_csv("content\movies.csv")
```

In [38]:
```
movies_data.head()
```

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 [39]:
```
movies_data.shape
```

Out[39]:
```
(4803, 24)
```

Selecting the Relevant Features only:

In [40]:
```
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 [41]:
```
for feature in selected_features:
    movies_data[feature] = movies_data[feature].fillna('')
```

Combining all 5 selected features:

In [42]:
```
combined_features = movies_data['genres']+' '+movies_data['keywords']+' '+movie
```

In [43]:
```
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 [44]:
```
vectorizer = TfidfVectorizer()
```

In [45]:
```
feature_vectors = vectorizer.fit_transform(combined_features)
```

In [46]:
```
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 [47]:
```
similarity = cosine_similarity(feature_vectors)
```

In [48]:
```
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 [49]:
```
print(similarity.shape)
```

```
(4803, 4803)
```

User Input

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

In [50]:
```
movie_name = input('Enter your favourite movie name : ')
```

```
Enter your favourite movie name : san andreas
```

In [51]:
```
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 [52]:
```
find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)
```

```
['San Andreas', 'Bananas', 'In Dreams']
```

In [53]:
```
close_match = find_close_match[0]
print(close_match)
```

```
San Andreas
```

In [54]:
```
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].val
print(index_of_the_movie)
```

```
244
```

Now we need to get a list of similar movies:

In [55]:
```
similarity_score = list(enumerate(similarity[index_of_the_movie]))
print(similarity_score)
```

```
00515046729455662), (63, 0.0), (64, 0.0), (65, 0.012976440791539487), (66,
0.0), (67, 0.0), (68, 0.004878034676241361), (69, 0.02877933541574536), (7
0, 0.005107390706621841), (71, 0.006838110446624989), (72, 0.00504719269927
0938), (73, 0.0), (74, 0.0052251169601376), (75, 0.004923285578353946), (7
6, 0.010138240814070873), (77, 0.002611427442492165), (78, 0.00294518706379
7183), (79, 0.00470003657406075), (80, 0.012165826774132558), (81, 0.004010
980199540196), (82, 0.01341833930389244), (83, 0.014177117732539435), (84,
0.006390162661781981), (85, 0.00574181076390052 3), (86, 0.02585156158691514
7), (87, 0.031564037106100756), (88, 0.005090420512023019), (89, 0.0), (90,
0.01798340197914145), (91, 0.005031140017674971), (92, 0.0), (93, 0.0096693
26420352582), (94, 0.00505697459572858), (95, 0.0024523423310483148), (96,
0.018528242374858023), (97, 0.006119937021219629), (98, 0.00489206895500359
8), (99, 0.00965214331569112), (100, 0.02941490051143029), (101, 0.00560239
27835184345), (102, 0.03761639120661381), (103, 0.00809038015840444), (104,
0.013001705966146044), (105, 0.0), (106, 0.0), (107, 0.005510304878763324),
(108, 0.010027339626529744), (109, 0.0), (110, 0.03213725919582014), (111,
0.005493716114656824), (112, 0.007784828289985279), (113, 0.0), (114, 0.0),
(115, 0.0050067321450073515), (116, 0.019140254237471756), (117, 0.00782802
7023107333), (118, 0.0), (119, 0.008285199641026721), (120, 0.0), (121, 0.0
1294165285527 3883), (122, 0.010877972928340862), (123, 0.026127420363229 38
```

In [56]:
```
len(similarity_score)
```

Out[56]:
```
4803
```

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

In [57]:
```
sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse =
print(sorted_similar_movies)
```

```
[(244, 1.0), (1986, 0.17737231847297472), (495, 0.16407892928860532), (931,
0.15365906412340638), (2486, 0.1368090051334056), (976, 0.1360747048446901
1), (1628, 0.1225619379481559), (4379, 0.11953647792509807), (215, 0.116315
47043840379), (2610, 0.11445471744652673), (2805, 0.1100679439058498), (199
9, 0.10584426690642933), (3252, 0.10538138175349315), (418, 0.1031456621624
0892), (1814, 0.10300826815832093), (163, 0.0981568323822735), (2137, 0.093
41832211960313), (1271, 0.0930891319907 7606), (1002, 0.09253142583543221),
(3924, 0.0925295465496 0074), (341, 0.0895421019024 9928), (59, 0.08898587176
239896), (808, 0.0874408118921 3575), (3796, 0.0874323910566 8184), (4403, 0.
0854376777323956), (1302, 0.0840772957158 0014), (761, 0.0825206940594 0558),
(835, 0.0821925724882 6811), (3382, 0.0816327931918 9834), (1798, 0.081506321
26422283), (243, 0.0811829287115 4388), (166, 0.0806365636945 3827), (368, 0.
07987406689442605), (393, 0.0783855594762 6892), (1792, 0.076388355920782 5
7), (914, 0.0762748141610 7168), (1363, 0.0762255453113 3362), (437, 0.076168
46911933614), (3828, 0.0751648391586 5378), (715, 0.0745033655907 8971), (251
4, 0.0734989607981 0303), (548, 0.0732299590163 1023), (2363, 0.0729835534561
8611), (1155, 0.0729456251752 1733), (1036, 0.0720463650144 5937), (4399, 0.0
7017296026552318), (1469, 0.0693038522290 9247), (44, 0.069027283910 9573),
(2442, 0.0686974410747 1247), (4213, 0.0679160062976 4216), (402, 0.067423011
```

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

In [58]:
```
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 . San Andreas
2 . Faster
3 . Journey 2: The Mysterious Island
4 . Race to Witch Mountain
5 . Meteor
6 . Escape from L.A.
7 . Sanctum
8 . I Origins
9 . Fantastic 4: Rise of the Silver Surfer
10 . A Mighty Heart
11 . The Land Before Time
12 . The Adventurer: The Curse of the Midas Box
13 . New Nightmare
14 . Cats & Dogs 2 : The Revenge of Kitty Galore
15 . W.
16 . Watchmen
17 . Texas Chainsaw 3D
18 . Pandorum
19 . The One
20 . East Is East
21 . Percy Jackson & the Olympians: The Lightning Thief
22 . 2012
23 . Walking Tall
24 . The Last Time I Committed Suicide
25 . The Jimmy Show
26 . Spy Kids 2: The Island of Lost Dreams
27 . Righteous Kill
28 . Mr. Popper's Penguins
29 . The Mighty Macs
```

Movie Recommendation System

In [59]:
```
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 : interstellar

Some movies suggested for you :

1 . Interstellar
2 . The Dark Knight Rises
3 . The Matrix
4 . The Martian
5 . Dear Frankie
6 . Argo
7 . The Matrix Revolutions
8 . The Matrix Reloaded
9 . The Terminator
10 . Armageddon
11 . Terminator Genisys
12 . Contact
13 . Terminator Salvation
14 . The Killer Inside Me
15 . Gandhi, My Father
16 . The Tree of Life
17 . Get Smart
18 . Back to the Future
19 . Terminator 3: Rise of the Machines
20 . The Prestige
21 . Batman Begins
22 . Dragonslayer
23 . WarGames
24 . Little Nicky
25 . Superman III
26 . The Other Side of Heaven
27 . House at the End of the Street
28 . Good Deeds
29 . Mortal Kombat: Annihilation
```

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 [ ]: