Movie Recommendation System Using Cosine Similarity

Name - Gautam Kumar Mahar
Branch - CSE

Overview -

In this project, we have built a movie recommendation system using cosine similarity. The dataset used for this project is movies.csv which contains various features related to movies such as title, genres, keywords, tagline, cast, and director. We have preprocessed the data by replacing the null values with a null string and combining the selected features to create a feature vector using TfidfVectorizer. Cosine similarity is then used to calculate the similarity score between the movies.

Description About This Project -

The first step of the project is to import the required libraries including numpy, pandas, difflib, TfidfVectorizer, and cosine_similarity. We have then loaded the movie dataset (movies.csv) using pandas and selected relevant features such as genres, keywords, tagline, cast, and director.

Next, we have replaced the null values with a null string and combined the selected features to create a feature vector using TfidfVectorizer. We have then used cosine similarity to calculate the similarity score between the movies.

The user can enter the name of their favorite movie, and the system will suggest similar movies based on the cosine similarity score. The system uses difflib library to find the closest match for the user's input if there is a typo or incorrect spelling.

Overall, this project demonstrates the implementation of cosine similarity to build a movie recommendation system and can be further improved by incorporating additional features or algorithms.

Codes

jupyter Movie_Recommendation_System_new Last Checkpoint: 6 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3 (ipykernel)

Run Markdown

NAME - GAUTAM KUMAR MAHAR

BRANCH - CSE

PROJECT - MOVIE RECOMMENDATION SYSTEM

Importing Useful Libraries

In [1]:
```
import numpy as np
import pandas as pd
import difflib # this is used for most closest name of the movie (this is useful when user doing some mistake in mo
from sklearn.feature_extraction.text import TfidfVectorizer # this is used for text value to numerical values
from sklearn.metrics.pairwise import cosine_similarity # this is used for similarity find in between movies
```

Data Collection and Pre-Processing <- A MAJOR STEP -> in this i use Movie.csv Dataset

In [2]:
```
# loading the data from the csv file to apandas dataframe (moviea_data)
movies_data = pd.read_csv('movies.csv') # this data function used for uploading data form movies.csv
```

In [4]:
```
# number of rows and columns in the data frame

movies_data.shape
```

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

In [5]:
```
# selecting the relevant features for recommendation

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

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

In [6]:
```
# replacing the null valuess with null string

for feature in selected_features:
  movies_data[feature] = movies_data[feature].fillna('') # Fill Null string
```

In [7]:
```
# combining all the 5 selected features

combined_features = movies_data['genres']+' '+movies_data['keywords']+' '+movies_data['tagline']+' '+movies_data['c
```

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
```

In [9]:
```
# Now, converting the text data to feature vectors

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
```

Cosine Similarity

In [12]:
```
# getting the similarity scores using cosine similarity

similarity = cosine_similarity(feature_vectors)
```

In [ ]:
```
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 [13]:
```
print(similarity.shape)
```

```
(4803, 4803)
```

Getting the movie name from the user

In [14]:
```
# getting the movie name from the user

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

```
 Enter your favourite movie name : ironman
```

In [15]:
```python
# creating a list with all the movie names given in the dataset

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

```
['Avatar', "Pirates of the Caribbean: At World's End", 'Spectre', 'The Dark Knight Rises', 'John Carter', 'Spide
r-Man 3', 'Tangled', 'Avengers: Age of Ultron', 'Harry Potter and the Half-Blood Prince', 'Batman v Superman: Da
wn of Justice', 'Superman Returns', 'Quantum of Solace', "Pirates of the Caribbean: Dead Man's Chest", 'The Lone
Ranger', 'Man of Steel', 'The Chronicles of Narnia: Prince Caspian', 'The Avengers', 'Pirates of the Caribbean:
On Stranger Tides', 'Men in Black 3', 'The Hobbit: The Battle of the Five Armies', 'The Amazing Spider-Man', 'Ro
bin Hood', 'The Hobbit: The Desolation of Smaug', 'The Golden Compass', 'King Kong', 'Titanic', 'Captain Americ
a: Civil War', 'Battleship', 'Jurassic World', 'Skyfall', 'Spider-Man 2', 'Iron Man 3', 'Alice in Wonderland',
'X-Men: The Last Stand', 'Monsters University', 'Transformers: Revenge of the Fallen', 'Transformers: Age of Ext
inction', 'Oz: The Great and Powerful', 'The Amazing Spider-Man 2', 'TRON: Legacy', 'Cars 2', 'Green Lantern',
'Toy Story 3', 'Terminator Salvation', 'Furious 7', 'World War Z', 'X-Men: Days of Future Past', 'Star Trek Into
Darkness', 'Jack the Giant Slayer', 'The Great Gatsby', 'Prince of Persia: The Sands of Time', 'Pacific Rim', 'T
ransformers: Dark of the Moon', 'Indiana Jones and the Kingdom of the Crystal Skull', 'The Good Dinosaur', 'Brav
e', 'Star Trek Beyond', 'WALL·E', 'Rush Hour 3', '2012', 'A Christmas Carol', 'Jupiter Ascending', 'The Legend o
f Tarzan', 'The Chronicles of Narnia: The Lion, the Witch and the Wardrobe', 'X-Men: Apocalypse', 'The Dark Knig
ht', 'Up', 'Monsters vs Aliens', 'Iron Man', 'Hugo', 'Wild Wild West', 'The Mummy: Tomb of the Dragon Emperor',
'Suicide Squad', 'Evan Almighty', 'Edge of Tomorrow', 'Waterworld', 'G.I. Joe: The Rise of Cobra', 'Inside Out',
'The Jungle Book', 'Iron Man 2', 'Snow White and the Huntsman', 'Maleficent', 'Dawn of the Planet of the Apes',
'The Lovers', '47 Ronin', 'Captain America: The Winter Soldier', 'Shrek Forever After', 'Tomorrowland', 'Big Her
o 6', 'Wreck-It Ralph', 'The Polar Express', 'Independence Day: Resurgence', 'How to Train Your Dragon', 'Termin
```

In [16]:
```python
# finding the close match for the movie name given by the user

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

```
['Birdman', 'Iron Man', 'Hitman']
```

In [17]:
```python
close_match = find_close_match[0]
print(close_match)
```

```
Birdman
```

In [18]:
```python
# finding the index of the movie with title

index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index_of_the_movie)
```

```
2334
```

In [19]:
```python
# getting a list of similar movies

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

```
[(0, 0.01197786514546125), (1, 0.024889093650457314), (2, 0.0), (3, 0.019164395473404258), (4, 0.0), (5, 0.01240
8130527450528), (6, 0.014861866300457638), (7, 0.047413408084477145), (8, 0.046301615467016784), (9, 0.058648880
62558872), (10, 0.041448299406387175), (11, 0.0), (12, 0.0), (13, 0.03128749803316471), (14, 0.04627621096366
6), (15, 0.014018945927412131), (16, 0.06492678226543958), (17, 0.026550484817288954), (18, 0.043834350807916
9), (19, 0.014342121709958952), (20, 0.0683474453402311), (21, 0.0), (22, 0.010817964290352918), (23, 0.00788713
2316192853), (24, 0.08983920215721228), (25, 0.0028934236258832064), (26, 0.033638253743915804), (27, 0.00415446
0686929251), (28, 0.004561107567588781), (29, 0.0), (30, 0.014951016491191267), (31, 0.008963797473188375), (32,
0.0), (33, 0.02946972449654809), (34, 0.0), (35, 0.010244934534l6439), (36, 0.00923150357963087), (37, 0.0393441
2131788671), (38, 0.057820027727481516), (39, 0.004294985065153953), (40, 0.02342181679930078), (41, 0.031105237
355685298), (42, 0.047432224783708433), (43, 0.009271533413127789), (44, 0.0), (45, 0.0026069465825476 4), (46,
0.018120179691678118), (47, 0.0), (48, 0.0), (49, 0.012428049357252953), (50, 0.004408985373446817), (51, 0.0),
(52, 0.012908839035279786), (53, 0.04174524126448959), (54, 0.0), (55, 0.033233160287077214), (56, 0.0), (57, 0.
009417955258864113), (58, 0.009588305982935943), (59, 0.011312453801547682), (60, 0.002996851498203749), (61, 0.
0), (62, 0.0), (63, 0.004214577735468984), (64, 0.047353552429201515), (65, 0.012850989834505014), (66, 0.008007
137230149444), (67, 0.0), (68, 0.03027982461704874), (69, 0.019022248939075205), (70, 0.018253623358020457), (7
1, 0.019302048572208155), (72, 0.0), (73, 0.025111829540028826), (74, 0.0), (75, 0.022529744619866073), (76, 0.
0), (77, 0.01108730943012549), (78, 0.003168220857496144), (79, 0.04153679683416592), (80, 0.0080488329662425 0
8), (81, 0.003516514023747092 5), (82, 0.003030102712230224 7), (83, 0.0), (84, 0.002332418509901328 4), (85, 0.035
641612750722466), (86, 0.003335728961587607), (87, 0.0), (88, 0.015545434319853703), (89, 0.013621867739699428),
```

In [24]:
```python
len(similarity_score)
```

Out[24]:
```
4803
```

In [21]:
```python
# sorting the movies based on their similarity score

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

```
[(2334, 1.0000000000000004), (2187, 0.2998155025057544), (176, 0.24213774422961998), (4089, 0.231429390872954 3
6), (2133, 0.21907204840916583), (1462, 0.1652910915978594), (2356, 0.15262553142921276), (2276, 0.1318721942846
826), (2086, 0.130114134021975 3), (4014, 0.12962845272693396), (2629, 0.12560295877308605), (615, 0.123764961937
19226), (4580, 0.12320799538467012), (174, 0.11836562703720376), (996, 0.11276027222690123), (1685, 0.1099355543
367162), (3276, 0.1097962334785412), (3792, 0.10271371292563988), (2635, 0.10108185232931062), (4016, 0.09849578
308468476), (3865, 0.0971674294002309), (234, 0.09641161678174084), (283, 0.09323044652409204), (1409, 0.0928573
0262046883), (427, 0.09187282794601656), (712, 0.09178380354974053), (2461, 0.09128404230225326), (3819, 0.08983
927878281779), (24, 0.08983920215721228), (1055, 0.08916948766953589), (641, 0.08889335722786529), (3057, 0.0878
7930780811568), (2633, 0.08741824797337483), (321, 0.08725404504832059), (256, 0.08722249751580598), (3384, 0.08
569314117185 24), (623, 0.08550290753455585), (991, 0.08490296917607716), (871, 0.08450775943479905), (924, 0.084
06944784025717), (975, 0.0839264787739876), (3083, 0.08376265011128958), (2955, 0.08359108655560336), (1146, 0.0
8339273833701039), (2769, 0.08239938344770231), (1356, 0.08207693630292505), (3951, 0.08155182308925735), (2737,
0.0812063054656299), (3430, 0.08077856931192888), (114, 0.08038608719323742), (1781, 0.079094287927068), (187,
0.07868981159278363), (497, 0.07833523057025875), (953, 0.07809814306887973), (966, 0.07792789559693104), (935,
0.07578072243581403), (1230, 0.07506736779116562), (2217, 0.07454914600572282), (2204, 0.07447137111819087), (29
37, 0.07390702302080983), (1428, 0.07376765537668943), (2627, 0.07356909056709704), (3482, 0.07297091685436623),
(4352, 0.07222638222505728), (3837, 0.0720949940519056), (1255, 0.07178715144769418), (3899, 0.0713678971904623
7), (3934, 0.0712230639864948), (3975, 0.07113186731235706), (867, 0.07111465304379673), (428, 0.07070453649309 9
02), (2600, 0.07038766497443451), (3033, 0.07026472037562781), (2759, 0.07023469525931923), (3649, 0.07013537188
```

In [22]:
```
# print the name of similar movies based on the index

print('So I suggest Some Movie 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[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

```
So I suggest Some Movie for you :

1 . Birdman
2 . 21 Grams
3 . The Revenant
4 . Amores perros
5 . Babel
6 . Biutiful
7 . Youth in Revolt
8 . The Painted Veil
9 . Home Alone 2: Lost in New York
10 . Kevin Hart: Let Me Explain
11 . You Will Meet a Tall Dark Stranger
12 . Money Train
13 . Quinceañera
14 . The Incredible Hulk
15 . Death to Smoochy
16 . Keeping the Faith
17 . It's Kind of a Funny Story
18 . Adore
19 . Funny Games
20 . Garden State
21 . Whiplash
22 . The Croods
23 . The Taking of Pelham 1 2 3
24 . J. Edgar
25 . The Hangover Part II
26 . Any Given Sunday
27 . Moonrise Kingdom
28 . Defendor
29 . King Kong
```

Movie Recommendation Sytem

```
In [23]: movie_name = input('First Please 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'].values[0]

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

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

print('So, Now I Suggest Some Movies 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[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

```
First Please Enter your favourite movie name : ironman2
So, Now I Suggest Some Movies For You :

1 . Iron Man 2
2 . Iron Man 3
3 . Avengers: Age of Ultron
4 . Iron Man
5 . The Avengers
6 . Captain America: Civil War
7 . Ant-Man
8 . X-Men: Apocalypse
9 . X-Men
10 . Captain America: The Winter Soldier
11 . Deadpool
12 . X2
13 . X-Men: Days of Future Past
14 . Thor: The Dark World
```

```
First Please Enter your favourite movie name : ironman2
So, Now I Suggest Some Movies For You :

1 . Iron Man 2
2 . Iron Man 3
3 . Avengers: Age of Ultron
4 . Iron Man
5 . The Avengers
6 . Captain America: Civil War
7 . Ant-Man
8 . X-Men: Apocalypse
9 . X-Men
10 . Captain America: The Winter Soldier
11 . Deadpool
12 . X2
13 . X-Men: Days of Future Past
14 . Thor: The Dark World
15 . The Incredible Hulk
16 . X-Men: The Last Stand
17 . Man of Steel
18 . The Amazing Spider-Man 2
19 . The Image Revolution
20 . Superman II
21 . X-Men: First Class
22 . Batman v Superman: Dawn of Justice
23 . Sin City
24 . The Jungle Book
25 . The Spirit
26 . Made
27 . X-Men Origins: Wolverine
28 . Spawn
29 . Red Sonja
```

 # NAME - GAUTAM KUMAR MAHAR
 # BRANCH - CSE
 # PROJECT - MOVIE RECOMMENDATION SYSTEM
Importing Useful Libraries

import numpy as np
import pandas as pd
import difflib # this is used for most closest name of the movie (this is useful when user doing some mistake in movie name)
from sklearn.feature_extraction.text import TfidfVectorizer # this is used for text value to numerical values
from sklearn.metrics.pairwise import cosine_similarity # this is used for similarity find in between movies

```
## Data Collection and Pre-Processing <- A MAJOR STEP -> in this i use Movie.csv
Dataset

# loading the data from the csv file to apandas dataframe (moviea_data)
movies_data = pd.read_csv('movies.csv') # this data function used for uploading data
form movies.csv

# printing the first 5 rows of the all dataframe(<- movies.csv)
movies_data.head()

# number of rows and columns in the data frame

movies_data.shape

# selecting the relevant features for recommendation

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

# replacing the null valuess with null string

for feature in selected_features:
  movies_data[feature] = movies_data[feature].fillna('') # Fill Null string

# combining all the 5 selected features

combined_features = movies_data['genres']+' '+movies_data['keywords']+'
'+movies_data['tagline']+' '+movies_data['cast']+' '+movies_data['director']

print(combined_features)

# Now, converting the text data to feature vectors

vectorizer = TfidfVectorizer()

feature_vectors = vectorizer.fit_transform(combined_features)

print(feature_vectors)

## Cosine Similarity

# getting the similarity scores using cosine similarity
```

```
similarity = cosine_similarity(feature_vectors)

print(similarity)

print(similarity.shape)

### Getting the movie name from the user

# getting the movie name from the user

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

# creating a list with all the movie names given in the dataset

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

# finding the close match for the movie name given by the user

find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)

close_match = find_close_match[0]
print(close_match)

# finding the index of the movie with title

index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index_of_the_movie)

# getting a list of similar movies

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

len(similarity_score)

# sorting the movies based on their similarity score

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

# print the name of similar movies based on the index
```

```
print('So I suggest Some Movie 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[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1

## Movie Recommendation Sytem

movie_name = input('First Please 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'].values[0]

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

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

print('So, Now I Suggest Some Movies 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[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

Thank You