# *Recommender Systems*

> About this Project

The provided code implements a basic movie recommendation system using content-based filtering and cosine similarity. Here's a summary of the code and a conclusion:

1. **Data Collection and Pre-Processing:**
   - The code starts by importing necessary libraries and loading the 'movies.csv' dataset.
   - The relevant features for recommendation are selected, including 'genres', 'keywords', 'tagline', 'cast', and 'director'.
   - Null values in the selected features are filled with empty strings.
   - The selected features are combined into a single string called 'combined_features'.
   - The text data is converted to numerical vectors using TF-IDF vectorization.
2. **Similarity Calculation:**
   - Cosine similarity is computed using the TF-IDF vectors.
   - The similarity matrix is generated, showing the similarity between all pairs of movies.
3. **Movie Recommendation:**
   - The user is prompted to enter their favorite movie name.
   - The code finds close matches to the entered movie name from the dataset.
   - The most similar movies are determined based on the cosine similarity scores.
   - A list of suggested movies is displayed, ordered by their similarity to the user's favorite movie.
4. **Conclusion:** The provided code demonstrates the implementation of a basic movie recommendation system using content-based filtering. It leverages textual features such as genres, keywords, taglines, cast, and directors to calculate the similarity between movies using TF-IDF vectors. The system then suggests movies that are most similar to the user's input, providing a list of recommended movies.

Overall, this code serves as a starting point for building a movie recommendation system, but further enhancements and refinements are necessary to create a more sophisticated and user-centered recommendation solution.

## Importing the dependencies

```
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

```
# Loading The data
movies_data = pd.read_csv('/content/movies.csv')
```

```
#first 5 rows
movies_data.head()
```

| | index | budget | genres | homepage | id | keywords | original_langua |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 237000000 | Action Adventure Fantasy Science Fiction | http://www.avatarmovie.com/ | 19995 | culture clash future space war space colony so... | |

```
# number of rows & Columns
movies_data.shape
```

```
(4803, 24)
```

```
# relevant Features for recommendation
selected_features = ['genres','keywords','tagline','cast','director']
print(selected_features)
```

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

```
# replacing the null values
for feature in selected_features:
  movies_data[feature] = movies_data[feature].fillna('')
```

```
# combined all the 5 features

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

```
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
```

```
# convert the text to  vectors
vectorizer = TfidfVectorizer()
```

```
feature_vectors = vectorizer.fit_transform(combined_features)


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
```

```
# similarity Scores
similarity = cosine_similarity(feature_vectors)
```

```
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.        ]]
```

```
print(similarity.shape)
```

```
(4803, 4803)
```

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

```
 Enter your favourite movie name : iron man
```

```
# give the list of the movies in 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', 'Spider-Man 3', 'Tangled', 'Avengers: Age of Ultron', 'Harry Potter ar
```

```
# finding the close match the givien by user
find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)
```

```
['Iron Man', 'Iron Man 3', 'Iron Man 2']
```

```
# for the only one value
close_match = find_close_match[0]
print(close_match)
```

```
Iron Man
```

```
# finding the index the value movie title
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index_of_the_movie)
```

```
68
```

```
# find the similar movie based on index value
similarity_score = list(enumerate(similarity[index_of_the_movie]))
print(similarity_score)
```

```
[(0, 0.033570748780675445), (1, 0.0546448279236134), (2, 0.013735500604224323), (3, 0.006468756104392058), (4, 0.03268943310073386), (5, 0.013907256685755473), (6, 0.0769283757
```

```
len(similarity_score)
```

```
4803
```

```
# higher similary ( sorting based on similarity)
sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)
print(sorted_similar_movies)
```

```
[(68, 1.0000000000000002), (79, 0.40890433998005965), (31, 0.31467052449477506), (7, 0.23944423963486405), (16, 0.22704403782296803), (26, 0.21566241096831154), (85, 0.2061586
```

```
# based on index of the movie
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[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

```
Movies suggested for you :

1 . Iron Man
2 . Iron Man 2
3 . Iron Man 3
4 . Avengers: Age of Ultron
5 . The Avengers
6 . Captain America: Civil War
7 . Captain America: The Winter Soldier
8 . Ant-Man
9 . X-Men
10 . Made
11 . X-Men: Apocalypse
12 . X2
13 . The Incredible Hulk
14 . The Helix... Loaded
15 . X-Men: First Class
```

```
16 . X-Men: Days of Future Past
17 . Captain America: The First Avenger
18 . Kick-Ass 2
19 . Guardians of the Galaxy
20 . Deadpool
21 . Thor: The Dark World
22 . G-Force
23 . X-Men: The Last Stand
24 . Duets
25 . Mortdecai
26 . The Last Airbender
27 . Southland Tales
28 . Zathura: A Space Adventure
29 . Sky Captain and the World of Tomorrow
```

## Test & Recommendation

```
#
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'].values[0]

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

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

# based on index of the movie
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[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

```
 Enter your favourite movie name : avatar
Movies suggested for you :

1 . Avatar
2 . Alien
3 . Aliens
```

4 . Guardians of the Galaxy
5 . Star Trek Beyond
6 . Star Trek Into Darkness
7 . Galaxy Quest
8 . Alien³
9 . Cargo
10 . Trekkies
11 . Gravity
12 . Moonraker
13 . Jason X
14 . Pocahontas
15 . Space Cowboys
16 . The Helix... Loaded
17 . Lockout
18 . Event Horizon
19 . Space Dogs
20 . Machete Kills
21 . Gettysburg
22 . Clash of the Titans
23 . Star Wars: Clone Wars: Volume 1
24 . The Right Stuff
25 . Terminator Salvation
26 . The Astronaut's Wife
27 . Planet of the Apes
28 . Star Trek
29 . Wing Commander