Project Overview: Movie Recommendation System

The Movie Recommendation System project is a content-based recommendation engine that suggests movies to users based on their preferences. The system leverages machine learning techniques, specifically Natural Language Processing (NLP) and Cosine Similarity, to provide personalized recommendations by analyzing various features of the movies such as genres, keywords, taglines, cast, and director.

Key Features of the Project:

Data Collection & Preprocessing:

1] The dataset consists of over 4800 movies with 24 features, including genres, cast, and directors.

2] We select the relevant features and handle missing data by replacing null values with empty strings.

3] These selected features are then combined to form a single descriptive string for each movie.

Text Vectorization:

The combined text features are transformed into numerical vectors using the TF-IDF (Term Frequency-Inverse Document Frequency) technique. This approach converts textual data into feature vectors that represent the importance of each word in relation to other movies in the dataset.

Cosine Similarity:

Cosine Similarity is used to measure the similarity between different movies based on their vector representations. It computes the cosine of the angle between two non-zero vectors, allowing the system to identify how similar two movies are based on their descriptions.

Recommendation Engine:

1] The user inputs the name of their favorite movie, and the system finds the closest match using difflib to handle potential typographical errors.

2] The system then computes and ranks the movies with the highest similarity scores, providing a list of the top 30 recommended movies that are most similar to the one the user entered.

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

In [2]:
```
# Data Collection and Preprocessing
# Load the dataset into a pandas dataframe
movies_data = pd.read_csv('movies.csv')
```

In [3]:
```
# Displaying the first 5 rows of the dataset
movies_data.head()
```

Out[3]:

	index	budget	genres	homepage	i
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	1999
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	28
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	20664
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	4902
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	4952

5 rows × 24 columns

In [4]:
```
# Checking the shape of the dataset (rows, columns)
print("Dataset Shape:", movies_data.shape)
```

```
Dataset Shape: (4803, 24)
```

In [5]:
```
# Selecting only the relevant features for building the recommendation engir
selected_features = ['genres', 'keywords', 'tagline', 'cast', 'director']
```

In [6]:
```
# Handling missing values by replacing null entries with empty strings
for feature in selected_features:
    movies_data[feature] = movies_data[feature].fillna('')
```

In [7]:
```
# Combining selected features into a single string for each movie
combined_features = movies_data['genres'] + ' ' + movies_data['keywords'] +
```

In [8]:

```
# Converting the combined text data into numerical feature vectors using TF-
vectorizer = TfidfVectorizer()
feature_vectors = vectorizer.fit_transform(combined_features)
```

In [9]:

```
# Displaying feature vectors
print("Feature Vectors:\n", feature_vectors)
```

```
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
```

In [10]:

```
# Computing Cosine Similarity between all movies
similarity = cosine_similarity(feature_vectors)
```

In [11]:
```
# Displaying the shape of the similarity matrix
print("Similarity Matrix Shape:", similarity.shape)
```

```
Similarity Matrix Shape: (4803, 4803)
```

In [14]:
```
# Function to get movie recommendations based on a given movie
def get_movie_recommendations(movie_name):
    # List of all movie titles
    list_of_all_titles = movies_data['title'].tolist()

    # Finding the closest match for the movie name given by the user
    close_matches = difflib.get_close_matches(movie_name, list_of_all_titles

    if not close_matches:
        return "No similar movies found."

    close_match = close_matches[0]

    # Fetching the index of the movie in the dataset
    index_of_movie = movies_data[movies_data.title == close_match].index[0]

    # Getting similarity scores for the selected movie
    similarity_scores = list(enumerate(similarity[index_of_movie]))

    # Sorting movies based on similarity scores in descending order
    sorted_similar_movies = sorted(similarity_scores, key=lambda x: x[1], re

    # Displaying the top 30 similar movie recommendations
    print("Movies recommended for you:\n")
    for i, movie in enumerate(sorted_similar_movies[1:31], start=1):
        index = movie[0]
        recommended_movie = movies_data.loc[index, 'title']
        print(f"{i}. {recommended_movie}")

# Input: Movie name from user
user_movie = input("Enter your favorite movie: ")

# Output: Recommended movies based on the input
get_movie_recommendations(movie_name=user_movie)
```

```
Enter your favorite movie: Iron man
Movies recommended for you:

1. Iron Man 2
2. Iron Man 3
3. Avengers: Age of Ultron
4. The Avengers
5. Captain America: Civil War
6. Captain America: The Winter Soldier
7. Ant-Man
8. X-Men
9. Made
10. X-Men: Apocalypse
11. X2
12. The Incredible Hulk
13. The Helix... Loaded
14. X-Men: First Class
15. X-Men: Days of Future Past
16. Captain America: The First Avenger
17. Kick-Ass 2
18. Guardians of the Galaxy
19. Deadpool
20. Thor: The Dark World
21. G-Force
22. X-Men: The Last Stand
23. Duets
24. Mortdecai
25. The Last Airbender
26. Southland Tales
27. Zathura: A Space Adventure
28. Sky Captain and the World of Tomorrow
29. The Amazing Spider-Man 2
30. The Good Night
```

Conclusion:

This project demonstrates how simple yet powerful machine learning techniques can be applied to build an effective recommendation engine. By utilizing NLP techniques and cosine similarity, this system is capable of offering accurate and relevant movie suggestions, providing users with an enhanced movie-watching experience.