Project 2 (Task 1)

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import difflib
```

```
df = pd.read_csv('/content/movies.csv')
```

```
df.head(5)
```

	index	budget	genres	homepage	id	keywords	original_language	original_title	overview	pc
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	1
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india trad...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	1
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel mi6	en	Spectre	A cryptic message from Bond’s past sends him o...	1
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc comics crime fighter terrorist secret ident...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	1
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	based on novel mars medallion space travel pri...	en	John Carter	John Carter is a war-weary, former military ca...	4

5 rows × 24 columns

```
df.shape
```

```
(4803, 24)
```

```
df = df.fillna('Null')
df
```

	index	budget	genres	homepage	id	keywords	original_language	origina
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	en	
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india trad...	en	Pira Carib Wo
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel mi6	en	
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc comics crime fighter terrorist secret ident...	en	The Da
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	based on novel mars medallion space travel pri...	en	Jol
...	...	...	...	...	...	...	...	
4798	4798	220000	Action Crime Thriller	Null	9367	united states\u2013mexico barrier legs arms pa...	es	El
4799	4799	9000	Comedy Romance	Null	72766	Null	en	Ne
4800	4800	0	Comedy Drama Romance TV Movie	http://www.hallmarkchannel.com/signedsealeddel...	231617	date love at first sight narration investigati...	en	Signed
4801	4801	0	Null	http://shanghaicalling.com/	126186	Null	en	Shangh
4802	4802	0	Documentary	Null	25975	obsession camcorder crush dream girl	en	My

4803 rows × 24 columns

```
import difflib  # For partial matching

# Load the movie dataset
df = pd.read_csv('movies.csv')
```

```
# Fill missing values with empty strings
for feature in ['overview', 'keywords', 'genres', 'cast', 'director']:
    df[feature] = df[feature].fillna('')

# Combine relevant features into a single string
df['combined_features'] = df['overview'] + ' ' + df['keywords'] + ' ' + df['genres'] + ' ' + df['cast'] + ' ' + df['director']

# Use TF-IDF to convert text data into numerical vectors
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(df['combined_features'])

# Calculate cosine similarity based on the TF-IDF matrix
cosine_sim = cosine_similarity(tfidf_matrix)

# Function to get recommendations based on movie title (with partial matching)
def get_recommendations(title, cosine_sim=cosine_sim):
    # Find close matches for the movie title in the dataset
    close_matches = difflib.get_close_matches(title, df['title'], n=5, cutoff=0.4)

    # If no close matches are found, inform the user
    if not close_matches:
        print(f"Sorry, no movies found matching '{title}'.")
        return

    # If there are multiple close matches, prompt the user to select the correct one
    print(f"\nDid you mean one of these movies?")
    for i, match in enumerate(close_matches, 1):
        print(f"{i}. {match}")

    # Ask the user to select the correct movie by number
    try:
        choice = int(input("\nEnter the number of the correct movie (1, 2, 3, etc.): "))
        selected_title = close_matches[choice - 1]
    except (ValueError, IndexError):
        print("Invalid selection.")
        return

    # Get index of the selected movie
    idx = df[df['title'] == selected_title].index[0]

    # Get similarity scores for all movies
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort movies based on similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the indices of the top 10 similar movies (excluding the input movie)
    movie_indices = [i[0] for i in sim_scores[1:11]]

    # Print recommendations based on the selected movie
    print(f"\nBased on the movie '{selected_title}', here are some recommended movies for you:\n")
    recommendations = df['title'].iloc[movie_indices]

    for i, movie in enumerate(recommendations, 1):
        print(f"{i}. {movie}")

# Ask the user to input a movie title
user_movie = input("Enter a movie title for recommendations: ")

# Get recommendations based on the user's input
get_recommendations(user_movie)
```

```
Enter a movie title for recommendations: Insidious

Did you mean one of these movies?
1. Insidious
2. Inside Out
3. Inside Job
4. Insidious: Chapter 3
5. Insidious: Chapter 2

Enter the number of the correct movie (1, 2, 3, etc.): 1

Based on the movie 'Insidious', here are some recommended movies for you:

1. Insidious: Chapter 2
```

```
2. The Conjuring 2
3. The Conjuring
4. Childless
5. Insidious: Chapter 3
6. Niagara
7. The Phantom of the Opera
8. Poltergeist
9. The Last Exorcism
10. The Haunting
```

```
# Function to combine selected features for a given movie
def combine_features(row):
    combined = (
        f"Title: {row['title']}\n"
        f"Overview: {row['overview']}\n"
        f"Keywords: {row['keywords']}\n"
        f"Genres: {row['genres']}\n"
        f"Cast: {row['cast']}\n"
        f"Director: {row['director']}\n"
        f"Popularity: {row['popularity']}\n"
        f"Vote Average: {row['vote_average']}\n"
    )
    return combined

# Apply the function to each movie row and store combined features
df['combined_features'] = df.apply(combine_features, axis=1)

# Print the combined features for each movie (you can limit this to a few for readability)
for i in range(5):  # Print for first 5 movies
    print(f"Movie {i+1} Combined Features:\n{df['combined_features'].iloc[i]}")
    print('-' * 80)
```

```
Movie 1 Combined Features:
Title: Avatar
Overview: In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn between fol
Keywords: culture clash future space war space colony society
Genres: Action Adventure Fantasy Science Fiction
Cast: Sam Worthington Zoe Saldana Sigourney Weaver Stephen Lang Michelle Rodriguez
Director: James Cameron
Popularity: 150.437577
Vote Average: 7.2

--------------------------------------------------------------------------------
Movie 2 Combined Features:
Title: Pirates of the Caribbean: At World's End
Overview: Captain Barbossa, long believed to be dead, has come back to life and is headed to the edge of the Earth with Will Turner an
Keywords: ocean drug abuse exotic island east india trading company love of one's life
Genres: Adventure Fantasy Action
Cast: Johnny Depp Orlando Bloom Keira Knightley Stellan Skarsg\u00e5rd Chow Yun-fat
Director: Gore Verbinski
Popularity: 139.082615
Vote Average: 6.9

--------------------------------------------------------------------------------
Movie 3 Combined Features:
Title: Spectre
Overview: A cryptic message from Bond’s past sends him on a trail to uncover a sinister organization. While M battles political forces
Keywords: spy based on novel secret agent sequel mi6
Genres: Action Adventure Crime
Cast: Daniel Craig Christoph Waltz L\u00e9a Seydoux Ralph Fiennes Monica Bellucci
Director: Sam Mendes
Popularity: 107.376788
Vote Average: 6.3

--------------------------------------------------------------------------------
Movie 4 Combined Features:
Title: The Dark Knight Rises
Overview: Following the death of District Attorney Harvey Dent, Batman assumes responsibility for Dent's crimes to protect the late at
Keywords: dc comics crime fighter terrorist secret identity burglar
Genres: Action Crime Drama Thriller
Cast: Christian Bale Michael Caine Gary Oldman Anne Hathaway Tom Hardy
Director: Christopher Nolan
Popularity: 112.31295
Vote Average: 7.6

--------------------------------------------------------------------------------
Movie 5 Combined Features:
Title: John Carter
Overview: John Carter is a war-weary, former military captain who's inexplicably transported to the mysterious and exotic planet of Ba
```

```
Keywords: based on novel mars medallion space travel princess
Genres: Action Adventure Science Fiction
Cast: Taylor Kitsch Lynn Collins Samantha Morton Willem Dafoe Thomas Haden Church
Director: Andrew Stanton
Popularity: 43.926995
Vote Average: 6.1

--------------------------------------------------------------------------------
```

```
tfidf_matrtfidf = TfidfVectorizer(stop_words='english')
ix = tfidf.fit_transform(df['combined_features']) # Use 'df' instead of 'movies_data'


# Assuming your movie data is in a CSV file named 'movies.csv'
movies_df = pd.read_csv('movies.csv') # Load the movie data

def get_movie_index(movies_df, movie_title):
    # Find the index of the movie based on its title
    result = movies_df[movies_df['title'].str.contains(movie_title, case=False, na=False)]
    if not result.empty:
        return result.index[0]
    else:
        return None

# Function to get user input and calculate similarity between two movies
def calculate_similarity(movies_df, cosine_sim):
    movie1 = input("Enter the title or part of the title of the first movie: ")
    movie2 = input("Enter the title or part of the title of the second movie: ")

    # Get the indices for the two movies
    idx1 = get_movie_index(movies_df, movie1)
    idx2 = get_movie_index(movies_df, movie2)

    if idx1 is None or idx2 is None:
        print("One or both movies not found in the dataset.")
        return

    # Calculate and display the cosine similarity
    similarity_score = cosine_sim[idx1, idx2]
    print(f"The cosine similarity between '{movie1}' and '{movie2}' is: {similarity_score:.4f}")

# Now the user can call this function
calculate_similarity(movies_df, cosine_sim)
```

```
Enter the title or part of the title of the first movie: Insidious
Enter the title or part of the title of the second movie: Insidious: Chapter 2
The cosine similarity between 'Insidious' and 'Insidious: Chapter 2' is: 0.1445
```

```
def get_movie_name_from_user(movies_df):
    # Get movie name input from the user
    movie_name = input("Enter the movie title (or part of the title): ")

# Run the function
get_movie_name_from_user(movies_df)
```

```
Enter the movie title (or part of the title): Insidious
```

```
# Create a list with all the movie names from the dataset
movie_names = movies_df['title'].tolist()

# Display the list of movie names
print(movie_names)
```

```
['Avatar', "Pirates of the Caribbean: At World's End", 'Spectre', 'The Dark Knight Rises', 'John Carter', 'Spider-Man 3', 'Tangled', '
```

```
def find_closest_movie(movie_name, movie_list):
    # Use difflib.get_close_matches to find the closest matches
    close_matches = difflib.get_close_matches(movie_name, movie_list, n=3, cutoff=0.6)

    if close_matches:
        print("Did you mean one of these movies?")
        for match in close_matches:
```

```
            print(match)
    else:
        print("No close matches found.")

# Create a list of all movie names from the dataset
movie_list = movies_df['title'].tolist()

# Get the movie name from the user
user_input = input("Enter a movie title: ")

# Find and print the closest matches
find_closest_movie(user_input, movie_list)
```

```
Enter a movie title: Insidious
Did you mean one of these movies?
Insidious
Inside Out
Inside Job
```

```
def find_movie_index_by_title(title, movies_df):
    # Use boolean indexing to locate the row with the given title
    movie = movies_df[movies_df['title'] == title]

    # If the movie is found, return its index, otherwise return a message
    if not movie.empty:
        return movie['index'].values[0]
    else:
        return "Movie not found."

# Take input from the user
movie_title = input("Enter the movie title: ")

# Find and display the movie index
movie_index = find_movie_index_by_title(movie_title, movies_df)
print(f"Index of the movie '{movie_title}': {movie_index}")
```

```
Enter the movie title: Insidious
Index of the movie 'Insidious': 4224
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd

def find_similar_movies(movie_title, movies_df, top_n=5):
    # Handle missing values in the 'genres' column
    movies_df['genres'] = movies_df['genres'].fillna('')  # Replace NaN with empty string

    # Create a count matrix for genres
    count_vectorizer = CountVectorizer(tokenizer=lambda x: x.split(' '))
    genre_matrix = count_vectorizer.fit_transform(movies_df['genres'])

    # Calculate the cosine similarity
    cosine_sim = cosine_similarity(genre_matrix, genre_matrix)

    # Find the index of the movie the user is looking for
    movie_index = movies_df[movies_df['title'] == movie_title].index.values[0]

    # Get similarity scores for the target movie
    similarity_scores = list(enumerate(cosine_sim[movie_index]))

    # Sort the movies based on the similarity scores
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)

    # Get the indices of the top N similar movies
    similar_movies_indices = [i[0] for i in similarity_scores[1:top_n+1]]  # Exclude the first one as it's the same movie

    # Return the titles of the similar movies
    return movies_df['title'].iloc[similar_movies_indices]

# Example usage
movie_title = input("Enter the movie title: ")
# Assuming you have a DataFrame named 'movies_df' with 'title' and 'genres' columns
# movies_df = pd.read_csv('movies.csv')  # Example loading dataset
```

```
similar_movies = find_similar_movies(movie_title, movies_df, top_n=5)
print(f"Movies similar to '{movie_title}':")
print(similar_movies)
```

```
Enter the movie title: Insidious
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:521: UserWarning: The parameter 'token_pattern' will not be
  warnings.warn(
Movies similar to 'Insidious':
1210                  Gothika
1265               FearDotCom
1598          Drag Me to Hell
1604         30 Days of Night
1605    The Cabin in the Woods
Name: title, dtype: object
```

```
# Function to sort movies based on similarity score to a given movie
def sort_movies_by_similarity(movie_title, movies_df):
    # Create a count matrix for genres
    count_vectorizer = CountVectorizer(tokenizer=lambda x: x.split(' '))
    genre_matrix = count_vectorizer.fit_transform(movies_df['genres'])

    # Calculate the cosine similarity
    cosine_sim = cosine_similarity(genre_matrix, genre_matrix)

    # Find the index of the movie the user is looking for
    if movie_title not in movies_df['title'].values:
        return "Movie not found."

    movie_index = movies_df[movies_df['title'] == movie_title].index.values[0]

    # Get similarity scores for the target movie
    similarity_scores = list(enumerate(cosine_sim[movie_index]))

    # Sort the movies based on the similarity scores
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)

    # Get the indices of the sorted movies
    sorted_movie_indices = [i[0] for i in similarity_scores]

    # Return the titles and similarity scores of the sorted movies
    sorted_movies = movies_df.iloc[sorted_movie_indices]
    sorted_movies['similarity_score'] = [similarity_scores[i][1] for i in range(len(similarity_scores))]

    return sorted_movies[['title', 'similarity_score']]

# Example usage
movie_title = input("Enter the movie title: ")
sorted_movies = sort_movies_by_similarity(movie_title, movies_df)

print(f"Movies sorted by similarity to '{movie_title}':")
print(sorted_movies)
```

```
Enter the movie title: Insidious
Movies sorted by similarity to 'Insidious':
                         title  similarity_score
666             I, Frankenstein               1.0
1210                    Gothika               1.0
1265                 FearDotCom               1.0
1598             Drag Me to Hell               1.0
1604            30 Days of Night               1.0
...                          ...               ...
4795                        Bang               0.0
4799                   Newlyweds               0.0
4800  Signed, Sealed, Delivered               0.0
4801           Shanghai Calling               0.0
4802          My Date with Drew               0.0

[4803 rows x 2 columns]
<ipython-input-37-6f6408ba803e>:27: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  sorted_movies['similarity_score'] = [similarity_scores[i][1] for i in range(len(similarity_scores))]
```

```
# Function to find similar movies based on the index
def find_similar_movies_by_index(movie_index, movies_df, top_n=5):
    # Handle missing values in the 'genres' column by replacing them with an empty string
    movies_df['genres'] = movies_df['genres'].fillna('')

    # Create a count matrix for genres
    count_vectorizer = CountVectorizer(tokenizer=lambda x: x.split(' '))
    genre_matrix = count_vectorizer.fit_transform(movies_df['genres'])

    # Calculate the cosine similarity
    cosine_sim = cosine_similarity(genre_matrix, genre_matrix)

    # Get similarity scores for the target movie based on its index
    similarity_scores = list(enumerate(cosine_sim[movie_index]))

    # Sort the movies based on the similarity scores
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)

    # Get the indices of the top N similar movies (excluding the first as it's the same movie)
    similar_movies_indices = [i[0] for i in similarity_scores[1:top_n+1]]

    # Return the titles of the similar movies
    return movies_df['title'].iloc[similar_movies_indices]

# Example usage
movie_index = int(input("Enter the movie index: "))
similar_movies = find_similar_movies_by_index(movie_index, movies_df, top_n=5)
print(f"Movies similar to the movie with index {movie_index}:")
print(similar_movies)
```

```
Enter the movie index: 4224
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:521: UserWarning: The parameter 'token_pattern' will not be
  warnings.warn(
Movies similar to the movie with index 4224:
1210                  Gothika
1265               FearDotCom
1598          Drag Me to Hell
1604         30 Days of Night
1605    The Cabin in the Woods
Name: title, dtype: object
```