Group 4 - Netflix Movie Recommender | Jay-An, Huu Huy Anh, Luis and Anmol

1. Installing Packages and Installing Libraries

In [1]:
```
# Necessary Packages:

#!pip install rake-nltk
#!pip install pandas
#!pip install nltk
#!pip install scikit-learn
#!install requests
#!pip install numpy
#!pip install rake-nltk
#!pip install gensim
#nltk.download('punkt')
#nltk.download('wordnet')
#nltk.download('stopwords')
#nltk.download('omw-1.4')
```

In [2]:
```
import pandas as pd
import ast
import nltk
import string
import re
import requests
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, H
from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances, ma
from nltk.stem import WordNetLemmatizer
from scipy.sparse import hstack
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from rake_nltk import Rake
from gensim.utils import tokenize
from gensim.parsing.preprocessing import remove_stopwords
from pprint import PrettyPrinter
from nltk.tokenize import RegexpTokenizer
from gensim.models import Word2Vec
```

2. Importing DataFrames

This section of code imports the data we need and displays a few rows for checking purposes.

In [3]:
```
# Reads .csv files:
credits_df = pd.read_csv('data/tmdb_5000_credits.csv')
movies_df = pd.read_csv('data/tmdb_5000_movies.csv')
```

In [4]:
```
movies_df.head(3)
```

Out[4]:

	budget	genres	homepage	id	keywords	origi
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	

In [5]:
```python
credits_df.head(3)
```

Out[5]:

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...

3. Joining DataFrames

This section of code merges the "credits" dataframe and "movies" dataframe that were imported earlier into a new dataframe called "df_not_cleaned" for further use.

In [6]:
```python
# Renames movie_id column to id
credits_df = credits_df.rename(columns={'movie_id': 'id'})

# Joins dataframes
df_not_cleaned = pd.merge(movies_df, credits_df, on='id', how='inner')
df_not_cleaned = df_not_cleaned[['id', 'genres', 'original_title', 'title_x', '

# Displays dataframe
df_not_cleaned.head(3)
```

Out[6]:

	id	genres	original_title	title_x	title_y	keywords	overview	popularity
0	19995	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	Avatar	Avatar	Avatar	[{"id": 1463, "name": "culture clash"}, {"id":...	In the 22nd century, a paraplegic Marine is di...	150.437577
1	285	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	Pirates of the Caribbean: At World's End	Pirates of the Caribbean: At World's End	Pirates of the Caribbean: At World's End	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	Captain Barbossa, long believed to be dead, ha...	139.082615
2	206647	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	Spectre	Spectre	Spectre	[{"id": 470, "name": "spy"}, {"id": 818, "name...	A cryptic message from Bond's past sends him o...	107.376788

4. Cleaning DataFrame

This section of code is responsible for cleaning the dataset. First, we check for null data and rows with 0 runtime, using an API key to fill in missing runtime and overview information from the TMDB website. Since there was only one null data point for the release date, we decided to drop it.

Next, we created functions to extract specific data from multiple columns, convert text to lowercase, remove spaces from certain column values, combine relevant data into new columns, and rename/drop columns for clarity. We then created a new dataframe with the cleaned data. Finally, we joined the "tag", "tag_genres", and "tag_ppl" lists into single string values for further processing.

4.a. Droppping runtime and release_date null rows

In [7]:
```
# Displays movies that has 0 or null in 'runtime' and 'vote_average'
for index, row in df_not_cleaned.iterrows():
    if pd.isnull(row['runtime']) or row['runtime'] == 0 or pd.isnull(row['vote_
        print(row['id'], row['title_x'], row['runtime'], row['vote_average'])
```

```
53953 The Tooth Fairy 0.0 4.3
310706 Black Water Transit 100.0 0.0
370980 Chiamatemi Francesco - Il Papa della gente nan 7.3
41894 Blood Done Sign My Name 0.0 6.0
113406 Should've Been Romeo 0.0 0.0
447027 Running Forever 88.0 0.0
158150 How to Fall in Love 0.0 5.2
395766 The Secret 200.0 0.0
370662 Time to Choose 100.0 0.0
281230 Fort McCoy 0.0 6.3
170480 The Deported 90.0 0.0
79587 Four Single Fathers 100.0 0.0
346081 Sardaarji 0.0 9.5
433715 8 Days 90.0 0.0
364083 Mi America 126.0 0.0
371085 Sharkskin 0.0 0.0
325140 Hum To Mohabbat Karega 0.0 0.0
459488 To Be Frank, Sinatra at 100 nan 0.0
386826 A Beginner's Guide to Snuff 87.0 0.0
66468 N-Secure 0.0 4.3
74084 Dil Jo Bhi Kahey... 0.0 0.0
51820 The Salon 0.0 3.5
280381 House at the End of the Drive 91.0 0.0
218500 The Ballad of Gregorio Cortez 104.0 0.0
295914 Queen of the Mountains 135.0 0.0
357834 The Algerian 99.0 0.0
114065 Down & Out With The Dolls 88.0 0.0
49951 Certifiably Jonathan 85.0 0.0
355629 The Blade of Don Juan 98.0 0.0
107315 Below Zero 0.0 4.4
310933 Bleeding Hearts 0.0 2.0
102840 Sex With Strangers 0.0 5.0
202604 The Vatican Exorcisms 0.0 4.4
219716 Sparkler 96.0 0.0
357441 Karachi se Lahore 0.0 8.0
323270 The Horror Network Vol. 1 0.0 5.0
146882 Elza 78.0 0.0
279759 Harrison Montgomery 0.0 0.0
146269 The Young Unknowns 87.0 0.0
302579 Naturally Native 107.0 0.0
51955 Hav Plenty 92.0 0.0
296943 The Hadza:  Last of the First 70.0 0.0
181940 Carousel of Revenge 103.0 0.0
263503 Water & Power 0.0 3.0
331493 Light from the Darkroom 0.0 0.0
70875 The Harvest (La Cosecha) 80.0 0.0
294550 The Outrageous Sophie Tucker 96.0 0.0
380097 America Is Still the Place 0.0 0.0
119657 El Rey de Najayo 101.0 0.0
285743 Alleluia! The Devil's Carnival 0.0 6.0
362765 The Sound and the Shadow 90.0 0.0
94072 Straight Out of Brooklyn 0.0 4.3
325579 Diamond Ruff 0.0 2.4
198370 Mutual Friends 0.0 0.0
328307 Rise of the Entrepreneur: The Search for a Better Way 0.0 8.0
281189 Gory Gory Hallelujah 0.0 1.0
189711 Love in the Time of Monsters 0.0 5.0
43743 Fabled 84.0 0.0
162396 The Big Swap 0.0 0.0
47534 Fighting Tommy Riley 0.0 5.3
```

```
426067 Midnight Cabaret 94.0 0.0
324352 Anderson's Cross 98.0 0.0
300327 Death Calls 0.0 0.0
378237 Amidst the Devil's Wings 90.0 0.0
46252 Rust 94.0 0.0
320435 UnDivided 0.0 0.0
150211 The Frozen 0.0 4.2
274758 Give Me Shelter 90.0 0.0
40963 Little Big Top 0.0 10.0
376010 Western Religion 106.0 0.0
194588 Short Cut to Nirvana: Kumbh Mela 85.0 0.0
402515 American Beast 89.0 0.0
361398 Theresa Is a Mother 105.0 0.0
288927 Archaeology of a Woman 94.0 0.0
253290 Butterfly Girl 77.0 0.0
344466 The World Is Mine 104.0 0.0
354624 Heroes of Dirt 98.0 0.0
335244 Antarctic Edge: 70° South 72.0 0.0
282128 An American in Hollywood 89.0 0.0
38786 The Blood of My Brother: A Story of Death in Iraq 90.0 0.0
266857 The Work and The Story 70.0 0.0
272726 Dude Where's My Dog? 0.0 0.0
69382 The Legend of God's Gun 78.0 0.0
220490 Her Cry: La Llorona Investigation 89.0 0.0
366967 Dutch Kills 90.0 0.0
287625 Stories of Our Lives 60.0 0.0
286939 Sanctuary: Quite a Conundrum 82.0 0.0
```

In [8]:
```python
# Defines function to fetch data for a single movie by ID

# Defines your TMDB API key
api_key = 'a38b4a8ed24b9edec801f0bc153f0177'

def get_movie_data(movie_id, column_name):
    url = f'https://api.themoviedb.org/3/movie/{movie_id}?api_key={api_key}&lar
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        return data[column_name]
    else:
        return None
```

In [9]:
```python
# Loops through each movie and fill in 'runtime' and 'vote_avearge' using get_r
for i, row in df_not_cleaned.iterrows():
    if pd.isnull(row['runtime']) or row['runtime'] == 0:
        runtime = get_movie_data(row['id'], 'runtime')
        if runtime:
            df_not_cleaned.at[i, 'runtime'] = runtime

    if pd.isnull(row['vote_average']) or row['vote_average'] == 0:
        vote_average = get_movie_data(row['id'], 'vote_average')
        if vote_average:
            df_not_cleaned.at[i, 'vote_average'] = vote_average
```

In [10]:
```python
# Checks if there are movies with 'runtime' equal 0 or null
for index, row in df_not_cleaned.iterrows():
    if pd.isnull(row['runtime']) or row['runtime'] == 0:
        print(row['id'], row['title_x'], row['runtime'])
```

```
41894 Blood Done Sign My Name 0.0
113406 Should've Been Romeo 0.0
281230 Fort McCoy 0.0
51820 The Salon 0.0
310933 Bleeding Hearts 0.0
325579 Diamond Ruff 0.0
328307 Rise of the Entrepreneur: The Search for a Better Way 0.0
320435 UnDivided 0.0
```

In [11]:
```
# Checks if there are movies with 'vote_average' equal 0 or null
for index, row in df_not_cleaned.iterrows():
    if pd.isnull(row['vote_average']) or row['vote_average'] == 0:
        print(row['id'], row['title_x'], row['vote_average'])
```

```
395766 The Secret 0.0
170480 The Deported 0.0
364083 Mi America 0.0
371085 Sharkskin 0.0
296943 The Hadza:  Last of the First 0.0
181940 Carousel of Revenge 0.0
331493 Light from the Darkroom 0.0
43743 Fabled 0.0
300327 Death Calls 0.0
378237 Amidst the Devil's Wings 0.0
320435 UnDivided 0.0
376010 Western Religion 0.0
194588 Short Cut to Nirvana: Kumbh Mela 0.0
361398 Theresa Is a Mother 0.0
288927 Archaeology of a Woman 0.0
354624 Heroes of Dirt 0.0
282128 An American in Hollywood 0.0
266857 The Work and The Story 0.0
366967 Dutch Kills 0.0
```

In [12]:
```
# There are no data online for the remaining 0s in 'runtime' and 'vote_average
# Drops movies that have 'runtime' equal 0
# Drops movies that have 'release_date' null
# Drops movies that have 'vote_average' equal 0

df_not_cleaned = df_not_cleaned[df_not_cleaned['runtime'] != 0]
df_not_cleaned.dropna(subset=['release_date'], inplace=True)
df_not_cleaned.drop(df_not_cleaned[df_not_cleaned['vote_average'] == 0].index,

df_not_cleaned.isnull().sum() #checking if any null data left
```

Out[12]:
```
id                       0
genres                   0
original_title           0
title_x                  0
title_y                  0
keywords                 0
overview                31
popularity               0
release_date             0
runtime                  0
vote_average             0
vote_count               0
cast                     0
crew                     0
production_companies     0
dtype: int64
```

In [13]:
```python
# Displays dataframe
df_not_cleaned.head(3)
```

Out[13]:

	id	genres	original_title	title_x	title_y	keywords	overview	popularity
0	19995	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	Avatar	Avatar	Avatar	[{"id": 1463, "name": "culture clash"}, {"id":...	In the 22nd century, a paraplegic Marine is di...	150.437577
1	285	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	Pirates of the Caribbean: At World's End	Pirates of the Caribbean: At World's End	Pirates of the Caribbean: At World's End	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	Captain Barbossa, long believed to be dead, ha...	139.082615
2	206647	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	Spectre	Spectre	Spectre	[{"id": 470, "name": "spy"}, {"id": 818, "name...	A cryptic message from Bond's past sends him o...	107.376788

4.b. Dealing with "overview" null rows

In [14]:
```python
# Creates new dataframe with movies that are null in 'overview' column
null_overview = df_not_cleaned[df_not_cleaned["overview"].isnull()]

# Upon further analysis, we decided to only keep 'title_x' as 'title_y' is the
# and 'original_title' shows names that are not in english
# Drops 'title_y' and 'original_title' columns
df_not_cleaned = df_not_cleaned.drop(columns=['title_y','original_title'])

# Prints out all movie titles that have NAN in 'overview' column
null_overview["title_x"]
```

Out[14]:
```
65                                         The Dark Knight
77                                              Inside Out
94                                 Guardians of the Galaxy
95                                            Interstellar
96                                               Inception
262     The Lord of the Rings: The Fellowship of the Ring
287                                       Django Unchained
298                                 The Wolf of Wall Street
329         The Lord of the Rings: The Return of the King
330                 The Lord of the Rings: The Two Towers
494                                          The Lion King
634                                             The Matrix
662                                             Fight Club
690                                         The Green Mile
809                                           Forrest Gump
1553                                                Se7en
1818                                      Schindler's List
1881                              The Shawshank Redemption
1990                               The Empire Strikes Back
2091                              The Silence of the Lambs
2285                                    Back to the Future
2294                                         Spirited Away
2522                                    The Imitation Game
2656             Chiamatemi Francesco - Il Papa della gente
2731                                The Godfather: Part II
2912                                             Star Wars
3232                                          Pulp Fiction
3337                                         The Godfather
3865                                              Whiplash
4140                          To Be Frank, Sinatra at 100
4431                                           Food Chains
Name: title_x, dtype: object
```

In [15]:
```python
# Loops through each movie and fill in 'overview' using get_movie_data functio
for i, row in df_not_cleaned.iterrows():
    if pd.isnull(row['overview']):
        overview = get_movie_data(row['id'], 'overview')
        if overview:
            df_not_cleaned.at[i, 'overview'] = overview
```

In [16]:
```python
# Checks if there are any null data in overview column
null_overview = df_not_cleaned[df_not_cleaned["overview"].isnull()]
null_overview["title_x"]  # No Null data on overview.
```

Out[16]:
```
Series([], Name: title_x, dtype: object)
```

4.c. Creating Functions to Clean Rows

In [17]:
```python
# Cleaning Functions:

def extract(lst): # Function to extract values from a dictionary
    feat = []
    for i in ast.literal_eval(lst):
        feat.append(i['name'])
    return feat

def get_names (lst): # Function to extract first 3 values from a dictionary
```

```
    feat = []
    counter = 0
    for i in ast.literal_eval(lst):
        if counter != 3:
            feat.append(i['name'])
            counter += 1
        else:
            break
    return feat

def get_director(lst): # Function to extract director name from crew column
    feat = []
    for i in ast.literal_eval(lst):
        if i['job'] == 'Director':
            feat.append(i['name'])
    return feat
```

4.d. Cleaning Cast, Crew, Genres, Keywords, and Production Company Columns

In [18]:

```
# Extracts data from multiple columns and converts to lowercase:

# 1 Cast - ONLY TOP 3 Actor/Actress
df_not_cleaned['cast_names'] = df_not_cleaned['cast'].apply(lambda x: get_names
df_not_cleaned = df_not_cleaned.drop(columns=['cast']) #cast is now cast_names

# 2 Crew - ONLY Director
df_not_cleaned['crew_names'] = df_not_cleaned['crew'].apply(get_director)
df_not_cleaned['crew_names'] = df_not_cleaned['crew_names'].apply(lambda x: [na
df_not_cleaned = df_not_cleaned.drop(columns=['crew']) #crew is now crew_names

# 3 Genres - ONLY TOP 3 Genres
df_not_cleaned['genres_names'] = df_not_cleaned['genres'].apply(lambda x: get_n
df_not_cleaned = df_not_cleaned.drop(columns=['genres']) #genres is now genres_

# 4 Keywords - All
df_not_cleaned['keywords_names'] = df_not_cleaned['keywords'].apply(lambda x: e
df_not_cleaned = df_not_cleaned.drop(columns=['keywords'])#keywords is now keyw

# 5 Production Companies - All
df_not_cleaned['production_companies_names'] = df_not_cleaned['production_compa
df_not_cleaned = df_not_cleaned.drop(columns=['production_companies'])#producti

# Sets the maximum column width to display full column contents
pd.set_option('max_colwidth', None)

# Displays dataframe
df_not_cleaned.head(3)
```

Out[18]:

	id	title_x	overview	popularity	release_date	runtime	vote_average	vote_cou
0	19995	Avatar	In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization.	150.437577	2009-12-10	162.0	7.2	118(
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, has come back to life and is headed to the edge of the Earth with Will Turner and Elizabeth Swann. But nothing is quite as it seems.	139.082615	2007-05-19	169.0	6.9	45(
2	206647	Spectre	A cryptic message from Bond's past sends him on a trail to uncover a sinister organization. While M battles political forces to keep the secret service alive, Bond peels back the layers of deceit to reveal the terrible truth behind SPECTRE.	107.376788	2015-10-26	148.0	6.3	44(

In [19]:

```
# Converts overview text to lowercase and split into a list of words
df_not_cleaned['overview'] = df_not_cleaned['overview'].apply(lambda x: x.lower
# Removes spaces from genre, keyword, and production company names
df_not_cleaned['genres_names'] = df_not_cleaned['genres_names'].apply(lambda x:
df_not_cleaned['keywords_names'] = df_not_cleaned['keywords_names'].apply(lambd
df_not_cleaned['production_companies_names'] = df_not_cleaned['production_compa

# Displays dataframe
df_not_cleaned.head(3)
```

Out[19]:

	id	title_x	overview	popularity	release_date	runtime	vote_average	vote_cou
0	19995	Avatar	[in, the, 22nd, century,, a, paraplegic, marine, is, dispatched, to, the, moon, pandora, on, a, unique, mission,, but, becomes, torn, between, following, orders, and, protecting, an, alien, civilization.]	150.437577	2009-12-10	162.0	7.2	118
1	285	Pirates of the Caribbean: At World's End	[captain, barbossa,, long, believed, to, be, dead,, has, come, back, to, life, and, is, headed, to, the, edge, of, the, earth, with, will, turner, and, elizabeth, swann., but, nothing, is, quite, as, it, seems.]	139.082615	2007-05-19	169.0	6.9	45
2	206647	Spectre	[a, cryptic, message, from, bond's, past, sends, him, on, a, trail, to, uncover, a, sinister, organization., while, m, battles, political, forces, to, keep, the, secret, service, alive,, bond, peels, back, the, layers, of, deceit, to, reveal, the, terrible, truth, behind, spectre.]	107.376788	2015-10-26	148.0	6.3	44

4.e. Organizing DataFrame

In [20]:
```
# Combines overview, genres, keywords, and production company into one column '
# Renames "genres_names" to "tag_genres"
# Combines "cast_names", "crew_names" into one column "tag_ppl"

df_not_cleaned['tag'] = df_not_cleaned['overview']+df_not_cleaned['genres_names
df_not_cleaned['tag_genres'] = df_not_cleaned['genres_names']
df_not_cleaned['tag_ppl'] = df_not_cleaned['cast_names']+df_not_cleaned['crew_n

# Displays dataframe
df_not_cleaned.head(3)
```

Out[20]:

	id	title_x	overview	popularity	release_date	runtime	vote_average	vote_cou
0	19995	Avatar	[in, the, 22nd, century,, a, paraplegic, marine, is, dispatched, to, the, moon, pandora, on, a, unique, mission,, but, becomes, torn, between, following, orders, and, protecting, an, alien, civilization.]	150.437577	2009-12-10	162.0	7.2	118
1	285	Pirates of the Caribbean: At World's End	[captain, barbossa,, long, believed, to, be, dead,, has, come, back, to, life, and, is, headed, to, the, edge, of, the, earth, with, will, turner, and, elizabeth, swann., but, nothing, is, quite, as, it, seems.]	139.082615	2007-05-19	169.0	6.9	45
2	206647	Spectre	[a, cryptic, message, from, bond’s, past, sends, him, on, a, trail, to, uncover, a, sinister, organization., while, m, battles, political, forces, to, keep, the, secret, service, alive,, bond, peels, back, the, layers, of, deceit, to, reveal, the, terrible, truth, behind, spectre.]	107.376788	2015-10-26	148.0	6.3	44

In [21]:
```
# Creates new dataframe with cleaned columns we needed for recommendation syste
df_cleaned = df_not_cleaned[['id','title_x','tag','tag_genres','tag_ppl','runti

# Joins 'tag', 'tag_genres', and 'tag_ppl' lists into single string values
df_cleaned['tag'] = df_cleaned['tag'].apply(lambda x:' '.join(x))
df_cleaned['tag_genres'] = df_cleaned['tag_genres'].apply(lambda x:' '.join(x))
df_cleaned['tag_ppl'] = df_cleaned['tag_ppl'].apply(lambda x:' '.join(x))

# Renames 'title_x' to 'title'
df_cleaned = df_cleaned.rename(columns={'title_x': 'title'})

# Displays new dataframe
df_cleaned.head(3)
```

```
/var/folders/c6/c3dgh49x4wx_n7ctzrv_6nzh0000gn/T/ipykernel_3074/2013941429.py:
5: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['tag'] = df_cleaned['tag'].apply(lambda x:' '.join(x))
/var/folders/c6/c3dgh49x4wx_n7ctzrv_6nzh0000gn/T/ipykernel_3074/2013941429.py:
6: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['tag_genres'] = df_cleaned['tag_genres'].apply(lambda x:' '.join
(x))
/var/folders/c6/c3dgh49x4wx_n7ctzrv_6nzh0000gn/T/ipykernel_3074/2013941429.py:
7: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['tag_ppl'] = df_cleaned['tag_ppl'].apply(lambda x:' '.join(x))
```

Out[21]:

	id	title	tag	tag_genres	tag_ppl	runtime	vote_
0	19995	Avatar	in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization. action adventure fantasy cultureclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations mindandsoul 3d ingeniousfilmpartners twentiethcenturyfoxfilmcorporation duneentertainment lightstormentertainment	action adventure fantasy	sam worthington zoe saldana sigourney weaver james cameron	162.0	
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believed to be dead, has come back to life and is headed to the edge of the earth with will turner and elizabeth swann. but nothing is quite as it seems. adventure fantasy action ocean drugabuse exoticisland eastindiatradingcompany loveofone'slife traitor shipwreck strongwoman ship alliance calypso afterlife fighter pirate swashbuckler aftercreditsstinger waltdisneypictures jerrybruckheimerfilms secondmateproductions	adventure fantasy action	johnny depp orlando bloom keira knightley gore verbinski	169.0	
2	206647	Spectre	a cryptic message from bond’s past sends him on a trail to uncover a sinister organization. while m battles political forces to keep the secret service alive, bond peels back the layers of deceit to reveal the terrible truth behind spectre. action adventure crime spy basedonnovel secretagent sequel mi6 britishsecretservice unitedkingdom columbiapictures danjaq b24	action adventure crime	daniel craig christoph waltz léa seydoux sam mendes	148.0	

5. Text Mining Preprocessing Techniques

This section of code is responsible for text mining processing. The techniques included in this section are remove punctuation, tokenization, stop words, and lemmatization. In the previous section, we already applied lower case while extracting the data.

5.a. Removing Punctuation

```
In [22]: # Creates copy of cleaned dataframe
         df_clean_rp = df_cleaned

         # Defines function to remove punctuation
         def remove_punctuation(text):
             if isinstance(text, str):
                 return text.translate(str.maketrans('', '', string.punctuation))
             return text

         # Applies the function to 'tag', 'tag_genres', and 'tag_ppl' columns
         df_clean_rp[['RP_tag','RP_tag_genres','RP_tag_ppl']] = df_clean_rp[['tag','tag_

         # Displays dataframe
         df_clean_rp.head(3)
```

Out[22]:

	id	title	tag	tag_genres	tag_ppl	runtime	vote_
0	19995	Avatar	in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization. action adventure fantasy cultureclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations mindandsoul 3d ingeniousfilmpartners twentiethcenturyfoxfilmcorporation duneentertainment lightstormentertainment	action adventure fantasy	sam worthington zoe saldana sigourney weaver james cameron	162.0	
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believed to be dead, has come back to life and is headed to the edge of the earth with will turner and elizabeth swann. but nothing is quite as it seems. adventure fantasy action ocean drugabuse exoticisland eastindiatradingcompany loveofone'slife traitor shipwreck strongwoman ship alliance calypso afterlife fighter pirate swashbuckler aftercreditsstinger waltdisneypictures jerrybruckheimerfilms secondmateproductions	adventure fantasy action	johnny depp orlando bloom keira knightley gore verbinski	169.0	
2	206647	Spectre	a cryptic message from bond's past sends him on a trail to uncover a sinister organization. while m battles political forces to keep the secret service alive, bond peels back the layers of deceit to reveal the terrible truth behind spectre. action adventure crime spy basedonnovel secretagent sequel mi6 britishsecretservice unitedkingdom columbiapictures danjaq b24	action adventure crime	daniel craig christoph waltz léa seydoux sam mendes	148.0	

5.b. Tokenization

In [23]:

```python
# Creates copy of cleaned dataframe
df_clean_token = df_cleaned

# Splits the text based on commas, Removes empty strings/extra whitespace, and
def tokenize_string(text):
    tokens = re.split(r',\s*', str(text))
    tokens = [token.strip() for token in tokens if token.strip()]
    tokens = [token.lower() for token in tokens]
    return tokens
```

```
# Applies function to 'tag', 'tag_genres', and 'tag_ppl' columns
df_clean_token[['tokenized_tag','tokenized_tag_genres','tokenized_tag_ppl']] =

# Displays dataframe
df_clean_token.head(3)
```

Out[23]:

	id	title	tag	tag_genres	tag_ppl	runtime	vote_
0	19995	Avatar	in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization. action adventure fantasy cultureclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations mindandsoul 3d ingeniousfilmpartners twentiethcenturyfoxfilmcorporation duneentertainment lightstormentertainment	action adventure fantasy	sam worthington zoe saldana sigourney weaver james cameron	162.0	
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believed to be dead, has come back to life and is headed to the edge of the earth with will turner and elizabeth swann. but nothing is quite as it seems. adventure fantasy action ocean drugabuse exoticisland eastindiatradingcompany loveofone'slife traitor shipwreck strongwoman ship alliance calypso afterlife fighter pirate swashbuckler aftercreditsstinger waltdisneypictures jerrybruckheimerfilms secondmateproductions	adventure fantasy action	johnny depp orlando bloom keira knightley gore verbinski	169.0	
2	206647	Spectre	a cryptic message from bond's past sends him on a trail to uncover a sinister organization. while m battles political forces to keep the secret service alive, bond peels back the layers of deceit to reveal the terrible truth behind spectre. action adventure crime spy basedonnovel secretagent sequel mi6 britishsecretservice unitedkingdom columbiapictures danjaq b24	action adventure crime	daniel craig christoph waltz léa seydoux sam mendes	148.0	

5.c. Removing Stop Words

In [24]:
```
# Creates copy of cleaned dataframe
df_clean_stopwords = df_cleaned

# Creates set of stopwords for English language
```

```
stop_words = set(stopwords.words('english'))

# Defines function to remove stopwords
def remove_stopwords(text):
    tokens = word_tokenize(text.lower())
    filtered_tokens = [token for token in tokens if not token in stop_words]
    return ' '.join(filtered_tokens)

# Applies Stop Words to Overview Column
df_clean_stopwords['tag_sw'] = df_clean_stopwords['tag'].apply(remove_stopwords

# Displays dataframe
df_clean_stopwords.head(3)
```

Out[24]:

	id	title	tag	tag_genres	tag_ppl	runtime	vote_
0	19995	Avatar	in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization. action adventure fantasy cultureclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations mindandsoul 3d ingeniousfilmpartners twentiethcenturyfoxfilmcorporation duneentertainment lightstormentertainment	action adventure fantasy	sam worthington zoe saldana sigourney weaver james cameron	162.0	
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believed to be dead, has come back to life and is headed to the edge of the earth with will turner and elizabeth swann. but nothing is quite as it seems. adventure fantasy action ocean drugabuse exoticisland eastindiatradingcompany loveofone'slife traitor shipwreck strongwoman ship alliance calypso afterlife fighter pirate swashbuckler aftercreditsstinger waltdisneypictures jerrybruckheimerfilms secondmateproductions	adventure fantasy action	johnny depp orlando bloom keira knightley gore verbinski	169.0	
2	206647	Spectre	a cryptic message from bond’s past sends him on a trail to uncover a sinister organization. while m battles political forces to keep the secret service alive, bond peels back the layers of deceit to reveal the terrible truth behind spectre. action adventure crime spy basedonnovel secretagent sequel mi6 britishsecretservice unitedkingdom columbiapictures danjaq b24	action adventure crime	daniel craig christoph waltz léa seydoux sam mendes	148.0	

5.d. Lemmatization

In [25]:
```
# Creates copy of cleaned dataframe
df_clean_lm = df_cleaned

# Creates instances for lemmatization
lemmatizer = WordNetLemmatizer()

# Defines funtion to lemmatize text
def lemmatize(tokens):
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return ' '.join(lemmatized_tokens)

# Applies Lemmatization to 'tag_sw', 'tokenized_tag_genres', and 'tokenized_tag
df_clean_lm['tag_sw_lem'] = df_clean_lm['tag_sw'].apply(word_tokenize).apply(le
df_clean_lm['tag_genres_lem'] = df_clean_lm['tokenized_tag_genres'].apply(lemma
df_clean_lm['tag_ppl_lem'] = df_clean_lm['tokenized_tag_ppl'].apply(lemmatize)

# Displays dataframe
df_clean_lm.head(3)
```

Out[25]:

	id	title	tag	tag_genres	tag_ppl	runtime	vote_
0	19995	Avatar	in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization. action adventure fantasy cultureclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations mindandsoul 3d ingeniousfilmpartners twentiethcenturyfoxfilmcorporation duneentertainment lightstormentertainment	action adventure fantasy	sam worthington zoe saldana sigourney weaver james cameron	162.0	
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believed to be dead, has come back to life and is headed to the edge of the earth with will turner and elizabeth swann. but nothing is quite as it seems. adventure fantasy action ocean drugabuse exoticisland eastindiatradingcompany loveofone'slife traitor shipwreck strongwoman ship alliance calypso afterlife fighter pirate swashbuckler aftercreditsstinger waltdisneypictures jerrybruckheimerfilms secondmateproductions	adventure fantasy action	johnny depp orlando bloom keira knightley gore verbinski	169.0	
2	206647	Spectre	a cryptic message from bond's past sends him on a trail to uncover a sinister organization. while m battles political forces to keep the secret service alive, bond peels back the layers of deceit to reveal the terrible truth behind spectre. action adventure crime spy basedonnovel secretagent sequel mi6 britishsecretservice unitedkingdom columbiapictures danjaq b24	action adventure crime	daniel craig christoph waltz léa seydoux sam mendes	148.0	

5.e. Cleaning Text

In [26]:
```python
# Creates copy of cleaned dataframe
df_clean_lm_sw_cleaned = df_cleaned

# Defines function to remove punctuation
def remove_punctuation(text):
    text = re.sub(r'[^\w\s]', '', text)
    return text

# Applies the function to 'tag_sw_lem','tag_genres_lem', and 'tag_ppl_lem' colu
df_clean_lm_sw_cleaned['tag_lem_2x_sw'] = df_clean_lm_sw_cleaned['tag_sw_lem'].
```

```
df_clean_lm_sw_cleaned['tag_lem_2x_genres'] = df_clean_lm_sw_cleaned['tag_genre
df_clean_lm_sw_cleaned['tag_lem_2x_ppl'] = df_clean_lm_sw_cleaned['tag_ppl_lem'

# Displays dataframe
df_clean_lm_sw_cleaned.head(3)
```

Out[26]:

	id	title	tag	tag_genres	tag_ppl	runtime	vote_
0	19995	Avatar	in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization. action adventure fantasy cultureclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations mindandsoul 3d ingeniousfilmpartners twentiethcenturyfoxfilmcorporation duneentertainment lightstormentertainment	action adventure fantasy	sam worthington zoe saldana sigourney weaver james cameron	162.0	
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believed to be dead, has come back to life and is headed to the edge of the earth with will turner and elizabeth swann. but nothing is quite as it seems. adventure fantasy action ocean drugabuse exoticisland eastindiatradingcompany loveofone'slife traitor shipwreck strongwoman ship alliance calypso afterlife fighter pirate swashbuckler aftercreditsstinger waltdisneypictures jerrybruckheimerfilms secondmateproductions	adventure fantasy action	johnny depp orlando bloom keira knightley gore verbinski	169.0	
2	206647	Spectre	a cryptic message from bond’s past sends him on a trail to uncover a sinister organization. while m battles political forces to keep the secret service alive, bond peels back the layers of deceit to reveal the terrible truth behind spectre. action adventure crime spy basedonnovel secretagent sequel mi6 britishsecretservice unitedkingdom columbiapictures danjaq b24	action adventure crime	daniel craig christoph waltz léa seydoux sam mendes	148.0	

5.f. Choosing and Preparing Final Dataframe

In [27]:
```
# Selects relevant rows to new dataframe, Renames and Displays
df_ready_to_vector = df_clean_lm[['id', 'title', 'tag_lem_2x_sw','tag_lem_2x_ge
df_ready_to_vector = df_ready_to_vector.rename(columns={'tag_lem_2x_sw': 'tag',
df_ready_to_vector.head(3)
```

Out[27]:

	id	title	tag	genres_tag	ppl_tag	runtime	vote_
0	19995	Avatar	22nd century paraplegic marine dispatched moon pandora unique mission becomes torn following order protecting alien civilization action adventure fantasy cultureclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations mindandsoul 3d ingeniousfilmpartners twentiethcenturyfoxfilmcorporation duneentertainment lightstormentertainment	action adventure fantasy	sam worthington zoe saldana sigourney weaver james cameron	162.0	
1	285	Pirates of the Caribbean: At World's End	captain barbossa long believed dead come back life headed edge earth turner elizabeth swann nothing quite seems adventure fantasy action ocean drugabuse exoticisland eastindiatradingcompany loveofoneslife traitor shipwreck strongwoman ship alliance calypso afterlife fighter pirate swashbuckler aftercreditsstinger waltdisneypictures jerrybruckheimerfilms secondmateproductions	adventure fantasy action	johnny depp orlando bloom keira knightley gore verbinski	169.0	
2	206647	Spectre	cryptic message bond past sends trail uncover sinister organization battle political force keep secret service alive bond peel back layer deceit reveal terrible truth behind spectre action adventure crime spy basedonnovel secretagent sequel mi6 britishsecretservice unitedkingdom columbiapictures danjaq b24	action adventure crime	daniel craig christoph waltz léa seydoux sam mendes	148.0	

5.g. Creating Genres Matrix (Genre Feature)

In [28]:
```
# Extracts movie ID and genres columns
genre_df = df_ready_to_vector[['id', 'genres_tag']]

# Creates dictionary of merged genres
merged_genres = {'romanc': 'romance', 'rom': 'romance', 'thril': 'thriller', 'n
                 'sciencefiction': 'sciencefict', 'act': 'action', 'famili': 'f
                 'documentari': 'documentary', 'anim': 'animation', 'crim': 'cr
                 'adventur': 'adventure', 'comedi': 'comedy', 'histori':'histor

# Merges similar genres
genre_df['genres_tag'] = genre_df['genres_tag'].apply(lambda x: [merged_genres.
# Creates set of unique genres
unique_genres = set(g for genres in genre_df['genres_tag'] for g in genres)

# Creates matrix with movie ID as row index and genre as column index
genre_matrix = pd.DataFrame(index=genre_df['id'], columns=list(unique_genres))

# Fills matrix with binary values based on the genres of each movie
for i, row in genre_df.iterrows():
    for genre in row['genres_tag']:
        genre_matrix.loc[row['id'], genre] = 1

# Fills NaN values with 0 and Displays
genre_matrix.fillna(0, inplace=True)
genre_matrix.head(3)
```

```
/var/folders/c6/c3dgh49x4wx_n7ctzrv_6nzh0000gn/T/ipykernel_3074/3216987734.py:
11: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  genre_df['genres_tag'] = genre_df['genres_tag'].apply(lambda x: [merged_genr
es.get(genre, genre) for genre in x.split()])
```

Out[28]:

	romance	action	horror	adventure	war	comedy	foreign	mystery	music	document
id										
19995	0	1	0	1	0	0	0	0	0	
285	0	1	0	1	0	0	0	0	0	
206647	0	1	0	1	0	0	0	0	0	

6. Vectorization and Similarity Measurement

This section of code is our movie recommender. System has two possible paths: "Path 1" is when the user inputs a movie title whereas "Path 2" is triggered when user inputs a name of a person (actress, director, etc.)

Path 1: When the user inputs a movie name, the system filter the movie by genre first. It looks for other movies that match at least 70% of the genres of the other movies. After that, we manually filtered the ouput to only show results above 85% genre-matching, which essentially filters the output by half. Once the genre portion is done, we compute TdifVectorization along with Cosine similarity on the "tag" of filtered movies ("tag" is a combination of overview, keywords, production company, and other metadata). Lastly, the function adds a "runtime" filter, in the sense that we don't want to suggest movies that are either too short or too long compared to the inputted movie. The system then provides the recommendations.

Path 2: If the user inputs a person's name, we assume that the person more concerned about seeing their favorite actress, actor, or director, and runtime nor genre necessarily matter as much. Therefore, the system first computes TdifVectorization along with Cosine similarity on the "tag" of the movies containing the person. Second, the system will calculate and sort for vote average, which represents the overall "liking" or "rating" of the movie. Finally, the system shows the recommendations.

In [29]:
```
# Movie Recommender Function:

# Creates TfidfVectorizer object
vectorizer = TfidfVectorizer()

def get_movie_recommendation(movie_input, top_n=5):

    # PATH 1
    if movie_input in df_ready_to_vector['title'].values:
        # Gets id and then genres
        movie_id = df_ready_to_vector[df_ready_to_vector['title'] == movie_inpu
        movie_genres = genre_matrix.loc[movie_id]

        # Gets % of Genre Matching, then Computes Scores
        genre_match_percentages = (genre_matrix == movie_genres).sum(axis=1) /
        score = 100 * (1 - (len(movie_genres) - 1) / (genre_match_percentages.m

        # Selects movies with Genre Matching Percentage > 70% (our decision fac
        recommended_movies = genre_match_percentages[genre_match_percentages >
        sorted_recommendations = recommended_movies.sort_values(ascending=False

        # Gets all recommended movies and Creates in dataframe
        recommended_movies_based_on_genres = []
        for movie_id, score in sorted_recommendations.items():
            score_percentage = int(score * 100)
            if score_percentage >= 85:  #85% eliminates about half of the outpu
                recommended_movies_based_on_genres.append({'id': movie_id, 'sco
            else:
                continue

        recommended_movies_based_on_genres = pd.DataFrame(recommended_movies_ba

        # Saves output to recommended_movies_df
        recommended_movies_df = recommended_movies_based_on_genres

        # Merges two dataframes into one
        genre_filtered_output = pd.merge(recommended_movies_df[['id', 'score_pe

        # Create Bag of Words then computes pairwise cosine similarities based
        tag_bow = vectorizer.fit_transform(genre_filtered_output['tag'] + ' ' +
        bow_features = tag_bow.toarray()
        movie_similarities = cosine_similarity(bow_features, bow_features)

        # Calculates original movie's runtime range
        original_movie_title = movie_input
        original_movie_runtime = genre_filtered_output.loc[genre_filtered_outpu
        runtime_range = (original_movie_runtime - 30, original_movie_runtime +

        # Applies runtime filter and Recommend movies
        if original_movie_title in genre_filtered_output['title'].values:
            # Gets the index of the movie in the feature matrix, computes Cosir
            movie_index = genre_filtered_output.index[genre_filtered_output['ti
            movie_similarities_matrix = cosine_similarity(bow_features[movie_ir
            movie_indices = np.argsort(movie_similarities_matrix.squeeze())[::-
            print(f"\nRecommended movies within 30 minutes of {original_movie_t
            for i in movie_indices:
                movie_title = genre_filtered_output.iloc[i].title
                movie_runtime = genre_filtered_output.iloc[i].runtime
                print(f"{movie_title} (Runtime: {movie_runtime} minutes)")
        else:
            print("Movie input is not in genre_filtered_output dataframe")

    # PATH 2
    else: # If user inserts a person's name that is not a movie, this code (Pat
        tag_bow = vectorizer.fit_transform(df_ready_to_vector['tag'])
        ppl_bow = vectorizer.fit_transform(df_ready_to_vector['ppl_tag'])
        bow_features = np.hstack((tag_bow.toarray(), ppl_bow.toarray()))

        # Computes pairwise cosine similarities between movies
        movie_similarities = cosine_similarity(bow_features, bow_features)

        # Filters by actor/actress/director name, gets indices, computes cosine
        movies_by_cast = df_ready_to_vector[df_ready_to_vector['ppl_tag'].apply
        movie_indices = movies_by_cast.index
        distance_cast = cosine_similarity(bow_features[movie_indices], bow_feat
        similarity_scores = distance_cast.sum(axis=0)
        movie_indices = np.argsort(similarity_scores.squeeze())[::-1][:top_n]
        movie_indices = sorted(movie_indices, key=lambda i: df_ready_to_vector.

        print(f"Recommended movies based on {movie_input}:")
        for i in movie_indices:
            movie_title = df_ready_to_vector.loc[i, 'title']
            movie_vote_average = df_ready_to_vector.loc[i, 'vote_average']
            print(f"{movie_title} (Vote Average: {movie_vote_average})")
```

7. Recommendations

This section highlights our movie recommendations.

In [30]:
```
get_movie_recommendation('Frozen', top_n=5)
```

```
Recommended movies within 30 minutes of Frozen (Runtime: 102.0 minutes)
The Snow Queen (Runtime: 76.0 minutes)
Aladdin (Runtime: 90.0 minutes)
Delgo (Runtime: 94.0 minutes)
Snow White and the Seven Dwarfs (Runtime: 83.0 minutes)
Brave (Runtime: 93.0 minutes)
```

In [31]:
```
get_movie_recommendation('The Dark Knight', top_n=5)
```

```
Recommended movies within 30 minutes of The Dark Knight (Runtime: 152.0 minute
s)
The Dark Knight Rises (Runtime: 165.0 minutes)
Batman Begins (Runtime: 140.0 minutes)
Batman Returns (Runtime: 126.0 minutes)
Batman: The Dark Knight Returns, Part 2 (Runtime: 78.0 minutes)
Batman Forever (Runtime: 121.0 minutes)
```

In [32]:
```
get_movie_recommendation('The Shawshank Redemption', top_n=5)
```

```
Recommended movies within 30 minutes of The Shawshank Redemption (Runtime: 14
2.0 minutes)
Prison (Runtime: 102.0 minutes)
Civil Brand (Runtime: 95.0 minutes)
Penitentiary (Runtime: 99.0 minutes)
The Longest Yard (Runtime: 113.0 minutes)
Mean Machine (Runtime: 99.0 minutes)
```

In [33]:
```
get_movie_recommendation('Christopher Nolan', top_n=5)
```

```
Recommended movies based on Christopher Nolan:
The Dark Knight (Vote Average: 8.2)
Interstellar (Vote Average: 8.1)
The Prestige (Vote Average: 8.0)
The Dark Knight Rises (Vote Average: 7.6)
Batman Begins (Vote Average: 7.5)
```

8. Different Vectorizations Comparison

This section of code compares the movie recommendation result using a combination of different vectorizer (CountVectorizer, TfidfVectorizer, HashingVectorizer) and similarity measurement (cosine_similarity, euclidean_distances).

In [34]:
```
def compare_get_movie_recommendation(movie_input, top_n=5):
    if movie_input in df_ready_to_vector['title'].values:
        # Gets id and then genres
        movie_id = df_ready_to_vector[df_ready_to_vector['title'] == movie_inpu
        movie_genres = genre_matrix.loc[movie_id]

        # Gets % of Genre Matching, then Computes Scores
        genre_match_percentages = (genre_matrix == movie_genres).sum(axis=1) /
        score = 100 * (1 - (len(movie_genres) - 1) / (genre_match_percentages.m

        # Selects movies with Genre Matching Percentage > 70% (our decision fac
        recommended_movies = genre_match_percentages[genre_match_percentages >
        sorted_recommendations = recommended_movies.sort_values(ascending=False

        # Get all recommended movies and Creates dataframe
        recommended_movies_based_on_genres = []
        for movie_id, score in sorted_recommendations.items():
            score_percentage = int(score * 100)
            if score_percentage >= 85:  #85% eliminates about half of the outpu
                recommended_movies_based_on_genres.append({'id': movie_id, 'sco
            else:
                continue

        recommended_movies_based_on_genres = pd.DataFrame(recommended_movies_ba

        # Saves the output to recommended_movies_df
        recommended_movies_df = recommended_movies_based_on_genres

        # Merges two dataframes into one
        genre_filtered_output = pd.merge(recommended_movies_df[['id', 'score_pe

        # Defines list of vectorizers and list of distance metrics
        vectorizers = [CountVectorizer(), TfidfVectorizer(), HashingVectorizer(
        distance_metrics = [cosine_similarity, euclidean_distances]
        # Iterates through the combinations of vectorizers and distance metrics
        for vectorizer in vectorizers:
            for distance_metric in distance_metrics:
                tag_bow = vectorizer.fit_transform(genre_filtered_output['tag']
                bow_features = tag_bow.toarray()

                # Computes pairwise similarities between movies
                movie_similarities = distance_metric(bow_features, bow_features

                # Applies runtime filter and Recommend movies
                if movie_input in genre_filtered_output['title'].values:
                    # Gets the index of the movie in the feature matrix, comput
                    movie_index = genre_filtered_output.index[genre_filtered_ou
                    movie_similarities_matrix = distance_metric(bow_features[mo
                    movie_indices = np.argsort(movie_similarities_matrix.squeez

                    print(f"Recommended movies based on {movie_input} using {ve
                    for i in movie_indices:
                        movie_title = genre_filtered_output.iloc[i]['title']
                        similarity_score = round(movie_similarities_matrix[0][i
                        print(f"{movie_title}")

                    print()
```

In [35]:
```
compare_get_movie_recommendation('Frozen', top_n=5)
```

```
Recommended movies based on Frozen using CountVectorizer and cosine_similarit
y:
Aladdin
Delgo
Spirit: Stallion of the Cimarron
The Princess and the Frog
Mulan

Recommended movies based on Frozen using CountVectorizer and euclidean_distanc
es:
Star Wars: Clone Wars: Volume 1
Alpha and Omega
Henry & Me
Dear Frankie
Madagascar 3: Europe's Most Wanted

Recommended movies based on Frozen using TfidfVectorizer and cosine_similarit
y:
The Snow Queen
Aladdin
Delgo
Snow White and the Seven Dwarfs
Brave

Recommended movies based on Frozen using TfidfVectorizer and euclidean_distanc
es:
The Looking Glass
Harrison Montgomery
The Algerian
Sardaarji
Sparkler

Recommended movies based on Frozen using HashingVectorizer and cosine_similari
ty:
Aladdin
Delgo
Spirit: Stallion of the Cimarron
The Princess and the Frog
Mulan

Recommended movies based on Frozen using HashingVectorizer and euclidean_dista
nces:
Harrison Montgomery
Lisa Picard Is Famous
The Algerian
The Looking Glass
Sardaarji
```

In [36]:
```
compare_get_movie_recommendation('The Dark Knight', top_n=5)
```

```
Recommended movies based on The Dark Knight using CountVectorizer and cosine_s
imilarity:
The Dark Knight Rises
Batman Begins
Batman Returns
Batman: The Dark Knight Returns, Part 2
Batman

Recommended movies based on The Dark Knight using CountVectorizer and euclidea
n_distances:
Fight Valley
Gladiator
Thank You for Smoking
Tae Guk Gi: The Brotherhood of War
Pocketful of Miracles

Recommended movies based on The Dark Knight using TfidfVectorizer and cosine_s
imilarity:
The Dark Knight Rises
Batman Begins
Batman Returns
Batman: The Dark Knight Returns, Part 2
Batman Forever

Recommended movies based on The Dark Knight using TfidfVectorizer and euclidea
n_distances:
Crowsnest
The Book of Mormon Movie, Volume 1: The Journey
The Outrageous Sophie Tucker
Childless
The Looking Glass

Recommended movies based on The Dark Knight using HashingVectorizer and cosine
_similarity:
The Dark Knight Rises
Batman Begins
Batman Returns
Batman: The Dark Knight Returns, Part 2
Batman

Recommended movies based on The Dark Knight using HashingVectorizer and euclid
ean_distances:
Cotton Comes to Harlem
Of Gods and Men
Sugar Hill
The Ballad of Jack and Rose
The Book of Mormon Movie, Volume 1: The Journey
```

In [37]:
```
compare_get_movie_recommendation('The Shawshank Redemption', top_n=5)
```

```
Recommended movies based on The Shawshank Redemption using CountVectorizer and
cosine_similarity:
Civil Brand
Prison
Penitentiary
Mean Machine
The Longest Yard

Recommended movies based on The Shawshank Redemption using CountVectorizer and
euclidean_distances:
Solomon and Sheba
Tadpole
Semi-Pro
Slacker Uprising
The Midnight Meat Train

Recommended movies based on The Shawshank Redemption using TfidfVectorizer and
cosine_similarity:
Prison
Civil Brand
Penitentiary
The Longest Yard
Mean Machine

Recommended movies based on The Shawshank Redemption using TfidfVectorizer and
euclidean_distances:
Sisters in Law
Baggage Claim
The FP
Hostel: Part II
Dream with the Fishes

Recommended movies based on The Shawshank Redemption using HashingVectorizer a
nd cosine_similarity:
Prison
Civil Brand
Penitentiary
The Last Station
The Longest Yard

Recommended movies based on The Shawshank Redemption using HashingVectorizer a
nd euclidean_distances:
Dwegons
Private Benjamin
The Ten
Devil's Due
Grand Theft Parsons
```

```
In [ ]:
```