Movie Recomender by Nicholas Felcher

In this project, I used CountVectorizer, TfidVectorizer, stopwords, CosineSimilarity, and nltk stemming. There is a LOT of data cleaning for the data I used, but I ended up with a pretty accurate recommender

GitHub Link: https://github.com/NicholasFelcher/CISB63-Midterm (https://github.com/NicholasFelcher/CISB63-Midterm)

Data from: https://www.themoviedb.org/?language=en-US (https://www.themoviedb.org/?language=en-US)

In [244]:
```
import requests
import pandas as pd
import sys
import numpy as np
import nltk
from nltk.tokenize import word_tokenize
```

Gathering & Exploring Data

I will be retrieving my data using TMDB's api. Here i can get the most popular movies and their features, like rating, release date, genre, etc.

In [33]:
```
#my API key (keep confidential please)
api_key = "022b3ce5e0177af0f6322607c3c26cf3"
```

In [319]:
```
results = []
#get result of all movies for each year
for year in [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021,
    page_number = 1
    response =  requests.get('https://api.themoviedb.org/3/discover/movi
    total_pages = response.json()['total_pages']
    results.extend(response.json()['results'])
    while page_number < total_pages:
        page_number += 1
        response = requests.get('https://api.themoviedb.org/3/discover/m
        results.extend(response.json()['results'])
```

In [135]:
```
#check the first entry
results[0]
```

Out[135]:
```
{'adult': False,
 'backdrop_path': '/sWulI556AJZ1DAQnxyAvnAAY6nd.jpg',
 'genre_ids': [16, 10751],
 'id': 73723,
 'original_language': 'en',
 'original_title': 'The Lorax',
 'overview': 'A 12-year-old boy searches for the one thing that will en
able him to win the affection of the girl of his dreams. To find it he
must discover the story of the Lorax, the grumpy yet charming creature
who fights to protect his world.',
 'popularity': 116.06,
 'poster_path': '/tePFnZFw5JvjwjQjaKkqDPNMLPU.jpg',
 'release_date': '2012-03-01',
 'title': 'The Lorax',
 'video': False,
 'vote_average': 6.5,
 'vote_count': 3230}
```

In [142]:
```
#explore the data
df= pd.DataFrame(results)
df
```

Out[142]:

```
                           poster_path  release_date                title  video  vote_average  vote_count

FnZFw5JvjwjQjaKkqDPNMLPU.jpg    2012-03-01            The Lorax  False           6.5        3230

lX2wcKCBAr24UyPD7xwmjaTn.jpg    2012-04-25                  The  False           7.7       29257
                                                       Avengers

nZCtHJyDLncBUarfM5h5mrppx.jpg   2012-05-30           Prometheus  False           6.5       11323

                                                            The
QNARYyERqRAq1p1c8xgePp4.jpg     2012-06-23              Amazing  False           6.7       16414
                                                     Spider-Man

DtMWpL0sYSFK0R6EZate2Ux.jpg     2012-06-21                Brave  False           7.0       12628

                           ...           ...                  ...    ...           ...         ...

v2BAjY0nBhXNHdummHczDnI.jpg     2023-07-19           Navigators  False           0.0           0

25UFMzouzjz9SHohpP0NMpoJ.jpg    2023-06-30           To Nowhere  False           0.0           0

2pXyZsW8t43Y5dKcpLyH2XNx.jpg    2023-08-25            The Grand  False           0.0           0
                                                         Bolero

                                                         What I
Pml7ihVOMKvVOspu0tnQ3Lqz.jpg    2023-06-14          Want You to  False           0.0           0
                                                           Know

                          None  2023-09-29            Beware of  False           0.0           0
                                                       Paul Pry
```

In [143]:

```
#remove unnecessary columns
df = pd.DataFrame(results, columns=['genre_ids', 'id', 'original_languag
       'overview', 'popularity', 'release_date', 'title', 'vote_average'
df
```

Out[143]:

	genre_ids	id	original_language	overview	popularity	release_date	
0	[16, 10751]	73723	en	A 12-year-old boy searches for the one thing t...	116.060	2012-03-01	The
1	[878, 28, 12]	24428	en	When an unexpected enemy emerges and threatens...	97.981	2012-04-25	Ave
2	[878, 12, 9648]	70981	en	A team of explorers discover a clue to the ori...	86.521	2012-05-30	Prome
3	[28, 12, 14]	1930	en	Peter Parker is an outcast high schooler aband...	82.064	2012-06-23	Am Spide
4	[16, 12, 35, 10751, 28, 14]	62177	en	Brave is set in the mystical Scottish Highland...	64.923	2012-06-21	
...	...	...	...	...	...	...	
23434	[99, 36]	940525	en	December 1919. The American government deports...	1.033	2023-07-19	Navi
23435	[18]	863800	en	Two self-destructive teenage friends embark on...	0.600	2023-06-30	To No
23436	[53, 18, 10749]	795254	en	During the covid-19 lockdown in Italy, Roxanne...	0.646	2023-08-25	The (E
23437	[99]	1197261	en	What I Want You To Know is a gripping, intimat...	0.600	2023-06-14	V Want
23438	[53]	1196892	en	Two Chapter Movie Showing The Terror of Paul P...	1.400	2023-09-29	Bew Pa

23439 rows × 9 columns

In [145]:
```
df.to_csv('2012-2023.csv')
```

Using IMBD's formula for weighted movie ratings, I created a weighted rating system.

Weighted Rating = ((v/v+m).R)+((m/v+m).C)

v = # of votes

m = minimum # of votes required to be charted

R = average rating of movie

C = mean vote across all movies

In [147]:
```
#Create variables and method for adding weighted rating to the dataframe
vote_counts = df[df['vote_count'].notnull()]['vote_count'].astype('int')
vote_averages = df[df['vote_average'].notnull()]['vote_average'].astype(
C = vote_averages.mean()

m = vote_counts.quantile(0.95)

def weighted_rating(x):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)
```

In [148]:
```
#Add weighted rating to the dataframe
df['wr'] = df.apply(weighted_rating, axis=1)
```

In [149]:
```
#Check dataframe
df.head()
```

Out[149]:

	genre_ids	id	original_language	overview	popularity	release_date	title
0	[16, 10751]	73723	en	A 12-year-old boy searches for the one thing t...	116.060	2012-03-01	The Lorax
1	[878, 28, 12]	24428	en	When an unexpected enemy emerges and threatens...	97.981	2012-04-25	The Avengers
2	[878, 12, 9648]	70981	en	A team of explorers discover a clue to the ori...	86.521	2012-05-30	Prometheus
3	[28, 12, 14]	1930	en	Peter Parker is an outcast high schooler aband...	82.064	2012-06-23	The Amazing Spider-Man
4	[16, 12, 35, 10751, 28, 14]	62177	en	Brave is set in the mystical Scottish Highland...	64.923	2012-06-21	Brave

I originally wanted to use weighted rating in my recommendation model, but I had to drop it due to time

In [150]:
```
#Sort by weighted rating
df.sort_values('wr', ascending=False).head()
```

Out[150]:

	genre_ids	id	original_language	overview	popularity	release_date	ti
5745	[12, 18, 878]	157336	en	The adventures of a group of explorers who mak...	142.249	2014-11-05	Interstel
15341	[12, 28, 878]	299536	en	As the Avengers and their allies have continue...	168.600	2018-04-25	Avenge Infir W
17109	[12, 878, 28]	299534	en	After the devastating events of Avengers: Infi...	106.047	2019-04-24	Avenge Endgar
15345	[28, 12, 16, 878]	324857	en	Struggling to find his place in the world whil...	94.947	2018-12-06	Spid Man: Ir t Spid Ver
5769	[18, 10402]	244786	en	Under the direction of a ruthless instructor, ...	57.487	2014-10-10	Whipla

In [177]:
```
#Put all movie id's in a list to retrieve more info via api calls (see be
ids = df['id'].tolist()
```

Retrieving cast members and credits to the database to improve recommendations

In [181]:

```
#adding cast and credits to the database
#CAUTION - TAKES A LONG TIME: ~ 45 MINS ON MY MACHINE
progress = 0
done_items = 0
credits = []

def get_credit(id):
    url = f"https://api.themoviedb.org/3/movie/{id}/credits?language=en-U

    headers = {
        "accept": "application/json",
        "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIwMjJiM2N
    }
    response = requests.get(url, headers=headers)
    #if the response isn't 200, it's an error and skip it
    if response.status_code != 200:
        print ('error')
        return []
    response = response.json()
    #check if any errors are in the request
    if 'errors' in response.keys():
        print('api error')
        return credits
    credits.append(response)

for item in ids:
    credits.append(get_credit(item))
    if done_items == 100:
        print('api in progress')
    done_items += 1
```

```
api in progress
```

In [185]:
```
#Look at the first item
credits[0]
```

```
   'gender': 1,
   'id': 71727,
   'known_for_department': 'Acting',
   'name': 'Betty White',
   'original_name': 'Betty White',
   'popularity': 7.677,
   'profile_path': '/eYDjR4ajOkzYDyEIMzOoSJz8za2.jpg',
   'cast_id': 15,
   'character': 'Grammy Norma (voice)',
   'credit_id': '52fe48a9c3a368484e104603',
   'order': 6},
  {'adult': False,
   'gender': 1,
   'id': 476163,
   'known_for_department': 'Acting',
   'name': 'Nasim Pedrad',
   'original_name': 'Nasim Pedrad',
   'popularity': 8.624,
   'profile_path': '/43rcDWcZHQXhC9COWQKdkLMufRj.jpg',
   'cast_id': 17,
```

In [323]:
```
#convert to dataframe object
credits2 = []
for i in credits:
    if i != None:
        credits2.append(i)
cast_df = pd.DataFrame(credits2)
cast_df.head()
```

Out[323]:

	id	cast	crew
0	73723	[{'adult': False, 'gender': 2, 'id': 518, 'kno...	[{'adult': False, 'gender': 2, 'id': 5720, 'kn...
1	24428	[{'adult': False, 'gender': 2, 'id': 3223, 'kn...	[{'adult': False, 'gender': 2, 'id': 37, 'know...
2	70981	[{'adult': False, 'gender': 1, 'id': 87722, 'k...	[{'adult': False, 'gender': 1, 'id': 2952, 'kn...
3	1930	[{'adult': False, 'gender': 2, 'id': 37625, 'k...	[{'adult': False, 'gender': 1, 'id': 6410, 'kn...
4	62177	[{'adult': False, 'gender': 1, 'id': 9015, 'kn...	[{'adult': False, 'gender': 2, 'id': 7, 'known...

In [198]:
```
#save as csv
cast_df.to_csv('cast.csv')
```

Retrieving movie keywords to improve recommendations

In [187]:
```
#CAUTION - TAKES A LONG TIME: ~ 45 MINS ON MY MACHINE
total_len = len(ids)
progress = 0
done_items = 0
keywords = []

def get_keywords(id):
    url = f"https://api.themoviedb.org/3/movie/{id}/keywords"

    headers = {
        "accept": "application/json",
        "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIwMjJiM2
    }
    response = requests.get(url, headers=headers)
    if response.status_code != 200:
        print ('error')
        return []
    response = response.json()
    if 'errors' in response.keys():
        print('api error !!!')
        return keywords
    keywords.append(response)

for item in ids:
    keywords.append(get_keywords(item))
```

In [193]:
```
#Look at third item
keywords[2]
```

Out[193]:
```
{'id': 24428,
 'keywords': [{'id': 242, 'name': 'new york city'},
  {'id': 5539, 'name': 'shield'},
  {'id': 9715, 'name': 'superhero'},
  {'id': 9717, 'name': 'based on comic'},
  {'id': 14909, 'name': 'alien invasion'},
  {'id': 155030, 'name': 'superhero team'},
  {'id': 179430, 'name': 'aftercreditsstinger'},
  {'id': 179431, 'name': 'duringcreditsstinger'},
  {'id': 180547, 'name': 'marvel cinematic universe (mcu)'}]}
```

In [324]:
```
#convert to dataframe object
keywords2 = []
for i in keywords:
    if i != None:
        keywords2.append(i)
keywords_df = pd.DataFrame(keywords2)
keywords_df.head()
```

Out[324]:

	id	keywords
0	73723	[{'id': 3352, 'name': 'tree'}, {'id': 5308, 'n...
1	24428	[{'id': 242, 'name': 'new york city'}, {'id': ...
2	70981	[{'id': 803, 'name': 'android'}, {'id': 9882, ...
3	1930	[{'id': 697, 'name': 'loss of loved one'}, {'i...
4	62177	[{'id': 388, 'name': 'scotland'}, {'id': 526, ...

In [326]:
```
#save as csv
keywords_df.to_csv('keywords.csv')
```

combining the dataframes

In [328]:
```
df_merged = df.merge(keywords_df, on='id')
df_merged = df_merged.merge(cast_df, on='id')
```

In [284]:
```
df_merged.head()
```

Out[284]:

	index	genre_ids	id	original_language	overview	popularity	release_date	
0	0	[16, 10751]	73723	en	A 12-year-old boy searches for the one thing t...	116.060	2012-03-01	The
1	1	[878, 28, 12]	24428	en	When an unexpected enemy emerges and threatens...	97.981	2012-04-25	Ave
2	2	[878, 12, 9648]	70981	en	A team of explorers discover a clue to the ori...	86.521	2012-05-30	Prome
3	3	[28, 12, 14]	1930	en	Peter Parker is an outcast high schooler aband...	82.064	2012-06-23	An Spide
4	4	[16, 12, 35, 10751, 28, 14]	62177	en	Brave is set in the mystical Scottish Highland...	64.923	2012-06-21	

In [285]:
```
df_merged.drop(['index'],axis = 1)
```

Out[285]:

overview	popularity	release_date	title	vote_average	vote_count	wr	keyw
A 12-year-old boy searches for the one thing t...	116.060	2012-03-01	The Lorax	6.5	3230	5.740740	3 'na 't {'id': 5
When an unexpected enemy emerges and threatens...	97.981	2012-04-25	The Avengers	7.7	29257	7.536319	[{'id': 'na 'new city'},
A team of explorers discover a clue to the ori...	86.521	2012-05-30	Prometheus	6.5	11323	6.227479	[{'id': 'na 'andr {'id': 9
Peter Parker is an outcast high schooler aband...	82.064	2012-06-23	The Amazing Spider-Man	6.7	16414	6.491106	[{'id': 'na 'lo l one'}
Brave is set in the mystical Scottish Highland...	64.923	2012-06-21	Brave	7.0	12628	6.706303	[{'id': 'na 'scotla {'id':
...	...	...	...	...	...	...	
December 1919. The American government deports...	1.033	2023-07-19	Navigators	0.0	0	3.856009	
Two self-destructive teenage friends embark on...	0.600	2023-06-30	To Nowhere	0.0	0	3.856009	
During the covid-19 lockdown in Italy, Roxanne...	0.646	2023-08-25	The Grand Bolero	0.0	0	3.856009	
What I Want You To Know is a gripping, intimat...	0.600	2023-06-14	What I Want You to Know	0.0	0	3.856009	

overview	popularity	release_date	title	vote_average	vote_count	wr	keyw
Two Chapter Movie Showing The Terror of Paul P...	1.400	2023-09-29	Beware of Paul Pry	0.0	0	3.856009	

In [214]:
```
df_merged.to_csv('merged.csv')
```

Basic description based recommendation system

Cosine Similiarity

In [333]:
```
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer, CountVector
```

In [334]:
```
#create TfidfVectorizer object
tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=1, stop_w
tfidf_matrix = tf.fit_transform(df['overview'])
```

In [335]:
```
#create cosine similarity object
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

In [332]:
```
#check first element
cosine_sim[0]
```

Out[332]:
```
array([1.        , 0.00310972, 0.00904846, ..., 0.00486915, 0.        ,
       0.        ])
```

In [159]:
```
#create a df full of only titles orderd properly
df = df.reset_index()
titles = df['title']
indices = pd.Series(df.index, index=df['title'])
```

In [160]:
```
#method to get most similar movies based on cosine similarity scores
def get_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    #sortfrom top to bottom
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:31]
    #provide a list of indicies
    movie_indices = [i[0] for i in sim_scores]
    return titles.iloc[movie_indices]
```

It's not perfect, but we get some super hero and action movies here

In [163]:
```
get_recommendations('The Avengers').head(10)
```

Out[163]:
```
4127               The Dawn of Aquarius
17114          Spider-Man: Far From Home
13432        Kingsman: The Golden Circle
11549                          Allegiant
10500      Billy Fury: The Sound of Fury
17226        Escape Plan: The Extractors
10129                         One by One
7372                     Against The Jab
8729             Avengers: Age of Ultron
11751                    Into the Forest
Name: title, dtype: object
```

In [165]:
```
get_recommendations('Interstellar').head(10)
```

Out[165]:
```
15721                             The Beyond
15878                                  Astro
14271                 The Coming Convergence
5798                     Dumb and Dumber To
1718             The Lebanese Rocket Society
21775                       Last Exit: Space
13438                 Star Wars: The Last Jedi
9026                                400 Days
13024                                   Tomb
9307      Steve McQueen: The Man & Le Mans
Name: title, dtype: object
```

In [223]:
```
a = df_merged['crew'][0]
```

In [224]:
```
a[3]
```

Out[224]:
```
{'adult': False,
 'gender': 2,
 'id': 8063,
 'known_for_department': 'Editing',
 'name': 'Ken Schretzmann',
 'original_name': 'Ken Schretzmann',
 'popularity': 0.7,
 'profile_path': '/oI4qy2nUz8Bjd5LQp9NuHEoYz87.jpg',
 'credit_id': '563b5ec89251414cc90043aa',
 'department': 'Editing',
 'job': 'Editor'}
```

Cast & keywords based system

Preprocess the data for crew, cast, and keywords

In [286]:
```
#get length of cast so I can remove the less import names from the model
df_merged['cast_size'] = df_merged['cast'].apply(lambda x: len(x))
df_merged['crew_size'] = df_merged['crew'].apply(lambda x: len(x))
#pick out the director from the crew
def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
#create director column
df_merged['director'] = df_merged['crew'].apply(get_director)
#cut the cast
df_merged['cast'] = df_merged['cast'].apply(lambda x: [i['name'] for i i
df_merged['cast'] = df_merged['cast'].apply(lambda x: x[:3] if len(x) >=
```

In [287]:
```
df_merged['director']
```

Out[287]:
```
0             Chris Renaud
1              Joss Whedon
2             Ridley Scott
3                Marc Webb
4           Brenda Chapman
                ...
23434        Noah Teichner
23435     Sian Astor-Lewis
23436     Gabriele Fabbro
23437       Catie Foertsch
23438                  NaN
Name: director, Length: 23439, dtype: object
```

In [288]:
```
#get rid of spacing in cast
df_merged['cast'] = df_merged['cast'].apply(lambda x: [str.lower(i.repla
#mention director 3 times to add to their weight in the model
df_merged['director'] = df_merged['director'].astype('str').apply(lambda
df_merged['director'] = df_merged['director'].apply(lambda x: [x, x, x])
```

In [289]:
```
#remove the ids and dictionary structure from keywords
def remove_keyword_ids(df):
    if df != None:
        words = []
        for i in df:
            for x in i:
                if x == 'name':
                    words.append(i[x])
        return words
df_merged['keywords'] = df_merged['keywords'].apply(remove_keyword_ids)
```

In [290]:
```
#check to see if they are removed correctly
df_merged['keywords']
```

Out[290]:
```
0        [tree, aftermath, family business , tragic vil...
1        [new york city, shield, superhero, based on co...
2        [android, space, alien, creature, spin off, cr...
3        [loss of loved one, experiment, vigilante, sup...
4        [scotland, rebel, bravery, kingdom, archer, wi...
                                ...                        
23434                                                   []
23435                                                   []
23436                                                   []
23437                                                   []
23438                                                   []
Name: keywords, Length: 23439, dtype: object
```

In [291]:
```
#prepare words for value_counts
key_freq = df_merged.apply(lambda x: pd.Series(x['keywords']),axis=1).st
key_freq.name = 'keyword'
```

```
C:\Users\nicho\AppData\Local\Temp\ipykernel_12132\4156778925.py:1: Futu
reWarning: The default dtype for empty Series will be 'object' instead
of 'float64' in a future version. Specify a dtype explicitly to silence
this warning.
  key_freq = df_merged.apply(lambda x: pd.Series(x['keywords']),axis=
1).stack().reset_index(level=1, drop=True)
```

In [292]:
```
#get frequency of keywords
key_freq = key_freq.value_counts()
key_freq[:5]
```

Out[292]:
```
woman director           1368
based on novel or book    486
murder                    428
biography                 402
based on true story       341
Name: keyword, dtype: int64
```

In [293]:
```
#remove keywords that only occur once
key_freq = key_freq[key_freq > 1]
```

In [294]:
```
from nltk.stem.porter import *
```

In [295]:
```
#create stemmer object
stemmer = PorterStemmer()
```

In [296]:
```
#get rid of the dictionary so it's just the words and there are no id's
def filter_keywords(x):
    words = []
    for i in x:
        if i in key_freq:
            words.append(i)
    return words
```

In [298]:
```
#convert keywords into stemmed version so they are the same
df_merged['keywords'] = df_merged['keywords'].apply(filter_keywords)
df_merged['keywords'] = df_merged['keywords'].apply(lambda x: [stemmer.st
df_merged['keywords'] = df_merged['keywords'].apply(lambda x: [str.lower
```

In [299]:
```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVector
```

In [310]:
```
#transform genre ids into words
def transform_genre(x):
    genres = {28:'Action',12:'Adventure',16:'Animation',35:'Comedy',80:'(
              18:'Drama', 10751:'Family', 14:'Fantasy',36:'Fantasy', 36:'H
              10402:'Music', 9648:'Mystery', 10749:'Romance', 878:'Scienc
              53:'Thriller', 10752:'War', 37:'Western'}
    list1 = []
    for i in x:
        list1.append(genres[i])
    return list1
```

In [311]:
```
df_merged['genre_ids'] = df_merged['genre_ids'].apply(transform_genre)
```

In [312]:
```
#check genre conversion
df_merged['genre_ids'].head()
```

Out[312]:
```
0                                  [Animation, Family]
1                    [Science Fiction, Action, Adventure]
2                   [Science Fiction, Adventure, Mystery]
3                           [Action, Adventure, Fantasy]
4    [Animation, Adventure, Comedy, Family, Action,...
Name: genre_ids, dtype: object
```

In [313]:
```
#create a column 'soup' that is the aggregate of all relevant data
df_merged['soup'] = df_merged['keywords'] + df_merged['cast'] + df_merge
df_merged['soup'] = df_merged['soup'].apply(lambda x: ' '.join(x))
```

Using cosine similarity on cast and keywords

Here, I create another recommendation system using the metadata from genre, cast, crew, and keywords

In [314]:
```
#use countvectorizer to transform words into frequency vectors
count = CountVectorizer(analyzer='word',ngram_range=(1, 2),min_df=1, sto
count_matrix = count.fit_transform(df_merged['soup'])
```

In [315]:
```
#use a cosine similarity model to determine the 'closeness' of the resul
cosine_sim = cosine_similarity(count_matrix, count_matrix)
```

In [336]:
```
#make sure everything is ordered correctly
df_merged = df_merged.reset_index()
titles = df_merged['title']
indices = pd.Series(df_merged.index, index=df_merged['title'])
```

In [317]:
```
#get movie recommendations
get_recommendations('The Avengers').head(10)
```

Out[317]:
```
8729                  Avengers: Age of Ultron
13419                         Justice League
17109                      Avengers: Endgame
5803        Captain America: The Winter Soldier
3039                    Much Ado About Nothing
22500         Ant-Man and the Wasp: Quantumania
11472               Captain America: Civil War
17113                          Captain Marvel
15363                           Black Panther
8745                                  Ant-Man
Name: title, dtype: object
```

In [318]:
```
get_recommendations('Interstellar').head(10)
```

Out[318]:
```
9315                         Quay
18711                       Tenet
9          The Dark Knight Rises
22485                Oppenheimer
13479                    Dunkirk
11483                    Arrival
20083                   Stowaway
13404                  Budhayaan
8739                 The Martian
3475               Supercollider
Name: title, dtype: object
```

In [322]:
```
get_recommendations('A Million Ways to Die in the West').head(10)
```

Out[322]:
```
8730                                  Ted 2
25                                      Ted
342                       Casa De Mi Padre
11476                                  Sing
15631                          Action Point
123                              Wanderlust
2953                  Pawn Shop Chronicles
21408    Paws of Fury: The Legend of Hank
22612                             Champions
22800                   Outlaw Johnny Black
Name: title, dtype: object
```

This system of recommending movies looks much more accurate and relevant.

I think it's pretty impressive that it was able to reccommend arrival and the martian on interstellar.

In [ ]:
```

```