```
import pandas as pd
import numpy as np
print("Import Successful!!")
```

```
Import Successful!!
```

```
df1 = pd.read_csv('Data-Asset/archive/tmdb_5000_credits.csv')
df2 = pd.read_csv('Data-Asset/archive/tmdb_5000_movies.csv')

print('Data Read Successful!!')
print('Data Shape : {},{}'.format(df1.shape,df2.shape))
```

```
Data Read Successful!!
Data Shape : (4803, 4),(4803, 20)
```

```
df1.columns = ['id','title_','cast','crew']
df2= df2.merge(df1,on='id')
```

We will compute pairwise similarity scores for all movies based on their plot descriptions and recommend movies based on that similarity score.

```
df2["overview"].head()
```

```
0    In the 22nd century, a paraplegic Marine is di...
1    Captain Barbossa, long believed to be dead, ha...
2    A cryptic message from Bond’s past sends him o...
3    Following the death of District Attorney Harve...
4    John Carter is a war-weary, former military ca...
Name: overview, dtype: object
```

Now if you are wondering what is term frequency , it is the relative frequency of a word in a document and is given as **(term instances/total instances).** Inverse Document Frequency is the relative count of documents containing the term is given as **log(number of documents/documents with term)** The overall importance of each word to the documents in which they appear is equal to **TF * IDF**

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(stop_words='english')
df2['overview'] = df2['overview'].fillna('')

tfidf_matrix = tfidf.fit_transform(df2['overview'])

print("Matrix Shape : {}".format(tfidf_matrix.shape))
```

```
Matrix Shape : (4803, 20978)
```

```
from sklearn.metrics.pairwise import linear_kernel

cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
print("Shape : {}".format((len(cosine_sim),len(cosine_sim))))
```

```
Shape : (4803, 4803)
```

```
indices = pd.Series(df2.index,
                    index = df2["title"]).drop_duplicates()
print(indices.head())
```

```
title
Avatar                                         0
Pirates of the Caribbean: At World's End       1
Spectre                                        2
The Dark Knight Rises                          3
John Carter                                    4
dtype: int64
```

```
def get_recommendations(title,
                        cosine_sim=cosine_sim):

    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key= lambda x: x[1], reverse=True)

    sim_scores = sim_scores[1:11]

    movie_indices = [i[0] for i in sim_scores]
    return df2['title'].iloc[movie_indices]
```

```
df2[['title',
     'director_',
     'genres']].loc[list(get_recommendations('The Dark Knight
Rises').index)]
```

```
                                       title          director_  \
65                          The Dark Knight  Christopher Nolan
299                          Batman Forever    Joel Schumacher
428                          Batman Returns         Tim Burton
1359                                Batman         Tim Burton
3854  Batman: The Dark Knight Returns, Part 2          Jay Oliva
119                           Batman Begins  Christopher Nolan
2507                              Slow Burn        Wayne Beach
9          Batman v Superman: Dawn of Justice        Zack Snyder
1181                                  JFK       Oliver Stone
210                          Batman & Robin    Joel Schumacher

                           genres
65          [drama, action, crime]
299       [action, crime, fantasy]
428              [action, fantasy]
1359             [fantasy, action]
3854           [action, animation]
119          [action, crime, drama]
2507       [mystery, crime, drama]
```

```
9     [action, adventure, fantasy]
1181    [drama, thriller, history]
210       [action, crime, fantasy]
```

```
df2[['title',
     'director_',
     'genres']].loc[list(get_recommendations('The Avengers').index)]
```

```
                                title         director_  \
7             Avengers: Age of Ultron       Joss Whedon
3144                          Plastic     Julian Gilbey
1715                          Timecop       Peter Hyams
4124               This Thing of Ours  Danny Provenzano
3311            Thank You for Smoking     Jason Reitman
3033                    The Corruptor       James Foley
588   Wall Street: Money Never Sleeps      Oliver Stone
2136       Team America: World Police       Trey Parker
1468                     The Fountain  Darren Aronofsky
1286                      Snowpiercer      Bong Joon-ho

                                    genres
7     [action, adventure, sciencefiction]
3144              [drama, action, comedy]
1715   [thriller, sciencefiction, action]
4124            [drama, action, thriller]
3311                      [comedy, drama]
3033             [action, crime, mystery]
588                       [drama, crime]
2136        [music, adventure, animation]
1468   [drama, adventure, sciencefiction]
1286      [action, sciencefiction, drama]
```

While our system has done a decent job of finding movies with similar plot descriptions, the quality of recommendations is not that great. "The Dark Knight Rises" returns all Batman movies while it is more likely that the people who liked that movie are more inclined to enjoy other Christopher Nolan movies. This is something that cannot be captured by the present system.

Using Cast, Crew & Keywords for better recommendations

```
from ast import literal_eval

features = ['cast','crew','keywords','genres']
for feature in features:
    df2[feature] = df2[feature].apply(literal_eval)

def get_director(x):

    for i in x:
        if i['job'] == 'Director':
```

```
            return i['name']
    return np.nan

def get_list(x):

    if isinstance(x, list):

        names = [i['name'] for i in x]

        if len(names) > 3:
            names = names[:3]
        return names

    return []

df2['director'] = df2['crew'].apply(get_director)

features = ['cast','keywords','genres']
for feature in features:
    df2[feature] = df2[feature].apply(get_list)

df2[['title','cast','director','keywords','genres']].head(3)
```

	title	cast	director	keywords	genres
0	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	James Cameron	[culture clash, future, space war]	[Action, Adventure, Fantasy]
1	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley]	Gore Verbinski	[ocean, drug abuse, exotic island]	[Adventure, Fantasy, Action]
2	Spectre	[Daniel Craig, Christoph Waltz, Léa Seydoux]	Sam Mendes	[spy, based on novel, secret agent]	[Action, Adventure, Crime]

```
def clean_data(x):

    if isinstance(x, list):
        return [str.lower(i.replace(" ","")) for i in x]
    else:

        if isinstance(x, str):
            return str.lower(x.replace(" ",""))
        else:
            return ""

features = ['cast','keywords','director','genres']
```

```
for feature in features:
    df2[feature] = df2[feature].apply(clean_data)
```

```
df2[['title','cast','director','keywords','genres']].head(3)
```

```
                                     title  \
0                                   Avatar
1  Pirates of the Caribbean: At World's End
2                                  Spectre

                                          cast       director  \
0  [samworthington, zoesaldana, sigourneyweaver]   jamescameron
1     [johnnydepp, orlandobloom, keiraknightley]  goreverbinski
2      [danielcraig, christophwaltz, léaseydoux]      sammendes

                          keywords                          genres
0  [cultureclash, future, spacewar]  [action, adventure, fantasy]
1  [ocean, drugabuse, exoticisland]  [adventure, fantasy, action]
2  [spy, basedonnovel, secretagent]    [action, adventure, crime]
```

We are now in a position to create our **metadata soup**, which is a string that contains all the metadata that we want to feed to our vectorizer (namely actors, director and keywords).

```
def create_soup(x):

    return ' '.join(x['keywords']) + ' ' + ' '.join(x['cast']) + ' ' +
x['director'] + ' ' + x['director'] + ' ' + x['director'] + ' ' + '
'.join(x['genres'])

df2['soup'] = df2.apply(create_soup,
                        axis = 1)
```

The next steps are the same as what we did with our plot description based recommender. One important difference is that we use the **CountVectorizer()** instead of TF-IDF. This is because we do not want to down-weight the presence of an actor/director if he or she has acted or directed in relatively more movies. It doesn't make much intuitive sense.

```
from sklearn.feature_extraction.text import CountVectorizer

count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(df2['soup'])

print("Count Matrix Shape : {}".format(count_matrix.shape))
```

```
Count Matrix Shape : (4803, 11520)
```

```
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
```

```
print("shape of cosine_sim matrix : {},
{}".format(len(cosine_sim2),len(cosine_sim2)))

shape of cosine_sim matrix : 4803,4803

df2 = df2.reset_index()
indices = pd.Series(df2.index,
                    index = df2['title'])

recommended_index = list(get_recommendations('The Dark Knight Rises',
                    cosine_sim2).index)

df2['director_'] = df2['crew'].apply(get_director)

df2[['title',
     'director_',
     'genres']].loc[recommended_index][:7]

                title          director_
genres
65    The Dark Knight  Christopher Nolan              [drama, action,
crime]
119     Batman Begins  Christopher Nolan              [action, crime,
drama]
1196     The Prestige  Christopher Nolan          [drama, mystery,
thriller]
95       Interstellar  Christopher Nolan  [adventure, drama,
sciencefiction]
1033         Insomnia  Christopher Nolan          [crime, mystery,
thriller]
96          Inception  Christopher Nolan  [action, thriller,
sciencefiction]
3573          Memento  Christopher Nolan                 [mystery,
thriller]

recommended_index2 = list(get_recommendations('The Godfather',
                                              cosine_sim2).index)

df2[['title',
     'director_',
     'genres']].loc[recommended_index2][:7]

                       title            director_
genres
867   The Godfather: Part III  Francis Ford Coppola  [crime, drama,
thriller]
2731   The Godfather: Part II  Francis Ford Coppola            [drama,
crime]
1525           Apocalypse Now  Francis Ford Coppola
[drama, war]
1018          The Cotton Club  Francis Ford Coppola     [music, drama,
crime]
```

1209 The Rainmaker Francis Ford Coppola [drama, crime, thriller]
3012 The Outsiders Francis Ford Coppola [crime, drama]
4209 The Conversation Francis Ford Coppola [crime, drama, mystery]