Machine Learning

__NOTE : This notebook is where the data was analyzed and the recommender system was built for the app. For App source code, click blue text below :__

> **Source Code**

Topic: Movie Recommendation System

Using recommendation systems may have been a thing of complexity and even luxury for companies in the past, but in the increasingly high-speed, high-tech world we live in today it has become a necessity to most. Recommender systems have revolutionized e-commerce, video/music streaming services, and even online dating. Corporations like Netflix, Amazon and Youtube, have vaulted themselves into being among the most valuable companies in the world due, in very large part, to the recommendation systems that consumers are so reliant on.

In this project, we will build a content-based movie recommendation system based on features such as genre, movie overview, and cast and crew, among others.

This dataset was generated from The Movie Database API, sourced from Kaggle and can be found here :

> **Kaggle Dataset**

The raw dataset contains 5000 movies, with release dates ranging from the year 1916 up until February 2017.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import ast
from sklearn.feature_extraction.text import CountVectorizer
import nltk
from nltk.stem.porter import PorterStemmer
from sklearn.metrics.pairwise import cosine_similarity
import pickle

import warnings
%load_ext watermark
```

```
In [2]: warnings.filterwarnings("ignore")
```

Datasets

```
In [3]: movies = pd.read_csv(r"C:\Users\ferna\OneDrive\Desktop\streamlit apps\app_4_Movie_RecSys\Datasets\tmdb_5000_movies.csv"
        credits = pd.read_csv(r"C:\Users\ferna\OneDrive\Desktop\streamlit apps\app_4_Movie_RecSys\Datasets\tmdb_5000_credits.cs
```

```
In [4]: movies.shape

Out[4]: (4803, 20)
```

```
In [5]: credits.shape

Out[5]: (4803, 4)
```

Merging Datasets

```
In [6]: movies = movies.merge(credits, on='title')
        movies.shape

Out[6]: (4809, 23)
```

```
In [7]: movies.head()
```

Out[7]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popular
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond’s past sends him o...	107.376
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	43.926

5 rows × 23 columns

Dataset Overview

```
In [8]: print('Number of records:',movies.shape[0])
        print(' — — — — -')
        print('Number of features:', movies.shape[1])
        print(' — — — — -')
        print(movies.info())
```

```
Number of records: 4809
 — — — — -
Number of features: 23
 — — — — -
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4809 entries, 0 to 4808
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4809 non-null   int64  
 1   genres                4809 non-null   object 
 2   homepage              1713 non-null   object 
 3   id                    4809 non-null   int64  
 4   keywords              4809 non-null   object 
 5   original_language     4809 non-null   object 
 6   original_title        4809 non-null   object 
 7   overview              4806 non-null   object 
 8   popularity            4809 non-null   float64
 9   production_companies  4809 non-null   object 
 10  production_countries  4809 non-null   object 
 11  release_date          4808 non-null   object 
 12  revenue               4809 non-null   int64  
 13  runtime               4807 non-null   float64
 14  spoken_languages      4809 non-null   object 
 15  status                4809 non-null   object 
 16  tagline               3965 non-null   object 
 17  title                 4809 non-null   object 
 18  vote_average          4809 non-null   float64
 19  vote_count            4809 non-null   int64  
 20  movie_id              4809 non-null   int64  
 21  cast                  4809 non-null   object 
 22  crew                  4809 non-null   object 
dtypes: float64(3), int64(5), object(15)
memory usage: 901.7+ KB
None
```

In [9]:
```
movies.describe().transpose()
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
budget	4809.0	2.902780e+07	4.070473e+07	0.0	780000.00000	1.500000e+07	4.000000e+07	3.800000e+08
id	4809.0	5.712057e+04	8.865337e+04	5.0	9012.00000	1.462400e+04	5.859500e+04	4.594880e+05
popularity	4809.0	2.149166e+01	3.180337e+01	0.0	4.66723	1.292159e+01	2.835053e+01	8.755813e+02
revenue	4809.0	8.227511e+07	1.628379e+08	0.0	0.00000	1.917000e+07	9.291317e+07	2.787965e+09
runtime	4807.0	1.068823e+02	2.260254e+01	0.0	94.00000	1.030000e+02	1.180000e+02	3.380000e+02
vote_average	4809.0	6.092514e+00	1.193989e+00	0.0	5.60000	6.200000e+00	6.800000e+00	1.000000e+01
vote_count	4809.0	6.903317e+02	1.234187e+03	0.0	54.00000	2.350000e+02	7.370000e+02	1.375200e+04
movie_id	4809.0	5.712057e+04	8.865337e+04	5.0	9012.00000	1.462400e+04	5.859500e+04	4.594880e+05

Truncated Dataframe

A content-based recommendation system like the one we're building requires features that will help us create tags to compare films with. Eg: movie budget is not important for a recommender system, because it is not a given that if a person likes Interstellar, that they will also like other high budget movies like Marvel movies.

COLUMNS TO BE KEPT

title
overview - for content based similarity
genre
keywords - basically tags to describe and recommend similar movies, this will be useful in creating our system.
production_companies - some companies stick to producing certain types of movies, like Pixar or Marvel Studios.
cast - we often recommend movies on the basis of actors
crew - we often recommend movies based on directors, among other crew members

In [10]:
```
movies = movies[['movie_id', 'title', 'overview','genres', 'keywords','production_companies', 'cast','crew']]
movies.head()
```

Out[10]:

	movie_id	title	overview	genres	keywords	production_companies	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"name": "Ingenious Film Partners", "id": 289...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond’s past sends him o...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 470, "name": "spy"}, {"id": 818, "name...	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	[{"id": 849, "name": "dc comics"}, {"id": 853,...	[{"name": "Legendary Pictures", "id": 923}, {"...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 818, "name": "based on novel"}, {"id":...	[{"name": "Walt Disney Pictures", "id": 2}]	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

In [11]:
```
movies.shape
```

Out[11]:
```
(4809, 8)
```

Preprocessing our data

Now we will preprocess our data by checking for null values as well as duplicated variables. We can also see the from the 'genres' column through the 'crew' column, the names of those features, which we need for creating tags, are tucked away inside lists of dictionaries. We will parse these columns to retrieve the names we are looking for.

In [12]:
```
movies.isnull().sum()
```

Out[12]:
```
movie_id                0
title                   0
overview                3
genres                  0
keywords                0
production_companies    0
cast                    0
crew                    0
dtype: int64
```

In [13]:
```
movies.dropna(inplace=True)
```

In [14]:
```
movies.duplicated().sum()
```

Out[14]:
```
0
```

In [15]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4806 entries, 0 to 4808
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   movie_id              4806 non-null   int64 
 1   title                 4806 non-null   object
 2   overview              4806 non-null   object
 3   genres                4806 non-null   object
 4   keywords              4806 non-null   object
 5   production_companies  4806 non-null   object
 6   cast                  4806 non-null   object
 7   crew                  4806 non-null   object
dtypes: int64(1), object(7)
memory usage: 337.9+ KB
```

Column Conversion

We will use the literal_eval function from the ast (Abstract Syntax Tree) library to create functions to parse through the necessary columns in order to retrieve the necessary attributes for our system.

> The **ast library** provides a way to parse and analyze the code written in Python. It can be used to transform code, check for errors, or extract information about the code.
>
> The **literal_eval function** is a function that evaluates a string containing a Python literal (e.g., a string, tuple, list, dictionary, number, or boolean value) and returns the corresponding Python object.

Genres and Keywords

An example of what genres look like ↓

In [16]:
```
movies['genres'][0]
```

Out[16]:
```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Sc
ience Fiction"}]'
```

Now to get the genre names and keywords ...

In [17]:
```
def convert(obj) :
    li = []
    for i in ast.literal_eval(obj) :
        li.append(i['name'])
    return li
```

In [18]:
```
movies['genres'] = movies['genres'].apply(convert)
movies['genres'][0:6]
```

Out[18]:
```
0    [Action, Adventure, Fantasy, Science Fiction]
1                      [Adventure, Fantasy, Action]
2                        [Action, Adventure, Crime]
3                  [Action, Crime, Drama, Thriller]
4              [Action, Adventure, Science Fiction]
5                      [Fantasy, Action, Adventure]
Name: genres, dtype: object
```

In [19]:
```
movies['keywords'] = movies['keywords'].apply(convert)
movies['keywords'][0:6]
```

```
Out[19]: 0    [culture clash, future, space war, space colon...
         1    [ocean, drug abuse, exotic island, east india ...
         2    [spy, based on novel, secret agent, sequel, mi...
         3    [dc comics, crime fighter, terrorist, secret i...
         4    [based on novel, mars, medallion, space travel...
         5    [dual identity, amnesia, sandstorm, love of on...
         Name: keywords, dtype: object
```

Here we can see that our dataframe is starting to look a little better ↓

```
In [20]: movies.head()
```

Out[20]:

	movie_id	title	overview	genres	keywords	production_companies	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[{"name": "Ingenious Film Partners", "id": 289...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[{"name": "Legendary Pictures", "id": 923}, {"...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[{"name": "Walt Disney Pictures", "id": 2}]	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

Production Companies

```
In [21]: movies['production_companies'][0]
```

```
Out[21]: '[{"name": "Ingenious Film Partners", "id": 289}, {"name": "Twentieth Century Fox Film Corporation", "id": 306}, {"nam
e": "Dune Entertainment", "id": 444}, {"name": "Lightstorm Entertainment", "id": 574}]'
```

```
In [22]: def convert_prod(obj) :
             li = []
             counter = 0
             for i in ast.literal_eval(obj) :
                 if counter < 4 :
                     li.append(i['name'])
                     counter += 1
             return li
```

```
In [23]: movies['production_companies'] = movies['production_companies'].apply(convert_prod)
         movies['production_companies'][0:6]
```

```
Out[23]: 0    [Ingenious Film Partners, Twentieth Century Fo...
         1    [Walt Disney Pictures, Jerry Bruckheimer Films...
         2                      [Columbia Pictures, Danjaq, B24]
         3    [Legendary Pictures, Warner Bros., DC Entertai...
         4                                [Walt Disney Pictures]
         5    [Columbia Pictures, Laura Ziskin Productions, ...
         Name: production_companies, dtype: object
```

Cast

```
In [24]: movies['cast'][0][:500]
```

```
Out[24]: '[{"cast_id": 242, "character": "Jake Sully", "credit_id": "5602a8a7c3a3685532001c9a", "gender": 2, "id": 65731, "nam
e": "Sam Worthington", "order": 0}, {"cast_id": 3, "character": "Neytiri", "credit_id": "52fe48009251416c750ac9cb", "ge
nder": 1, "id": 8691, "name": "Zoe Saldana", "order": 1}, {"cast_id": 25, "character": "Dr. Grace Augustine", "credit_i
d": "52fe48009251416c750aca39", "gender": 1, "id": 10205, "name": "Sigourney Weaver", "order": 2}, {"cast_id": 4, "char
acter": "Col. Quaritch", "c'
```

```
In [25]: def convert_cast(obj) :
             li = []
             counter = 0
             for i in ast.literal_eval(obj) :
                 if counter < 3 :
                     li.append(i['name'])
```

```
            counter += 1
    return li
```

In [26]:
```
movies['cast'] = movies['cast'].apply(convert_cast)
movies['cast'][0:6]
```

Out[26]:
```
0     [Sam Worthington, Zoe Saldana, Sigourney Weaver]
1        [Johnny Depp, Orlando Bloom, Keira Knightley]
2         [Daniel Craig, Christoph Waltz, Léa Seydoux]
3         [Christian Bale, Michael Caine, Gary Oldman]
4       [Taylor Kitsch, Lynn Collins, Samantha Morton]
5         [Tobey Maguire, Kirsten Dunst, James Franco]
Name: cast, dtype: object
```

Crew

In [27]:
```
movies['crew'][0][:500]
```

Out[27]:
```
'[{"credit_id": "52fe48009251416c750aca23", "department": "Editing", "gender": 0, "id": 1721, "job": "Editor", "name":
"Stephen E. Rivkin"}, {"credit_id": "539c47ecc3a36810e3001f87", "department": "Art", "gender": 2, "id": 496, "job": "Pr
oduction Design", "name": "Rick Carter"}, {"credit_id": "54491c89c3a3680fb4001cf7", "department": "Sound", "gender": 0,
"id": 900, "job": "Sound Designer", "name": "Christopher Boyes"}, {"credit_id": "54491cb70e0a267480001bd0", "departmen
t": "Sound", "gender": 0,'
```

In [28]:
```
def convert_crew(obj):
    crew_set = set()
    crew_list = []

    for i in ast.literal_eval(obj):
        if i['job'] in ['Director', 'Screenplay', 'Producer']:
            name = i['name']
            if name not in crew_set:
                crew_set.add(name)
                crew_list.append(name)

    return crew_list
```

In [29]:
```
movies['crew'] = movies['crew'].apply(convert_crew)
movies['crew'][0:6]
```

```
Out[29]: 0                          [James Cameron, Jon Landau]
         1    [Gore Verbinski, Jerry Bruckheimer, Ted Elliot...
         2    [Sam Mendes, John Logan, Barbara Broccoli, Rob...
         3    [Charles Roven, Christopher Nolan, Jonathan No...
         4    [Andrew Stanton, Colin Wilson, Jim Morris, Lin...
         5    [Sam Raimi, Laura Ziskin, Avi Arad, Alvin Sarg...
         Name: crew, dtype: object
```

```
In [30]: movies.head()
```

Out[30]:

	movie_id	title	overview	genres	keywords	production_companies	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Ingenious Film Partners, Twentieth Century Fo...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron, Jon Landau]
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Walt Disney Pictures, Jerry Bruckheimer Films...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski, Jerry Bruckheimer, Ted Elliot...
2	206647	Spectre	A cryptic message from Bond’s past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Columbia Pictures, Danjaq, B24]	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes, John Logan, Barbara Broccoli, Rob...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Legendary Pictures, Warner Bros., DC Entertai...	[Christian Bale, Michael Caine, Gary Oldman]	[Charles Roven, Christopher Nolan, Jonathan No...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Walt Disney Pictures]	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton, Colin Wilson, Jim Morris, Lin...

Overview

This will convert our movie overviews into a list of strings, in other words, tokens. This will help us in measuring similarities between movies.

```
In [31]: movies['overview'] = movies['overview'].apply(lambda x:x.split())
         movies['overview'][0:6]
```

```
Out[31]: 0    [In, the, 22nd, century,, a, paraplegic, Marin...
         1    [Captain, Barbossa,, long, believed, to, be, d...
         2    [A, cryptic, message, from, Bond's, past, send...
         3    [Following, the, death, of, District, Attorney...
         4    [John, Carter, is, a, war-weary,, former, mili...
         5    [The, seemingly, invincible, Spider-Man, goes,...
         Name: overview, dtype: object
```

Our dataframe looks much better and is easier to read now.

We will save a copy of this dataframe to create our website ↓

```
In [32]: cleaned_movies = movies.copy()
         cleaned_movies.to_csv('cleaned_movies.csv')
```

Feature Transformation

Now we will remove the spaces between strings for each value in 'genres', 'keywords', 'production_companies', 'cast', and 'crew'.

The purpose of this is to create only one tag per feature instead of two or more.

Example:
'Daniel Craig' will be 'DanielCraig'

If we don't do this, then another actor with the first name Daniel might get recommended to the user (eg: Daniel Day-Lewis).

```
In [33]: movies['genres'] = movies['genres'].apply(lambda x:[i.replace(' ', '') for i in x])
         movies['keywords'] = movies['keywords'].apply(lambda x:[i.replace(' ', '') for i in x])
         movies['production_companies'] = movies['production_companies'].apply(lambda x:[i.replace(' ','') for i in x])
         movies['cast'] = movies['cast'].apply(lambda x:[i.replace(' ', '') for i in x])
         movies['crew'] = movies['crew'].apply(lambda x:[i.replace(' ', '') for i in x])
```

```
In [34]: movies.head()
```

Out[34]:

	movie_id	title	overview	genres	keywords	production_companies	cast	crew
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[IngeniousFilmPartners, TwentiethCenturyFoxFil...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron, JonLandau]
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[WaltDisneyPictures, JerryBruckheimerFilms, Se...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski, JerryBruckheimer, TedElliott, ...
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel, mi6, ...	[ColumbiaPictures, Danjaq, B24]	[DanielCraig, ChristophWaltz, LéaSeydoux]	[SamMendes, JohnLogan, BarbaraBroccoli, Robert...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Action, Crime, Drama, Thriller]	[dccomics, crimefighter, terrorist, secretiden...	[LegendaryPictures, WarnerBros., DCEntertainme...	[ChristianBale, MichaelCaine, GaryOldman]	[CharlesRoven, ChristopherNolan, JonathanNolan...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Action, Adventure, ScienceFiction]	[basedonnovel, mars, medallion, spacetravel, p...	[WaltDisneyPictures]	[TaylorKitsch, LynnCollins, SamanthaMorton]	[AndrewStanton, ColinWilson, JimMorris, Lindse...

Creating Our Final Dataframe

First, we will create a 'tags' column that joins overview, genres, keywords, cast, and crew.

In [35]:
```
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['production_companies'] + movies['
movies
```

Out[35]:

	movie_id	title	overview	genres	keywords	production_companies	cast	crew	
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[IngeniousFilmPartners, TwentiethCenturyFoxFil...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron, JonLandau]	[Ir centu parap M
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[WaltDisneyPictures, JerryBruckheimerFilms, Se...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski, JerryBruckheimer, TedElliott, ...	[Ca Barb believe b
2	206647	Spectre	[A, cryptic, message, from, Bond’s, past, send...	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel, mi6, ...	[ColumbiaPictures, Danjaq, B24]	[DanielCraig, ChristophWaltz, LéaSeydoux]	[SamMendes, JohnLogan, BarbaraBroccoli, Robert...	[A, cr mes Bond’s, s
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Action, Crime, Drama, Thriller]	[dccomics, crimefighter, terrorist, secretiden...	[LegendaryPictures, WarnerBros., DCEntertainme...	[ChristianBale, MichaelCaine, GaryOldman]	[CharlesRoven, ChristopherNolan, JonathanNolan...	[Follo the, c of, Di Attor
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Action, Adventure, ScienceFiction]	[basedonnovel, mars, medallion, spacetravel, p...	[WaltDisneyPictures]	[TaylorKitsch, LynnCollins, SamanthaMorton]	[AndrewStanton, ColinWilson, JimMorris, Lindse...	[Carter war-w fo
...	...	...	...	...	...	...	...	...	
4804	9367	El Mariachi	[El, Mariachi, just, wants, to, play, his, gui...	[Action, Crime, Thriller]	[unitedstates–mexicobarrier, legs, arms, paper...	[ColumbiaPictures]	[CarlosGallardo, JaimedeHoyos, PeterMarquardt]	[RobertRodriguez, CarlosGallardo]	[El, Mar just, v to, pla
4805	72766	Newlyweds	[A, newlywed, couple's, honeymoon, is, upended...	[Comedy, Romance]	[]	[]	[EdwardBurns, KerryBishé, MarshaDietlein]	[EdwardBurns, WilliamRexer, AaronLubin]	newly cou honeyn upen
4806	231617	Signed, Sealed, Delivered	["Signed,, Sealed,, Delivered",	[Comedy, Drama,	[date, loveatfirstsight,	[FrontStreetPictures, MuseEntertainmentEnterpr...	[EricMabius, KristinBooth, CrystalLowe]	[HarveyKahn, ScottSmith]	["Sig Se Delive

	movie_id	title	overview	genres	keywords	production_companies	cast	crew	
			introduces, a,...	Romance, TVMovie]	narration, investigat...				introc
4807	126186	Shanghai Calling	[When, ambitious, New, York, attorney, Sam, is...	[]	[]	[]	[DanielHenney, ElizaCoupe, BillPaxton]	[DanielHsia]	[V ambi New, attc Sar
4808	25975	My Date with Drew	[Ever, since, the, second, grade, when, he, fi...	[Documentary]	[obsession, camcorder, crush, dreamgirl]	[rustybearentertainment, luckycrowfilms]	[DrewBarrymore, BrianHerzlinger, CoreyFeldman]	[BrianHerzlinger, JonGunn, BrettWinn]	[Ever, the, se g whe

4806 rows × 9 columns

Final Dataframe

Since the newly-created tags column already contains all the necessary information for creating our recommendation system, our dataframe will only contain this column, past the title column.

In [36]:
```
movies_df = movies[['movie_id','title','tags']]
movies_df
```

Out[36]:

	movie_id	title	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond’s, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...
...	...	...	...
4804	9367	El Mariachi	[El, Mariachi, just, wants, to, play, his, gui...
4805	72766	Newlyweds	[A, newlywed, couple's, honeymoon, is, upended...
4806	231617	Signed, Sealed, Delivered	["Signed,, Sealed,, Delivered", introduces, a,...
4807	126186	Shanghai Calling	[When, ambitious, New, York, attorney, Sam, is...
4808	25975	My Date with Drew	[Ever, since, the, second, grade, when, he, fi...

4806 rows × 3 columns

Now we can convert each list in tags column to a string using join function ...

In [37]:
```python
movies_df['tags'] = movies_df['tags'].apply(lambda x:' '.join(x))
movies_df.head()
```

Out[37]:

	movie_id	title	tags
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...
2	206647	Spectre	A cryptic message from Bond’s past sends him o...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...
4	49529	John Carter	John Carter is a war-weary, former military ca...

convert them to lowercase (so as not to confuse same words with different capitalization as different) ...

```
In [38]: movies_df['tags'] = movies_df['tags'].apply(lambda x:x.lower())
```

And finally, this is what the tags will look like.

```
In [39]: movies_df['tags'][0]
```

```
Out[39]: 'in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn betwe
en following orders and protecting an alien civilization. action adventure fantasy sciencefiction cultureclash future s
pacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle love
affair antiwar powerrelations mindandsoul 3d ingeniousfilmpartners twentiethcenturyfoxfilmcorporation duneentertainment
lightstormentertainment samworthington zoesaldana sigourneyweaver jamescameron jonlandau'
```

Preparing our System

The CountVectorizer function from sklearn converts a collection of text documents to a matrix of token counts, that way we can see the most occuring features in our data.

We chose 5000 features as our max since our dataframe contains information for 5000 movies and 'english' for the stop_words parameter since our dataframe is in english. This will cause the Vectorizer to ignore words that don't really add meaning to a sentence, such as, 'the', 'and', etc.

```
In [40]: cv = CountVectorizer(max_features=5000, stop_words='english')
         vectors = cv.fit_transform(movies_df['tags']).toarray()
         vectors
```

```
Out[40]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
In [41]: vectors.shape
```

```
Out[41]: (4806, 5000)
```

This will show the 100 most occuring values in numeric-alphabetical order ↓

```
In [42]: cv.get_feature_names()[:101]
```

```
Out[42]: ['000',
          '007',
          '10',
          '100',
          '11',
          '12',
          '13',
          '14',
          '1492pictures',
          '15',
          '16',
          '17',
          '18',
          '18th',
          '19',
          '1930s',
          '1940s',
          '1950s',
          '1960s',
          '1970s',
          '1980',
          '1980s',
          '1985',
          '1990s',
          '19th',
          '19thcentury',
          '20',
          '200',
          '2009',
          '20th',
          '21lapsentertainment',
          '24',
          '25',
          '2929productions',
          '30',
          '300',
          '3artsentertainment',
          '3d',
          '40',
          '40acres',
          '50',
          '500',
          '60',
          '60s',
          '70',
          'aaron',
          'aaroneckhart',
          'abandoned',
          'abducted',
          'abigailbreslin',
          'abilities',
          'ability',
          'able',
          'aboard',
          'abrams',
          'abuse',
          'abusive',
          'academy',
          'accept',
          'accepted',
          'accepts',
          'access',
          'accident',
          'accidental',
          'accidentally',
          'accompanied',
          'accomplish',
          'account',
          'accountant',
          'accused',
          'ace',
          'achieve',
          'act',
          'acting',
          'action',
          'actionhero',
          'actions',
          'activist',
          'activities',
          'activity',
          'actor',
          'actors',
          'actress',
          'acts',
          'actual',
          'actually',
           'adam',
           'adammckay',
           'adams',
           'adamsandler',
           'adamshankman',
           'adaptation',
           'adapted',
           'addict',
           'addicted',
           'addiction',
           'adolescence',
           'adopt',
           'adopted',
           'adoption',
           'adopts']
```

Stemming Features

We will use the PorterStemmer function from the NLTK (Natural Language Toolkit) library to reduce words down to their root word. This will keep words that mean the same thing, like 'actions' and 'action', to be counted as different words.

> The **NLTK (Natural Language Toolkit)** library is the go-to API for Natural Language Processing with Python. It is a really powerful tool to preprocess text data for further analysis like with recommendation systems for instance.
>
> The **PorterStemmer** is a function that removes any prefixes or suffixes from words, leaving only the word stem, hence the name.

In [43]:
```
ps = PorterStemmer()
```

In [44]:
```
def stemming(text):
    li=[]
    for i in text.split():
        li.append(ps.stem(i))

    return ' '.join(li)
```

```
In [45]: movies_df['tags'] = movies_df['tags'].apply(stemming)
```

Similarities

Using the cosine_similarity function from sklearn, we obtain the cosine distance between each movie vector. Cosine_similarity is frequently used in natural language processing and machine learning to compare the similarity of documents, text, or other high-dimensional data. That is to say, the angle between each vector. The smaller the angle, the more similar the data points, in this case movies, are.

```
In [46]: similarity = cosine_similarity(vectors)
         similarity
```

```
Out[46]: array([[1.        , 0.08006408, 0.05337605, ..., 0.02414023, 0.02668803,
                 0.        ],
                [0.08006408, 1.        , 0.05555556, ..., 0.02512595, 0.        ,
                 0.        ],
                [0.05337605, 0.05555556, 1.        , ..., 0.02512595, 0.        ,
                 0.        ],
                ...,
                [0.02414023, 0.02512595, 0.02512595, ..., 1.        , 0.07537784,
                 0.04956816],
                [0.02668803, 0.        , 0.        , ..., 0.07537784, 1.        ,
                 0.05479966],
                [0.        , 0.        , 0.        , ..., 0.04956816, 0.05479966,
                 1.        ]])
```

```
In [47]: similarity.shape
```

```
Out[47]: (4806, 4806)
```

↑ 4806 comparisons for 4806 movies.

Here ↓ we will enumerate and sort the similarities in descending order to get the top 5 similar movies.

Enumerating allows us to keep the index order of the movies.

Using the *lambda* function, we sort using the second value in each tuple, those being the similarity scores.

In [48]:
```
sorted(list(enumerate(similarity[0])), reverse=True, key=lambda x:x[1])[1:6]
```

Out[48]:
```
[(539, 0.2668802563418119),
 (1216, 0.2656722567395829),
 (507, 0.26148818018424536),
 (2409, 0.2470831055537004),
 (220, 0.2300789234172203)]
```

Recommendation Function

Finally, we have prepared our dataset for final use and we can use it to build our new movie recommendation system.

In [49]:
```
def recommend(movie):
    movies_index = movies_df[movies_df['title'] == movie].index[0]
    distances = similarity[movies_index]
    movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x:x[1])[1:6]

    for i in movies_list:
        print(movies_df.iloc[i[0]].title)
```

Example 1

In [50]:
```
recommend('Avatar')
```

```
Titan A.E.
Aliens vs Predator: Requiem
Independence Day
Aliens
Prometheus
```

Example 2

```
In [51]: recommend('Batman Begins')
```

```
The Dark Knight
The Dark Knight Rises
Batman v Superman: Dawn of Justice
Batman
Batman & Robin
```

Pickling

Now we wil pickle our final dataframe and our similarities function containing the vectors for our recommendations. This will be used to create our website.

```
In [70]: pickle.dump(movies_df.to_dict(), open('movies_dict.pkl', 'wb'))
```

```
In [71]: pickle.dump(similarity, open('similarity.pkl', 'wb'))
```