Movie Recommendations with Document Similarity

*** Code Adapted from:

- Chapter 7 of Text Analytics with Python (2016) by Dipanjan Sarkar
- Week 9 lecture of Dr. Olga Scrivner's course: *Introduction to NLP* at Indiana University ***

A popular application of machine learning is a recommender system. While these systems can be applied in a variety of ways, we will use machine learning to provide movie suggestions.

Typically recommender systems can be implemented in three ways:

- Rule Based: based on metrics such as popularity or rating
- Content Based: recommendations are based on entities that contain similar metadata
- Collaborative Filtering: provide recommendations based on past ratings of different users

The goal of this recommendation system is to analyze the text-based data for movies to recommend a similar movie using a content based recommender. With the focus on NLP processes, the data used is limited to a small set of 5,000 movies and their descriptions.

Import Libraries and Load Dataset

In [6]:
```
import numpy as np
import pandas as pd
import nltk
import re

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

#Declare stopwords
stop_words = nltk.corpus.stopwords.words('english')

df = pd.read_csv('tmdb_5000_movies.csv.gz', compression='gzip')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   budget                  4803 non-null   int64
 1   genres                  4803 non-null   object
 2   homepage                1712 non-null   object
 3   id                      4803 non-null   int64
 4   keywords                4803 non-null   object
 5   original_language       4803 non-null   object
 6   original_title          4803 non-null   object
 7   overview                4800 non-null   object
 8   popularity              4803 non-null   float64
 9   production_companies    4803 non-null   object
 10  production_countries    4803 non-null   object
 11  release_date            4802 non-null   object
 12  revenue                 4803 non-null   int64
 13  runtime                 4801 non-null   float64
 14  spoken_languages        4803 non-null   object
 15  status                  4803 non-null   object
 16  tagline                 3959 non-null   object
 17  title                   4803 non-null   object
 18  vote_average            4803 non-null   float64
 19  vote_count              4803 non-null   int64
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

In [7]:
```
#Check that data was imported correctly
df.head()
```

Out[7]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_com
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...	150.437577	[{"name": "Ing Film Partner
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...	139.082615	[{"name": "Walt Pictures", "id":
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...	107.376788	[{"name": "Co Pictures",
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...	112.312950	[{"name": "Leg Pictures", "id
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...	43.926995	[{"name": "Walt Pictures",

In [8]:
```
#Change column names
df = df[['title', 'tagline', 'overview', 'genres', 'popularity']]

#Fill NA values for the tagline column with an empty space
df.tagline.fillna('', inplace=True)

#create description column that contains both tagline and overview
df['description'] = df['tagline'].map(str) + ' ' + df['overview']

#Drop NA values
df.dropna(inplace=True)

#Check that we have no NA values after dropping
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4800 entries, 0 to 4802
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   title        4800 non-null   object
 1   tagline      4800 non-null   object
 2   overview     4800 non-null   object
 3   genres       4800 non-null   object
 4   popularity   4800 non-null   float64
 5   description  4800 non-null   object
dtypes: float64(1), object(5)
memory usage: 262.5+ KB
```

In [9]:
```
#Check that columns were renamed correctly
df.head()
```

Out[9]:

	title	tagline	overview	genres	popularity	description
0	Avatar	Enter the World of Pandora.	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	150.437577	Enter the World of Pandora. In the 22nd century,...
1	Pirates of the Caribbean: At World's End	At the end of the world, the adventure begins.	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	139.082615	At the end of the world, the adventure begins....
2	Spectre	A Plan No One Escapes	A cryptic message from Bond's past sends him o...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	107.376788	A Plan No One Escapes A cryptic message from Bo...
3	The Dark Knight Rises	The Legend Ends	Following the death of District Attorney Harve...	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	112.312950	The Legend Ends Following the death of District...
4	John Carter	Lost in our world, found in another.	John Carter is a war-weary, former military ca...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	43.926995	Lost in our world, found in another. John Cart...

Text pre-processing

In [11]:
```
#Function to preprocess
def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z0-9\s]', '', doc, re.I|re.A)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = nltk.word_tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc

#Vectorize document
normalize_corpus = np.vectorize(normalize_document)

#Create a list(corpus) of normalized descriptions from the descriptions in our dataframe
norm_corpus = normalize_corpus(list(df['description']))

#Check corpus length
len(norm_corpus)
```

Out[11]:
```
4800
```

Extract TF-IDF Features

In [12]:
```
#Extract features using bigrams
tf = TfidfVectorizer(ngram_range=(1, 2), min_df=2)
tfidf_matrix = tf.fit_transform(norm_corpus)
tfidf_matrix.shape
```

Out[12]:
```
(4800, 20667)
```

Compute Pairwise Document Similarity

In [13]:
```
#Calculate cosine similarity
doc_sim = cosine_similarity(tfidf_matrix)

#Convert to dataframe
doc_sim_df = pd.DataFrame(doc_sim)

#Check head of cosine similarity matrix
doc_sim_df.head()
```

Out[13]:

	0	1	2	3	4	5	6	7	8	9	...	4790	4791	4792	4793	4794	47
0	1.000000	0.010701	0.000000	0.019030	0.028687	0.024901	0.000000	0.026516	0.000000	0.007420	...	0.009702	0.0	0.023336	0.033549	0.000000	0.0000
1	0.010701	1.000000	0.011891	0.000000	0.041623	0.000000	0.014564	0.027122	0.034688	0.007614	...	0.009956	0.0	0.004818	0.000000	0.000000	0.0125
2	0.000000	0.011891	1.000000	0.000000	0.000000	0.000000	0.000000	0.022242	0.015854	0.004891	...	0.042617	0.0	0.000000	0.000000	0.016519	0.0000
3	0.019030	0.000000	0.000000	1.000000	0.008793	0.000000	0.015976	0.023172	0.027452	0.073610	...	0.000000	0.0	0.009667	0.000000	0.000000	0.0000
4	0.028687	0.041623	0.000000	0.008793	1.000000	0.000000	0.022912	0.028676	0.000000	0.023538	...	0.014880	0.0	0.000000	0.000000	0.000000	0.0107

5 rows × 4800 columns

Get List of Movie Titles

In [14]:
```
#Create list of movie titles from data
movies_list = df['title'].values

#Print list, get shape
movies_list, movies_list.shape
```

Out[14]:
```
(array(['Avatar', 'Pirates of the Caribbean: At World's End',
        'Spectre', ..., 'Signed, Sealed, Delivered',
        'Shanghai Calling', 'My Date With Drew'], dtype=object),
 (4800,))
```

Find Top Similar Movies for a Sample Movie

Taking the movie **Minions** we can find the most similar movies to recommend.

Find movie ID

In [21]:
```
#Get the index for the movie 'Minions'
movie_idx = np.where(movies_list == 'Minions')[0][0]
movie_idx
```

Out[21]:
```
546
```

Get movie similarities

In [23]:
```
#Create and print an array of values most similar to "Minions"
movie_similarities = doc_sim_df.iloc[movie_idx].values
movie_similarities
```

Out[23]:
```
array([0.0104544 , 0.01072835, 0.        , ..., 0.00690954, 0.        ,
       0.        ])
```

Get top 5 similar movie IDs

In [24]:
```
#Get the index values of the movies most similar to "Minions"
similar_movie_idxs = np.argsort(-movie_similarities)[1:6]
similar_movie_idxs
```

Out[24]:
```
array([506, 614, 241, 813, 154])
```

Get top 5 similar movies

In [18]:
```
#Get the title of movies based on their index
similar_movies = movies_list[similar_movie_idxs]
similar_movies
```

Out[18]:
```
array(['Despicable Me 2', 'Despicable Me',
       'Teenage Mutant Ninja Turtles: Out of the Shadows', 'Superman',
       'Rise of the Guardians'], dtype=object)
```

Build a function to recommend 5 most similar movies.

Now that we have built a recommender system step by step, we create a function to combine these steps.

In [32]:
```
def movie_recommender(movie_title, movies=movies_list, doc_sims=doc_sim_df):
    # find movie id
    movie_idx = np.where(movies == movie_title)[0][0]
    # get movie similarities
    movie_similarities = doc_sims.iloc[movie_idx].values
    # get top 5 similar movie IDs
    similar_movie_idxs = np.argsort(-movie_similarities)[1:6]
    # get top 5 movies
    similar_movies = movies[similar_movie_idxs]
    # return the top 5 movies
    return similar_movies
```

Get Popular Movie Recommendations

We can test the recommendation system by getting the 5 movies most similar to "Star Wars"

In [39]:
```
movie = 'Star Wars'
recommendation = movie_recommender(movie_title = movie)

i = 0

print("Movie Recommendations similar to {}".format(movie))

for movie in recommendation:
    print("{}: {}".format(i+1,recommendation[i]))
    i = i + 1
```

```
Movie Recommendations similar to Star Wars
1: The Empire Strikes Back
2: Return of the Jedi
3: Shrek the Third
4: The Ice Pirates
5: The Tale of Despereaux
```