In [1]:
```
import pandas as pd
```

In [2]:
```
movies = pd.read_csv("movies.csv")
```

In [3]:
```
movies.head()
```

Out[3]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [4]:
```
import re

def clean_title(title):
    title = re.sub("[^a-zA-Z0-9 ]", "", title)
    return title
movies["clean_title"] = movies["title"].apply(clean_title)
movies
```

Out[4]:

	movieId	title	genres	clean_title
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	Toy Story 1995
1	2	Jumanji (1995)	Adventure|Children|Fantasy	Jumanji 1995
2	3	Grumpier Old Men (1995)	Comedy|Romance	Grumpier Old Men 1995
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	Waiting to Exhale 1995
4	5	Father of the Bride Part II (1995)	Comedy	Father of the Bride Part II 1995
...	...	...	...	...
62418	209157	We (2018)	Drama	We 2018
62419	209159	Window of the Soul (2001)	Documentary	Window of the Soul 2001
62420	209163	Bad Poems (2018)	Comedy|Drama	Bad Poems 2018
62421	209169	A Girl Thing (2001)	(no genres listed)	A Girl Thing 2001
62422	209171	Women of Devil's Island (1962)	Action|Adventure|Drama	Women of Devils Island 1962

62423 rows × 4 columns

In [5]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1,2))

tfidf = vectorizer.fit_transform(movies["clean_title"])
```

In [6]:
```
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

def search(title):
    title = clean_title(title)
    query_vec = vectorizer.transform([title])
    similarity = cosine_similarity(query_vec, tfidf).flatten()
    indices = np.argpartition(similarity, -5)[-5:]
    results = movies.iloc[indices].iloc[::-1]

    return results
```

In [7]:
```
# pip install ipywidgets
#jupyter labextension install @jupyter-widgets/jupyterlab-manager
import ipywidgets as widgets
from IPython.display import display

movie_input = widgets.Text(
    value='Toy Story',
    description='Movie Title:',
    disabled=False
)
movie_list = widgets.Output()

def on_type(data):
    with movie_list:
        movie_list.clear_output()
        title = data["new"]
        if len(title) > 5:
            display(search(title))

movie_input.observe(on_type, names='value')


display(movie_input, movie_list)
```

```
Text(value='Toy Story', description='Movie Title:')
Output()
```

In [8]:
```
movie_id = 89745

#def find_similar_movies(movie_id):
movie = movies[movies["movieId"] == movie_id]
```

In [9]:
```
ratings = pd.read_csv("ratings.csv")
ratings.dtypes
```

Out[9]:
```
userId          int64
movieId         int64
rating        float64
timestamp       int64
dtype: object
```

In [10]:
```
similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] > 4)]["userId"].unique()
similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings["rating"] > 4)]["movieId"]
similar_user_recs = similar_user_recs.value_counts() / len(similar_users)

similar_user_recs = similar_user_recs[similar_user_recs > .10]
all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"] > 4)]
all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].unique())
rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
rec_percentages.columns = ["similar", "all"]
rec_percentages
```

Out[10]:

	similar	all
1	0.236083	0.126250
32	0.103877	0.101516
47	0.203115	0.146232
50	0.211067	0.202959
110	0.182240	0.162835
...	...	...
134853	0.198641	0.036444
152081	0.133532	0.020652
164179	0.128728	0.029124
166528	0.124751	0.014411
168252	0.132538	0.016469

193 rows × 2 columns

In [11]:
```
rec_percentages["score"] = rec_percentages["similar"] / rec_percentages["all"]
rec_percentages = rec_percentages.sort_values("score", ascending=False)
rec_percentages.head(10).merge(movies, left_index=True, right_on="movieId")
```

Out[11]:

	similar	all	score	movieId	title	genres	clean_title
17067	1.000000	0.040459	24.716368	89745	Avengers, The (2012)	Action|Adventure|Sci-Fi|IMAX	Avengers The 2012
20513	0.103711	0.005289	19.610199	106072	Thor: The Dark World (2013)	Action|Adventure|Fantasy|IMAX	Thor The Dark World 2013
25058	0.241054	0.012367	19.491770	122892	Avengers: Age of Ultron (2015)	Action|Adventure|Sci-Fi	Avengers Age of Ultron 2015
19678	0.216534	0.012119	17.867419	102125	Iron Man 3 (2013)	Action|Sci-Fi|Thriller|IMAX	Iron Man 3 2013
16725	0.215043	0.012052	17.843074	88140	Captain America: The First Avenger (2011)	Action|Adventure|Sci-Fi|Thriller|War	Captain America The First Avenger 2011
16312	0.175447	0.010142	17.299824	86332	Thor (2011)	Action|Adventure|Drama|Fantasy|IMAX	Thor 2011
21348	0.287608	0.016737	17.183667	110102	Captain America: The Winter Soldier (2014)	Action|Adventure|Sci-Fi|IMAX	Captain America The Winter Soldier 2014
25071	0.214049	0.012856	16.649399	122920	Captain America: Civil War (2016)	Action|Sci-Fi|Thriller	Captain America Civil War 2016
25061	0.136017	0.008573	15.865628	122900	Ant-Man (2015)	Action|Adventure|Sci-Fi	AntMan 2015
14628	0.242876	0.015517	15.651921	77561	Iron Man 2 (2010)	Action|Adventure|Sci-Fi|Thriller|IMAX	Iron Man 2 2010

In [12]:
```
def find_similar_movies(movie_id):
    similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] > 4)]["userId"].unique()
    similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings["rating"] > 4)]["movieId"]
    similar_user_recs = similar_user_recs.value_counts() / len(similar_users)

    similar_user_recs = similar_user_recs[similar_user_recs > .10]
    all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"] > 4)]
    all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].unique())
    rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
    rec_percentages.columns = ["similar", "all"]

    rec_percentages["score"] = rec_percentages["similar"] / rec_percentages["all"]
    rec_percentages = rec_percentages.sort_values("score", ascending=False)
    return rec_percentages.head(10).merge(movies, left_index=True, right_on="movieId")[["score", "title", "genres"]]
```

In [13]:
```
import ipywidgets as widgets
from IPython.display import display

movie_name_input = widgets.Text(
    value='Toy Story',
    description='Movie Title:',
    disabled=False
)
recommendation_list = widgets.Output()

def on_type(data):
    with recommendation_list:
        recommendation_list.clear_output()
        title = data["new"]
        if len(title) > 5:
            results = search(title)
            movie_id = results.iloc[0]["movieId"]
            display(find_similar_movies(movie_id))

movie_name_input.observe(on_type, names='value')

display(movie_name_input, recommendation_list)
```

```
Text(value='Toy Story', description='Movie Title:')
Output()
```

In [ ]:
```
```