In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import collections
import seaborn as sns
%matplotlib inline
```

In [2]:

```
ratings_data = pd.read_csv('ratings.csv')
movies_df = pd.read_csv("movies.csv")
tags = pd.read_csv('tags.csv')
scores = pd.read_csv('genome-scores.csv')
links = pd.read_csv('links.csv')
genome_tags = pd.read_csv('genome-tags.csv')
```

In [3]:

```
movies_df.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]:

```
#Using regular expressions to find a year stored between parentheses
#We specify the parantheses so we don't conflict with movies that have years in their titles
movies_df['year'] = movies_df.title.str.extract('(\(\d\d\d\d\))',expand=False)
#Removing the parentheses
movies_df['year'] = movies_df.year.str.extract('(\d\d\d\d)',expand=False)
#Removing the years from the 'title' column
movies_df['title'] = movies_df.title.str.replace('(\(\d\d\d\d\))', '')
#Applying the strip function to get rid of any ending whitespace characters that may have appeared
movies_df['title'] = movies_df['title'].apply(lambda x: x.strip())
movies_df.head()
```

Out[4]:

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

In [5]:

```
#Every genre is separated by a | so we simply have to call the split function on |
movies_df['genres'] = movies_df.genres.str.split('|')
movies_df.head()
```

Out[5]:

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

In [6]:

```
#Copying the movie dataframe into a new one since we won't need to use the genre information in ou
r first case.
moviesWithGenres_df = movies_df.copy()

#For every row in the dataframe, iterate through the list of genres and place a 1 into the corresp
onding column
for index, row in movies_df.iterrows():
    for genre in row['genres']:
        moviesWithGenres_df.at[index, genre] = 1
#Filling in the NaN values with 0 to show that a movie doesn't have that column's genre
moviesWithGenres_df = moviesWithGenres_df.fillna(0)
moviesWithGenres_df.head()
```

Out[6]:

	movieId	title	genres	year	Adventure	Animation	Children	Comedy	Fantasy	Romance	...	Horror	Mystery	Sci-Fi	IMAX
0	1	Toy Story	[Adventure, Animation, Children, Comedy, Fantasy]	1995	1.0	1.0	1.0	1.0	1.0	0.0	...	0.0	0.0	0.0	0.0
1	2	Jumanji	[Adventure, Children, Fantasy]	1995	1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0
2	3	Grumpier Old Men	[Comedy, Romance]	1995	0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
3	4	Waiting to Exhale	[Comedy, Drama, Romance]	1995	0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
4	5	Father of the Bride Part II	[Comedy]	1995	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0

5 rows × 24 columns

In [7]:

```
ratings_data.head()

userinput_list = []

user_id = 1
user_movies = ratings_data[ratings_data['userId'] == user_id]['movieId']

print(len(user_movies))

user_ratings = ratings_data[ratings_data['userId'] == user_id]['rating']
inputMovies = movies_df[movies_df['movieId'].isin(user_movies)].copy()
inputMovies['rating'] = pd.Series(user_ratings)
inputMovies.dropna(inplace= True)
inputMovies.drop(columns = ['movieId', 'genres'], inplace = True)
inputMovies
```

```
175
```

Out[7]:

	title	year	rating
1	Jumanji	1995	3.5
28	City of Lost Children, The (Cité des enfants p...	1995	3.5
31	Twelve Monkeys (a.k.a. 12 Monkeys)	1995	4.5
46	Seven (a.k.a. Se7en)	1995	3.5
49	Usual Suspects, The	1995	4.0
110	Rumble in the Bronx (Hont faan kui)	1995	3.0
149	Rob Roy	1995	4.0

In [8]:

```
#Filtering out the movies by title
inputId = movies_df[movies_df['title'].isin(inputMovies['title'].tolist())]
#Then merging it so we can get the movieId. It's implicitly merging it by title.
inputMovies = pd.merge(inputId, inputMovies)
#Dropping information we won't use from the input dataframe
inputMovies = inputMovies.drop('genres', 1).drop('year', 1)
#Final input dataframe
#If a movie you added in above isn't here, then it might not be in the original
#dataframe or it might spelled differently, please check capitalisation.
inputMovies
```

Out[8]:

	movieId	title	rating
0	2	Jumanji	3.5
1	29	City of Lost Children, The (Cité des enfants p...	3.5
2	32	Twelve Monkeys (a.k.a. 12 Monkeys)	4.5
3	47	Seven (a.k.a. Se7en)	3.5
4	50	Usual Suspects, The	4.0
5	112	Rumble in the Bronx (Hont faan kui)	3.0
6	151	Rob Roy	4.0

In [9]:

```
#Filtering out the movies from the input
userMovies = moviesWithGenres_df[moviesWithGenres_df['movieId'].isin(inputMovies['movieId'].tolist(
))]
userMovies
```

Out[9]:

	movieId	title	genres	year	Adventure	Animation	Children	Comedy	Fantasy	Romance	...	Horror	Mystery	Sci-Fi	IM
1	2	Jumanji	[Adventure, Children, Fantasy]	1995	1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	
28	29	City of Lost Children, The (Cité des enfants p...	[Adventure, Drama, Fantasy, Mystery, Sci-Fi]	1995	1.0	0.0	0.0	0.0	1.0	0.0	...	0.0	1.0	1.0	
31	32	Twelve Monkeys (a.k.a. 12 Monkeys)	[Mystery, Sci-Fi, Thriller]	1995	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	1.0	
46	47	Seven (a.k.a. Se7en)	[Mystery, Thriller]	1995	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	

	movieId	title	genres	year	Adventure	Animation	Children	Comedy	Fantasy	Romance	...	Horror	Mystery	Sci-Fi
49	50	Usual Suspects, The	[Crime, Mystery, Thriller]	1995	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0
110	112	Rumble in the Bronx (Hont faan kui)	[Action, Adventure, Comedy, Crime]	1995	1.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0
149	151	Rob Roy	[Action, Drama, Romance, War]	1995	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0

7 rows × 24 columns

In [10]:

```
#Resetting the index to avoid future issues
userMovies = userMovies.reset_index(drop=True)

#Dropping unnecessary issues due to save memory and to avoid issues
userGenreTable = userMovies.drop('movieId', 1).drop('title', 1).drop('genres', 1).drop('year', 1)
userGenreTable
```

Out[10]:

	Adventure	Animation	Children	Comedy	Fantasy	Romance	Drama	Action	Crime	Thriller	Horror	Mystery	Sci-Fi	IMAX
0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0
5	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

In [11]:

```
inputMovies['rating']
```

Out[11]:

```
0    3.5
1    3.5
2    4.5
3    3.5
4    4.0
5    3.0
6    4.0
Name: rating, dtype: float64
```

In [12]:

```
#Dot produt to get weights
userProfile = userGenreTable.transpose().dot(inputMovies['rating'])
#The user profile
userProfile
```

Out[12]:

```
Adventure             10.0
Animation              0.0
Children               3.5
Comedy                 3.0
Fantasy                7.0
Romance                4.0
Drama                  7.5
Action                 7.0
```

```
Crime                  7.0
Thriller              12.0
Horror                 0.0
Mystery               15.5
Sci-Fi                 8.0
IMAX                   0.0
Documentary            0.0
War                    4.0
Musical                0.0
Western                0.0
Film-Noir              0.0
(no genres listed)     0.0
dtype: float64
```

In [13]:

```
#Now let's get the genres of every movie in our original dataframe
genreTable = moviesWithGenres_df.set_index(moviesWithGenres_df['movieId'])
#And drop the unnecessary information
genreTable = genreTable.drop('movieId', 1).drop('title', 1).drop('genres', 1).drop('year', 1)
genreTable.head()
```

Out[13]:

	Adventure	Animation	Children	Comedy	Fantasy	Romance	Drama	Action	Crime	Thriller	Horror	Mystery	Sci-Fi	IMAX
movieId														
1	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [14]:

```
genreTable.shape
```

Out[14]:

```
(27278, 20)
```

In [15]:

```
#Multiply the genres by the weights and then take the weighted average
recommendationTable_df = ((genreTable*userProfile).sum(axis=1))/(userProfile.sum())
recommendationTable_df.head()
```

Out[15]:

```
movieId
1    0.265537
2    0.231638
3    0.079096
4    0.163842
5    0.033898
dtype: float64
```

In [16]:

```
#Sort our recommendations in descending order
recommendationTable_df = recommendationTable_df.sort_values(ascending=False)
#Just a peek at the values
recommendationTable_df.head()
```

Out[16]:

```
movieId
```

```
5018      0.790960
6902      0.711864
81132     0.700565
128985    0.666667
59844     0.666667
dtype: float64
```

In [17]:

```
#The final recommendation table
Final_recommendation = movies_df.loc[movies_df['movieId'].isin(recommendationTable_df.head(20).key
s())]
Final_recommendation
```

Out[17]:

	movieId	title	genres	year
196	198	Strange Days	[Action, Crime, Drama, Mystery, Sci-Fi, Thriller]	1995
2329	2414	Young Sherlock Holmes	[Action, Adventure, Children, Fantasy, Mystery...	1985
4922	5018	Motorama	[Adventure, Comedy, Crime, Drama, Fantasy, Mys...	1991
6792	6902	Interstate 60	[Adventure, Comedy, Drama, Fantasy, Mystery, S...	2002
8886	26504	Cloak & Dagger	[Action, Adventure, Children, Crime, Mystery, ...	1984
8996	26701	Patlabor: The Movie (Kidô keisatsu patorebâ: T...	[Action, Animation, Crime, Drama, Film-Noir, M...	1989
9791	31921	Seven-Per-Cent Solution, The	[Adventure, Comedy, Crime, Drama, Mystery, Thr...	1976
10862	43932	Pulse	[Action, Drama, Fantasy, Horror, Mystery, Sci-...	2006
12704	59844	Honor Among Thieves (Adieu l'ami) (Farewell, F...	[Action, Adventure, Crime, Drama, Mystery, Thr...	1968
13532	67070	Army of One (Joshua Tree)	[Action, Adventure, Crime, Drama, Mystery, Thr...	1993
14374	71999	Aelita: The Queen of Mars (Aelita)	[Action, Adventure, Drama, Fantasy, Romance, S...	1924
14975	75408	Lupin III: Sweet Lost Night (Rupan Sansei: Swe...	[Action, Animation, Comedy, Crime, Drama, Myst...	2008
15047	76153	Lupin III: First Contact (Rupan Sansei: Faasut...	[Action, Animation, Comedy, Crime, Drama, Myst...	2002
15534	79132	Inception	[Action, Crime, Drama, Mystery, Sci-Fi, Thrill...	2010
16024	81132	Rubber	[Action, Adventure, Comedy, Crime, Drama, Film...	2010
16473	83266	Kaho Naa... Pyaar Hai	[Action, Adventure, Comedy, Drama, Mystery, Ro...	2000
18318	91542	Sherlock Holmes: A Game of Shadows	[Action, Adventure, Comedy, Crime, Mystery, Th...	2011
24334	115333	Charlie Chan in Panama	[Adventure, Comedy, Crime, Drama, Mystery, Thr...	1940
24371	115479	Whip Hand, The	[Action, Adventure, Crime, Drama, Sci-Fi, Thri...	1951
26847	128985	The Stone Council	[Adventure, Crime, Drama, Fantasy, Mystery, Th...	2006

In [ ]: