Demo

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

In [2]:
```
movies = pd.read_csv('data/movies_cleaned.csv')
movies.drop('Unnamed: 0', axis=1, inplace=True)

ratings = pd.read_csv('data/ratings_cleaned.csv')
ratings.drop('Unnamed: 0', axis=1, inplace=True)

tags = pd.read_csv('data/tags_cleaned.csv')
tags.drop('Unnamed: 0', axis=1, inplace=True)

ratings.head()
```

Out[2]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [3]:
```
ratings.drop('timestamp', axis=1, inplace=True)

from surprise import Reader, Dataset
from surprise.prediction_algorithms import BaselineOnly
from surprise import accuracy
```

Content-based filter

In [4]:
```
import nltk
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

In [5]:
```
item_content = pd.read_csv('data/item_content.csv')
```

In [6]:
```
# combine titles with item content

item_content2 = pd.merge(item_content, movies, how='left', on='movieId')
item_content2 = item_content2.drop('genres', axis=1)
item_content2.head()
```

Out[6]:

	movieId	bow	title
0	1	adventure animation children comedy fantasy pi...	Toy Story (1995)
1	2	adventure children fantasy fantasy magic board...	Jumanji (1995)
2	3	comedy romance moldy old	Grumpier Old Men (1995)

	movieId	bow	title
3	4	comedy drama romance	Waiting to Exhale (1995)
4	5	comedy pregnancy remake	Father of the Bride Part II (1995)

In [7]:
```
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(item_content2['bow'])

cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
print(cosine_sim)
```

```
[[1.         0.08806834 0.01349231 ... 0.         0.15083694 0.09576975]
 [0.08806834 1.         0.         ... 0.         0.         0.        ]
 [0.01349231 0.         1.         ... 0.         0.         0.14088282]
 ...
 [0.         0.         0.         ... 1.         0.         0.        ]
 [0.15083694 0.         0.         ... 0.         1.         0.        ]
 [0.09576975 0.         0.14088282 ... 0.         0.         1.        ]]
```

In [8]:
```
# Series containing titles
indices = pd.Series(item_content2['title'])

# Return list of all movies sorted by similarity
def recommendations(title, cosine_sim=cosine_sim):
    recommended_movies = []
    idx = indices[indices == title].index[0]
    sorted_scores = pd.Series(cosine_sim[idx]).sort_values(ascending=False)
    sorted_indexes = list(sorted_scores.index)
    for i in sorted_indexes:
        recommended_movies.append(indices[i])
    return recommended_movies
```

Additional Functions

In [9]:
```
# Movie rater for content-based recommendations

def movie_rater(movie_list, userId):
    rating_list = []
    for movie in movie_list:
        entry = movies[movies['title'] == movie]
        print('\n')
        print('*****************************************************************
        print(entry)
        rating = input('How do you rate this movie on a scale of 1-5, press n if
        if rating == 'n':
            continue
        else:
            rating_one_movie = {'userId': userId, 'movieId': entry['movieId'].va
                                'rating': rating}
            rating_list.append(rating_one_movie)
            if len(rating_list) == 5:
                break
    return rating_list
```

In [10]:
```
def collab_recommendations(user_ratings, movie_title_df, n):
    for idx, rec in enumerate(user_ratings):
        title = movie_title_df.loc[movie_title_df['movieId'] == int(rec[0])]['ti
        print('Recommendation #', idx + 1, ': ', title, '\n')
```

```
        n -= 1
        if n == 0:
            break
```

Demo

The following cells should be run in order. A new user searches for 3 films of interest and selects them. Content-based filtering returns the most similary films to their searched films. The user rates these films and the data is updated. Collaborative filtering predicts the user's ratings for unseen films and returns the top n recommendations.

In [15]:

```
print('Welcome to your profile!\n')
userId = int(input('Enter UserId:'))
print('\n')

# obtain list of three movies
cold_start_list = []
while len(cold_start_list) < 3:
    search = input('Enter a movie you like:')
    print('\n')
    search_results = item_content2[item_content2['title'].str.contains(search, c
    search_results = search_results.reset_index().drop('index', axis=1)

    # select and append film names from searches
    search_results['Enter for film:'] = search_results.index + 1
    display(search_results[['movieId', 'title', 'Enter for film:']])
    film_num = input('Enter number, press n if search is empty:')
    if film_num == 'n':
        print('\n')
        continue
    else:
        idx = int(film_num) - 1
        cold_start_list.append(search_results.iloc[idx]['title'])
        print('\n')

# Print search results
print('Selected Movies')
print('------------------------')
for idx, movie in enumerate(cold_start_list):
    print('Movie {}: {}'.format((idx+1), movie))
print('\n')

# Content-based recommendations in 3 lists
cold_recs = []
for movie in cold_start_list:
    print('Movies like', movie)
    rec_list = recommendations(movie)
    cold_recs.append(rec_list)
    # display top 6 most similar movies
    temp = rec_list[:6]
    print([i for i in temp if i not in movie])
    print('\n')

# Rate seen films
print('Rate Films')
print('------------------------')
user_ratings_nested = []
for rec_list in cold_recs:
```

```
    # user rates at least 5 most similar movies for each search
    user_ratings = movie_rater(rec_list, userId)
    user_ratings_nested.append(user_ratings)

# flattened list of 15 new ratings
user_ratings_all = []
for element in user_ratings_nested:
    user_ratings_all.extend(element)
```

```
Welcome to your profile!
```

	movieId	title	Enter for film:
0	2	Jumanji (1995)	1
1	179401	Jumanji: Welcome to the Jungle (2017)	2

	movieId	title	Enter for film:
0	3535	American Psycho (2000)	1
1	27473	American Psycho II: All American Girl (2002)	2

	movieId	title	Enter for film:
0	1	Toy Story (1995)	1
1	3114	Toy Story 2 (1999)	2
2	78499	Toy Story 3 (2010)	3

```
Selected Movies
------------------------
Movie 1: Jumanji (1995)
Movie 2: American Psycho (2000)
Movie 3: Toy Story (1995)


Movies like Jumanji (1995)
['Tomb Raider (2018)', 'Night at the Museum (2006)', 'Pan (2015)', 'Return to Oz
(1985)', 'Seventh Son (2014)']


Movies like American Psycho (2000)
['Saw VI (2009)', "Bird with the Crystal Plumage, The (Uccello dalle piume di cr
istallo, L') (1970)", 'Book of Shadows: Blair Witch 2 (2000)', 'Testament of Dr.
Mabuse, The (Das Testament des Dr. Mabuse) (1933)', 'From Hell (2001)']


Movies like Toy Story (1995)
["Bug's Life, A (1998)", 'Toy Story 2 (1999)', 'Guardians of the Galaxy 2 (201
```

```
7)', 'Monsters, Inc. (2001)', 'Turbo (2013)']


Rate Films
------------------------



*******************************************************************************
   movieId           title                    genres
1        2  Jumanji (1995)  Adventure|Children|Fantasy


*******************************************************************************
      movieId               title                        genres
9689   184471  Tomb Raider (2018)  Action|Adventure|Fantasy


*******************************************************************************
      movieId                       title                           genres
6252    46972  Night at the Museum (2006)  Action|Comedy|Fantasy|IMAX


*******************************************************************************
      movieId      title                        genres
8798   130450  Pan (2015)  Adventure|Children|Fantasy


*******************************************************************************
      movieId                title                           genres
1556     2093  Return to Oz (1985)  Adventure|Children|Fantasy


*******************************************************************************
      movieId          title                            genres
7171    72129  Saw VI (2009)  Crime|Horror|Mystery|Thriller


*******************************************************************************
      movieId                                              title  \
6070    41014  Bird with the Crystal Plumage, The (Uccello da...

                            genres
6070  Crime|Horror|Mystery|Thriller


*******************************************************************************
      movieId                   title                            genres
2641     3535  American Psycho (2000)  Crime|Horror|Mystery|Thriller


*******************************************************************************
      movieId                                  title  \
2964     3973  Book of Shadows: Blair Witch 2 (2000)

                            genres
2964  Crime|Horror|Mystery|Thriller


*******************************************************************************
      movieId                                              title  \
5271     8670  Testament of Dr. Mabuse, The (Das Testament de...

                            genres
5271  Crime|Horror|Mystery|Thriller
```

```
**************************************************************************
   movieId             title                                      genres
0        1  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy


**************************************************************************
      movieId                 title                                genres
1757     2355  Bug's Life, A (1998)  Adventure|Animation|Children|Comedy


**************************************************************************
      movieId               title                                       genres
2355     3114  Toy Story 2 (1999)  Adventure|Animation|Children|Comedy|Fantasy


**************************************************************************
      movieId                              title                       genres
8693   122918  Guardians of the Galaxy 2 (2017)  Action|Adventure|Sci-Fi


**************************************************************************
      movieId                  title  \
3568     4886  Monsters, Inc. (2001)

                                           genres
3568  Adventure|Animation|Children|Comedy|Fantasy
```

In [16]:

```python
# train algo on updated ratings data
new_ratings_df = ratings.append(user_ratings_all, ignore_index=True)

reader = Reader()
data = Dataset.load_from_df(new_ratings_df, reader)
trainset = data.build_full_trainset()

bsl_options = {'method': 'als',
               'n_epochs': 50,
               'reg_u': 4,
               'reg_i': 3
               }
algo = BaselineOnly(bsl_options=bsl_options)
algo.fit(trainset)
```

```
Estimating biases using als...
```

Out[16]:

```
<surprise.prediction_algorithms.baseline_only.BaselineOnly at 0x7ff3f4c6f4f0>
```

In [17]:

```python
# predictions/ collaborative filtering recommendations
list_of_movies = []
for m_id in new_ratings_df['movieId'].unique():
    list_of_movies.append((m_id, algo.predict(userId, m_id)[3]))

ranked_movies = sorted(list_of_movies, key = lambda x:x[1], reverse=True)

print('Based on Your Ratings:')
print('------------------------\n')
collab_recommendations(ranked_movies, movies, n=5)
```

```
Based on Your Ratings:
------------------------

Recommendation # 1 :  9615    Three Billboards Outside Ebbing, Missouri (2017)
Name: title, dtype: object

Recommendation # 2 :  277    Shawshank Redemption, The (1994)
```

```
Name: title, dtype: object

Recommendation # 3 :  2582    Guess Who's Coming to Dinner (1967)
Name: title, dtype: object

Recommendation # 4 :  906    Lawrence of Arabia (1962)
Name: title, dtype: object

Recommendation # 5 :  602    Dr. Strangelove or: How I Learned to Stop Worr...
Name: title, dtype: object
```