Hybrid Recommenders

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

In [2]:
```python
#Import or compute the cosine_sim matrix
cosine_sim = pd.read_csv('../data/cosine_sim.csv')
```

In [3]:
```python
#Import or compute the cosine sim mapping matrix
cosine_sim_map = pd.read_csv('../data/cosine_sim_map.csv', header=None)

#Convert cosine_sim_map into a Pandas Series
cosine_sim_map = cosine_sim_map.set_index(0)
cosine_sim_map = cosine_sim_map[1]
```

In [4]:
```python
#Build the SVD based Collaborative filter
from surprise import SVD, Reader, Dataset

reader = Reader()
ratings = pd.read_csv('../data/ratings_small.csv')
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
data.split(n_folds=5)
svd = SVD()
trainset = data.build_full_trainset()
svd.train(trainset)
```

In [5]:
```python
#Build title to ID and ID to title mappings
id_map = pd.read_csv('../data/movie_ids.csv')
id_to_title = id_map.set_index('id')
title_to_id = id_map.set_index('title')
```

In [6]:
```python
#Import or compute relevant metadata of the movies
smd = pd.read_csv('../data/metadata_small.csv')
```

In [7]:
```python
def hybrid(userId, title):
    #Extract the cosine_sim index of the movie
    idx = cosine_sim_map[title]

    #Extract the TMDB ID of the movie
    tmdbId = title_to_id.loc[title]['id']

    #Extract the movie ID internally assigned by the dataset
    movie_id = title_to_id.loc[title]['movieId']

    #Extract the similarity scores and their corresponding index for every movie from the cosine_sim matrix
    sim_scores = list(enumerate(cosine_sim[str(int(idx))]))

    #Sort the (index, score) tuples in decreasing order of similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    #Select the top 25 tuples, excluding the first
    #(as it is the similarity score of the movie with itself)
    sim_scores = sim_scores[1:26]

    #Store the cosine_sim indices of the top 25 movies in a list
    movie_indices = [i[0] for i in sim_scores]

    #Extract the metadata of the aforementioned movies
    movies = smd.iloc[movie_indices][['title', 'vote_count', 'vote_average', 'year', 'id']]

    #Compute the predicted ratings using the SVD filter
    movies['est'] = movies['id'].apply(lambda x: svd.predict(userId, id_to_title.loc[x]['movieId']).est)

    #Sort the movies in decreasing order of predicted rating
    movies = movies.sort_values('est', ascending=False)

    #Return the top 10 movies as recommendations
    return movies.head(10)
```

In [8]:
```python
hybrid(1, 'Avatar')
```

Out[8]:

	title	vote_count	vote_average	year	id	est
1011	The Terminator	4208.0	7.4	1984	218	3.140748
974	Aliens	3282.0	7.7	1986	679	3.126947
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	3.079551
7705	Alice in Wonderland	8.0	5.4	1933	25694	3.054995
3060	Sinbad and the Eye of the Tiger	39.0	6.3	1977	11940	3.028386
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	2.997411
2014	Fantastic Planet	140.0	7.6	1973	16306	2.957614
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	2.914548
1621	Darby O'Gill and the Little People	35.0	6.7	1959	18887	2.844940
1668	Return from Witch Mountain	38.0	5.6	1978	14822	2.804012

In [9]:
```python
hybrid(2, 'Avatar')
```

Out[9]:

	title	vote_count	vote_average	year	id	est
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	3.943639
2834	Predator	2129.0	7.3	1987	106	3.866272
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	3.858491
1011	The Terminator	4208.0	7.4	1984	218	3.856029
7705	Alice in Wonderland	8.0	5.4	1933	25694	3.701565
922	The Abyss	822.0	7.1	1989	2756	3.676465
974	Aliens	3282.0	7.7	1986	679	3.672303
1621	Darby O'Gill and the Little People	35.0	6.7	1959	18887	3.628234
1668	Return from Witch Mountain	38.0	5.6	1978	14822	3.614118
2014	Fantastic Planet	140.0	7.6	1973	16306	3.602051

In [ ]:

In [ ]: