Movie Recommendation System

This project has been made as the course project for the CS328-data science at IIT Gandhinagar

Contributors:

Harshit Singh Chauhan

Md. Amir Shohail

Nitin Kakraliya

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

In [2]:
```
#Reading the CSV data files
movies=pd.read_csv('movies.csv')
ratings=pd.read_csv('ratingsmovie.csv')
```

In [3]:
```
movies.head(17)
```

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
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children
8	9	Sudden Death (1995)	Action

	movieId	title	genres
9	10	GoldenEye (1995)	Action|Adventure|Thriller
10	11	American President, The (1995)	Comedy|Drama|Romance
11	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
12	13	Balto (1995)	Adventure|Animation|Children
13	14	Nixon (1995)	Drama
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
15	16	Casino (1995)	Crime|Drama
16	17	Sense and Sensibility (1995)	Drama|Romance

In [4]:

```
ratings
```

Out[4]:

	userId	movieId	rating	timestamp
0	1	296	5.0	1147880044
1	1	306	3.5	1147868817
2	1	307	5.0	1147868828
3	1	665	5.0	1147878820
4	1	899	3.5	1147868510
...	...	...	...	...
25000090	162541	50872	4.5	1240953372
25000091	162541	55768	2.5	1240951998
25000092	162541	56176	2.0	1240950697
25000093	162541	58559	4.0	1240953434
25000094	162541	63876	5.0	1240952515

25000095 rows × 4 columns

In [5]:
```
# movies.iloc[15881]
# movies['title'].iloc[15881]
```

In [6]:
```
#dropping all the movies which had 'no genres listed' in the given dataset and resetting the index
lst=movies.loc[movies['genres']=='(no genres listed)'].index
movies.drop(lst,inplace=True)
movies.reset_index(drop=True,inplace=True)
```

In [7]:
```
#Dropped 5062 movies and reindexed the dataset
movies
```

Out[7]:

	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
...	...	...	...
57356	209155	Santosh Subramaniam (2008)	Action|Comedy|Romance
57357	209157	We (2018)	Drama
57358	209159	Window of the Soul (2001)	Documentary
57359	209163	Bad Poems (2018)	Comedy|Drama
57360	209171	Women of Devil's Island (1962)	Action|Adventure|Drama

57361 rows × 3 columns

In [8]:
```
# movies.iloc[15881]
# movies['title'].iloc[15881]
```

In [9]:
```
ratings.shape
```
Out[9]:
```
(25000095, 4)
```

In [10]:
```
movies.shape
```
Out[10]:
```
(57361, 3)
```

In [11]:
```
# Forming a new column for the release year of the movie
movies_new=movies
movies_new['release_year']=''
```

In [12]:
```
movies_new
```
Out[12]:

	movieId	title	genres	release_year
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	
...	...	...	...	...
57356	209155	Santosh Subramaniam (2008)	Action|Comedy|Romance	
57357	209157	We (2018)	Drama	
57358	209159	Window of the Soul (2001)	Documentary	
57359	209163	Bad Poems (2018)	Comedy|Drama	
57360	209171	Women of Devil's Island (1962)	Action|Adventure|Drama	

57361 rows × 4 columns

In [13]:
```
#Separating the release year from the title of the movie
#Also,separating the
for i in range(movies_new.shape[0]):
    a=movies_new['title'].iloc[i]
    movies_new.title[i]=a[:-7]
    movies_new['release_year'][i]=a[-5:-1]
    movies_new.genres[i]=movies_new.genres[i].replace('|',' ')
```

```
C:\Users\Dell\AppData\Local\Temp/ipykernel_42668/3028130940.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy
  movies_new.title[i]=a[:-7]
C:\Users\Dell\AppData\Local\Temp/ipykernel_42668/3028130940.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy
  movies_new['release_year'][i]=a[-5:-1]
C:\Users\Dell\AppData\Local\Temp/ipykernel_42668/3028130940.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy
  movies_new.genres[i]=movies_new.genres[i].replace('|',' ')
```

In [14]:
```
movies_new['genres']=movies_new['genres'].apply(lambda x:x.lower())
```

In [15]:
```
# movies_new.head(60)
```

In [16]:
```
movies_new
```

Out[16]:

	movieId	title	genres	release_year
0	1	Toy Story	adventure animation children comedy fantasy	1995
1	2	Jumanji	adventure children fantasy	1995

	movieId	title	genres	release_year
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
...	...	...	...	...
57356	209155	Santosh Subramaniam	action comedy romance	2008
57357	209157	We	drama	2018
57358	209159	Window of the Soul	documentary	2001
57359	209163	Bad Poems	comedy drama	2018
57360	209171	Women of Devil's Island	action adventure drama	1962

57361 rows × 4 columns

In [17]:
```
#Checking for any duplicate entry in the database
ratings.duplicated().sum()
```

Out[17]:
```
0
```

In [18]:
```
#Dropping the 'timestamp' column from the dataset
ratings.drop(['timestamp'],axis=1,inplace=True)
```

In [19]:
```
ratings
```

Out[19]:

	userId	movieId	rating
0	1	296	5.0
1	1	306	3.5
2	1	307	5.0
3	1	665	5.0

	userId	movieId	rating
4	1	899	3.5
...	...	...	...
25000090	162541	50872	4.5
25000091	162541	55768	2.5
25000092	162541	56176	2.0
25000093	162541	58559	4.0
25000094	162541	63876	5.0

25000095 rows × 3 columns

In [20]:
```
#Merging the movies and ratings dataset on the basis of the column 'movieId'
movierating=movies_new.merge(ratings,on='movieId')
```

In [21]:
```
movierating
```

Out[21]:

	movieId	title	genres	release_year	userId	rating
0	1	Toy Story	adventure animation children comedy fantasy	1995	2	3.5
1	1	Toy Story	adventure animation children comedy fantasy	1995	3	4.0
2	1	Toy Story	adventure animation children comedy fantasy	1995	4	3.0
3	1	Toy Story	adventure animation children comedy fantasy	1995	5	4.0
4	1	Toy Story	adventure animation children comedy fantasy	1995	8	4.0
...	...	...	...	...	...	...
24973463	209155	Santosh Subramaniam	action comedy romance	2008	134916	5.0
24973464	209157	We	drama	2018	119571	1.5
24973465	209159	Window of the Soul	documentary	2001	115835	3.0
24973466	209163	Bad Poems	comedy drama	2018	6964	4.5

	movieId	title	genres	release_year	userId	rating
24973467	209171	Women of Devil's Island	action adventure drama	1962	119571	3.0

24973468 rows × 6 columns

In [22]:

```
temp1=movierating.groupby('title').sum()
```

In [23]:

```
#Grouping together the data with the same 'userId' and storing only the data of those users who have rated more than 1000 movies
x=movierating.groupby('userId').count()['rating']>1000
#Performing Boolean indexing to get the index of the users who have rated more than 1000 movies
userwithrating=x[x].index
```

In [24]:

```
#Storing the data of the users who have rated more than 1000 movies in a new dataset
filteredratings=movierating[movierating['userId'].isin(userwithrating)]
```

In [25]:

```
filteredratings
```

Out[25]:

	movieId	title	genres	release_year	userId	rating
53	1	Toy Story	adventure animation children comedy fantasy	1995	187	3.5
130	1	Toy Story	adventure animation children comedy fantasy	1995	426	2.5
170	1	Toy Story	adventure animation children comedy fantasy	1995	541	5.0
172	1	Toy Story	adventure animation children comedy fantasy	1995	548	4.5
199	1	Toy Story	adventure animation children comedy fantasy	1995	626	4.5
...	...	...	...	...	...	...
24973441	209057	The Somme	drama	2005	115548	3.0
24973442	209069	Snapshots	drama romance	2002	30779	3.0
24973449	209121	Adrenalin: The BMW Touring Car Story	documentary	2014	53808	4.0
24973454	209135	Jane B. by Agnès V.	documentary fantasy	1988	154484	3.5
24973460	209147	The Carpet of Horror	crime horror	1962	83426	3.5

	movieId	title	genres	release_year	userId	rating

4169404 rows × 6 columns

In [26]:

```
#List of all the movies whihc are rated by the user with userId=548
#              filteredratings.loc[filteredratings['userId']==548]
```

In [27]:

```
#Grouping together the data with the same 'movieId' and storing only those movies which have been rated by at least 50 users
y=filteredratings.groupby('movieId').count()['rating']>50
#Performing Boolean indexing to get the index of the movies which have been rated by more than 50 users
ratingsonmovies=y[y].index
```

In [28]:

```
#Storing the data of the movies which have been rated by more than 50 users in a new dataset
filteredmovies=filteredratings[filteredratings['movieId'].isin(ratingsonmovies)]
```

In [29]:

```
filteredmovies
```

Out[29]:

	movieId	title	genres	release_year	userId	rating
53	1	Toy Story	adventure animation children comedy fantasy	1995	187	3.5
130	1	Toy Story	adventure animation children comedy fantasy	1995	426	2.5
170	1	Toy Story	adventure animation children comedy fantasy	1995	541	5.0
172	1	Toy Story	adventure animation children comedy fantasy	1995	548	4.5
199	1	Toy Story	adventure animation children comedy fantasy	1995	626	4.5
...	...	...	...	...	...	...
24971045	205383	El Camino: A Breaking Bad Movie	crime drama thriller	2019	149294	4.0
24971046	205383	El Camino: A Breaking Bad Movie	crime drama thriller	2019	149890	4.0
24971048	205383	El Camino: A Breaking Bad Movie	crime drama thriller	2019	151009	4.0
24971050	205383	El Camino: A Breaking Bad Movie	crime drama thriller	2019	152937	2.5
24971066	205383	El Camino: A Breaking Bad Movie	crime drama thriller	2019	161560	2.5

movieId	title	genres	release_year	userId	rating

3854296 rows × 6 columns

In [30]:
```
# Counting the number of movies we are left with in the dataset after the two filterings
#filteredmovies.movieId.nunique()
```

In [31]:
```
filteredmovies_new=filteredmovies.groupby('title').first()
```

In [32]:
```
filteredmovies_new.reset_index(inplace=True)
```

In [33]:
```
filteredmovies_new
```

Out[33]:

	title	movieId	genres	release_year	userId	rating
0		189577	sci-fi thriller	Ta	3013	3.0
1	'71	117867	action drama thriller war	2014	426	2.5
2	'Round Midnight	26564	drama musical	1986	9391	3.0
3	'Salem's Lot	27751	drama horror mystery thriller	2004	548	3.0
4	'Til There Was You	779	drama romance	1997	2177	4.0
...	...	...	...	...	...	...
9765	xXx	5507	action crime thriller	2002	187	3.5
9766	xXx: Return of Xander Cage	167738	action adventure crime thriller	2017	2389	2.5
9767	xXx: State of the Union	33158	action crime thriller	2005	187	3.0
9768	¡Three Amigos!	2478	comedy western	1986	626	2.0
9769	À nous la liberté (Freedom for Us)	5560	comedy musical	1931	847	2.5

9770 rows × 6 columns

In [34]:
```
#For checking the genres of any respective movie
filteredmovies_new.loc[filteredmovies_new['title']=='Forrest Gump',['title','genres']]
```

Out[34]:

	title	genres
3136	Forrest Gump	comedy drama romance war

Content Based Recommender System

In [35]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer()
```

In [36]:
```
#Converting the genre terms into arrays, these arrays would be used for camparing the genres of the movie
vectors=cv.fit_transform(filteredmovies_new['genres']).toarray()
```

In [37]:
```
#For finding out the similarity between the arrays, which are formed using the genres of the movies.
from sklearn.metrics.pairwise import cosine_similarity
similarity = cosine_similarity(vectors)
```

In [51]:
```
similarity[0][9768]
```

Out[51]:
```
0.0
```

In [38]:
```
#filteredmovies_new.loc[filteredmovies_new['title']=='Titanic','release_year']
```

In [39]:
```
#Takes a movie name and recommends the most similar movies as per its the genre tags.
def contentrecommend(movie):
    movieindex=filteredmovies_new[filteredmovies_new['title']==movie].index[0]
    distances=similarity[movieindex]
    movieslist=sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:20]
    for i in movieslist:
```

```
        print(filteredmovies_new.iloc[i[0]].title)
    return
```

In [40]:

```
contentrecommend('Salaam Bombay!')
```

```
12 Angry Men
12 Years a Slave
13 Hours
2 ou 3 choses que je sais d'elle (2 or 3 Things I Know About Her)
20th Century Women
28 Days
3 Women (Three Women)
4 Months, 3 Weeks and 2 Days (4 luni, 3 saptamâni si 2 zile)
42
45 Years
54
61*
71 Fragments of a Chronology of Chance (71 Fragmente einer Chronologie des Zufalls)
8 Mile
8 Seconds
99 Homes
A Ghost Story
Aberdeen
Accattone
```

In [ ]:

```
#Star Trek: Nemesis
#Harry Potter and the Chamber of Secrets
#Salaam Bombay!
#S.W.A.T.
#Fast & Furious 6 (Fast and the Furious 6, The)
#A.I. Artificial Intelligence
#Spy Kids
#Revenge of the Nerds II: Nerds in Paradise
#Goonies, The
#Zootopia
```

Collaborative Filtering Based Recommender System

In [52]:

```
#Creating a matrix with movie title as rows, userId as columns, and the values as the ratings given to the movie by the correspond
ratingmatrix=filteredmovies.pivot_table(index='title',columns='userId',values='rating')
```

In [53]:
```
ratingmatrix.shape
```

Out[53]:
```
(9770, 2665)
```

In [54]:
```
#This matrix shows rating given to a movie by a user with corresponding userId
ratingmatrix
```

Out[54]:

userId	187	426	541	548	626	653	757	803	846	847	...	161560	161586	161675	161826	161928	162047	162271	162495	162508	1
title																					
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
'71	NaN	2.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
'Round Midnight	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
'Salem's Lot	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
'Til There Was You	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
xXx	3.5	NaN	NaN	3.0	3.0	NaN	NaN	3.0	1.5	NaN	...	2.0	1.5	NaN	1.0	3.0	NaN	NaN	NaN	NaN	
xXx: Return of Xander Cage	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
xXx: State of the Union	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	0.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
¡Three Amigos!	NaN	NaN	NaN	NaN	2.0	4.0	3.0	NaN	0.5	2.5	...	NaN	1.0	NaN	NaN	NaN	NaN	2.5	4.0	NaN	
À nous la liberté	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.5	...	NaN	NaN	NaN	NaN	NaN	NaN	2.5	NaN	NaN	

userId	187	426	541	548	626	653	757	803	846	847	...	161560	161586	161675	161826	161928	162047	162271	162495	162508	1
title																					
(Freedom for Us)																					

9770 rows × 2665 columns

In [55]:

```
#Replacing the missing values with 0
ratingmatrix.fillna(0,inplace=True)
```

In [56]:

```
ratingmatrix
```

Out[56]:

userId	187	426	541	548	626	653	757	803	846	847	...	161560	161586	161675	161826	161928	162047	162271	162495	162508	16251(
title																					
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
'71	0.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
'Round Midnight	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5
'Salem's Lot	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
'Til There Was You	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
xXx	3.5	0.0	0.0	3.0	3.0	0.0	0.0	3.0	1.5	0.0	...	2.0	1.5	0.0	1.0	3.0	0.0	0.0	0.0	0.0	2.5
xXx: Return of Xander Cage	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
xXx: State of	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(

userId	187	426	541	548	626	653	757	803	846	847	...	161560	161586	161675	161826	161928	162047	162271	162495	162508	16251(
title																					
the Union																					
¡Three Amigos!	0.0	0.0	0.0	0.0	2.0	4.0	3.0	0.0	0.5	2.5	...	0.0	1.0	0.0	0.0	0.0	0.0	2.5	4.0	0.0	2.(
À nous la liberté (Freedom for Us)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5	...	0.0	0.0	0.0	0.0	0.0	0.0	2.5	0.0	0.0	0.(

9770 rows × 2665 columns

In [57]:

```
from sklearn.metrics.pairwise import cosine_similarity
```

In [58]:

```
#Finding out the cosine similarity between the movies
similarityscores=cosine_similarity(ratingmatrix)
```

In [65]:

```
similarityscores.shape
```

Out[65]:

```
(9770, 9770)
```

In [59]:

```
similarityscores[0]
```

Out[59]:

```
array([1.        , 0.06339446, 0.02196289, ..., 0.13518302, 0.07176835,
       0.03885632])
```

In [126…

```
#Function which takes a movie name as input and recommends the top 20 similar movies
def recommend(moviename):
    index=np.where(ratingmatrix.index==moviename)[0][0]
    similaritems=sorted(list(enumerate(similarityscores[index])),key=lambda x:x[1],reverse=True)[1:21]
    #distances=similarityscores[index]
    for i in similaritems:
```

```
        print(ratingmatrix.index[i[0]])
    return
```

In [127…

```
recommend('Titanic')
```

```
Forrest Gump
Sixth Sense, The
Matrix, The
Shawshank Redemption, The
Back to the Future
Jurassic Park
Silence of the Lambs, The
Men in Black (a.k.a. MIB)
Star Wars: Episode IV - A New Hope
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)
Gladiator
Truman Show, The
Toy Story
Pulp Fiction
Groundhog Day
E.T. the Extra-Terrestrial
Star Wars: Episode V - The Empire Strikes Back
Terminator, The
Saving Private Ryan
Terminator 2: Judgment Day
```

In [128…

```
#Star Trek: Nemesis
#Harry Potter and the Chamber of Secrets
#Salaam Bombay!
#S.W.A.T.
#Fast & Furious 6 (Fast and the Furious 6, The)
#A.I. Artificial Intelligence
#Spy Kids
#Revenge of the Nerds II: Nerds in Paradise
#Titanic
```

Enhancing the Collabortive filtering based recommendation system:

In [134…

```
#Considering the movie genres even for recommending the movies
def enhancedrecommend(moviename):
```

```
    index=np.where(ratingmatrix.index==moviename)[0][0]
    similaritems=sorted(list(enumerate(similarityscores[index])),key=lambda x:x[1],reverse=True)[1:21]
    #Forming a dictionary to store the index of movies which are similar to the given movie based on user rating and genres.
    hybridsimilaritems={}
    for j in similaritems:
        hybridsimilaritems.__setitem__(j[0],similarity[index][j[0]])
    keys = list(hybridsimilaritems.keys())
    values = list(hybridsimilaritems.values())
    sorted_value_index = np.argsort(values)
    sorthybridsimilaritems = {keys[i]: values[i] for i in sorted_value_index}
    revsorthybridsimilaritems = dict(reversed(list(sorthybridsimilaritems.items())))
    for k in list(revsorthybridsimilaritems.keys()):
        print(ratingmatrix.index[k])
    return
```

In [135…

```
enhancedrecommend('Titanic')
```

```
Forrest Gump
Shawshank Redemption, The
Sixth Sense, The
Groundhog Day
Gladiator
Saving Private Ryan
E.T. the Extra-Terrestrial
Pulp Fiction
Truman Show, The
Silence of the Lambs, The
Matrix, The
Back to the Future
Jurassic Park
Terminator 2: Judgment Day
Men in Black (a.k.a. MIB)
Star Wars: Episode IV - A New Hope
Toy Story
Star Wars: Episode V - The Empire Strikes Back
Terminator, The
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)
```

In [ ]:

```
#Star Trek: Nemesis
#Harry Potter and the Chamber of Secrets
#Salaam Bombay!
#S.W.A.T.
```

```
#Fast & Furious 6 (Fast and the Furious 6, The)
#A.I. Artificial Intelligence
#Spy Kids
#Revenge of the Nerds II: Nerds in Paradise
#Titanic
```