Recomendador de películas (con análisis de sentimiento de twitter)

Proyecto mineria de datos - 2021

Ela Katherine Shepherd Arévalo - Pablo Daurell Marina

In [1]:

```
!pip install tweepy
```

```
Requirement already satisfied: tweepy in /usr/local/lib/python3.7/dist-pac
kages (3.10.0)
Requirement already satisfied: requests[socks]>=2.11.1 in /usr/local/lib/p
ython3.7/dist-packages (from tweepy) (2.23.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dis
t-packages (from tweepy) (1.15.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/
python3.7/dist-packages (from tweepy) (1.3.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python
3.7/dist-packages (from requests[socks]>=2.11.1->tweepy) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.11.1->twee
py) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.7/dist-packages (from requests[socks]>=2.11.1->tweepy) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/di
st-packages (from requests[socks]>=2.11.1->tweepy) (2.10)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6; extra == "socks" in
/usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.11.1->twee
py) (1.7.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.
7/dist-packages (from requests-oauthlib>=0.7.0->tweepy) (3.1.0)
```

In [2]:

```
import pandas as pd
import numpy as np
import os
import re
import string
import pickle
import random

import tweepy
from tweepy import OAuthHandler

import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import losses
from tensorflow.keras import preprocessing
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization
```

In [3]:

```
'''Importar dataset de Drive'''
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

- El objetivo de esta parte es implementar un recomendador de películas (de la web IMDb (https://www.imdb.com/)), que en función de los género de películas más vistos por un usuario, le recomiende películas que no haya visto pero otros usuarios con gustos similares si. *(Recommender based on Collaborative Filtering)*
- Además, extrayendo tweets del usuario, se analizará su sentimiento y se filtraran las peliculas recomendadas para adaptarse a ese estado de ánimo.

1. Dataset para el sistema recomendador

- Para simular un entorno en el que conocemos las películas vistas por varios usuarios y en el que tenemos acceso a sus tweets, usaremos el siguiente dataset: MovieTweetings (https://github.com/sidooms/MovieTweetings), que contiene toda la información necesaria para este propósito

> - **ratings.dat**: Contiene una lista que relaciona ids usuarios con ids de reviews que estos mismos han hecho sobre ciertas películas (De aquí podemos extraer que películas ha visto cada usuario)
> - **movies.dat**: Relaciona el id de una película con su título y sus géneros (Con esto podemos ver que géneros son los que más ha visto cada usuario)
> - **users.dat**: Relaciona el id de un usuario con el id de su cuenta de twitter (A partir de aquí podemos obtener los tweets de cada usuario para el análisis de sentimiento)

In [4]:

```
ratings_file = "/content/drive/MyDrive/Mineria/MovieTweetings/ratings.txt"
movies_file = "/content/drive/MyDrive/Mineria/MovieTweetings/movies.txt"
users_file = "/content/drive/MyDrive/Mineria/MovieTweetings/users.txt"

ratings_ds = pd.read_csv(ratings_file, sep='::', names=['user id', 'movie id', 'rating'
, 'rating timestamp'], dtype=object, engine='python')
movies_ds = pd.read_csv(movies_file, sep='::', names=['movie id', 'movie title', 'genr
e'], dtype=object, engine='python')
users_ds = pd.read_csv(users_file, sep='::', names=['user id', 'twitter id'], dtype=obj
ect, engine='python')
```

In [5]:

```
## Ratings dataset example
print(ratings_ds.shape)
ratings_ds.head()
```

```
(903946, 4)
```

Out[5]:

	user id	movie id	rating	rating timestamp
0	1	0114508	8	1381006850
1	2	0499549	9	1376753198
2	2	1305591	8	1376742507
3	2	1428538	1	1371307089
4	3	0075314	1	1595468524

In [6]:

```
## Movies dataset example
print(movies_ds.shape)
movies_ds.head()
```

```
(37248, 3)
```

Out[6]:

	movie id	movie title	genre
0	0000008	Edison Kinetoscopic Record of a Sneeze (1894)	Documentary|Short
1	0000010	La sortie des usines Lumière (1895)	Documentary|Short
2	0000012	The Arrival of a Train (1896)	Documentary|Short
3	25	The Oxford and Cambridge University Boat Race ...	NaN
4	0000091	Le manoir du diable (1896)	Short|Horror

In [7]:

```
# Limpiamos el dataset de películas de valores vacíos
print(np.count_nonzero(movies_ds.isnull()))
movies_ds = movies_ds.dropna()
movies_ds.shape
```

```
100
```

Out[7]:

```
(37148, 3)
```

In [8]:

```
movies_ds.head()
```

Out[8]:

	movie id	movie title	genre
0	0000008	Edison Kinetoscopic Record of a Sneeze (1894)	Documentary|Short
1	0000010	La sortie des usines Lumière (1895)	Documentary|Short
2	0000012	The Arrival of a Train (1896)	Documentary|Short
4	0000091	Le manoir du diable (1896)	Short|Horror
5	0000131	Une nuit terrible (1896)	Short|Comedy|Horror

In [9]:

```
## Users dataset example
print(users_ds.shape)
users_ds.head()
```

```
(70550, 2)
```

Out[9]:

	user id	twitter id
0	1	139564917
1	2	17528189
2	3	522540374
3	4	475571186
4	5	215022153

2. Preprocesamiento del dataset

- Para construir el recomendador tendremos en cuenta el número de películas de cada género que ha visto cada usuario.
- Para ello, a partir del dataset original construiremos una tabla con una fila por cada usuario y tantas columnas como géneros haya.

In [10]:

```
## Primero recorremos todas las peliculas del dataset para obtener todos los géneros
genres_dict = {'user id':[]}
for movie in movies_ds.to_numpy():
  for genre in movie[2].split('|'):
    genres_dict.update({genre:0})

print(genres_dict)
```

```
{'user id': [], 'Documentary': 0, 'Short': 0, 'Horror': 0, 'Comedy': 0, 'A
ction': 0, 'Adventure': 0, 'Fantasy': 0, 'Sci-Fi': 0, 'Crime': 0, 'Wester
n': 0, 'Drama': 0, 'Romance': 0, 'History': 0, 'Family': 0, 'War': 0, 'Spo
rt': 0, 'Biography': 0, 'Mystery': 0, 'Thriller': 0, 'Animation': 0, 'Musi
c': 0, 'Musical': 0, 'Film-Noir': 0, 'Adult': 0, 'Talk-Show': 0, 'News':
0, 'Reality-TV': 0, 'Game-Show': 0}
```

In [11]:

```
# Comenzamos a construir el DataFrame
user_genres_ds = pd.DataFrame(data=genres_dict)
user_genres_ds
```

Out[11]:

user id	Documentary	Short	Horror	Comedy	Action	Adventure	Fantasy	Sci-Fi	Crime	We

In [ ]:

```
### Recorremos todos los usuarios y sus peliculas vistas ###

# Recorrer usuarios
for i in range(users_ds.count()[0]):
  user_dict = genres_dict.copy()
  user_dict.update({'user id':i})

  # Recorrer peliculas vistas por cada usuario
  for movie in ratings_ds[ratings_ds['user id'] == i].to_numpy()[:,1]:
    if movies_ds[movies_ds['movie id'] == movie]['movie id'].empty:
      continue

    # Recorrer generos de cada pelicula
    for genre in movies_ds[movies_ds['movie id'] == movie]['genre'].to_numpy()[0].split
('|'):
      aux = user_dict.get(genre)
      user_dict.update({genre:aux+1})

  user_genres_ds.loc[i] = user_dict
```

In [ ]:

```
user_genres_ds
```

In [ ]:

```
# Almacenamos el DataFrame resultante en un fichero csv para no tener que volver a gene
rarlo más
user_genres_ds['user id'] = user_genres_ds['user id'].astype(int)
user_genres_ds.to_csv('/content/drive/MyDrive/Mineria/MovieTweetings/user_genres.csv',
index=False)
```

3. Sistema recomendador (Método K-NN)

- Con el DataFrame creado anteriormente podemos entrenar un modelo de K-vecinos-más-cercanos (K-NN) que para cada usuario encuentre los usuarios que más se asemejen a sus intereses.
- Este método tratará los elementos de la lista de géneros más vistos por cada usuario como vectores numéricos y calculará la distancia entre ellos, haciendo que los usuarios con gustos más similares se encuentren a poca distancia.

In [12]:

```
## Cargamos los datos
data = pd.read_csv('/content/drive/MyDrive/Mineria/MovieTweetings/user_genres.csv')
data = data.drop([0])
data
```

Out[12]:

	user id	Documentary	Short	Horror	Comedy	Action	Adventure	Fantasy	Sci-Fi	Crim
1	1	0	0	1	0	1	0	0	1	
2	2	0	0	1	0	2	2	2	2	
3	3	0	0	3	6	3	3	3	1	
4	4	0	0	0	0	0	0	0	0	
5	5	0	0	0	3	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	
70545	70545	0	0	50	17	26	31	19	34	
70546	70546	0	0	0	0	1	0	0	1	
70547	70547	0	0	1	1	0	0	0	0	
70548	70548	0	0	8	24	23	11	7	6	1
70549	70549	0	0	0	0	1	1	0	0	

70549 rows × 29 columns

- Con los datos cargados, creamos y entrenamos el modelo:

In [13]:

```
### Entrenamos el modelo K-NN
from sklearn.neighbors import NearestNeighbors

X = data.iloc[:,1:].values # Lista del número de géneros vistos, quitando el user_id

classifier = NearestNeighbors()
classifier.fit(X)
```

Out[13]:

```
NearestNeighbors(algorithm='auto', leaf_size=30, metric='minkowski',
                 metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                 radius=1.0)
```

- Una vez entrenado el modelo podemos probarlo:

In [14]:

```
uid = 3
li = classifier.kneighbors([X[uid-1]],n_neighbors=5,return_distance=False)
li = np.delete(li[0], 0) # Eliminamos al propio usuario
li = list(map(lambda x : x + 1, li)) # Sumar 1 para hacer coincidir los indices de la t
abla con el user_id
```

In [15]:

```
print('Usuarios similares al usuario nº{}: {}'.format(uid, li))
```

```
Usuarios similares al usuario nº3: [6101, 2616, 47348, 6063]
```

In [16]:

```
data[data['user id'] == uid]
```

Out[16]:

	user id	Documentary	Short	Horror	Comedy	Action	Adventure	Fantasy	Sci-Fi	Crime	W
3	3	0	0	3	6	3	3	3	1	11	

In [17]:

```
data[data['user id'] == li[0]]
```

Out[17]:

	user id	Documentary	Short	Horror	Comedy	Action	Adventure	Fantasy	Sci-Fi	Crime
6101	6101	0	0	3	6	3	2	1	1	7

In [18]:

```
current_user = ratings_ds[ratings_ds['user id']==str(uid)]['movie id'].values
similar_user = ratings_ds[ratings_ds['user id']==str(li[0])]['movie id'].values

print('Ids peliculas vistas por usuario {}: \n{}'.format(uid, current_user))
print('\nIds peliculas vistas por usuario similar ({}): \n{}'.format(li[0], similar_use
r))
```

```
Ids peliculas vistas por usuario 3:
['0075314' '0102926' '0114369' '0118715' '0120737' '0208092' '0358273'
 '0477348' '10039344' '1051906' '1568346' '2278388' '6199572' '6723592'
 '6751668' '7131622' '7975244' '7984734' '8367814' '8579674' '8946378']

Ids peliculas vistas por usuario similar (6101):
['0118971' '0154506' '0166924' '0256380' '1038988' '1206543' '1229340'
 '1242422' '1291150' '1294970' '1535109' '1614989' '1800241' '1939659'
 '2106476' '2140373' '2193215' '2226417' '2278388']
```

In [19]:

```
movies_list = np.array([movie for movie in similar_user if movie not in current_user])
print('\nPeliculas a recomendar: \n{}'.format(movies_list))
```

```
Peliculas a recomendar:
['0118971' '0154506' '0166924' '0256380' '1038988' '1206543' '1229340'
 '1242422' '1291150' '1294970' '1535109' '1614989' '1800241' '1939659'
 '2106476' '2140373' '2193215' '2226417']
```

In [20]:

```
print('Titulos (y géneros) de las peliculas recomendadas: \n')
movie_titles = []
movie_genres = []
for id in movies_list:
  title = movies_ds[movies_ds['movie id']==id]['movie title'].values
  genre = movies_ds[movies_ds['movie id']==id]['genre'].values
  movie_titles.append(title[0])
  movie_genres.append(genre[0])

for i in range(len(movie_titles)):
  print('{}: \t({})'.format(movie_titles[i], movie_genres[i]))
```

```
Titulos (y géneros) de las peliculas recomendadas:

The Devil's Advocate (1997):    (Drama|Mystery|Thriller)
Following (1998):       (Crime|Mystery|Thriller)
Mulholland Dr. (2001):  (Drama|Mystery|Thriller)
Shallow Hal (2001):     (Comedy|Drama|Fantasy|Romance)
[Rec] (2007):   (Horror|Mystery|Thriller)
Out of the Furnace (2013):      (Crime|Drama|Thriller)
Anchorman 2: The Legend Continues (2013):       (Comedy)
Celda 211 (2009):       (Action|Crime|Drama|Thriller)
Teenage Mutant Ninja Turtles (2014):    (Action|Adventure|Comedy|Sci-Fi)
The Angriest Man in Brooklyn (2014):    (Comedy|Drama)
Captain Phillips (2013):        (Biography|Drama|Thriller)
Hodejegerne (2011):     (Action|Crime|Thriller)
American Hustle (2013):         (Crime|Drama)
Carrie (2013):  (Drama|Horror)
Jagten (2012):  (Drama)
Saving Mr. Banks (2013):        (Biography|Comedy|Drama|Music)
The Counselor (2013):   (Crime|Drama|Thriller)
Insidious: Chapter 2 (2013):    (Horror|Mystery|Thriller)
```

- Podemos agrupar todo esto en una funcion:

In [21]:

```
'''
Funcion que recibe un id de usuario del dataset y el numero de usuarios similares que s
e desea,
ejecuta el modelo K-NN para conseguir los usuarios mas similares
y devuelve una lista con las peliculas que los usuarios similares han visto pero el usu
ario principal no.
'''
def movie_recommender(id, num_neighbors):
  X = data.iloc[:,1:].values # Generos por usuario (sin user id)

  ## K-NN
  li = classifier.kneighbors([X[int(id)-1]], n_neighbors=num_neighbors, return_distance
=False)
  li = np.delete(li[0], 0)
  li = list(map(lambda x : x + 1, li))

  movie_list = []
  current_user = ratings_ds[ratings_ds['user id']==str(id)]['movie id'].values

  ## Recorrer las peliculas vistas de todos los usuarios similares
  for i in range(len(li)):
      similar_user = ratings_ds[ratings_ds['user id']==str(li[i])]['movie id'].values
      movies = np.array([movie for movie in similar_user if movie not in current_user])
      for movie in movies:
        movie_list.append(movie)

  movie_list = list(dict.fromkeys(movie_list)) # Eliminar peliculas duplciadas

  movie_titles = []
  movie_genres = []
  for id in movie_list:
    title = movies_ds[movies_ds['movie id']==id]['movie title'].values
    genre = movies_ds[movies_ds['movie id']==id]['genre'].values

    movie_titles.append(title[0])
    movie_genres.append(genre[0])


  return movie_list, movie_titles, movie_genres
```

Con todo lo anterior, tenemos un sitema recomendador que recomienda películas en función de lo visto por usuarios con gustos similares

El siguiente paso es obtener el estado de ánimo del usuario a traves de sus tweets

4. Análisis de sentimiento de tweets (tweepy y red neuronal)

- El primer paso será ser capaces de extraer tweets de cada usuario. Para ello usaremos el archivo movies.dat que relaciona cada usuario con su id de twitter.
- Usando la API de Twitter, mediante la biblioteca Tweepy, podremos extraer tweets de una cuenta a través de su id

In [22]:

```
### Autentificarse para usar la app de tweepy
consumer_key = '----'
consumer_secret = '----'
access_token = '----'
access_token_secret = '----'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)
```

In [23]:

```
### Funciones auxiliares para trabajar con el dataset y Tweepy

# Devuelve el id de twitter correspondiente a un usuario del dataset
def get_twitterID(user_id):
  return users_ds[users_ds['user id']==user_id]['twitter id'].values[0]

# Devuelve el nombre de usuario (@ + user) asociado a un id de twitter
def get_user(id):
  try:
    u = api.get_user(int(get_twitterID(id)))
    return u.screen_name
  except:
    print('User not available')
```

- Para el análisis de sentimiento usaremos la red neuronal que entrenamos en otro notebook a parte

> Para usarla simplemente tenemos que importar la capa de vectorizacion de textos y el modelo de la red:

In [24]:

```
## A pesar de usar un modelo importado, es necesario volver a definir la funcion de est
andarizacion
@tf.keras.utils.register_keras_serializable()
def custom_standardization(input_data):
  lowercase = tf.strings.lower(input_data)

  # Eliminar menciones (@username), enlaces, o caracteres especiales
  clean_tweet = tf.strings.regex_replace(lowercase,
                            '(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|(www.\S+)'
, ' ')
  return clean_tweet

## Importamos la capa de TextVectorization
from_disk = pickle.load(open('/content/drive/MyDrive/Mineria/tweet-vectorizeLayer.pkl',
'rb'))
new_v = TextVectorization.from_config(from_disk['config'])
# Aunque ya está entrenada, debido a problemas con keras, es necesario llamar a adapt()
con datos cualquiera para que funcione
new_v.adapt(tf.data.Dataset.from_tensor_slices(["xyz"]))
new_v.set_weights(from_disk['weights'])

## Importamos el modelo de red neuronal
loaded_model = tf.keras.models.load_model('/content/drive/MyDrive/Mineria/tweet-sentime
nt-NN')
```

- Con todo esto ya podemos extraer tweets de un usuario con Tweepy e introducirlos en la red neuronal para inferir un sentimiento

In [25]:

```
'''
Funcion que recibe un id de usuario del dataset, lo convierte a el correspondiente id d
e twitter,
extrae el numero de tweets indicado, analiza el sentimiento de todos ellos y finalment
e,
devuelve el nombre del usuario y el sentimiento mayoritario entre los tweets analizado
'''
def sentiment_from_tweets(id, tweet_num):
  user = get_user(str(id))

  if user == None:
    return (None, None)

  else:
    ## Extraer tweets
    try:
      tweets = api.user_timeline(screen_name=user, count=100, include_rts=False)
    except tweepy.TweepError:
      print('Private user')
      return(None, None)

    if tweets[0].lang != 'en': # Evitamos tweets en otro idioma. Nuestro clasificador e
stá entrenado en inglés
      print('Not in english')
      return(None, None)

    examples = [tweet.text for tweet in tweets[:tweet_num]]

    ## Analisis de sentimiento
    predictions = loaded_model.predict(examples)
    pos = 0
    neg = 0
    for p in predictions:
      if p >= 0.5:
        pos += 1
      else:
        neg += 1

    user_sentiment = 1 if pos > neg else -1


  return (user, user_sentiment)
```

In [26]:

```
id = input('User id: ')
num = int(input('Number of tweets: '))
user, sentiment = sentiment_from_tweets(id, num)

### MOSTRAR RESULTADOS
print('- User name: @{}'.format(user))
if sentiment == 1:
  print('- User sentiment: {}'.format('Positive'))
else:
  print('- User sentiment: {}'.format('Negative'))
```

```
User id: 1
Number of tweets: 10
- User name: @Waffaboy
- User sentiment: Positive
```

- Podemos ver los **tweets** extraidos para comparar el resultado:

In [27]:

```
user = get_user(str(id))
tweets = api.user_timeline(screen_name=user, count=100, include_rts=False)
for t in tweets[:10]:
  print(t.text, '\n')
```

```
@LeeDawsonPT The duality of entertainment

@alanswan a receipt of all the non essential buys from amazon- mainly the
harmonica and cocktail mixers. A lethal combo

Fanatastic campaign launched by SIRO. Life is different now and we're all
adjusting as best we can! #StayHome https://t.co/eeuZ0Ig0ai

@hoeyannie @labour Congrats!!

@NianticHelp What about the special research tasks with time limits? Hard
to get out of the home to watch very spec… https://t.co/NgIyNKZWAI

@LeeDawsonPT go to town with it https://t.co/j4B9L8jdi1

@Ghost_Fl0wer Hope that it turns out you don't have OC. Here if you need a
nything

@LfcHarsh Repetitive music boosts productivity for both so that's prolly w
hy!

Brave and bold statement from British TV icon Phillip Schofield, nothing b
ut support to him and his family should b… https://t.co/wbhTnb2oMj

I've had people call me all sorts of things but I think my favourite has t
o be the person who thought "Airline" was… https://t.co/PoQUb0Ad85
```

5. Recomendador + Analisis de sentimiento

- Ya tenemos el recomendador y la forma de analizar el sentimiento del usuario
- Con esto podemos juntar ambas técnicas para filtrar las recomendaciones por sentimiento, pero neceesitamos una forma de asignar sentimiento a las películas. Probaremos dos formas:

> - **Enfoque principal**: Asignar a cada género un sentimiento (ej: Drama -> negativo, Comedia -> positivo) y asignar a cada peli una puntuación en función de sus géneros
> - **Enfoque "experimental"**: Conseguir la sinopsis de cada peli y utilizar la red neuronal para clasificar cada sinopsis por sentimiento, asignando ese sentimiento a la pelicula.

- **Enfoque principal**:

> Este enfoque consitirá en asignar a cada género un valor (1: positivo, -1: negativo, 0: neutro) y para cada pelicula asignar un sentimiento en funcion de la suma de los valores de cada uno de sus generos
> - Si la suma resulta > 0, la pelicula será positiva; si resulta \< 0, será negativa y si fuese 0, sería una peli neutra que podemos recomendar a cualquier usuario

In [28]:

```
# Generos del dataset
generos = np.array(user_genres_ds.keys()[1:])
print(generos)
```

```
['Documentary' 'Short' 'Horror' 'Comedy' 'Action' 'Adventure' 'Fantasy'
 'Sci-Fi' 'Crime' 'Western' 'Drama' 'Romance' 'History' 'Family' 'War'
 'Sport' 'Biography' 'Mystery' 'Thriller' 'Animation' 'Music' 'Musical'
 'Film-Noir' 'Adult' 'Talk-Show' 'News' 'Reality-TV' 'Game-Show']
```

In [29]:

```
# Asignar a cada genero un valor (1: positivo, -1: negativo, 0: neutro)
valores = np.array([0, 0, -1, 1, 1, 1, 1, 1, -1, 1, -1, 1, 0, 1, -1, 1, 0, -1, -1, 0, 1
, 1, -1, -1, 1, 0, 1, 1])
```

In [30]:

```
sentimiento_generos = dict(zip(generos, valores))
print(sentimiento_generos)
```

```
{'Documentary': 0, 'Short': 0, 'Horror': -1, 'Comedy': 1, 'Action': 1, 'Ad
venture': 1, 'Fantasy': 1, 'Sci-Fi': 1, 'Crime': -1, 'Western': 1, 'Dram
a': -1, 'Romance': 1, 'History': 0, 'Family': 1, 'War': -1, 'Sport': 1, 'B
iography': 0, 'Mystery': -1, 'Thriller': -1, 'Animation': 0, 'Music': 1,
'Musical': 1, 'Film-Noir': -1, 'Adult': -1, 'Talk-Show': 1, 'News': 0, 'Re
ality-TV': 1, 'Game-Show': 1}
```

In [31]:

```
for g, s in sentimiento_generos.items():
  if s == 1:
    print('{} -> {}'.format(g, 'POS'))
  elif s == -1:
    print('{} -> {}'.format(g, 'NEG'))
  else:
    print('{} -> {}'.format(g, 'NEUTRO'))

print('\nPositivos: ', np.count_nonzero(valores == 1))
print('Negativos: ', np.count_nonzero(valores == -1))
print('Neutros: ', np.count_nonzero(valores == 0))
```

```
Documentary -> NEUTRO
Short -> NEUTRO
Horror -> NEG
Comedy -> POS
Action -> POS
Adventure -> POS
Fantasy -> POS
Sci-Fi -> POS
Crime -> NEG
Western -> POS
Drama -> NEG
Romance -> POS
History -> NEUTRO
Family -> POS
War -> NEG
Sport -> POS
Biography -> NEUTRO
Mystery -> NEG
Thriller -> NEG
Animation -> NEUTRO
Music -> POS
Musical -> POS
Film-Noir -> NEG
Adult -> NEG
Talk-Show -> POS
News -> NEUTRO
Reality-TV -> POS
Game-Show -> POS

Positivos:  14
Negativos:  8
Neutros:  6
```

In [32]:

```
'''
Funcion que recibe una lista de generos y un diccionario con el sentimiento asociado a
 cada genero.
Asigna a cada pelicula un genero en funcion de la puntuacion resultante de sus generos
Devuelve una lista con un sentimiento por pelicula
'''
def filter_genres_sentiment(movie_genres, genres_sentiment):
  movie_sentiment = []

  for movie in movie_genres:
    sum = 0
    for genre in movie.split('|'):
      sum += genres_sentiment.get(genre)

    sentiment = 1 if sum > 0 else -1 if sum < 0 else 0
    movie_sentiment.append(sentiment)

  return movie_sentiment
```

In [71]:

```
user_id = '5'

## Analisis de sentimiento
user, sentiment = sentiment_from_tweets(user_id, 10)

## Recomendador de pelis
movie_list, movie_titles, movie_genres = movie_recommender(user_id, 7)

## Sentimiento de peliculas (por generos)
movie_genre_sentiment = filter_genres_sentiment(movie_genres, sentimiento_generos)

print('- User name: @{}'.format(user))
if sentiment == 1:
  print('- User sentiment: {}'.format('Positive'))
else:
  print('- User sentiment: {}'.format('Negative'))

print('- Recomendations by similar users:')
for i in range(len(movie_titles)):
  print('\t -', movie_titles[i], ": (", movie_genres[i], "): ", movie_genre_sentiment[i
])

print('- Recomendations by users and genre sentiment:')
recomendations = []
for i in range(len(movie_titles)):
  if movie_genre_sentiment[i] == sentiment or movie_genre_sentiment[i] == 0:
    recomendations.append(movie_titles[i])

for i in recomendations:
  print('\t -', i)
```

```
- User name: @theashoxford
- User sentiment: Positive
- Recomendations by similar users:
         - Cast Away (2000) : ( Adventure|Drama|Romance ):  1
         - The Irishman (2019) : ( Biography|Crime|Drama|History|Thriller
):  -1
         - Once Upon a Time ...in Hollywood (2019) : ( Comedy|Drama ):  0
         - Marriage Story (2019) : ( Comedy|Drama ):  0
         - The King (2019) : ( Biography|Drama|History|Romance|War ):  -1
         - The Two Popes (2019) : ( Biography|Comedy|Drama ):  0
         - A Perfect World (1993) : ( Crime|Drama|Thriller ):  -1
         - Jersey Girl (2004) : ( Comedy|Drama|Romance ):  1
         - My One and Only (2009) : ( Adventure|Biography|Comedy|Drama|Rom
ance ):  1
         - Macbeth (2015) : ( Drama|History|War ):  -1
         - Amadeus (1984) : ( Biography|Drama|History|Music ):  0
         - Ard al-Khof (1999) : ( Thriller ):  -1
         - Hotel Rwanda (2004) : ( Biography|Drama|History|War ):  -1
         - Eat Pray Love (2010) : ( Drama|Romance ):  0
         - American Hustle (2013) : ( Crime|Drama ):  -1
         - Birdman (2014) : ( Comedy|Drama ):  0
         - Me and Earl and the Dying Girl (2015) : ( Comedy|Drama|Romance
):  1
         - Persepolis (2007) : ( Animation|Biography|Drama|History|War ):
-1
         - Me Before You (2016) : ( Drama|Romance ):  0
         - The Dressmaker (2015) : ( Comedy|Drama ):  0
         - A Hologram for the King (2016) : ( Comedy|Drama|Romance ):  1
         - Septembers of Shiraz (2015) : ( Thriller ):  -1
         - Lady Bird (2017) : ( Comedy|Drama ):  0
         - Silver Linings Playbook (2012) : ( Comedy|Drama|Romance ):  1
         - Captain Phillips (2013) : ( Biography|Drama|Thriller ):  -1
         - Crazy, Stupid, Love. (2011) : ( Comedy|Drama|Romance ):  1
         - 12 Years a Slave (2013) : ( Biography|Drama|History ):  -1
         - Nightcrawler (2014) : ( Crime|Drama|Thriller ):  -1
         - On the Waterfront (1954) : ( Crime|Drama|Thriller ):  -1
         - Annie Hall (1977) : ( Comedy|Romance ):  1
         - When Harry Met Sally... (1989) : ( Comedy|Drama|Romance ):  1
         - La vita è bella (1997) : ( Comedy|Drama|Romance|War ):  0
         - Lincoln (2012) : ( Biography|Drama|History|War ):  -1
         - The Master (2012) : ( Drama ):  -1
         - Predestination (2014) : ( Drama|Mystery|Sci-Fi|Thriller ):  -1
- Recomendations by users and genre sentiment:
         - Cast Away (2000)
         - Once Upon a Time ...in Hollywood (2019)
         - Marriage Story (2019)
         - The Two Popes (2019)
         - Jersey Girl (2004)
         - My One and Only (2009)
         - Amadeus (1984)
         - Eat Pray Love (2010)
         - Birdman (2014)
         - Me and Earl and the Dying Girl (2015)
         - Me Before You (2016)
         - The Dressmaker (2015)
         - A Hologram for the King (2016)
         - Lady Bird (2017)
         - Silver Linings Playbook (2012)
         - Crazy, Stupid, Love. (2011)
         - Annie Hall (1977)
         - When Harry Met Sally... (1989)
         - La vita è bella (1997)
```

- **Enfoque "experimental"**:

> En este enfoque probaremos a clasificar a las peliculas no por sus géneros, si no por el sentimiento que la red neuronal infiera de su sinopsis
>
> Para acceder a la sinopsis de una pelicula necesitaremos hacer uso de la API de IMDb, mediante la biblioteca IMDbpy (https://github.com/alberanid/imdbpy).
> - Usando la API podremos conseguir la sinopsis de una pelicula usando el id correspondiente de la pelicula en nuestro dataset

In [33]:

```
## Instalamos el modulo imdbpy desde su repositorio
!pip install git+https://github.com/alberanid/imdbpy
```

```
Collecting git+https://github.com/alberanid/imdbpy
  Cloning https://github.com/alberanid/imdbpy to /tmp/pip-req-build-hpfulu
px
  Running command git clone -q https://github.com/alberanid/imdbpy /tmp/pi
p-req-build-hpfulupx
Requirement already satisfied: SQLAlchemy in /usr/local/lib/python3.7/dist
-packages (from IMDbPY==2021.5.21) (1.4.15)
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packa
ges (from IMDbPY==2021.5.21) (4.2.6)
Requirement already satisfied: importlib-metadata; python_version < "3.8"
in /usr/local/lib/python3.7/dist-packages (from SQLAlchemy->IMDbPY==2021.
5.21) (4.0.1)
Requirement already satisfied: greenlet!=0.4.17; python_version >= "3" in
/usr/local/lib/python3.7/dist-packages (from SQLAlchemy->IMDbPY==2021.5.2
1) (1.1.0)
Requirement already satisfied: typing-extensions>=3.6.4; python_version <
"3.8" in /usr/local/lib/python3.7/dist-packages (from importlib-metadata;
python_version < "3.8"->SQLAlchemy->IMDbPY==2021.5.21) (3.7.4.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata; python_version < "3.8"->SQLAlchemy->IMD
bPY==2021.5.21) (3.4.1)
Building wheels for collected packages: IMDbPY
  Building wheel for IMDbPY (setup.py) ... done
  Created wheel for IMDbPY: filename=IMDbPY-2021.5.21-cp37-none-any.whl si
ze=299341 sha256=c99b6aff4d686be0dad22c2a43fda6a9e938d678daa5840d7376c3706
6f53cb7
  Stored in directory: /tmp/pip-ephem-wheel-cache-hye7rao4/wheels/0f/09/6
1/190df5e0276765680540f1562f2abca80e725a7e48595e993f
Successfully built IMDbPY
Installing collected packages: IMDbPY
Successfully installed IMDbPY-2021.5.21
```

In [34]:

```
from imdb import IMDb
```

In [69]:

```
'''
Funcion que recibe una lista de peliculas y un clasificador ya entrenado,
recupera la sinopsis de cada pelicula mediante la API de IMDb y, usando el clasificado
r,
analiza el sentimiento de cada sinopsis y asigna ese sentimiento a cada pelicula.
Devuelve una lista con un sentimiento por pelicula.
'''
def filter_plot_sentiment(movie_list, model):
  ia = IMDb() # API IMDb
  movie_sentiment = []

  for id in movie_list:
    plot = []
    try:
      movie = ia.get_movie(id)
      plot.append(movie.get('plot')[0])
    except:
      continue

    sentiment = loaded_model.predict(plot)[0][0]
    sentiment = 1 if sentiment >= 0.5 else -1

    movie_sentiment.append(sentiment)

  return movie_sentiment
```

In [76]:

```
user_id = '5'

## Analisis de sentimiento
user, sentiment = sentiment_from_tweets(user_id, 10)

## Recomendador de pelis
movie_list, movie_titles, movie_genres = movie_recommender(user_id, 7)

## Sentimiento de peliculas (por sinopsis)
movie_plot_sentiment = filter_plot_sentiment(movie_list, loaded_model)

print('- User name: @{}'.format(user))
if sentiment == 1:
  print('- User sentiment: {}'.format('Positive'))
else:
  print('- User sentiment: {}'.format('Negative'))

print('- Recomendations by similar users:')
for i in range(len(movie_titles)):
  print('\t -', movie_titles[i], ": (", movie_genres[i], "): ", movie_plot_sentiment[i
])

print('- Recomendations by users and plot sentiment:')
recomendations = []
for i in range(len(movie_titles)):
  if movie_plot_sentiment[i] == sentiment:
    recomendations.append(movie_titles[i])

for i in recomendations:
  print('\t -', i)
```

```
- User name: @theashoxford
- User sentiment: Positive
- Recomendations by similar users:
	- Cast Away (2000) : ( Adventure|Drama|Romance ):  1
	- The Irishman (2019) : ( Biography|Crime|Drama|History|Thriller
):  1
	- Once Upon a Time ...in Hollywood (2019) : ( Comedy|Drama ):  1
	- Marriage Story (2019) : ( Comedy|Drama ):  1
	- The King (2019) : ( Biography|Drama|History|Romance|War ):  -1
	- The Two Popes (2019) : ( Biography|Comedy|Drama ):  1
	- A Perfect World (1993) : ( Crime|Drama|Thriller ):  -1
	- Jersey Girl (2004) : ( Comedy|Drama|Romance ):  1
	- My One and Only (2009) : ( Adventure|Biography|Comedy|Drama|Rom
ance ):  -1
	- Macbeth (2015) : ( Drama|History|War ):  1
	- Amadeus (1984) : ( Biography|Drama|History|Music ):  -1
	- Ard al-Khof (1999) : ( Thriller ):  -1
	- Hotel Rwanda (2004) : ( Biography|Drama|History|War ):  -1
	- Eat Pray Love (2010) : ( Drama|Romance ):  -1
	- American Hustle (2013) : ( Crime|Drama ):  1
	- Birdman (2014) : ( Comedy|Drama ):  1
	- Me and Earl and the Dying Girl (2015) : ( Comedy|Drama|Romance
):  -1
	- Persepolis (2007) : ( Animation|Biography|Drama|History|War ):
1
	- Me Before You (2016) : ( Drama|Romance ):  1
	- The Dressmaker (2015) : ( Comedy|Drama ):  1
	- A Hologram for the King (2016) : ( Comedy|Drama|Romance ):  -1
	- Septembers of Shiraz (2015) : ( Thriller ):  -1
	- Lady Bird (2017) : ( Comedy|Drama ):  1
	- Silver Linings Playbook (2012) : ( Comedy|Drama|Romance ):  1
	- Captain Phillips (2013) : ( Biography|Drama|Thriller ):  1
	- Crazy, Stupid, Love. (2011) : ( Comedy|Drama|Romance ):  1
	- 12 Years a Slave (2013) : ( Biography|Drama|History ):  -1
	- Nightcrawler (2014) : ( Crime|Drama|Thriller ):  1
	- On the Waterfront (1954) : ( Crime|Drama|Thriller ):  1
	- Annie Hall (1977) : ( Comedy|Romance ):  1
	- When Harry Met Sally... (1989) : ( Comedy|Drama|Romance ):  -1
	- La vita è bella (1997) : ( Comedy|Drama|Romance|War ):  1
	- Lincoln (2012) : ( Biography|Drama|History|War ):  1
	- The Master (2012) : ( Drama ):  1
	- Predestination (2014) : ( Drama|Mystery|Sci-Fi|Thriller ):  1
- Recomendations by users and plot sentiment:
	- Cast Away (2000)
	- The Irishman (2019)
	- Once Upon a Time ...in Hollywood (2019)
	- Marriage Story (2019)
	- The Two Popes (2019)
	- Jersey Girl (2004)
	- Macbeth (2015)
	- American Hustle (2013)
	- Birdman (2014)
	- Persepolis (2007)
	- Me Before You (2016)
	- The Dressmaker (2015)
	- Lady Bird (2017)
	- Silver Linings Playbook (2012)
	- Captain Phillips (2013)
	- Crazy, Stupid, Love. (2011)
	- Nightcrawler (2014)
	- On the Waterfront (1954)
```

```
        - Annie Hall (1977)
        - La vita è bella (1997)
        - Lincoln (2012)
        - The Master (2012)
        - Predestination (2014)
```

6. Recomendador final

- Con todo ya listo, podemos comenzar a recomendar peliculas
- Introduciendo el id del usuario al que recomendar, se extraeran sus 10 ultimos tweets de los que se asignará un sentimiento al usuario (en base a la moda de los sentimientos de cada tweet), se buscarán usuarios con gustos similares y se devolverá una lista de peliculas vistas por los usuarios similares pero no por el principal. Finalmente, esta lista de películas se filtrará teniendo en cuenta el sentimiento del usuario y de cada pelicula (el sentimiento de las peliculas se analizará de las dos manera explicadas anteriormente).

In [75]:

```
def sistema_recomendador(user_id=None, max_recomendations=50, num_tweets=10, num_neighb
ors=7,
                         genres_sentiment=sentimiento_generos, model=loaded_model,
                         show_genres=False, show_sentiment=False):
  if user_id == None:
    user_id = input('User ID: ')

  ## Analisis de sentimiento
  user, sentiment = sentiment_from_tweets(user_id, num_tweets)
  if user == None:
    return

  ## Recomendador de pelis
  movie_list, movie_titles, movie_genres = movie_recommender(user_id, num_neighbors)

  ## Sentimiento de peliculas (por generos)
  movie_genre_sentiment = filter_genres_sentiment(movie_genres, genres_sentiment)

  ## Sentimiento de peliculas (por sinopsis)
  movie_plot_sentiment = filter_plot_sentiment(movie_list, model)


  ## Mostrar resultados:
  print('----------------------------------------------------------------')
  print('- User name: @{}'.format(user))
  if sentiment == 1:
    print('- User sentiment: {}'.format('Positive'))
  else:
    print('- User sentiment: {}'.format('Negative'))

  print('- Recomendations by similar users:')
  for i in range(len(movie_titles[:max_recomendations])):
    if show_genres and show_sentiment:
      print('\t -', movie_titles[i], ", GENRES: ", movie_genres[i], ", Genres Sentimen
t: ", movie_genre_sentiment[i], ", Plot Sentiment: ", movie_plot_sentiment[i])
    elif show_genres:
      print('\t -', movie_titles[i], ": ", movie_genres[i])
    elif show_sentiment:
      print('\t -', movie_titles[i], ", Genres Sentiment: ", movie_genre_sentiment[i],
", Plot Sentiment: ", movie_plot_sentiment[i])
    else:
      print('\t -', movie_titles[i])

  print('- Recomendations by users and genre sentiment:')
  recomendations1 = []
  for i in range(len(movie_titles)):
    if movie_genre_sentiment[i] == sentiment or movie_genre_sentiment[i] == 0:
      recomendations1.append(movie_titles[i])

  if max_recomendations < len(recomendations1):
    recomendations1 = random.sample(recomendations1, max_recomendations)

  for i in recomendations1:
    print('\t -', i)

  print('- Recomendations by users and plot sentiment:')
  recomendations2 = []
  for i in range(len(movie_titles)):
    if movie_plot_sentiment[i] == sentiment:
      recomendations2.append(movie_titles[i])

  if max_recomendations < len(recomendations2):
    recomendations2 = random.sample(recomendations2, max_recomendations)

  for i in recomendations2:
    print('\t -', i)

  print('--------------------------------------------------------------\n')
```

In [74]:

```
sistema_recomendador(user_id=1, max_recomendations=3)
```

```
--------------------------------------------------------------
- User name: @Waffaboy
- User sentiment: Positive
- Recomendations by similar users:
         - The Purge: Election Year (2016)
         - The Colony (2013)
         - Lake Placid: Legacy (2018)
- Recomendations by users and genre sentiment:
         - Lake Placid: Legacy (2018)
         - Cloverfield (2008)
         - The Purge: Election Year (2016)
- Recomendations by users and plot sentiment:
         - Cloverfield (2008)
         - Morgan (2016)
--------------------------------------------------------------
```

In [40]:

```
sistema_recomendador()
```

```
User ID: 71
--------------------------------------------------------------
- User name: @best6789
- User sentiment: Negative
- Recomendations by similar users:
         - Man of Steel (2013)
         - Riddick (2013)
         - Escape from L.A. (1996)
         - Star Trek Into Darkness (2013)
         - The Purge (2013)
         - X: First Class (2011)
         - The Wolverine (2013)
         - The Conjuring (2013)
         - Texas Chainsaw 3D (2013)
         - Aftershock (2012)
         - G.I. Joe: Retaliation (2013)
         - Pacific Rim (2013)
- Recomendations by users and genre sentiment:
         - The Purge (2013)
         - The Conjuring (2013)
         - Texas Chainsaw 3D (2013)
         - Aftershock (2012)
- Recomendations by users and plot sentiment:
         - Riddick (2013)
         - Escape from L.A. (1996)
         - X: First Class (2011)
         - The Conjuring (2013)
         - Aftershock (2012)
         - G.I. Joe: Retaliation (2013)
--------------------------------------------------------------
```

In [ ]:

```
for i in [1,10,16,19,21,23,24,25,89,97,9,33,35,71,91]:
  print('ID: ', i)
  sistema_recomendador(user_id=i, max_recomendations=3)
```

```
ID:  1
--------------------------------------------------------------
- User name: @Waffaboy
- User sentiment: Positive
- Recomendations by similar users:
         - The Purge: Election Year (2016)
         - The Colony (2013)
         - Lake Placid: Legacy (2018)
- Recomendations by users and genre sentiment:
         - Morgan (2016)
         - The Purge: Election Year (2016)
         - The Colony (2013)
- Recomendations by users and plot sentiment:
         - Cloverfield (2008)
         - Morgan (2016)
--------------------------------------------------------------

ID:  10
--------------------------------------------------------------
- User name: @LSBPsupport
- User sentiment: Positive
- Recomendations by similar users:
         - Night Train to Lisbon (2013)
         - The Loft (2014)
         - Vertigo (1958)
- Recomendations by users and genre sentiment:
- Recomendations by users and plot sentiment:
         - Night Train to Lisbon (2013)
         - Vertigo (1958)
--------------------------------------------------------------

ID:  16
--------------------------------------------------------------
- User name: @JoePranaitis
- User sentiment: Positive
- Recomendations by similar users:
         - The Last Stand (2013)
         - American Assassin (2017)
         - A Good Day to Die Hard (2013)
- Recomendations by users and genre sentiment:
         - A Good Day to Die Hard (2013)
         - 6 Underground (2019)
         - The Last Stand (2013)
- Recomendations by users and plot sentiment:
         - The Last Stand (2013)
         - American Assassin (2017)
         - 6 Underground (2019)
--------------------------------------------------------------

ID:  19
--------------------------------------------------------------
- User name: @ckerr84
- User sentiment: Positive
- Recomendations by similar users:
         - Nocturnal Animals (2016)
         - Temptation: Confessions of a Marriage Counselor (2013)
         - Kokuhaku (2010)
- Recomendations by users and genre sentiment:
- Recomendations by users and plot sentiment:
         - Nocturnal Animals (2016)
         - Kokuhaku (2010)
         - Miss Sloane (2016)
--------------------------------------------------------------

ID:  21
--------------------------------------------------------------
- User name: @AbdullahFarhad
- User sentiment: Positive
- Recomendations by similar users:
         - Forrest Gump (1994)
         - Brooklyn (2015)
         - Extraction (2020)
- Recomendations by users and genre sentiment:
         - The Gentlemen (2019)
         - El Camino: A Breaking Bad Movie (2019)
         - Wonder Boys (2000)
- Recomendations by users and plot sentiment:
         - Brooklyn (2015)
         - The Reunion (2011)
         - Before We Go (2014)
--------------------------------------------------------------

ID:  23
--------------------------------------------------------------
- User name: @julesrlavin
- User sentiment: Positive
- Recomendations by similar users:
         - Vaiana (2016)
         - Life (2017)
         - Love, Rosie (2014)
- Recomendations by users and genre sentiment:
         - Kingsman: The Secret Service (2014)
         - Who Framed Roger Rabbit (1988)
         - Home (2015)
- Recomendations by users and plot sentiment:
         - Toy Story 4 (2019)
         - Kingsman: The Secret Service (2014)
         - Home (2015)
--------------------------------------------------------------

ID:  24
--------------------------------------------------------------
- User name: @vlush
- User sentiment: Positive
- Recomendations by similar users:
         - Mission: Impossible - Rogue Nation (2015)
         - Salt (2010)
         - Fast & Furious 6 (2013)
- Recomendations by users and genre sentiment:
         - Mission: Impossible - Rogue Nation (2015)
         - Fast & Furious 6 (2013)
- Recomendations by users and plot sentiment:
         - Salt (2010)
         - Non-Stop (2014)
--------------------------------------------------------------

ID:  25
--------------------------------------------------------------
- User name: @MondayPewPew
- User sentiment: Positive
- Recomendations by similar users:
         - Jurassic World (2015)
         - Mad Max: Fury Road (2015)
         - The Revenant (2015)
- Recomendations by users and genre sentiment:
         - Ted 2 (2015)
         - Fast & Furious 6 (2013)
         - Batman Begins (2005)
- Recomendations by users and plot sentiment:
         - Batman Begins (2005)
         - Ted 2 (2015)
         - Jurassic World (2015)
--------------------------------------------------------------

ID:  89
--------------------------------------------------------------
- User name: @WRCretrocorner
- User sentiment: Positive
- Recomendations by similar users:
         - Once Upon a Time ...in Hollywood (2019)
         - Joker (2019)
         - Marriage Story (2019)
- Recomendations by users and genre sentiment:
         - Birds of Prey: And the Fantabulous Emancipation of One Harley Q
uinn (2020)
         - The Art of Racing in the Rain (2019)
         - Dawn of the Planet of the Apes (2014)
- Recomendations by users and plot sentiment:
         - LUV (2012)
         - The Gentlemen (2019)
         - Hunt for the Wilderpeople (2016)
--------------------------------------------------------------

ID:  97
--------------------------------------------------------------
- User name: @NeilSchloth
- User sentiment: Positive
- Recomendations by similar users:
         - Captain America: The First Avenger (2011)
         - Thor (2011)
         - The Avengers (2012)
- Recomendations by users and genre sentiment:
         - Ender's Game (2013)
         - Fantastic Four (2005)
         - The Avengers (2012)
- Recomendations by users and plot sentiment:
         - Thor (2011)
         - Transformers: Age of Extinction (2014)
         - Rogue One (2016)
--------------------------------------------------------------

ID:  9
--------------------------------------------------------------
- User name: @fahadturky
- User sentiment: Positive
- Recomendations by similar users:
         - Forrest Gump (1994)
         - The Usual Suspects (1995)
         - My Best Friend's Wedding (1997)
- Recomendations by users and genre sentiment:
         - My Best Friend's Wedding (1997)
         - Rush (2013)
         - One Piece: Stampede (2019)
- Recomendations by users and plot sentiment:
         - Ready or Not (2019)
         - Captain Phillips (2013)
         - Rush (2013)
--------------------------------------------------------------

ID:  33
--------------------------------------------------------------
- User name: @bailey__blaise
- User sentiment: Positive
- Recomendations by similar users:
         - The Last Stand (2013)
         - American Assassin (2017)
         - A Good Day to Die Hard (2013)
- Recomendations by users and genre sentiment:
         - American Assassin (2017)
         - Holiday (2014)
         - 6 Underground (2019)
- Recomendations by users and plot sentiment:
         - The Last Stand (2013)
         - American Assassin (2017)
         - 6 Underground (2019)
--------------------------------------------------------------

ID:  35
--------------------------------------------------------------
- User name: @AlexCapel
- User sentiment: Positive
- Recomendations by similar users:
         - Captain America: The First Avenger (2011)
         - The Revenant (2015)
         - Captain America: The Winter Soldier (2014)
- Recomendations by users and genre sentiment:
         - The Amazing Spider-Man (2012)
         - Ant-Man and the Wasp (2018)
         - Interstellar (2014)
- Recomendations by users and plot sentiment:
         - Interstellar (2014)
         - Pitch Perfect 2 (2015)
         - G.O.R.A. (2004)
--------------------------------------------------------------

ID:  71
--------------------------------------------------------------
- User name: @best6789
- User sentiment: Negative
- Recomendations by similar users:
         - Man of Steel (2013)
         - Riddick (2013)
         - Escape from L.A. (1996)
- Recomendations by users and genre sentiment:
         - Aftershock (2012)
         - The Conjuring (2013)
         - Texas Chainsaw 3D (2013)
- Recomendations by users and plot sentiment:
         - G.I. Joe: Retaliation (2013)
         - Aftershock (2012)
         - Escape from L.A. (1996)
--------------------------------------------------------------

ID:  91
```

```
--------------------------------------------------------------
- User name: @ItsGaryC
- User sentiment: Negative
- Recomendations by similar users:
         - 12 Years a Slave (2013)
         - Schindler's List (1993)
         - Dark Waters (2019)
- Recomendations by users and genre sentiment:
         - 12 Years a Slave (2013)
         - Schindler's List (1993)
         - Dark Waters (2019)
- Recomendations by users and plot sentiment:
         - 12 Years a Slave (2013)
--------------------------------------------------------------
```

In [ ]:

```
for i in np.random.randint(1, len(users_ds), 5):
  print('ID: ', i)
  sistema_recomendador(user_id=i, max_recomendations=3)
```

```
ID:  52773
Not in english
ID:  60689
User not available
ID:  25080

--------------------------------------------------------------
- User name: @ml_cfc1967
- User sentiment: Positive
- Recomendations by similar users:
         - The Cabin in the Woods (2011)
         - Evil Dead II (1987)
         - The Babysitter (2017)
- Recomendations by users and genre sentiment:
         - Evil Dead II (1987)
         - The Babysitter (2017)
         - The Cabin in the Woods (2011)
- Recomendations by users and plot sentiment:
         - The Cabin in the Woods (2011)
         - Evil Dead II (1987)
         - The Babysitter (2017)
--------------------------------------------------------------

ID:  5274
Not in english
ID:  57616

--------------------------------------------------------------
- User name: @jessejoeaz
- User sentiment: Positive
- Recomendations by similar users:
         - The Way (2010)
         - Y tu mamá también (2001)
         - Chef (2014)
- Recomendations by users and genre sentiment:
         - The Way (2010)
         - Chef (2014)
         - Hunt for the Wilderpeople (2016)
- Recomendations by users and plot sentiment:
         - Y tu mamá también (2001)
         - Chef (2014)
         - Hunt for the Wilderpeople (2016)
--------------------------------------------------------------
```