SISTEMA DE RECOMENDACIÓN Y PREDICCIÓN DE PELÍCULAS POR CONTENIDO Y VALORACIONES DE USUARIO

GRUPO_25 | ABP2_Tenores

Colmenares González, David Santiago

Sabucedo González, Alberto Mateo

Tenorio Costa, Juan

Introducción

El proyecto está enfocado hacia la recomendación y valoración de películas dada la gran demanda que éstas tienen en la actualidad gracias a plataformas de ocio como Netflix, HBO o Prime Video, quienes ofrecen y recomiendan películas a sus usuarios.

Se marcó como objetivo principal del proyecto la definición de una plataforma que fuese capaz de recomendar películas teniendo en cuenta las valoraciones que los usuarios pudieran hacer de éstas. Para cumplir con este objetivo, se desarrolló un sistema que permitiese recomendar en base a dos criterios distintos: el “análisis textual” y el “análisis de sentimientos”. La recomendación en base al análisis textual permite que el sistema recomiende películas teniendo en cuenta la similitud que éstas tienen entre sus sinopsis y género. Por otro lado, la recomendación en base al análisis de sentimientos, hace que el sistema sea capaz de recomendar teniendo en cuenta las valoraciones de las películas que los usuarios hayan ido realizando, ya sea mediante un comentario de texto o a través de una valoración positiva o negativa.

CRUDs asociados a datos

Carga y filtrado de datos

Para el desarrollo del trabajo, se optó por hacer uso de un dataset extraído de internet (url: https://www.kaggle.com/shivamb/netflix-shows) llamado *netflix_titles.csv*, el cual contiene películas y series de televisión de la plataforma de streaming Netflix.

Este archivo .csv lo incluimos dentro de la carpeta content para poder trabajar con él, cargándolo y guardándolo dentro de la variable originalData.

```
import pandas as pd
```

```
originalData = pd.read_csv('/content/netflix_titles.csv')
originalData.head(10)
```

	show_id	type	title	director	cast	country	date_added	release_
0	81145628	Movie	Norm of the North: King Sized Adventure	Richard Finn, Tim Maltby	Alan Marriott, Andrew Toth, Brian Dobson, Cole...	United States, India, South Korea, China	September 9, 2019	
1	80117401	Movie	Jandino: Whatever it Takes	NaN	Jandino Asporaat	United Kingdom	September 9, 2016	
					Peter			

Dado que el proyecto está enfocado únicamente a la recomendación de películas y el dataset contiene tanto películas como series de televisión, se debe realizar un filtrado sobre éste. Para ello, nos quedamos con aquellas filas que tengan como valor 'Movie' en la columna 'type'.

```
filteredMovies = (originalData["type"] == "Movie")
dataMovies = originalData[filteredMovies]
dataMovies.reset_index().head(10)
```

	index	show_id	type	title	director	cast	country	date_added
0	0	81145628	Movie	Norm of the North: King Sized Adventure	Richard Finn, Tim Maltby	Alan Marriott, Andrew Toth, Brian Dobson, Cole...	United States, India, South Korea, China	September 9, 2019
1	1	80117401	Movie	Jandino: Whatever it Takes	NaN	Jandino Asporaat	United Kingdom	September 9, 2016
2	4	80125979	Movie	#realityhigh	Fernando Lebrija	Nesta Cooper, Kate Walsh, John Michael Higgins...	United States	September 8, 2017
3	6	70304989	Movie	Automata	Gabe Ibáñez	Antonio Banderas, Dylan McDermott, Melanie Gri...	Bulgaria, United States, Spain, Canada	September 8, 2017
4	7	80164077	Movie	Fabrizio Copano: Solo pienso en mi	Rodrigo Toro, Francisco Schultz	Fabrizio Copano	Chile	September 8, 2017
						James Franco,	United States.	

Por otra parte, se optó por trabajar únicamente con las columnas show_id(id de la película), title (título de la película), listed_in (géneros de la película) y description (descripción de la película). Para ello, se eliminaron todas las columnas que no fueran a ser utilizadas. Finalmente, se renombraron los índices de las filas para que se adaptasen al nuevo dataset *dataMovies*

6	10	80169755	Movie	Reyes:	Miguel	Joaquin	NaN	September

```
dataMovies = dataMovies.drop(columns=['type','director','cast','country','date_added','rel
dataMovies = dataMovies.reset_index()
dataMovies = dataMovies.drop(columns=['index'])
dataMovies.head(10)
```

	show_id	title	listed_in	description
0	81145628	Norm of the North: King Sized Adventure	Children & Family Movies, Comedies	Before planning an awesome wedding for his gra...
1	80117401	Jandino: Whatever it Takes	Stand-Up Comedy	Jandino Asporaat riffs on the challenges of ra...
2	80125979	#realityhigh	Comedies	When nerdy high schooler Dani finally attracts...
3	70304989	Automata	International Movies, Sci-Fi & Fantasy, Thrillers	In a dystopian future, an insurance adjuster f...
4	80164077	Fabrizio Copano: Solo pienso en mi	Stand-Up Comedy	Fabrizio Copano takes audience participation t...
5	70304990	Good People	Action & Adventure, Thrillers	A struggling couple can't believe their luck w...
6	80169755	Joaquín Reyes: Una y no más	Stand-Up Comedy	Comedian and celebrity impersonator Joaquín Re...

El dataset ya filtrado se guarda dentro de la carpeta content como dataMovies.csv

Heineken Movies "Freddy" Heineken is ...

```
dataMovies.to_csv('dataMovies.csv')
print(dataMovies.head(10))
```

```
    show_id  ...                                        description
0  81145628  ...  Before planning an awesome wedding for his gra...
1  80117401  ...  Jandino Asporaat riffs on the challenges of ra...
2  80125979  ...  When nerdy high schooler Dani finally attracts...
3  70304989  ...  In a dystopian future, an insurance adjuster f...
4  80164077  ...  Fabrizio Copano takes audience participation t...
5  70304990  ...  A struggling couple can't believe their luck w...
6  80169755  ...  Comedian and celebrity impersonator Joaquín Re...
7  70299204  ...  When beer magnate Alfred "Freddy" Heineken is ...
8  80182480  ...  A team of minstrels, including a monkey, cat a...
9  80182483  ...  An artisan is cheated of his payment, a lion o...

[10 rows x 4 columns]
```

Procesado mediante la descripción y género de las películas

Para la recomendación de películas en base a las descripciones textuales, se ha optado por tener en cuenta tanto la descripción de las películas como el género de éstas. Para ello, se calculan dos matrices de distancias distintas. La matriz de distancias de la descripción textual guarda valores entre 0 y 1. La matriz de género, guarda distancias entre 0 y 0.125. Para el cálculo de distancias final se suman ambas matrices obteniendo así valores entre 0 y 1.125 para cada par posible de películas. A mayor es el valor, menor es la similitud entre las películas del par.

Inicialmente, cargamos el archivo .csv que habíamos guardado previamente y lo guardamos dentro de la variable originalData. Todo este proceso se hace a través de la función LoadData().

```
import pandas as pd

def LoadData():
  originalData = pd.read_csv('/content/dataMovies.csv')
  originalData = originalData.drop(columns=['Unnamed: 0'])
  return originalData

originalData = LoadData()
originalData.head(10)
```

	show_id	title	listed_in	description
0	81145628	Norm of the North: King Sized Adventure	Children & Family Movies, Comedies	Before planning an awesome wedding for his gra...
1	80117401	Jandino: Whatever it Takes	Stand-Up Comedy	Jandino Asporaat riffs on the challenges of ra...
2	80125979	#realityhigh	Comedies	When nerdy high schooler Dani finally attracts...
3	70304989	Automata	International Movies, Sci-Fi & Fantasy, Thrillers	In a dystopian future, an insurance adjuster f...
4	80164077	Fabrizio Copano: Solo pienso en mi	Stand-Up Comedy	Fabrizio Copano takes audience participation t...
5	70304990	Good People	Action & Adventure, Thrillers	A struggling couple can't believe their luck w...
6	80169755	Joaquín Reyes: Una y no más	Stand-Up Comedy	Comedian and celebrity impersonator Joaquín Re...
7	70299204	Kidnapping Mr. Heineken	Action & Adventure, Dramas, International Movies	When beer magnate Alfred "Freddy" Heineken is ...
8	80182480	Krish Trish and Baltiboy	Children & Family Movies	A team of minstrels, including a monkey, cat a...
9	80182483	Krish Trish and Baltiboy: Battle of Wits	Children & Family Movies	An artisan is cheated of his payment, a lion o...

Se obtiene el dataset preprocessedData, que contiene una nueva columna llamada processed_text en la que se guardan las palabras más relevantes de la descripción para la creación de una bolsa de palabras. Todo este proceso está contenido dentro de la función ProcessedText().

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

import nltk
nltk.download('punkt')
nltk.download('stopwords')

def ProcessedText():
```

```
def ProcessedText():
  ps = PorterStemmer()

  preprocessedText = []

  for row in originalData.itertuples():
    text = word_tokenize(row[4]) ## indice de la columna que contiene el texto
    ## Remove stop words
    stops = set(stopwords.words("english"))
    text = [ps.stem(w) for w in text if not w in stops and w.isalnum()]
    text = " ".join(text)
    preprocessedText.append(text)

  preprocessedData = originalData
  preprocessedData['processed_text'] = preprocessedText
  return preprocessedData

preprocessedData = ProcessedText()
preprocessedData.head(10)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

	show_id	title	listed_in	description	processed_text
0	81145628	Norm of the North: King Sized Adventure	Children & Family Movies, Comedies	Before planning an awesome wedding for his gra...	befor plan awesom wed grandfath polar bear kin...
1	80117401	Jandino: Whatever it Takes	Stand-Up Comedy	Jandino Asporaat riffs on the challenges of ra...	jandino asporaat riff challeng rais kid serena...
2	80125979	#realityhigh	Comedies	When nerdy high schooler Dani finally attracts...	when nerdi high schooler dani final attract in...
3	70304989	Automata	International Movies, Sci-Fi & Fantasy, Thrillers	In a dystopian future, an insurance adjuster f...	In dystopian futur insur adjust tech compani i...
4	80164077	Fabrizio Copano: Solo pienso en mi	Stand-Up Comedy	Fabrizio Copano takes audience participation t...	fabrizio copano take audienc particip next lev...
5	70304990	Good People	Action & Adventure, Thrillers	A struggling couple can't believe their luck w...	A struggl coupl ca believ luck find stash mone...
6	80169755	Joaquín Reyes: Una y no más	Stand-Up Comedy	Comedian and celebrity impersonator Joaquín Re...	comedian celebr imperson joaquín rey decid zes...

A partir de preprocessedData y su campo processed_text se crea una 'prematriz'. Esta prematriz estará compuesta por textsBows que se crean haciendo uso de la librería. Estos

textsBows almacenan la información de las frecuencias relativas de cada palabra. Todo este proceso se guarda dentro de la función TextsBow().

```
from sklearn.feature_extraction.text import TfidfVectorizer

def TextsBoW():
  bagOfWordsModel = TfidfVectorizer()
  bagOfWordsModel.fit(preprocessedData['processed_text'])
  textsBoW= bagOfWordsModel.transform(preprocessedData['processed_text'])
  return textsBoW

textsBoW = TextsBoW()
print("Finished")
```

```
Finished
```

```
print(textsBoW[10])
```

```
  (0, 8420)     0.1835787545856911
  (0, 6973)     0.28129166024763186
  (0, 6827)     0.3178209314072413
  (0, 5890)     0.3292897982928149
  (0, 5734)     0.26787770205547956
  (0, 5589)     0.28549201141576924
  (0, 4274)     0.22132941337485013
  (0, 4273)     0.20096460221097676
  (0, 3393)     0.23369419355147228
  (0, 3283)     0.3292897982928149
  (0, 2516)     0.29551102893678927
  (0, 1468)     0.2557032156227651
  (0, 987)      0.34545425828855103
```

Se calculan las distancias a través de un algoritmo. Se optó por hacer uso de la *Similitud Coseno* para obtener las distancias entre **vectores**. Como resultado del proceso se obtienen valores del 0 al 1, mayor es el valor, menor es la similitud.

Todo este proceso se hace dentro de la función Distance().

```
from sklearn.metrics import pairwise_distances

def Distance():
  distance_matrix= pairwise_distances(textsBoW,textsBoW ,metric='cosine')
  return distance_matrix

distance_matrix = Distance()
```

Con la prematriz con las frecuencias relativas crearemos una matriz NxN, donde N es el número de películas. Cada celda devuelve la distancia entre las películas del par, indicadas por sus índices en la matriz, representada con un valor del 0 al 1.

```
print(distance_matrix.shape)
print(type(distance_matrix))
```

```
(4265, 4265)
<class 'numpy.ndarray'>
```

Una vez calculadas las distancias en base a las descripciones de las películas, se deben calcular las distancias teniendo en cuenta el género.

Se obtiene el dataset preprocessedData, que contiene una nueva columna llamada processed_genre en la que se guardan las palabras más relevantes del género para la creación de una bolsa de palabras. Todo este proceso está contenido dentro de la función ProcessedGenre().

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

import nltk
nltk.download('punkt')
nltk.download('stopwords')

def ProcessedGenre():
  ps = PorterStemmer()

  preprocessedGenre = []

  for row in originalData.itertuples():
    text = word_tokenize(row[3]) ## indice de la columna que contiene el género
    ## Remove stop words
    stops = set(stopwords.words("english"))
    text = [ps.stem(w) for w in text if not w in stops and w.isalnum()]
    text = " ".join(text)
    preprocessedGenre.append(text)

  preprocessedData = originalData
  preprocessedData['processed_genre'] = preprocessedGenre
  return preprocessedData

preprocessedData = ProcessedGenre()
preprocessedData.head(10)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

	show_id	title	listed_in	description	processed_text	processed_genre
0	81145628	Norm of the North: King Sized Adventure	Children & Family Movies, Comedies	Before planning an awesome wedding for his gra...	befor plan awesom wed grandfath polar bear kin...	children famili movi comedi
1	80117401	Jandino: Whatever it Takes	Stand-Up Comedy	Jandino Asporaat riffs on the challenges of ra...	jandino asporaat riff challeng rais kid serena...	comedi
2	80125979	#realityhigh	Comedies	When nerdy high schooler Dani finally attracts...	when nerdi high schooler dani final attract in...	comedi
3	70304989	Automata	International Movies, Sci-Fi & Fantasy, Thrillers	In a dystopian future, an insurance adjuster f...	In dystopian futur insur adjust tech compani i...	intern movi fantasi thriller
4	80164077	Fabrizio Copano: Solo pienso en mi	Stand-Up Comedy	Fabrizio Copano takes audience participation t...	fabrizio copano take audienc particip next lev...	comedi

A partir de preprocessedData y su campo processed_genre se crea una 'prematriz'. Esta prematriz estará compuesta por textsBows que se crean haciendo uso de la librería. Estos textsBows almacenan la información de las frecuencias relativas de cada palabra. Todo este proceso se guarda dentro de la función TextsBowG().

6	80169755	Reyes: Una	Stand-Up	celebrity	imperson joaquín	comedi

```
from sklearn.feature_extraction.text import TfidfVectorizer

def TextsBoWG():
  bagOfWordsModel = TfidfVectorizer()
  bagOfWordsModel.fit(preprocessedData['processed_genre'])
  TextsBoWG= bagOfWordsModel.transform(preprocessedData['processed_genre'])
  return TextsBoWG

textsBoWG = TextsBoWG()
print("Finished")
```

```
Finished
```

```
print(textsBoWG[10])
```

```
  (0, 17)       0.2644716934023223
  (0, 10)       0.6819291471219383
```

```
  (0, 3)        0.6819291471219383
```

Se calculan las distancias a través de un algoritmo. Se optó por hacer uso de la *Similitud Coseno* para obtener las distancias entre **vectores**. Cabe destacar que las distancias se dividen entre 8 para que el género no tenga tanto peso como las descripciones de las películas. Todo este proceso se hace dentro de la función DistanceG().

```
from sklearn.metrics import pairwise_distances

def DistanceG():
  distance_matrixG= pairwise_distances(textsBoWG,textsBoWG,metric='cosine')/8
  return distance_matrixG

distance_matrixG = DistanceG()
```

Con la prematriz con las frecuencias relativas crearemos una matriz NxN, donde N es el número de películas. Cada celda devuelve la distancia entre los géneros de las películas del par representada con un valor del 0 al 0,125.

```
print(distance_matrixG.shape)
print(type(distance_matrixG))
```

```
 (4265, 4265)
 <class 'numpy.ndarray'>
```

Una vez calculadas las dos matrices de distancias, solo hace falta sumarlas entre sí para obtener una matriz de distancias final que tenga en cuenta tanto la descripción textual de las películas como el género de estas.

```
def add(distance_matrix, distance_matrixG):
    for i in range(distance_matrix.shape[0]):
        for j in range(distance_matrix.shape[1]):
            distance_matrix[i][j] = distance_matrix[i][j] + distance_matrixG[i][j]
    return distance_matrix

distance_matrixFinal = add(distance_matrix,distance_matrixG)
print(distance_matrixFinal[0:11])
```

```
 [[0.         1.07638423 1.07638423 ... 1.11741137 1.125      1.09454381]
  [1.07638423 0.         1.         ... 1.125      1.125      1.125     ]
  [1.07638423 1.         0.         ... 1.125      1.125      1.125     ]
  ...
  [1.00984138 1.125      1.125      ... 1.11676285 1.125      1.09194104]
  [1.00984138 1.125      1.125      ... 1.11676285 1.125      1.09194104]
  [1.00984138 1.125      1.125      ... 1.11676285 1.125      1.09194104]]
```

Añadir película

El añadido de una película se divide en tres partes. En la primera, se inserta la nueva película en el dataset de películas después de calcular su id con el hash. En la segunda, se vuelve a realizar el cálculo de distancias para que esta nueva película sea tenida en cuenta a la hora de recomendar. En la última, se almacena la nueva tabla generada con la película insertada.

Ha de tenerse en cuenta que para que todo el código que viene a continuación funcione, deben haberse ejecutado todos los bloques de código del apartado "**Procesado mediante la descripción y género de las películas**" de la sección CRUDs asociados a datos. Esto es debido a que en dicha sección se definen varias funciones que serán usadas en ésta.

1.Agregación de la Película al dataset

Primero de todo, debe cargarse el dataset que contiene todas las películas.

```
import pandas as pd

originalData = LoadData()
originalData.tail(10)
```

	show_id	title	listed_in	description
4255	80100054	Skins	Dramas, International Movies	Deformed, disfigured characters must find a wa...
4256	80097468	The Bad Kids	Documentaries	In this documentary, teachers at a Mojave Dese...
4257	80171439	The Human Factor: The Untold Story of the Bomb...	Documentaries, International Movies, Music & M...	A family of Parsi musicians collectively works...
4258	80104237	The Tenth Man	Dramas, International Movies	After spending much of his adult life in New Y...
4259	80093107	Toro	Dramas, International Movies, Thrillers	Ex-con Toro's brother and former partner in cr...
4260	80085438	Frank and Cindy	Documentaries	Frank was a rising pop star when he married Ci...
4261	80085439	Frank and Cindy	Comedies, Dramas, Independent Movies	A student filmmaker vengefully turns his camer...
4262	80011846	Iverson	Documentaries, Sports Movies	This unfiltered documentary follows the rocky ...

Se calcula a través del título el hash de la película que servirá como identificador, en caso de colisiones se opta por sumar uno al id. Una vez calculado el hash y cargado el dataset, se

añade la nueva fila y se comprueba que se haya agregado correctamente.

```
codigo = hash('Alberto') % 100000000
existe = False

while existe == False:
  if (originalData['show_id'] == codigo).any():
    codigo = (codigo + 1) % 100000000
  else:
    existe = True

originalData = originalData.append({'show_id' : codigo , 'title' : 'Alberto', 'listed_in'

filtered = (originalData["show_id"] == codigo)
comprobacion = originalData[filtered]
comprobacion
```

	show_id	title	listed_in	description
4265	17103665	Alberto	Comedy	This film is for testing

2.Cálculo de distancias

Ahora se debe actualizar la bolsa de palabras y realizar nuevamente el cálculo de distancias para que se tenga en cuenta la nueva película insertada.

```
preprocessedData = ProcessedText()
textsBoW = TextsBoW()
distance_matrix = Distance()

preprocessedData = ProcessedGenre()
textsBoWG = TextsBoWG()
distance_matrixG = DistanceG()

distance_matrixFinal = add(distance_matrix,distance_matrixG)

print(distance_matrixFinal.shape)
print(type(distance_matrixFinal))
```

```
(4266, 4266)
<class 'numpy.ndarray'>
```

A continuación, se incluye un ejemplo de cómo el sistema es capaz de recomendar películas semejantes a la nueva película agregada.

```
searchTitle = "Alberto" #Película base para las recomendaciones
indexOfTitle = preprocessedData[preprocessedData['title']==searchTitle].index.values[0]
distance_scores = list(enumerate(distance_matrixFinal[indexOfTitle]))
ordered_scores = sorted(distance_scores, key=lambda x: x[1])
top_scores = ordered_scores[0:11]
print(top_scores)
```

```
top_indexes = [i[0] for i in top_scores]
print(preprocessedData['title'].iloc[top_indexes])
```

```
[(4265, 0.0), (3554, 0.8007530757456257), (1800, 0.8512691067126563), (3719, 0.86626
4265                                        Alberto
3554                  Martin Lawrence Live: Runteldat
1800                                       No Entry
3719                                   One Last Shot
3510                                A Trip to Jamaica
2520                                  Mi Obra Maestra
2308                                Chris Tucker Live
1533    Undercover: How to Operate Behind Enemy Lines
45                                Marc Maron: Too Real
2528                                   The Good Catholic
444          Monty Python: Live at The Hollywood Bowl
Name: title, dtype: object
```

3.Almacenamiento del Dataset

Finalmente, se guarda el dataset con la nueva película agregada.

```
originalData.to_csv('originalData.csv')
print(originalData.tail(10))
```

```
       show_id  ...                      processed_genre
4256  80097468  ...                           documentari
4257  80171439  ...  documentari intern movi music music
4258  80104237  ...                    drama intern movi
4259  80093107  ...           drama intern movi thriller
4260  80085438  ...                           documentari
4261  80085439  ...           comedi drama independ movi
4262  80011846  ...               documentari sport movi
4263  80064521  ...                           documentari
4264  80116008  ...                                  movi
4265  17103665  ...                               comedi

[10 rows x 6 columns]
```

Modificar película

Al igual que pasaba en *Añadir Película*, la modificación de una película también se divide en tres partes: modificación de la película; actualizaciones necesarios (cálculo de distancias o nuevo cálculo de identificador); y guardado del dataset modificado.

Ha de tenerse en cuenta que para que todo el código que viene a continuación funcione, deben haberse ejecutado todos los bloques de código del apartado "**Procesado mediante la descripción y género de las películas**" de la sección de CRUDs asociados a datos. Esto es debido a que en dicha sección se definen varias funciones que serán usadas en ésta.

1.Modificación de la Película en el dataset

Primero de todo, debe cargarse el dataset que contiene todas las películas.

```
import pandas as pd

originalData = LoadData()
originalData.tail(10)
```

	show_id	title	listed_in	description
4255	80100054	Skins	Dramas, International Movies	Deformed, disfigured characters must find a wa...
4256	80097468	The Bad Kids	Documentaries	In this documentary, teachers at a Mojave Dese...
4257	80171439	The Human Factor: The Untold Story of the Bomb...	Documentaries, International Movies, Music & M...	A family of Parsi musicians collectively works...
4258	80104237	The Tenth Man	Dramas, International Movies	After spending much of his adult life in New Y...
4259	80093107	Toro	Dramas, International Movies, Thrillers	Ex-con Toro's brother and former partner in cr...
4260	80085438	Frank and Cindy	Documentaries	Frank was a rising pop star when he married Ci...
4261	80085439	Frank and Cindy	Comedies, Dramas, Independent Movies	A student filmmaker vengefully turns his camer...
4262	80011846	Iverson	Documentaries, Sports Movies	This unfiltered documentary follows the rocky ...

Se busca la película a editar y se modifican sus datos.

```
codigo = 80117401
existe = False

originalData.loc[originalData['show_id'] == codigo, ['description']] = 'The description\'s
originalData.head(10)
```

	show_id	title	listed_in	description
0	81145628	Norm of the North: King Sized Adventure	Children & Family Movies, Comedies	Before planning an awesome wedding for his gra...
1	80117401	Jandino: Whatever it Takes	Stand-Up Comedy	The description's value changed
2	80125979	#realityhigh	Comedies	When nerdy high schooler Dani finally attracts...
3	70304989	Automata	International Movies, Sci-Fi & Fantasy, Thrillers	In a dystopian future, an insurance adjuster f...
4	80164077	Fabrizio Copano: Solo pienso en mi	Stand-Up Comedy	Fabrizio Copano takes audience participation t...
5	70304990	Good People	Action & Adventure, Thrillers	A struggling couple can't believe their luck w

2.Cálculo de distancias

Ahora se debe actualizar la bolsa de palabras para que se tenga en cuenta la modificación realizada.

Heineken Movies Freddy "Heineken is ...

```
preprocessedData = ProcessedText()
textsBoW = TextsBoW()
distance_matrix = Distance()

preprocessedData = ProcessedGenre()
textsBoWG = TextsBoWG()
distance_matrixG = DistanceG()

distance_matrixFinal = add(distance_matrix,distance_matrixG)

print(distance_matrixFinal.shape)
print(type(distance_matrixFinal))
```

```
(4265, 4265)
<class 'numpy.ndarray'>
```

3.Almacenamiento del Dataset

Finalmente, se guarda en el dataset las modificaciones agregadas a la película.

```
originalData.to_csv('originalData.csv')
print(originalData.head(10))
```

```
    show_id  ...                    processed_genre
0  81145628  ...        children famili movi comedi
1  80117401  ...                             comedi
2  80125979  ...                             comedi
3  70304989  ...       intern movi fantasi thriller
4  80164077  ...                             comedi
5  70304990  ...           action adventur thriller
6  80169755  ...                             comedi
```

```
7  70299204  ...  action adventur drama intern movi
8  80182480  ...              children famili movi
9  80182483  ...              children famili movi

[10 rows x 6 columns]
```

Sistema de recomendación

Funcionamiento de sistema de recomendación en base a descripciones textuales

A continuación, se muestra un ejemplo de uso del sistema de Recomendación. Se seleccinó la película *Automata* para obtener una lista de recomendaciones de películas que guardan cierta similitud con ésta.

Ha de tenerse en cuenta que para que todo el código que viene a continuación funcione, deben haberse ejecutado todos los bloques de código del apartado "**Procesado mediante la descripción y género de las películas**" de la sección CRUDs asociados a datos. Aunque no aparezca en esta sección el Procesado mediante la descripción y género de las películas pertenece al Sistema de recomendación. La obligatoriedad de la ejecución de dicho apartado es debido a que es en este apartado donde se genera la matriz de distancias entre películas necesaria para realizar el proceso de recomendación.

```
searchTitle = "Automata" #Película base para las recomendaciones
indexOfTitle = preprocessedData[preprocessedData['title']==searchTitle].index.values[0]
indexOfTitle
```

```
3
```

Distancia entre la película Automata con el resto de la matriz. 1.5 -> Ningún parecido. 0.0 -> Completamente igual.

```
distance_scores = list(enumerate(distance_matrixFinal[indexOfTitle]))
distance_scores[0:11]
```

```
[(0, 1.1179838769274695),
 (1, 1.125),
 (2, 1.125),
 (3, 0.0),
 (4, 1.125),
 (5, 1.078857398522215),
 (6, 1.125),
 (7, 1.10397572808732),
 (8, 1.1173842834543346),
 (9, 1.1173842834543346),
 (10, 1.1173842834543346)]
```

Ordenadas las distancias. La primera es la propia película Automata, por eso su diferencia es 0.0.

```
ordered_scores = sorted(distance_scores, key=lambda x: x[1])
ordered_scores[0:11]
```

```
[(3, 0.0),
 (3128, 0.8803039037690801),
 (2176, 0.8986352017001517),
 (1200, 0.9042604560361982),
 (886, 0.9063044750703562),
 (2566, 0.9112390175427516),
 (1858, 0.9152425359825711),
 (3606, 0.9217303806387209),
 (1923, 0.9267826461474766),
 (3471, 0.9438910722036926),
 (2425, 0.9439620705159506)]
```

A continuación, se muestran las 10 películas más parecidas a Automata y se recogen sus índices para poder consultar su información..

```
top_scores = ordered_scores[0:11]
top_scores
```

```
[(3, 0.0),
 (3128, 0.8803039037690801),
 (2176, 0.8986352017001517),
 (1200, 0.9042604560361982),
 (886, 0.9063044750703562),
 (2566, 0.9112390175427516),
 (1858, 0.9152425359825711),
 (3606, 0.9217303806387209),
 (1923, 0.9267826461474766),
 (3471, 0.9438910722036926),
 (2425, 0.9439620705159506)]
```

```
top_indexes = [i[0] for i in top_scores]
top_indexes
```

```
[3, 3128, 2176, 1200, 886, 2566, 1858, 3606, 1923, 3471, 2425]
```

Títulos de las 10 películas más parecidas.

```
preprocessedData['title'].iloc[top_indexes]
```

```
3                              Automata
3128                             Master
2176                         Ex Machina
1200                               Anon
886                       The Rainmaker
2566    Roger Corman's Death Race 2050
```

```
1858                      Equilibrium
3606                       The Worthy
1923                              TAU
3471                         F.R.E.D.I.
2425                 Suicide (Hitabdut)
Name: title, dtype: object
```

```
originalData.iloc[top_indexes]
```

	show_id	title	listed_in	description	processed_text	processed_genre
3	70304989	Automata	International Movies, Sci-Fi & Fantasy, Thrillers	In a dystopian future, an insurance adjuster f...	In dystopian futur insur adjust tech compani i...	intern movi fantasi thriller
3128	80163352	Master	Action & Adventure, International Movies	Needing hard evidence to convict a company cha...	need hard evid convict compani chairman fraud ...	action adventur intern movi
2176	80023689	Ex Machina	Dramas, Independent Movies, International Movies	A coder at a tech company wins a week-long ret...	A coder tech compani win retreat compound comp...	drama independ movi intern movi
1200	80195964	Anon	Dramas, Sci-Fi & Fantasy, Thrillers	In a future where technology has rendered priv...	In futur technolog render privaci obsolet dete...	drama fantasi thriller
886	1181661	The Rainmaker	Dramas, Thrillers	A young attorney and a scrappy paralegal work ...	A young attorney scrappi paraleg work help par...	drama thriller
2566	80152003	Roger Corman's Death Race 2050	Action & Adventure, Sci-Fi & Fantasy	In this dystopian sequel, kills equal points i...	In dystopian sequel kill equal point violent g...	action adventur fantasi
1858	60024935	Equilibrium	Action & Adventure, Dramas, Sci-Fi & Fantasy	In a dystopian future, a totalitarian regime m...	In dystopian futur totalitarian regim maintain...	action adventur drama fantasi
3606	80191513	The	International Movies, Sci-Fi &	In the near future, with	In near futur civil thing past man	intern movi fantasi

Sistema de valoración manual y automática

Carga y procesado de los dataset

Para esta sección se usarán los datasets "ratings_small.csv", "Test.csv", "Train.csv", "ratings.csv" y "dataMovies.csv". Ambos datasets deben incluirse dentro de la carpeta content. El dataset "dataMovies.csv" se optiene ejecutando el apartado **"Carga y filtrado de datos"** de la sección CRUDs asociados a datos.

El dataset ratings_small.csv contiene valoraciones, del 1 al 5, de películas hechas por distintos usuarios. El dataset dataMovies contiene películas de netflix. El objetivo del procesado de los datos será sustituir los IDs de las películas de ratings_small por los IDs de las películas de dataMovies.

Además, también serán cambiadas las valoraciones a -1 y 1.

El dataset resultante se guarda en ratings.csv que será usado, posteriormente, para las valoraciones de películas y el filtrado colaborativo.

```
import pandas as pd
import random
import numpy as np

def LoadRatings():
  ratings = pd.read_csv('/content/ratings_small.csv')
  return ratings

def LoadDataMovies():
  dataMovies = pd.read_csv('/content/dataMovies.csv')
  return dataMovies

ratings = LoadRatings()
dataMovies = LoadDataMovies()

userIds = ratings["userId"].unique()
moviesIds = dataMovies["show_id"].unique()
userId = 1;
copiaIds = moviesIds
for i in ratings.index:
  if ratings.iloc[i]['userId'] != userId:
    userId = userId + 1
    copiaIds = moviesIds
  randomId = random.choice(copiaIds)
  ratings.at[i,'movieId'] = randomId
  # ratings.iloc[i]['movieId'] = randomId
  np.delete(copiaIds, np.where(copiaIds == randomId))

  if ratings.iloc[i]['rating'] < 2.5:
    ratings.at[i,'rating'] = -1
  else :
    ratings.at[i,'rating'] = 1

ratings = ratings.drop(columns=['timestamp'])
```

```
ratings = ratings.reset_index()
ratings = ratings.drop(columns=['index'])
ratings.to_csv('ratings.csv')
print(ratings.head(10))
print(" ")
print(ratings['rating'].value_counts())
```

```
   userId   movieId  rating
0       1  20764666     1.0
1       1  81037984     1.0
2       1  80994423     1.0
3       1  80158875    -1.0
4       1  70077556     1.0
5       1  80128277    -1.0
6       1  80197303    -1.0
7       1  80988048    -1.0
8       1  70284281     1.0
9       1  80135586    -1.0

 1.0    86619
-1.0    13385
Name: rating, dtype: int64
```

Para el aprendizaje automático, usaremos el dataset "Train.csv" que contiene comentarios de usurios y una clasificación de si ese comentario es positivo o negativo. Además, se cambiarán los valores 0 de la columna label por -1.

```
import pandas as pd

def LoadReviews():
  reviews = pd.read_csv('/content/Train.csv')
  return reviews

reviews = LoadReviews()

for i in reviews.index:
  if reviews.iloc[i]['label'] == 0:
    reviews.at[i,'label'] = -1

reviews = reviews.head(20000)
reviews.to_csv('reviews.csv')
print(reviews.head(10))
print(" ")
print(reviews['label'].value_counts())
```

```
                                                text  label
0  I grew up (b. 1965) watching and loving the Th...     -1
1  When I put this movie in my DVD player, and sa...     -1
2  Why do people who do not know what a particula...     -1
3  Even though I have great interest in Biblical ...     -1
4  Im a die hard Dads Army fan and nothing will e...      1
5  A terrible movie as everyone has said. What ma...     -1
6  Finally watched this shocking movie last night...      1
7  I caught this film on AZN on cable. It sounded...     -1
8  It may be the remake of 1987 Autumn's Tale aft...      1
```

```
9  My Super Ex Girlfriend turned out to be a plea...      1

-1    10024
 1     9976
Name: label, dtype: int64
```

Para comprobar que la preción del aprendizaje automático realizado ha sido correcta, usaremos el dataset de prueba Test. Este dataset ya contiene una clasificación de los comentarios, las cuales serán comparadas con los resultados obtenidos tras haber hecho la valoración automática de los comentarios. Al igual que en el dataset anterior, los valores 0 de la columna 'label' serán sustituídos por -1.

```
import pandas as pd

def LoadTest():
  test = pd.read_csv('/content/Test.csv')
  return test

test = LoadTest()

for i in test.index:
  if test.iloc[i]['label'] == 0:
    test.at[i,'label'] = -1

print(test.head(10))
print(" ")
print(test['label'].value_counts())
```

```
                                                text  label
0  I always wrote this series off as being a comp...     -1
1  1st watched 12/7/2002 - 3 out of 10(Dir-Steve ...     -1
2  This movie was so poorly written and directed ...     -1
3  The most interesting thing about Miryang (Secr...      1
4  when i first read about "berlin am meer" i did...     -1
5  I saw this film on September 1st, 2005 in Indi...      1
6  I saw a screening of this movie last night. I ...     -1
7  William Hurt may not be an American matinee id...      1
8  IT IS A PIECE OF CRAP! not funny at all. durin...     -1
9  I'M BOUT IT(1997)<br /><br />Developed & publi...     -1

 1    2505
-1    2495
Name: label, dtype: int64
```

Proceso de aprendizaje supervisado

A continuación se realizan los pasos para entrenar al algoritmo teniendo como dataset de entrada "reviews". Como resultado se obtiene un sistema entrenado que es capaz de clasificar comentarios en positivos (valor 1) o negativos (valor -1).

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
```

```
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import svm

import nltk
nltk.download('punkt')
nltk.download('stopwords')

ps = PorterStemmer()

preprocessedText = []

for row in reviews.itertuples():


    text = word_tokenize(row[1]) ## indice de la columna que contiene el texto
    ## Remove stop words
    stops = set(stopwords.words("english"))
    text = [ps.stem(w) for w in text if not w in stops and w.isalnum()]
    text = " ".join(text)

    preprocessedText.append(text)

preprocessedData = reviews
preprocessedData['processed_text'] = preprocessedText

bagOfWordsModel = TfidfVectorizer()
bagOfWordsModel.fit(preprocessedData['processed_text'])
textsBoW= bagOfWordsModel.transform(preprocessedData['processed_text'])

svc = svm.SVC(kernel='linear') #Modelo de clasificación
X_train = textsBoW #Documentos
Y_train = reviews['label'] #Etiquetas de los documentos
svc.fit(X_train, Y_train) #Entrenamiento
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Para comprobar que el sistema de clasificación de comentarios funciona correctamente, pasamos un test que contiene comentarios para que el sistema los clasifique. Este dataset también contiene una clasificación ya correcta de los comentarios que servirán para comparar con la clasificación hecha por el sistema.

```
from sklearn.metrics import classification_report

ps = PorterStemmer()
```

```
preprocessedText = []
for row in test.itertuples():
    text = word_tokenize(row[1]) ## indice de la columna que contiene el texto
    ## Remove stop words
    stops = set(stopwords.words("english"))
    text = [ps.stem(w) for w in text if not w in stops and w.isalnum()]
    text = " ".join(text)
    preprocessedText.append(text)
preprocessedDataTest = test
preprocessedDataTest['processed_text'] = preprocessedText

textsBoWTest= bagOfWordsModel.transform(preprocessedDataTest['processed_text'])

X_test = textsBoWTest #Documentos
predictions = svc.predict(X_test) #Se almacena en el array predictions las predicciones de

Y_test = test['label'] #Etiquetas reales de los documentos
print (classification_report(Y_test, predictions))
```

```
                  precision    recall  f1-score   support

              -1       0.89      0.87      0.88      2495
               1       0.87      0.90      0.89      2505

        accuracy                           0.88      5000
       macro avg       0.88      0.88      0.88      5000
    weighted avg       0.88      0.88      0.88      5000
```

Valorar de forma automática

Para realizar la valoración automática, creamos un dataset que incluye el usuario que realiza el comentario, la película que está valorando y la crítica que hace el usuario sobre la película. Además, se incluye un campo clasificación en el que se añadirán las predicciones realizadas por el sistema. Por defecto, el valor de clasificación se pone a 0.

Cabe destacar que tenemos dos dataset relacionados con las valoraciones. El primer dataset, **ratings**, continene información del usuario, la película que valoró y una valoración numérica (-1 ó 1). El segundo dataset, **comentaries**, contiene información del usuario que valora, la película que está valorando y el comentario con el cual valoró la película. El sistema se encargará de clasificar dicho comentario en positivo o negativo y escribir el valor en el dataset ratings.

```
import pandas as pd
comentaries = pd.DataFrame(columns=('userId', 'movieId', 'text', 'rating'))
comentaries.loc[len(comentaries)]=[1,81157840,'The writers got carried away, the directors
comentaries.loc[len(comentaries)]=[2,81157840,'Having read the first few Harry Potter book
comentaries.loc[len(comentaries)]=[3,81157840,'Some movies are so good that they are talke
comentaries.loc[len(comentaries)]=[4,81157840,'I am an avid reader of Twilight. I recommer
comentaries.loc[len(comentaries)]=[5,81157840,'Dont spend your time comparing it to the bo
comentaries.loc[len(comentaries)]=[6,81157840,'I dont think it was intended to be so absur
print(comentaries)
```

```
   userId   movieId                                               text rating
0       1  81157840  The writers got carried away, the directors ov...      0
1       2  81157840  Having read the first few Harry Potter books b...      0
2       3  81157840  Some movies are so good that they are talked a...      0
3       4  81157840  I am an avid reader of Twilight. I recommend i...      0
4       5  81157840  Dont spend your time comparing it to the book....      0
5       6  81157840  I dont think it was intended to be so absurdly...      0
```

```
from sklearn.metrics import classification_report

def calcularValoraciones(comentaries):
  ps = PorterStemmer()
  preprocessedText = []
  for row in comentaries.itertuples():
      text = word_tokenize(row[3]) ## indice de la columna que contiene el texto
      ## Remove stop words
      stops = set(stopwords.words("english"))
      text = [ps.stem(w) for w in text if not w in stops and w.isalnum()]
      text = " ".join(text)
      preprocessedText.append(text)
  preprocessedDataTest = comentaries
  preprocessedDataTest['processed_text'] = preprocessedText

  textsBoWTest= bagOfWordsModel.transform(preprocessedDataTest['processed_text'])

  X_test = textsBoWTest #Documentos
  predictions = svc.predict(X_test) #Se almacena en el array predictions las predicciones

  for i in comentaries.index:
      comentaries.at[i,'rating'] = predictions[i]
  comentaries = comentaries.drop(columns=['processed_text'])
  return comentaries

comentaries = calcularValoraciones(comentaries)
comentaries.to_csv('comentaries.csv')
print(comentaries)
```

```
   userId   movieId                                               text rating
0       1  81157840  The writers got carried away, the directors ov...     -1
1       2  81157840  Having read the first few Harry Potter books b...      1
2       3  81157840  Some movies are so good that they are talked a...     -1
3       4  81157840  I am an avid reader of Twilight. I recommend i...      1
4       5  81157840  Dont spend your time comparing it to the book....      1
5       6  81157840  I dont think it was intended to be so absurdly...     -1
```

Añadir Valoraciones

El añadido de una valoración se podrá realizar de dos formas: tanto escribiendo un comentario como clasificando la película de forma manual entre positiva (1) o negativa (-1).

Para insertar una valoración de forma manual a través de una clasificación positiva o negativa, se pedirá el id del usuario, el id de la película que desea valorar y la valoración que éste le desea

dar. Una vez se tienen esos tres datos, se inserta al final del dataset **ratings** la valoración.

Por otro lado, para valorar una película a través de un comentario, se pedirán el id del usuario, el id de la película que desea comentar y el comentario que desea escribir. Estos datos serán añadidos al dataset **comentaries**. Una vez añadido el comentario, el sistema estima si el comentario representa una valoración negativa o positiva y la añade al dataset **ratings**, junto con el id del usuario y el id de la película valorada.

```
#Valoracion a través de clasificación de forma manual entre positiva o negativa
import pandas as pd

ratings = pd.read_csv('/content/ratings.csv')
ratings = ratings.drop(columns=['Unnamed: 0'])

usuario = 15
película = 80220311
valoracion = -1

ratings = ratings.append({'userId' : usuario , 'movieId' : película, 'rating' : valoracior
ratings.tail(10)
```

	userId	movieId	rating
99995	671	81045065	1.0
99996	671	80103818	1.0
99997	671	80156939	1.0
99998	671	70123160	1.0
99999	671	81043345	1.0
100000	671	80242378	1.0
100001	671	81172902	1.0
100002	671	80004286	1.0
100003	671	70308135	1.0
100004	15	80220311	-1.0

```
#Valoracion a través de comentario
import pandas as pd

ratings = pd.read_csv('/content/ratings.csv')
ratings = ratings.drop(columns=['Unnamed: 0'])

comentaries = pd.read_csv('/content/comentaries.csv')
comentaries = comentaries.drop(columns=['Unnamed: 0'])

usuario = 16
película = 80220311
comentarie = 'I absolutely love this movie, partly because the acting is really good, but

comentaries = comentaries.append({'userId' : usuario , 'movieId' : película, 'text' : com
```

```
comentaries = comentaries.append({'userId' : usuario , 'movieId' : película, 'text' : come
comentaries = calcularValoraciones(comentaries)

ratings = ratings.append({'userId' : usuario , 'movieId' : película, 'rating' : comentarie

print(comentaries)
print(" ")
print(ratings)
```

```
   userId   movieId                                               text  rating
0       1  81157840  The writers got carried away, the directors ov...      -1
1       2  81157840  Having read the first few Harry Potter books b...       1
2       3  81157840  Some movies are so good that they are talked a...      -1
3       4  81157840  I am an avid reader of Twilight. I recommend i...       1
4       5  81157840  Dont spend your time comparing it to the book....       1
5       6  81157840  I dont think it was intended to be so absurdly...      -1
6      16  80220311  I absolutely love this movie, partly because t...       1

        userId   movieId  rating
0            1  20764666     1.0
1            1  81037984     1.0
2            1  80994423     1.0
3            1  80158875    -1.0
4            1  70077556     1.0
...        ...       ...     ...
100000     671  80242378     1.0
100001     671  81172902     1.0
100002     671  80004286     1.0
100003     671  70308135     1.0
100004      16  80220311     1.0

[100005 rows x 3 columns]
```

Mejoras y nuevas funcionalidades

Filtrado Colaborativo

Decidimos realizar un filtrado colaborativo para mejorar las recomendaciones de usuarios que ya hayan usado la aplicación. Para el cálculo de las recomendaciones, usamos el dataset **ratings.csv**, el cual contiene valoraciones de películas que han hecho los usuarios. Además, en este apartado también se hace uso del dataset "dataMovies.csv", el cual se obtiene ejecutando el apartado **"Carga y filtrado de datos"** de la sección CRUDs asociados a datos.

Se utiliza la librería surprise para realizar el filtrado colaborativo.

El filtrado colaborativo lo que nos permite es recomendar películas que les hayan gustado a otros usuarios con gustos similares al nuestro. Para ejemplificar, si un usuario ha visto tres películas y las ha valorado positivamente y otro usuario distinto ha visto cuatro películas que también valoró positivamente y de las cuales tres eran las que el primer usuario ya había visto, entonces es probable que la cuarta película que el primer usuario aún no vió le vaya a gustar.

Esta forma de recomendar tiene un problema para usuarios de los cuales tenemos pocas valoraciones, en estos casos se utiliza el Sistema de recomendación textual.

Ha de tenerse en cuenta que para que todo el código que viene a continuación funcione, deben haberse ejecutado los bloques de carga de los datasets del apartado "**Carga y procesado de los dataset**" en la sección Sistema de valoración manual y automática. Se necesita la ejecución de

```
!pip install surprise
```

```
Collecting surprise
  Downloading https://files.pythonhosted.org/packages/61/de/e5cba8682201fcf9c3719a6f
Collecting scikit-surprise
  Downloading https://files.pythonhosted.org/packages/97/37/5d334adaf5ddd65da99fc65f
     |████████████████████████████████| 11.8MB 5.7MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp36-cp36m-linux
  Stored in directory: /root/.cache/pip/wheels/78/9c/3d/41b419c9d2aff5b6e2b4c0fc8d25
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.1 surprise-0.1
```

Se calcula la matriz de estimaciones para el filtrado colaborativo haciendo uso del algoritmo SVD. Posteriormente, se calculan las 10 películas que probablemente le gusten más a cada uno de los usuarios.

```
from collections import defaultdict

from surprise import SVD, Reader
from surprise import Dataset
import pandas as pd


originalData = pd.read_csv('/content/dataMovies.csv')
originalData = originalData.drop(columns=['Unnamed: 0'])


def get_top_n(predictions, n=10):
    """Devuelve las 10 primeras recomendaciones para cada uno de los usuarios.
    """

    # Primer map las predicciones para cada usuario.
    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))

    # Ordenar las predicciones y escoger únicamente las k-mejores.
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
```

```
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]

    return top_n


# Entrenamiento del algorimos SVD.
reader = Reader(rating_scale=(-1, 1))
data = Dataset.load_from_df(ratings[['userId','movieId','rating']], reader)
trainset = data.build_full_trainset()
algo = SVD()
algo.fit(trainset)

# Predicciones para todos los pares (usuario, película) que no pertenecen al dataset de er
testset = trainset.build_anti_testset()
predictions = algo.test(testset)

top_n = get_top_n(predictions, n=10)
```

Una vez finalizado el proceso mostramos el que sería un posible resultado para los usuarios, en nuestro caso usamos las recomendaciones para el usuario con id = 1.

```
# Imprimir por pantalla las 10 mejores estimaciones para el usuario con id 1.
for uid, user_ratings in top_n.items():
  if uid <=1:
    print("Recomendaciones para el usuario ", uid, ":")
    print(" ")
    top_movies = ([iid for (iid, _) in user_ratings])
    for x in top_movies:
      indexOfMovie = originalData[originalData['show_id']==x].index.values[0]
      print(x, "  ", originalData['title'].iloc[indexOfMovie])
```

```
 Recomendaciones para el usuario  1 :

 80170875    God's Own Country
 70019506    Hitch
 81018120    Who Would You Take to a Deserted Island?
 70082268    Cloverfield
 80115844    John & Jane
 80109089    Unchained: The Untold Story of Freestyle Motocross
 80239503    Mo Amer: The Vagabond
 70108386    Naruto the Movie 3: Guardians of the Crescent Moon Kingdom
 80239629    Animas
 80135164    1922
```

Recursos y bibliografía

1.- Surprise; Documentos, FAQ (Frecuently Asked Questions); Fecha de recuperación: 28 de enero de 2021; Disponible en: https://surprise.readthedocs.io/en/stable/FAQ.html

2.- Usuario Rounak Banik en Kaggle; Movie Recommender Systems; Fecha de publicación: 2018; Fecha de recuperación: 28 de enero de 2021; Disponible en: https://www.kaggle.com/rounakbanik/movie-recommender-systems

3.- Foro público Stack Overflow; Python Pandas: Get index of rows which column matches certain value; Fecha de publicación: febrero de 2014; Fecha de recuperación: 28 de enero de 2021; Disponible en: https://stackoverflow.com/questions/21800169/python-pandas-get-index-of-rows-which-column-matches-certain-value

4.- Usuario adrpseara en GitHub; Tutorial Recomendador basado en contenido; Fecha de recuperación: 28 de enero de 2021; Disponible en: https://github.com/adrseara/abp_notebooks/blob/master/Tutorial_Recomendador_basado_en_contenido.ipynb

5.- Usuario adrpseara en GitHub; Tutorial Análisis de sentimientos; Fecha de recuperación: 28 de enero de 2021; Disponible en: https://github.com/adrseara/abp_notebooks/blob/master/Tutorial_an%C3%A1lisis_de_sentimientos.ipynb

6.- Velez-Langs, O., & Santos, C. (2006). Sistemas recomendadores: Un enfoque desde los algoritmos genéticos. Industrial data, 9(1), 23-31.

7.- García, R. (2013). SVD Aplicado a sistemas de recomendación basados en filtrado colaborativo. Trabajo Fin de Máster: Universidad Politécnica de Madrid.