Sistemas recomendadores

Técnicas Avanzadas de Análisis de Datos

Sonia Díaz Santos

Índice

Introducción

A continuación se va a describir el trabajo realizado, comenzando con el objetivo general del proyecto, la metodología que se ha seguido y los resultados esperados.

Objetivo

El objetivo de este proyecto es implementar un sistema de recomendación basado en contenido, que nos permita recomendar los mejores documentos para un cliente, mediante el algoritmo de clasificación KNN.

Metodología

Crear un software que reciba un archivo de texto plano con extensión .txt, que contenga el conjunto de posibles documentos a recomendar al usuario final. Cada documento viene representado en una línea del archivo.

En este caso, el fichero de texto plano con los posibles documentos a recomendar al usuario final es el siguiente:

```
7. Here's a bright, informal red that opens with aromas of
candied berry, white pepper and savory herb that carry over to
the the palate. It's balanced with fresh acidity and soft
tannins.
1. Aromas include tropical fruit, broom, brimstone and dried
herb. The palate isn't overly expressive, offering unripened
apple, citrus and dried sage alongside brisk acidity.
3. Tart and snappy, the flavors of lime flesh and rind
dominate. Some green pineapple pokes through, with crisp
acidityunderscoring the flavors. The wine was all stainless-
steel fermented.
4. Pineapple rind, lemon pith and orange blossom start off the
aromas. The palate is a bit more opulent, with notes of honey-
drizzled guava and mango giving way to a slightly astringent,
semidry finish.
2. This is ripe and fruity, a wine that is smooth while still
structured. Firm tannins are filled out with juicy red berry
fruits and freshened with acidity. It's already drinkable,
although it will certainly be better from 2016.
6. Blackberry and raspberry aromas show a typical Navarran
whiff of green herbs and, in this case, horseradish. In the
mouth, this is fairly full bodied, with tomatoes acidity.
Spicy, herbal flavors complement dark plum fruit, while the
finish is fresh but grabby.
5. Much like the regular bottling from 2012, this comes across
as rather rough and tannic, with rustic, earthy, herbal
characteristics. Nonetheless, if you think of it as a
pleasantly unfussy country wine, it's a good companion to a
hearty winter stew.
```

Se ha generado un fichero de texto con algunos documentos que le han gustado a dicho usuario como se puede observar en la siguiente figura:

```
Aromas have tropical fruit, broom, brimstone and dried herb.
Offer unripened apple, citrus and dried sage alongside brisk
acidity
The wine was all stainless-steel fermented. The flavors of lime
flesh and rind dominate. Some green pineapple pokes through
6I like blackberry and raspberry aromas, green herbs and
horseradish. In the mouth with tomatoes acidity. Spicy, herbal
flavors complement dark plum fruit
```

Resultados esperados

El software debe proporcionar como salida lo siguiente:

- Para cada documento, tabla con las siguientes columnas:
  - Índice del término.
  - Término.
  - TF.
  - IDF.
  - TF-IDF.
- Similaridad coseno entre cada par de documentos.

Análisis Exploratorio de Datos

En esta sección se va a llevar a cabo el análisis exploratorio de datos, incluyendo la instalación de las librerías necesarias, el preprocesado de los datos para leer los ficheros, la limpieza de dichos datos y la unión de los posibles documentos a recomendar con los que le han gustado al usuario.

Instalación de librerías

Se procede a instalar las librerías necesarias para llevar a cabo este proyecto:

```
pip install lenskit
```

```
import pandas as pd
import numpy as np
import re

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

%matplotlib inline
```

Preprocesado de datos

A continuación, se van a leer los ficheros de los posibles documentos a recomendar y los documentos que le han gustado al usuario. Posteriormente, se van a limpiar los datos y a unir ambos documentos.

Lectura de ficheros

En este apartado se van a leer los documentos necesarios para la realización de este proyecto. El fichero de los documentos a recomendar al usuario y el fichero de los documentos que le han gustado al usuario.

```
# Se vincula la cuenta de drive
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive;
```

```
# Se leen los datos
posibles_documentos_a_recomendar = pd.read_csv("/content/gdrive/My Drive/Master/TAAD 2022 (2)/documents.txt", sep='\n', header=None)
documentos_favoritos = pd.read_csv("/content/gdrive/My Drive/Master/TAAD 2022 (2)/documents_liked.txt", sep='\n', header=None)
```

Limpieza de datos

Ahora se van a limpiar los datos leídos anteriormente:

```
# Se asigna el nombre de Documentos a la columna
posibles_documentos_a_recomendar.columns = ["Documentos"]
documentos_favoritos.columns = ["Documentos"]
```

Como resultado se obtiene el siguiente resultado:

	Documentos
0	7. Here's a bright, informal red that opens wi...
1	1. Aromas include tropical fruit, broom, brims...
2	3. Tart and snappy, the flavors of lime flesh ...
3	4. Pineapple rind, lemon pith and orange bloss...
4	2. This is ripe and fruity, a wine that is smo...
5	6. Blackberry and raspberry aromas show a typi...
6	5. Much like the regular bottling from 2012, t...

El próximo paso es crear un bucle para recorrer los documentos y realizar dos funciones. La primera es crear un array de índices idx en el que se almacenan los números de los documentos, para posteriormente generar una columna con los números de los documentos. La segunda es eliminar de la columna de documentos el número y el punto al inicio de cada uno. También se quita el punto final de cada documento:

```
idx = []

for i in range(posibles_documentos_a_recomendar.shape[0]):
  idx.append(int(posibles_documentos_a_recomendar["Documentos"][i][0]))
  posibles_documentos_a_recomendar["Documentos"][i] = re.sub(r'^\d\.', '', posibles_documentos_a_recomendar["Documentos"][i])
  posibles_documentos_a_recomendar["Documentos"][i] = posibles_documentos_a_recomendar["Documentos"][i].replace(posibles_documentos_a_recomendar["Documentos"][i][-1],'')

idx
```

Obteniendo el array de índices idx:

```
[7, 1, 3, 4, 2, 6, 5]
```

```
# Se crea la columna que indica el número de cada documento
posibles_documentos_a_recomendar.insert(0,'Número de documento', idx)
```

	Número de documento	Documentos
0	7	Here's a bright, informal red that opens with...
1	1	Aromas include tropical fruit, broom, brimsto...
2	3	Tart and snappy, the flavors of lime flesh an...
3	4	Pineapple rind, lemon pith and orange blossom...
4	2	This is ripe and fruity, a wine that is smoot...
5	6	Blackberry and raspberry aromas show a typica...
6	5	Much like the regular bottling from 2012, thi...

Se ordena el *dataframe* por los valores de los números de documentos de valor más bajo al más alto:

```
posibles_documentos_a_recomendar_ordered = posibles_documentos_a_recomendar.sort_values(by='Número de documento', ascending=True)
```

	Número de documento	Documentos
1	1	Aromas include tropical fruit, broom, brimsto...
4	2	This is ripe and fruity, a wine that is smoot...
2	3	Tart and snappy, the flavors of lime flesh an...
3	4	Pineapple rind, lemon pith and orange blossom...
6	5	Much like the regular bottling from 2012, thi...
5	6	Blackberry and raspberry aromas show a typica...
0	7	Here's a bright, informal red that opens with...

Se coloca como índice el número del documento:

```
posibles_documentos_a_recomendar_ordered.set_index('Número de documento',inplace=True)
```

	Documentos
Número de documento	
1	Aromas include tropical fruit, broom, brimsto...
2	This is ripe and fruity, a wine that is smoot...
3	Tart and snappy, the flavors of lime flesh an...
4	Pineapple rind, lemon pith and orange blossom...
5	Much like the regular bottling from 2012, thi...
6	Blackberry and raspberry aromas show a typica...
7	Here's a bright, informal red that opens with...

Los documentos que le han gustado al usuario son los siguientes:

	Documentos
0	Aromas have tropical fruit, broom, brimstone a...
1	The wine was all stainless-steel fermented. Th...
2	6I like blackberry and raspberry aromas, green...

Unión de ambos documentos

El siguiente paso es unir los posibles documentos a recomendar y los documentos que le han gustado al usuario.

```
# Se unen los dataframes de todos los documentos y de los documentos que le han gustado al usuario, estando estos en las 3 últimas filas
documento_completo = pd.concat([posibles_documentos_a_recomendar_ordered, documentos_favoritos], ignore_index=True)
```

	Documentos
0	Aromas include tropical fruit, broom, brimsto...
1	This is ripe and fruity, a wine that is smoot...
2	Tart and snappy, the flavors of lime flesh an...
3	Pineapple rind, lemon pith and orange blossom...
4	Much like the regular bottling from 2012, thi...
5	Blackberry and raspberry aromas show a typica...
6	Here's a bright, informal red that opens with...
7	Aromas have tropical fruit, broom, brimstone a...
8	The wine was all stainless-steel fermented. Th...
9	6I like blackberry and raspberry aromas, green...

```
# Se modifican los índices para que comiencen en 1 y no en 0
documento_completo['Número de documento'] = ['1','2','3','4','5','6','7','8','9','10']
```

```
# Se le asigna como índice comenzando desde 1
documento_completo.set_index('Número de documento', inplace=True)
```

Número de documento	Documentos
1	Aromas include tropical fruit, broom, brimsto...
2	This is ripe and fruity, a wine that is smoot...
3	Tart and snappy, the flavors of lime flesh an...
4	Pineapple rind, lemon pith and orange blossom...
5	Much like the regular bottling from 2012, thi...
6	Blackberry and raspberry aromas show a typica...
7	Here's a bright, informal red that opens with...
8	Aromas have tropical fruit, broom, brimstone a...
9	The wine was all stainless-steel fermented. Th...
10	6I like blackberry and raspberry aromas, green...

Creación de la tabla

Ahora se va a crear la tabla de documentos que cuenta con el índice del término, el término y el TF-IDF de cada uno de los documentos, tanto de los posibles a recomendar como los que le han gustado al usuario.

Para ello, se crea un nuevo *dataframe* que cuenta con las mismas filas que números de documentos y con las columnas correspondientes: Índice del término, Término y TF-IDF:

```
tabla = pd.DataFrame(documento_completo, index = documento_completo.index, columns = ['Índice del término', 'Término', 'TF-IDF'], dtype = object)
```

Número de documento	Índice del término	Término	TF-IDF
1	NaN	NaN	NaN
2	NaN	NaN	NaN
3	NaN	NaN	NaN
4	NaN	NaN	NaN
5	NaN	NaN	NaN
6	NaN	NaN	NaN
7	NaN	NaN	NaN
8	NaN	NaN	NaN
9	NaN	NaN	NaN
10	NaN	NaN	NaN

Índice del término y término

En esta sección se van a calcular los datos de las columnas índice del término y término.

```
# Se rellena la columna de índice de término y término
for i in range(documento_completo["Documentos"].shape[0]):
  # Se cuenta el número de términos para cada documento
  num_of_terms = len(documento_completo["Documentos"][i].split())
  tabla["Término"][i] = documento_completo["Documentos"][i].split()
  tabla["Índice del término"][i] = np.array(range(num_of_terms))
```

El resultado obtenido es el que se muestra en la siguiente imagen:

Número de documento	Índice del término	Término	TF-IDF
1	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Aromas, include, tropical, fruit,, broom,, br...	NaN
2	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[This, is, ripe, and, fruity,, a, wine, that, ...	NaN
3	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Tart, and, snappy,, the, flavors, of, lime, f...	NaN
4	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Pineapple, rind,, lemon, pith, and, orange, b...	NaN
5	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Much, like, the, regular, bottling, from, 201...	NaN
6	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Blackberry, and, raspberry, aromas, show, a, ...	NaN
7	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Here's, a, bright,, informal, red, that, open...	NaN
8	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Aromas, have, tropical, fruit,, broom,, brims...	NaN
9	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[The, wine, was, all, stainless-steel, ferment...	NaN
10	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[6l, like, blackberry, and, raspberry, aromas,...	NaN

Se muestra el primer documento en la columna término como un array de todos los términos del documento:

```
tabla["Término"][0]
```

```
['Aromas',
 'include',
 'tropical',
 'fruit,',
 'broom,',
 'brimstone',
 'and',
 'dried',
 'herb',
 'The',
 'palate',
 "isn't",
 'overly',
 'expressive,',
 'offering',
 'unripened',
 'apple,',
 'citrus',
 'and',
 'dried',
 'sage',
 'alongside',
 'brisk',
 'acidity']
```

Se verifica que se ha realizado correctamente:

```
[documento_completo["Documentos"][0]]
```

```
[" Aromas include tropical fruit, broom, brimstone and dried herb The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity"]
```

TF-IDF

En esta sección se van a calcular los datos de la columna TF-IDF. TF-IDF es una abreviatura de Term Frequency-Inverse Document Frequency. Este es un algoritmo muy común para transformar el texto en una representación significativa de números que se utiliza para ajustar el algoritmo de la máquina para la predicción. Se genera la matriz TF-IDF y se añaden los valores a la tabla como una matriz densa.

Para ello, se crea un array con todos los documentos, se crea el vector TF-IDF para cada uno y se asigna el resultado a la tabla creada anteriormente:

```
total_tf_idf = []
for i in range(documento_completo.shape[0]):
  tf_idf = TfidfVectorizer(stop_words='english')
  tf_idf_matrix = tf_idf.fit_transform([documento_completo["Documentos"][i]])
  tabla["TF-IDF"][i] = tf_idf_matrix.todense()
  total_tf_idf.append(tf_idf.vocabulary_)
```

En la siguiente imagen se puede observar todos los términos del documento y su ubicación en el documento:

```
total_tf_idf
```

```
[{'aromas': 3,
  'include': 12,
  'tropical': 18,
  'fruit': 10,
  'broom': 6,
  'brimstone': 4,
  'dried': 8,
  'herb': 11,
  'palate': 16,
  'isn': 13,
  'overly': 15,
  'expressive': 9,
  'offering': 14,
  'unripened': 19,
  'apple': 2,
  'citrus': 7,
  'sage': 17,
  'alongside': 1,
  'brisk': 5,
  'acidity': 0},
 {'ripe': 13,
  'fruity': 10,
```

Se genera con éxito la matriz TF-IDF en la tabla:

Número de documento	Índice del término	Término	TF-IDF
1	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Aromas, include, tropical, fruit,, broom,, br...	[[[[0.20851441 0.20851441 0.20851441 0.208514...
2	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[This, is, ripe, and, fruity,, a, wine, that, ...	[[[[0.23570226 0.23570226 0.23570226 0.235702...
3	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Tart, and, snappy,, the, flavors, of, lime, f...	[[[[0.22941573 0.22941573 0.22941573 0.229415...
4	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Pineapple, rind,, lemon, pith, and, orange, b...	[[[[0.21320072 0.21320072 0.21320072 0.213200...
5	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Much, like, the, regular, bottling, from, 201...	[[[[0.21320072 0.21320072 0.21320072 0.213200...
6	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Blackberry, and, raspberry, aromas, show, a, ...	[[[[0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0...
7	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Here's, a, bright,, informal, red, that, open...	[[[[0.23570226 0.23570226 0.23570226 0.235702...
8	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[Aromas, have, tropical, fruit,, broom,, brims...	[[[[0.23570226 0.23570226 0.23570226 0.235702...
9	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[The, wine, was, all, stainless-steel, ferment...	[[[[0.28867513 0.28867513 0.28867513 0.288675...
10	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[6I, like, blackberry, and, raspberry, aromas,...	[[[[0.23570226 0.23570226 0.23570226 0.235702...

Similaridad del coseno

Primero, se calcula la matriz TF-IDF de todos los documentos:

```
tf_idf_liked = TfidfVectorizer(stop_words='english')
tf_idf_matrix_liked = tf_idf_liked.fit_transform(documento_completo['Documentos']);
```

```
# Se ven las dimensiones de la matriz
tf_idf_matrix_liked.shape
```

```
(10, 121)
```

```
# Se calcula la matriz de similaridad del coseno de la matriz TF-IDF consigo misma
cosine_similarity_matrix = cosine_similarity(tf_idf_matrix_liked, tf_idf_matrix_liked)
```

```
# Se ven las dimensiones de la matriz de similaridad del coseno generado
cosine_similarity_matrix.shape
```

```
(10, 10)
```

Como resultado se genera la siguiente matriz de similaridad del coseno:

```
array([[1.        , 0.01778818, 0.        , 0.04667929, 0.        ,
        0.05860527, 0.10846279, 0.77990298, 0.        , 0.07390319],
       [0.01778818, 1.        , 0.03066838, 0.        , 0.0240976 ,
        0.01698506, 0.16138702, 0.02090135, 0.04010935, 0.02141872],
       [0.        , 0.03066838, 1.        , 0.06975101, 0.02686441,
        0.08618358, 0.        , 0.        , 0.80933387, 0.10868035],
       [0.04667929, 0.        , 0.06975101, 1.        , 0.        ,
        0.05355276, 0.04969092, 0.01878787, 0.09122319, 0.01925292],
       [0.        , 0.0240976 , 0.02686441, 0.        , 1.        ,
        0.02855701, 0.        , 0.        , 0.03513436, 0.08305926],
       [0.05860527, 0.01698506, 0.08618358, 0.05355276, 0.02855701,
        1.        , 0.07995458, 0.068862  , 0.07514291, 0.67306898],
       [0.10846279, 0.16138702, 0.        , 0.04969092, 0.        ,
        0.07995458, 1.        , 0.08555014, 0.        , 0.0447356 ],
       [0.77990298, 0.02090135, 0.        , 0.01878787, 0.        ,
        0.068862  , 0.08555014, 1.        , 0.        , 0.08683726],
       [0.        , 0.04010935, 0.80933387, 0.09122319, 0.03513436,
        0.07514291, 0.        , 0.        , 1.        , 0.0947577 ],
       [0.07390319, 0.02141872, 0.10868035, 0.01925292, 0.08305926,
        0.67306898, 0.0447356 , 0.08683726, 0.0947577 , 1.        ]])
```

Como para la siguiente función el índice tiene que empezar en 0, se genera una lista de los índices de la tabla restándole un valor. De esta manera, el documento 0 realmente es el documento 1 y el documento 9 es el documento 10:

```
indice_sub = [elemento - 1 for elemento in list(map(int, tabla.index))]
```

```
similarity_scores = pd.DataFrame(cosine_similarity_matrix[indice_sub], columns=["1","2","3", "4","5", "6","7", "8","9", "10"])
similarity_scores.sort_values(by=["1","2","3", "4","5", "6","7", "8","9", "10"],ascending=False)
```

Se obtiene como resultado las medidas de similaridad entre documentos en forma de matriz:

	1	2	3	4	5	6	7	8	9	10
0	1.000000	0.017788	0.000000	0.046679	0.000000	0.058605	0.108463	0.779903	0.000000	0.073903
7	0.779903	0.020901	0.000000	0.018788	0.000000	0.068862	0.085550	1.000000	0.000000	0.086837
6	0.108463	0.161387	0.000000	0.049691	0.000000	0.079955	1.000000	0.085550	0.000000	0.044736
9	0.073903	0.021419	0.108680	0.019253	0.083059	0.673069	0.044736	0.086837	0.094758	1.000000
5	0.058605	0.016985	0.086184	0.053553	0.028557	1.000000	0.079955	0.068862	0.075143	0.673069
3	0.046679	0.000000	0.069751	1.000000	0.000000	0.053553	0.049691	0.018788	0.091223	0.019253
1	0.017788	1.000000	0.030668	0.000000	0.024098	0.016985	0.161387	0.020901	0.040109	0.021419
8	0.000000	0.040109	0.809334	0.091223	0.035134	0.075143	0.000000	0.000000	1.000000	0.094758
2	0.000000	0.030668	1.000000	0.069751	0.026864	0.086184	0.000000	0.000000	0.809334	0.108680
4	0.000000	0.024098	0.026864	0.000000	1.000000	0.028557	0.000000	0.000000	0.035134	0.083059

Como en la matriz se tienen los mismos valores en el triángulo superior como inferior se va a obtener solamente el inferior:

```
triangle = np.tril(cosine_similarity_matrix)
```

```
array([[1.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.01778818, 1.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.03066838, 1.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.04667929, 0.        , 0.06975101, 1.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.0240976 , 0.02686441, 0.        , 1.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.05860527, 0.01698506, 0.08618358, 0.05355276, 0.02855701,
        1.        , 0.        , 0.        , 0.        , 0.        ],
       [0.10846279, 0.16138702, 0.        , 0.04969092, 0.        ,
        0.07995458, 1.        , 0.        , 0.        , 0.        ],
       [0.77990298, 0.02090135, 0.        , 0.01878787, 0.        ,
        0.068862  , 0.08555014, 1.        , 0.        , 0.        ],
       [0.        , 0.04010935, 0.80933387, 0.09122319, 0.03513436,
        0.07514291, 0.        , 0.        , 1.        , 0.        ],
       [0.07390319, 0.02141872, 0.10868035, 0.01925292, 0.08305926,
        0.67306898, 0.0447356 , 0.08683726, 0.0947577 , 1.        ]])
```

Resultados y conclusiones

Para obtener los resultados se realiza la siguiente función:

```
j = 1
x = 0
for i in triangle[indice_sub]:
  for z in i:
    if(z >= 0.6 and z<1.0):
      print("Para el documento", j , "se recomienda el documento", x+1 , "con un coseno de similitud de", round(z, 2))
    x = x+1
  x = 0
  j = j+1
```

En esta función recorre el triángulo inferior de las medidas de similaridad entre documentos con los índices desde el 0 hasta el 9, que corresponde a un documento respecto al resto de los documentos. Posteriormente se recorre internamente cada uno de los documentos respecto a ese documento. Se ha optado por recomendar aquellos documentos que tengan un valor superior al 0,6 y menor a 1 (el documento consigo mismo).

Se obtiene como resultado las siguientes recomendaciones:

```
Para el documento 8 se recomienda el documento 1 con un coseno de similitud de 0.78
Para el documento 9 se recomienda el documento 3 con un coseno de similitud de 0.81
Para el documento 10 se recomienda el documento 6 con un coseno de similitud de 0.67
```

Se pueden comprobar los resultados con la matriz anteriormente creada o con el triángulo inferior, en este caso se verá con la matriz. Hay que tener en cuenta que en la función se le sumó un valor al índice, así que los documentos que se recomiendan realmente hay que restar un valor al índice. Con estas consideraciones se pueden verificar los resultados:

	1	2	3	4	5	6	7	8	9	10
0	1.000000	0.017788	0.000000	0.046679	0.000000	0.058605	0.108463	0.779903	0.000000	0.073903
7	0.779903	0.020901	0.000000	0.018788	0.000000	0.068862	0.085550	1.000000	0.000000	0.086837
6	0.108463	0.161387	0.000000	0.049691	0.000000	0.079955	1.000000	0.085550	0.000000	0.044736
9	0.073903	0.021419	0.108680	0.019253	0.083059	0.673069	0.044736	0.086837	0.094758	1.000000
5	0.058605	0.016985	0.086184	0.053553	0.028557	1.000000	0.079955	0.068862	0.075143	0.673069
3	0.046679	0.000000	0.069751	1.000000	0.000000	0.053553	0.049691	0.018788	0.091223	0.019253
1	0.017788	1.000000	0.030668	0.000000	0.024098	0.016985	0.161387	0.020901	0.040109	0.021419
8	0.000000	0.040109	0.809334	0.091223	0.035134	0.075143	0.000000	0.000000	1.000000	0.094758
2	0.000000	0.030668	1.000000	0.069751	0.026864	0.086184	0.000000	0.000000	0.809334	0.108680
4	0.000000	0.024098	0.026864	0.000000	1.000000	0.028557	0.000000	0.000000	0.035134	0.083059

Se puede concluir que se ha construido un sistema recomendador que, con dos ficheros de entrada, uno con los posibles documentos a recomendar al usuario con un formato específico y otro con los documentos que le han gustado al usuario, puede recomendar aquellos que tengan una similitud de al menos 0,6. Con lo que se ha logrado el objetivo inicial de recomendar a un usuario nuevos documentos conociendo algunos que le hayan gustado previamente en base a la similitud entre ellos.

Referencias

[1] Fuente del conjunto de datos, posibles documentos a recomendar (documents.txt): https://drive.google.com/file/d/1kJ91jBvhxpny2m3UfvI3YhY6T0BsliV5/view?usp=sharing

[2] Fuente del conjunto de datos, posibles documentos a recomendar (documents_liked.txt): https://drive.google.com/file/d/13iD9qvDBMd_gPeMr7NRm_-2DQigWepTP/view?usp=sharing

[3] Repositorio GitHub: https://github.com/alu0101061672/TAAD.git