SISTEMAS RECOMENDADORES

Técnicas Avanzadas de Análisis de Datos

Bloque I - Práctica 2

Descripción:

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.

Acceso al repositorio de GitHub: https://github.com/alu0100596113/TAAD

Máster: Ciberseguridad e Inteligencia de Datos

Asignatura: Técnicas Avanzadas de Análisis de Datos

Alumno: Carlos Barreda Falciano

Correos: alu0100596113@ull.edu.es

Universidad de La Laguna

Índice

Descripción de la práctica

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: Estudiar los modelos basados en el contenido en el material de clase. 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 irá representado en una línea del archivo.

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.

Acceso al repositorio de GitHub

Todos los archivos, y el código implementado pueden obtenerse en el siguiente repositorio: https://github.com/alu0100596113/TAAD

Herramientas

Todo el proyecto ha sido implementado sobre plataforma Google Colaboratory, herramienta introducida y utilizada durante todo el desarrollo del máster. Para efectuar este proyecto, se ha empleado como lenguaje de programación Python, ya que dispone de librerías específicas que facilitan el trabajo.

Librerías utilizadas

Se utilizan algunas librerías que son comunes cuando se trabaja con Python, como pueden ser panda y numpy. Para el desarrollo de esta práctica, se ha importado también:

nltk: es una librería que permite realizar procesamiento del lenguaje natural simbólico y estadísticos para el lenguaje de programación Python. Ha sido utilizada para definir las stopwords.

Sklearn: que es una librería para utilizada en Python para el aprendizaje automático de software, incluye varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte, bosques aleatorios, Gradient boosting, K-means y DBSCAN.

```
# ========================================================================
# Librerías utilizadas
# ========================================================================
import pandas as pd
import numpy as np
import re
import nltk

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.corpus import stopwords
nltk.download('stopwords')
```

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

Lectura y carga de datos

El almacenamiento del fichero que contiene todo el cuerpo de documentos a recomendar está almacenado en Google Drive, y para tener acceso al archivo debe realizarse el montaje de la unidad.

```
# ========================================================================
# Montaje del directorio de Google Drive donde se encuentra almacenado el archivo txt
# ========================================================================
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

Una vez se accede a la unidad, se procede a realizar la carga del archivo en un dataframe. El fichero contiene el catálogo completo de documentos a recomendar, y tal y como se ha explicado en el enunciado de la práctica, dispone de múltiples líneas, donde cada una de ellas hace referencia a un documento a recomendar. El archivo representa un catálogo de vinos, y cada documento es una descripción concreta de cada uno de ellos.

```
# ========================================================================
# Lectura de datos y almacenamiento en Dataframe
# ========================================================================
data = pd.read_csv("/content/gdrive/MyDrive/TAAD/SR/data.txt", sep='\n', header=None)
data.head(10)
```

	0
0	1.Aromas include tropical fruit, broom, brimst...
1	2.This is ripe and fruity, a wine that is smoo...
2	3.Tart and snappy, the flavors of lime flesh a...
3	4.Pineapple rind, lemon pith and orange blosso...
4	5.Much like the regular bottling from 2012, th...
5	6.Blackberry and raspberry aromas show a typic...
6	7.Here's a bright, informal red that opens wit...

Preprocesado de los datos

Para poder trabajar con el documento, deben aplicarse técnicas de preprocesado de datos que permitan leer los datos de forma correcta y clara, suprima posibles errores y permita obtener los resultados deseados en el estudio de la búsqueda del sistema recomendación. Las acciones de preprocesado que se han llevado a cabo son:

Se identifican las columnas y se le añade un nombre acorde a la información que aportan. Además, se añade una columna que hará referencia al identificador del documento.

```
# ========================================================================
# Se añade un nombre a la columna y se agrega una columna con el Id del documento
# ========================================================================
data.columns = ['Documento']
data['id'] = np.arange(len(data)) # Añadimos la col 'id' para identificar a cada documento
data = data[['id', 'Documento']] # Cambia el orden para que ID sea primera col
data.columns
```

Se elimina la enumeración inicial de los documentos.

```
# ========================================================================
# Se elimina la enumeración inicial de los documentos
# ========================================================================
data['Documento'] = data['Documento'].str.strip('123456789.')
```

Se realiza la eliminación de todos los símbolos de puntuación, excepto el símbolo de apostrofe, ya que el documento está escrito en inglés, y es necesario su utilización.

```
# ========================================================================
# Eliminación de signos de puntuación, execpto apostrophe
# ========================================================================
import re
#data['Documento'] = re.sub('[^A-Za-z0-9]+', ' ', str(data['Documento']))
data['Documento'] = data['Documento'].str.replace(r"[^0-9a-zA-Z:']+", ' ')
```

Se elimina el uso de mayúsculas, y se transforman todas las palabras del documento escritas únicamente en minúsculas.

```
# ================================================================================
# Convertimos todo el texto en minúsculas.
# ================================================================================
import re
data['Documento'] = data['Documento'].str.lower()
```

Construcción de la tabla solicitada

Una vez se ha ejecutado una limpieza de los datos, se procede a realizar el primer apartado de esta práctica, cuya finalidad es la construcción de una tabla que recoja la siguiente información.

id_doc: contiene el identificador del documento.

palabras_docuemnto: almacena un array con todas y cada una de las palabras que contiene el documento (sin eliminar las stopwords).

términos: es un array con cada uno de los términos que forman un documento. En este caso, sólo se almacenan las palabras que tienen valor significativo, es decir, se eliminan preposiciones, determinantes, conectores, etc. (stopwords).

ind_termino: corresponde al índice que ocupa el termino en el vocabulario de cada documento.

TF: es la frecuencia del término. Cada celda del dataframe almacena un array con tantos valores como términos tiene un documento. Cada valor indica la frecuencia en la que el término aparece en el documento.

IDF: es la frecuencia inversa del documento y se trata de la segunda medida que se combina con la frecuencia de los términos.

TF-IDF: (Term Frequency - Inverse Document Frequency), es un dato estadístico que muestra el grado de frecuencia de una palabra en una colección de documentos.

Palabras del documento

Una vez hemos se ha creado un dataframe con las columnas indicadas anteriormente, se procede a separar las palabras de cada uno de los documentos.

```
# ================================================================================
# Creamos un nuevo dataframe tal y como indica el enunciado del proyecto
# Es decir, una tabla que contenga: id_doc, Indice para cada termino, lista de terminos TF, IDF y TF-IDF
# ================================================================================
datos = pd.DataFrame(data, index = data.index, columns = ['id_doc','palabras_documento', 'terminos', 'ind_termino', 'TF', 'IDF','TF-IDF'], dtype = object)
datos['id_doc'] = data['id']
```

```
# ================================================================================
#  Se separan las palabras (términos) de cada documento y se almacenan en la tabla: datos['palabras_documento']
# ================================================================================
for i in range(len(data)):
   #print(data.iloc[i]['Documento'])
   #print(data.iloc[i]['Documento'].split( ' '))
   datos['palabras_documento'][i] = data.loc[i]['Documento'].split( ' ')

#datos['terminos'][0] # Para mostrar los términos del primer documento
datos['palabras_documento'] # Para mostrar todos los términos
datos
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
import sys
```

	id_doc	palabras_documento	terminos	ind_termino	TF	IDF	TF-IDF
0	0	[aromas, include, tropical, fruit, broom, brim...	NaN	NaN	NaN	NaN	NaN
1	1	[this, is, ripe, and, fruity, a, wine, that, i...	NaN	NaN	NaN	NaN	NaN
2	2	[tart, and, snappy, the, flavors, of, lime, fl...	NaN	NaN	NaN	NaN	NaN
3	3	[pineapple, rind, lemon, pith, and, orange, bl...	NaN	NaN	NaN	NaN	NaN
4	4	[much, like, the, regular, bottling, from, 201...	NaN	NaN	NaN	NaN	NaN
5	5	[blackberry, and, raspberry, aromas, show, a, ...	NaN	NaN	NaN	NaN	NaN
6	6	[here's, a, bright, informal, red, that, opens...	NaN	NaN	NaN	NaN	NaN

Términos del documento

Se procede a crear el vocabulario. Utilizando nltk se pueden definir las stopwords, y descartar palabras comunes del lenguaje. Estas palabras suelen aparecer numerosas veces en los textos, pero que no aportan información relevante.

```
# ================================================================================
#  Definición de stopwords utilizando nltk en el idioma inglés
# ================================================================================
myStopwords = set(stopwords.words('english'))
myStopwords
```

```
{'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
```

Una vez se ha creado el vocabulario, se procede a agregar los términos a la columna términos. Esta columna estará formada por un array para cada documento, y dicho array contiene exclusivamente las palabras clave. En la siguiente imagen se puede apreciar como desaparecen las palabras consideradas stopwords para el idioma inglés.

```
# ================================================================================
# Vocabulario de cada documento (para obtener los terminos) habiendo eliminado las stopwords
# ================================================================================
for i in range(data.shape[0]):
  vectorizer = TfidfVectorizer(analyzer = 'word', stop_words = myStopwords, lowercase=True, use_idf=True)
  tfidf = vectorizer.fit_transform([data["Documento"][i]])
  datos['terminos'][i] = list(vectorizer.vocabulary_.keys()) #Se obtienen solo los valores claves (palabras) del vocabulario
datos
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
import sys
```

	id_doc	palabras_documento	terminos	ind_termino	TF	IDF	TF-IDF
0	0	[aromas, include, tropical, fruit, broom, brim...	[aromas, include, tropical, fruit, broom, brim...	NaN	NaN	NaN	NaN
1	1	[this, is, ripe, and, fruity, a, wine, that, ...	[ripe, fruity, wine, smooth, still, structured...	NaN	NaN	NaN	NaN
2	2	[tart, and, snappy, the, flavors, of, lime, fl...	[tart, snappy, flavors, lime, flesh, rind, dom...	NaN	NaN	NaN	NaN
3	3	[pineapple, rind, lemon, pith, and, orange, bl...	[pineapple, rind, lemon, pith, orange, blossom...	NaN	NaN	NaN	NaN
4	4	[much, like, the, regular, bottling, from, 201...	[much, like, regular, bottling, 2012, comes, a...	NaN	NaN	NaN	NaN
5	5	[blackberry, and, raspberry, aromas, show, a, ...	[blackberry, raspberry, aromas, show, typical,...	NaN	NaN	NaN	NaN
6	6	[here's, a, bright, informal, red, that, opens...	[bright, informal, red, opens, aromas, candied...	NaN	NaN	NaN	NaN

Índice del término

Para obtener el índice de cada término, se obtiene la “posición” por orden alfabético que ocupa el termino en el vocabulario del documento.

```
# ================================================================================
# Vocabulario de cada documento (para obtener los indices del termino) habiendo eliminado las stopwords
# ================================================================================
for i in range(data.shape[0]):
  vectorizer = TfidfVectorizer(analyzer = 'word', stop_words = myStopwords, use_idf=True)
  tfidf = vectorizer.fit_transform([data["Documento"][i]])
  #datos["TF-IDF"][i] = matrix.todense()
  datos['ind_termino'][i] = list(vectorizer.vocabulary_.values()) # Se obtienen solo los valores (indices) del vocabulario

#datos['ind_termino']
datos
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

	id_doc	palabras_documento	terminos	ind_termino	TF	IDF	TF-IDF
0	0	[aromas, include, tropical, fruit, broom, brim...	[aromas, include, tropical, fruit, broom, brim...	[3, 12, 17, 10, 6, 4, 8, 11, 15, 14, 9, 13, 18...	NaN	NaN	NaN
1	1	[this, is, ripe, and, fruity, a, wine, that, i...	[ripe, fruity, wine, smooth, still, structured...	[15, 12, 20, 16, 17, 18, 9, 19, 8, 13, 14, 4, ...	NaN	NaN	NaN
2	2	[tart, and, snappy, the, flavors, of, lime, fl...	[tart, snappy, flavors, lime, flesh, rind, dom...	[14, 11, 4, 7, 5, 10, 2, 6, 8, 9, 1, 0, 15, 16...	NaN	NaN	NaN
3	3	[pineapple, rind, lemon, pith, and, orange, bl...	[pineapple, rind, lemon, pith, orange, blossom...	[15, 17, 9, 16, 13, 3, 20, 0, 14, 2, 12, 11, 8...	NaN	NaN	NaN
4	4	[much, like, the, regular, bottling, from, 201...	[much, like, regular, bottling, 2012, comes, a...	[12, 11, 16, 2, 0, 4, 1, 15, 17, 20, 18, 7, 10...	NaN	NaN	NaN
5	5	[blackberry, and, raspberry, aromas, show, a, ...	[blackberry, raspberry, aromas, show, typical,...	[2, 21, 1, 22, 25, 19, 26, 14, 16, 4, 17, 18, ...	NaN	NaN	NaN
6	6	[here's, a, bright, informal, red, that, opens...	[bright, informal, red, opens, aromas, candied...	[4, 9, 13, 10, 1, 5, 3, 17, 12, 14, 8, 6, 11, ...	NaN	NaN	NaN

Cálculo de la frecuencia del término (TF)

Para calcular la frecuencia del término se procede a contabilizar las veces que un termino determinado aparece en un documento, y a dividir la cantidad de apariciones entre el número total de términos del documento. El resultado es un array con valores entre [0-1] que

representa las veces que aparece una palabra en un documento determinado, dividido entre el número total de términos que tiene el documento.

```
# ================================================================================
# Contamos las veces que aparece una palabra en un documento: Frecuencia del Termino
# Se utiliza round para truncar a 3 decimales la frecuencia de cada termino.
# ================================================================================
for i in range(len(datos)):
  datos['TF'][i] = [] # Cada celda de TF de la tabla es un array con la frecuencia de cada termino en el documento
  for w in datos['terminos'][i]:
    #Cuenta las veces que aparece el termino en cada documento y lo divide por la longitud de terminos del documento
    datos['TF'][i].append(round(datos['palabras_documento'][i].count(w)/len(datos['terminos'][i]),3))
datos
#print(datos['terminos'][0])  # muestra los terminos del primer documento
#print(datos['TF'][0]) # muestra la frecuencia de cada termino del primer documento
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

	id_doc	palabras_documento	terminos	ind_termino	TF	IDF	TF-IDF
0	0	[aromas, include, tropical, fruit, broom, brim...	[aromas, include, tropical, fruit, broom, brim...	[3, 12, 17, 10, 6, 4, 8, 11, 15, 14, 9, 13, 18...	[0.053, 0.053, 0.053, 0.053, 0.053, 0.053, 0.1...	NaN	NaN
1	1	[this, is, ripe, and, fruity, a, wine, that, i...	[ripe, fruity, wine, smooth, still, structured...	[15, 12, 20, 16, 17, 18, 9, 19, 8, 13, 14, 4, ...	[0.048, 0.048, 0.048, 0.048, 0.048, 0.048, 0.0...	NaN	NaN
2	2	[tart, and, snappy, the, flavors, of, lime, fl...	[tart, snappy, flavors, lime, flesh, rind, dom...	[14, 11, 4, 7, 5, 10, 2, 6, 8, 9, 1, 0, 15, 16...	[0.059, 0.059, 0.118, 0.059, 0.059, 0.059, 0.0...	NaN	NaN
3	3	[pineapple, rind, lemon, pith, and, orange, bl...	[pineapple, rind, lemon, pith, orange, blossom...	[15, 17, 9, 16, 13, 3, 20, 0, 14, 2, 12, 11, 8...	[0.045, 0.045, 0.045, 0.045, 0.045, 0.045, 0.0...	NaN	NaN
4	4	[much, like, the, regular, bottling, from, 201...	[much, like, regular, bottling, 2012, comes, a...	[12, 11, 16, 2, 0, 4, 1, 15, 17, 20, 18, 7, 10...	[0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.0...	NaN	NaN
5	5	[blackberry, and, raspberry, aromas, show, a, ...	[blackberry, raspberry, aromas, show, typical,...	[2, 21, 1, 22, 25, 19, 26, 14, 16, 4, 17, 18, ...	[0.037, 0.037, 0.037, 0.037, 0.037, 0.037, 0.0...	NaN	NaN
6	6	[here's, a, bright, informal, red, that, opens...	[bright, informal, red, opens, aromas, candied...	[4, 9, 13, 10, 1, 5, 3, 17, 12, 14, 8, 6, 11, ...	[0.056, 0.056, 0.056, 0.056, 0.056, 0.056, 0.0...	NaN	NaN

Cálculo de la frecuencia inversa del documento (IDF)

Se procede a completar la columna correspondiente a la frecuencia inversa del documento, conocida como *IDF.* Su objetivo es reducir el peso de las palabras clave que aparecen muy a menudo en todos los documentos. La idea es que esas palabras que no son generalmente frecuentes son muy útiles para discriminar entre los documentos, y por lo tanto se debe dar más peso a las palabras que aparecen sólo en unos pocos documentos. Para calcularlo se aplica la formula:

$$IDF(i) = log\frac{N}{n(i)}$$

Donde ***N*** el número de todos los documentos recomendables y ***n(i)*** el número de los documentos de N en los que aparece la palabra clave ***i***.

Para calcular N simplemente obtenemos la longitud del dataframe, ya que cada fila representa a un documento, y, por tanto, la longitud del dataframe se corresponde con la cantidad de documentos de la colección.

Para calcular n(i), se hace uso de Tfidfvectorizer y se genera un vocabulario global que contiene todos los términos de todos los documentos. Esta lista de términos se recorre posteriormente sobre la columna que contiene los términos de cada documento, y se contabiliza en cuantos documentos aparece cada termino, es decir, n(i). Para facilitar el

procedimiento de calcular n(i), y poder visualizarlo mejor, se ha almacenado en un dataframe nuevo, que contiene dos columnas, el termino y la cantidad de documentos en las que aparece.

```
# ================================================================================
# Contamos ...
# La formula es N/ni, donde:
#    N es la cantidad total de documentos a recomendar
#    ni es la cantidad de documentos donde aparece el termino.
# ================================================================================
# ================================================================================
#Calculamos N:
# ================================================================================
N = len(datos.index)

# ================================================================================
#  Vocabulario total de todos los documentos (para obtener una lista de todos los terminos)
# ================================================================================
vectorizer = TfidfVectorizer(analyzer = 'word', stop_words = myStopwords, use_idf=True)
vectorizer
vocabulario_global = []
for i in range(data.shape[0]):
  tfidf = vectorizer.fit_transform(data['Documento'])
  vocabulario_global = pd.DataFrame(list(vectorizer.vocabulary_.keys()))
vocabulario_global
# ================================================================================
#  Se cuenta la cantidad de documentos en los que aparece cada termino
# ================================================================================
vocabulario_global.columns = ['Termino'] # Se asigna nombre a la col
#Añadimos la columna que recogerá la cantidad de docuemntos donde aparece el termino
vocabulario_global ['cantidad_doc'] = pd.Series()
#Calculamos el numero de documentos donde aparece cada termino
vocabulario_global
datos['terminos'][1]
for j in range(vocabulario_global.shape[0]): # Recorre cada termino del vocabulario global
#  print(vocabulario_global['Termino'][j])
  for w in range(datos.shape[0]): # Recorre todos los documentos
    cont = 0
    for i in range(len(datos['terminos'])): # Recorre toda la lista de terminos de cada documento
      #print(datos['terminos'][i])
      if vocabulario_global['Termino'][j] in datos['terminos'][i]:
        cont = cont+1
        vocabulario_global['cantidad_doc'][j] = cont

vocabulario_global
```

	Termino	cantidad_doc
0	aromas	4.0
1	include	1.0
2	tropical	1.0
3	fruit	2.0
4	broom	1.0
...	...	...
121	pepper	1.0
122	savory	1.0
123	carry	1.0
124	balanced	1.0
125	soft	1.0

126 rows × 2 columns

Una vez se almacenado en el dataframe anterior la cantidad de documentos en las que aparece un término, se procede a calcular la ecuación: `IDF = log(N/n(i))`

```
# ================================================================================
# Se calcula IDF. Acude a la tabla que dispone de la cantidad de documentos donde aparece un termino en la colección
# y luego realiza la operación para el calculo de IDFlog(N/ni)
# ================================================================================
from math import log
for i in range(datos.shape[0]): # Cantidad de documentos
  datos['IDF'][i] = []
  for j in (datos['terminos'][i]): #Cantidas de terminos en el documento
    for w in range(vocabulario_global.shape[0]): # longitud del vocabulario
      if j == vocabulario_global['Termino'][w]:
        datos['IDF'][i].append(round(log((N/vocabulario_global['cantidad_doc'][w])),3))
datos
```

	id_doc	palabras_documento	terminos	ind_termino	TF	IDF	TF-IDF
0	0	[aromas, include, tropical, fruit, broom, brim...	[aromas, include, tropical, fruit, broom, brim...	[3, 12, 17, 10, 6, 4, 8, 11, 15, 14, 9, 13, 18...	[0.053, 0.053, 0.053, 0.053, 0.053, 0.053, 0.1...	[0.56, 1.946, 1.946, 1.253, 1.946, 1.946, 1.94...	NaN
1	1	[this, is, ripe, and, fruity, a, wine, that, i...	[ripe, fruity, wine, smooth, still, structured...	[15, 12, 20, 16, 17, 18, 9, 19, 8, 13, 14, 4, ...	[0.048, 0.048, 0.048, 0.048, 0.048, 0.048, 0.0...	[1.946, 1.946, 0.847, 1.946, 1.946, 1.946, 1.9...	NaN
2	2	[tart, and, snappy, the, flavors, of, lime, fl...	[tart, snappy, flavors, lime, flesh, rind, dom...	[14, 11, 4, 7, 5, 10, 2, 6, 8, 9, 1, 0, 15, 16...	[0.059, 0.059, 0.118, 0.059, 0.059, 0.059, 0.0...	[1.946, 1.946, 1.253, 1.946, 1.946, 1.253, 1.9...	NaN
3	3	[pineapple, rind, lemon, pith, and, orange, bl...	[pineapple, rind, lemon, pith, orange, blossom...	[15, 17, 9, 16, 13, 3, 20, 0, 14, 2, 12, 11, 8...	[0.045, 0.045, 0.045, 0.045, 0.045, 0.045, 0.0...	[1.253, 1.253, 1.946, 1.946, 1.946, 1.946, 1.9...	NaN
4	4	[much, like, the, regular, bottling, from, 201...	[much, like, regular, bottling, 2012, comes, a...	[12, 11, 16, 2, 0, 4, 1, 15, 17, 20, 18, 7, 10...	[0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.0...	[1.946, 1.946, 1.946, 1.946, 1.946, 1.946, 1.9...	NaN
5	5	[blackberry, and, raspberry, aromas, show, a, ...	[blackberry, raspberry, aromas, show, typical,...	[2, 21, 1, 22, 25, 19, 26, 14, 16, 4, 17, 18, ...	[0.037, 0.037, 0.037, 0.037, 0.037, 0.037, 0.0...	[1.946, 1.946, 0.56, 1.946, 1.946, 1.946, 1.94...	NaN
6	6	[here's, a, bright, informal, red, that, opens...	[bright, informal, red, opens, aromas, candied...	[4, 9, 13, 10, 1, 5, 3, 17, 12, 14, 8, 6, 11, ...	[0.056, 0.056, 0.056, 0.056, 0.056, 0.056, 0.0...	[1.946, 1.946, 1.253, 1.946, 0.56, 1.946, 1.25...	NaN

Cálculo de TF-IDF

Básicamente, ***TF-IDF*** (Term Frequency - Inverse Document Frequency) es un dato estadístico que muestra el grado de frecuencia de una palabra en una colección de documentos. Para calcular para una palabra clave i en el documento j se calcula como el producto de estas dos medidas:

$$TF\text{-}IDF(i, j) = TF(i, j) * IDF(i)$$

```
# ================================================================================
# Se calcula TF-IDF.
# ================================================================================
#Se multiplica TF*IDF
for i in range(datos.shape[0]):
  datos['TF-IDF'][i] = [x*y for x,y in zip(datos['TF'][i],datos['IDF'][i])]
#Se redondea el valor de TF-IDF
for i in range(datos.shape[0]):
  datos['TF-IDF'][i] = [round(num, 4) for num in datos['TF-IDF'][i]]
datos
```

	id_doc	palabras_documento	terminos	ind_termino	TF	IDF	TF-IDF
0	0	[aromas, include, tropical, fruit, broom, brim...	[aromas, include, tropical, fruit, broom, brim...	[3, 12, 17, 10, 6, 4, 8, 11, 15, 14, 9, 13, 18...	[0.053, 0.053, 0.053, 0.053, 0.053, 0.053, 0.1...	[0.56, 1.946, 1.946, 1.253, 1.946, 1.946, 1.94...	[0.0297, 0.1031, 0.1031, 0.0664, 0.1031, 0.103...
1	1	[this, is, ripe, and, fruity, a, wine, that, i...	[ripe, fruity, wine, smooth, still, structured...	[15, 12, 20, 16, 17, 18, 9, 19, 8, 13, 14, 4, ...	[0.048, 0.048, 0.048, 0.048, 0.048, 0.048, 0.0...	[1.946, 1.946, 0.847, 1.946, 1.946, 1.946, 1.9...	[0.0934, 0.0934, 0.0407, 0.0934, 0.0934, 0.093...
2	2	[tart, and, snappy, the, flavors, of, lime, fl...	[tart, snappy, flavors, lime, flesh, rind, dom...	[14, 11, 4, 7, 5, 10, 2, 6, 8, 9, 1, 0, 15, 16...	[0.059, 0.059, 0.118, 0.059, 0.059, 0.059, 0.0...	[1.946, 1.946, 1.253, 1.946, 1.946, 1.253, 1.9...	[0.1148, 0.1148, 0.1479, 0.1148, 0.1148, 0.073...
3	3	[pineapple, rind, lemon, pith, and, orange, bl...	[pineapple, rind, lemon, pith, orange, blossom...	[15, 17, 9, 16, 13, 3, 20, 0, 14, 2, 12, 11, 8...	[0.045, 0.045, 0.045, 0.045, 0.045, 0.045, 0.0...	[1.253, 1.253, 1.946, 1.946, 1.946, 1.946, 1.9...	[0.0564, 0.0564, 0.0876, 0.0876, 0.0876, 0.087...
4	4	[much, like, the, regular, bottling, from, 201...	[much, like, regular, bottling, 2012, comes, a...	[12, 11, 16, 2, 0, 4, 1, 15, 17, 20, 18, 7, 10...	[0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.0...	[1.946, 1.946, 1.946, 1.946, 1.946, 1.946, 1.9...	[0.0778, 0.0778, 0.0778, 0.0778, 0.0778, 0.077...
5	5	[blackberry, and, raspberry, aromas, show, a, ...	[blackberry, raspberry, aromas, show, typical,...	[2, 21, 1, 22, 25, 19, 26, 14, 16, 4, 17, 18, ...	[0.037, 0.037, 0.037, 0.037, 0.037, 0.037, 0.0...	[1.946, 1.946, 0.56, 1.946, 1.946, 1.946, 1.94...	[0.072, 0.072, 0.0207, 0.072, 0.072, 0.072, 0....
6	6	[here's, a, bright, informal, red, that, opens...	[bright, informal, red, opens, aromas, candied...	[4, 9, 13, 10, 1, 5, 3, 17, 12, 14, 8, 6, 11, ...	[0.056, 0.056, 0.056, 0.056, 0.056, 0.056, 0.0...	[1.946, 1.946, 1.253, 1.946, 0.56, 1.946, 1.25...	[0.109, 0.109, 0.0702, 0.109, 0.0314, 0.109, 0...

El resultado de la imagen anterior corresponde al primer punto solicitado en el desarrollo de esta práctica, que se trata de una tabla que contenga una colección de documentos y donde se pueda obtener la información de los términos, índices, TF, IDF y TF-IDF de cada uno de ellos.

Similaridad coseno entre cada par de documentos

Como hemos visto, cada documento de texto puede representarse vectorialmente. Para realizar la comparación de los textos representados vectorialmente, se procede a utilizar la similitud de coseno, obteniendo los valores de TF-IDF con TfidfVectorizer (sklearn), en lugar de aplicar los valores almacenados en la tabla anteriormente creada, ya que estos valores están redondeados y truncados a unos pocos decimales.

```
# ================================================================================
#  Vocabulario total de todos los documentos
# ================================================================================
vectorizer = TfidfVectorizer(analyzer = 'word', stop_words = myStopwords, use_idf=True)
vectorizer
vocabulario = []
for i in range(data.shape[0]):
  tfidf = vectorizer.fit_transform(data['Documento'])
```

vectorizer.get_feature_names() nos permite visualizar el vocabulario que forma la colección de documentos, habiendo excluido las stopwords, símbolos de puntuación, etc.

```
# ================================================================================
#  Permite visualizar el vocabulario que forma la colección de documentos
# ================================================================================
vectorizer.get_feature_names()
```

```
['20',
 '2012',
 'acidity',
 'across',
 'alongside',
 'already',
 'although',
 'apple',
 'aromas',
 'astringent',
 'balanced',
 'berry',
 'better',
 'bit',
 'blackberry',
```

tfidf.toarray() representa cada uno de los documentos como vectores, con los valores asignados por TF-IDF para cada uno de los términos. Aquellos valores donde aparezca cero, significan que el término no está presente para ese documento.

```
# ================================================================================
#  Valores asignados por TF-IDF para cada uno de los términos
# ================================================================================
tfidf.toarray()
```

```
array([[0.        , 0.        , 0.12203573, 0.        , 0.22615301,
        0.        , 0.        , 0.22615301, 0.13931464, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.22615301,
        0.22615301, 0.22615301, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.22615301, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.45230602, 0.        , 0.        , 0.        , 0.22615301,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.18772642,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.18772642,
        0.        , 0.        , 0.        , 0.        , 0.22615301,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.22615301, 0.        , 0.        , 0.        ,
        0.22615301, 0.16046232, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.22615301, 0.        , 0.        , 0.        ,
```

Matriz de similitud entre documentos

Obtenemos la matriz de similitud entre cada par de documentos utilizando cosine_similarity. Para una mejor visualización de la matriz, el resultado se muestra en un dataframe. Como puede apreciarse, la diagonal de la matriz está compuesta por "1", ya que representa la similitud de un documento consigo mismo. De igual modo, los valores más próximos a "1", indicarán que existe una mayor similitud entre ambos documentos.

```
# ================================================================================
#  Obtención de la matriz de similitud entre documentos.
# ================================================================================
similarity_matrix = cosine_similarity(tfidf,tfidf)
#similarity_matrix
pd.DataFrame(similarity_matrix)
```

	0	1	2	3	4	5	6
0	1.000000	0.015163	0.016154	0.044704	0.000000	0.063025	0.110201
1	0.015163	1.000000	0.044883	0.000000	0.023567	0.013742	0.142019
2	0.016154	0.044883	1.000000	0.075686	0.025108	0.118571	0.018682
3	0.044704	0.000000	0.075686	1.000000	0.000000	0.049032	0.051699
4	0.000000	0.023567	0.025108	0.000000	1.000000	0.028713	0.000000
5	0.063025	0.013742	0.118571	0.049032	0.028713	1.000000	0.072887
6	0.110201	0.142019	0.018682	0.051699	0.000000	0.072887	1.000000

Sistema de recomendación basado en contenido

Para realizar el sistema de recomendación, se ha decidido realizar la carga de otro documento que contiene un producto favorito del usuario. Para cargar el archivo realiza el preprocesado de los datos, al igual que se ha realizado para el archivo que contiene la colección de documentos. Se eliminan signos de numeración, puntuación, etc.

```
# ================================================================================
# Lectura y preprocesado de datos del fichero que contiene el artículo favorito
# ================================================================================
recomd = pd.read_csv("/content/gdrive/MyDrive/TAAD/SR/favorito.txt", sep='\n', header=None)
recomd.columns = ['Documento']
recomd['Documento'] = recomd['Documento'].str.strip('123456789.')
recomd['Documento'] = recomd['Documento'].str.replace(r"[^0-9a-zA-Z:']+", ' ')
recomd['Documento'] = recomd['Documento'].str.lower()
recomd
```

	Documento
0	pale straw yellow colour intense fruity aroma...

Una vez se ha almacenado el documento en el mismo formato que la colección de documentos, se procede a añadirlo a la colección, crear el vocabulario global y calcular los valores tf-idf y la matriz de similitud entre documentos.

```
# ========================================================================
# Añadimos la colección inicial de documentos a recomendar
# ========================================================================
df_total = data.loc[0:]
# ========================================================================
# Añadimos el documento que deseamos comparar
# ========================================================================
df_total = df_total.append(recomd, ignore_index=True)
# ========================================================================
# Añadimos los index al dataframe
# ========================================================================
df_total['id'] = np.arange(len(df_total)) # Añadimos la col 'id' para identificar a cada documento
df_total
```

	id	Documento
0	0	aromas include tropical fruit broom brimstone ...
1	1	this is ripe and fruity a wine that is smooth ...
2	2	tart and snappy the flavors of lime flesh and ...
3	3	pineapple rind lemon pith and orange blossom s...
4	4	much like the regular bottling from 2012 this ...
5	5	blackberry and raspberry aromas show a typical...
6	6	here's a bright informal red that opens with a...
7	7	pale straw yellow colour intense fruity aroma...

```
# ========================================================================
#  Vocabulario total de todos los documentos
# ========================================================================
vectorizer = TfidfVectorizer(analyzer = 'word', stop_words = myStopwords, use_idf=True)
vectorizer
vocabulario = []
for i in range(data.shape[0]):
  tfidf = vectorizer.fit_transform(df_total['Documento'])

# ========================================================================
#  Obtención de la matriz de similitud entre documentos.
# ========================================================================
similarity_matrix = cosine_similarity(tfidf,tfidf)
#similarity_matrix
matrix_similitud = pd.DataFrame(similarity_matrix)
matrix_similitud
```

	0	1	2	3	4	5	6	7
0	1.000000	0.016797	0.017579	0.047563	0.000000	0.058711	0.116225	0.078128
1	0.016797	1.000000	0.047794	0.000000	0.024806	0.015243	0.148121	0.064156
2	0.017579	0.047794	1.000000	0.077253	0.025960	0.122656	0.020161	0.000000
3	0.047563	0.000000	0.077253	1.000000	0.000000	0.051532	0.054549	0.030932
4	0.000000	0.024806	0.025960	0.000000	1.000000	0.029579	0.000000	0.000000
5	0.058711	0.015243	0.122656	0.051532	0.029579	1.000000	0.067335	0.092110
6	0.116225	0.148121	0.020161	0.054549	0.000000	0.067335	1.000000	0.026804
7	0.078128	0.064156	0.000000	0.030932	0.000000	0.092110	0.026804	1.000000

Al observar la última fila, pueden apreciarse los valores de similitud entre el nuevo documento añadido y los documentos que estaban presentes en la colección a recomendar.

Viendo la fila anterior, se puede concluir que, de todos los documentos de la colección, el orden de recomendación con respecto al último documento añadido, que se trata de un artículo que el usuario ha colocado como favorito, es el siguiente:

```
# ========================================================================
# Ordenar por los valores de la columna
# ========================================================================
#recomendacion.sort_values(by= 7, ascending=False)
recomendacion = recomendacion.sort_values(by= 7, ascending=False)
recomendacion
```

	7
5	0.092110
0	0.078128
1	0.064156
3	0.030932
6	0.026804
2	0.000000
4	0.000000

Se decide mostrar únicamente los 3 primeros elementos a recomendar según orden de similitud, y se obtiene como salida que los tres documentos que presentan mayor similitud con respecto al contenido del último elemento añadido (archivo favorito.txt) son:

```
# ========================================================================
# Salida del Recomendador. Muestra los 3 elementos que presentan mayor similitud con respecto al último documento añadido.
# ========================================================================
recomendacion = recomendacion.rename(columns = {7:"Similitud"})
recomendacion = recomendacion.rename_axis('Documento').reset_index()
recomendacion.head(3)
```

	Documento	Similitud
0	5	0.092110
1	0	0.078128
2	1	0.064156

Conclusión

Si comparamos la descripción del documento favorito con la descripción del documento que presenta mayor similitud (documento 5), puede concluirse que, de la lista de productos de la colección, el documento recomendado es similar al documento añadido como favorito.

Documento favorito:

Pale straw yellow colour. Intense fruity aroma (ripe and tropical fruit). Light herbaceous notes, characteristic of the Listán Blanco variety from the Canary Islands. Its passage through the mouth is light, fresh, and the memory of fruit returns.

Documentos con mayor similitud (Documento 5):

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.

Como conclusión del resultado obtenido, se puede confirmar que el sistema de recomendación ha elegido un vino que presenta unas características similares a las marcadas por el usuario como favorito, eligiendo en las dos primeras opciones a recomendar un producto afrutado. Aunque las frutas de ambos productos no coinciden, cosa que inicialmente parece clave para recomendar un producto de degustación en función a otro, la descripción de ambos vinos y el resultado final tras la cata es similar.