PRÁCTICA 2 (2/2)
Data Mining

Inteligencia Artificial en las Organizaciones
Grado en Ingeniería Informática
Curso 2020/2021

INTRODUCCIÓN

Text Mining es un campo en el que muchos investigadores están prestando gran atención últimamente. Una de las principales características de este campo es que los datos procesados son no estructurados o semi-estructurados.

En esta práctica, nos centraremos en obtener una representación adecuada de una serie de documentos como un vector de frecuencias ponderadas, es decir, de frecuencias de las palabras del mismo ponderadas por su relevancia. Así, supongamos que disponemos de una serie de documentos, de los cuales queremos tanto obtener una representación adecuada, como poder comparar su similitud con nuevos documentos. Para realizar este objetivo, vamos a utilizar la herramienta Weka con la que se pueden realizar procesos para el tratamiento de textos (minería de datos) de forma rápida y eficiente.

Lo primero que debemos realizar es un proceso de limpieza (eliminando signos de puntuación, paréntesis, palabras muy frecuentes como artículos, pronombres, etc…) de cada uno de los documentos. De esta forma, obtenemos un corpus de palabras (vocabulario).

OBJETIVO

Para esta práctica, utilizaremos un corpus formado por una colección de textos de reseñas (reviews). Estas reseñas se tomaron del sitio web *Tripadvisor* como parte de un proyecto de extracción automática de sentimientos del texto[1].

[1] Alam, M. H., Ryu, W.-J., Lee, S., 2016. Joint multi-grain topic sentiment: modeling semantic aspects for online reviews. Information Sciences 339, 206–223.

A través de la plataforma *Tripadvisor*, viajeros de todo el mundo pueden consultar más de 859 millones de comentarios y opiniones de 8,6 millones de alojamientos, restaurantes, experiencias, vuelos y cruceros. Además del comentario, cada reseña va acompañada de una puntuación (entre 1 y 5 estrellas).

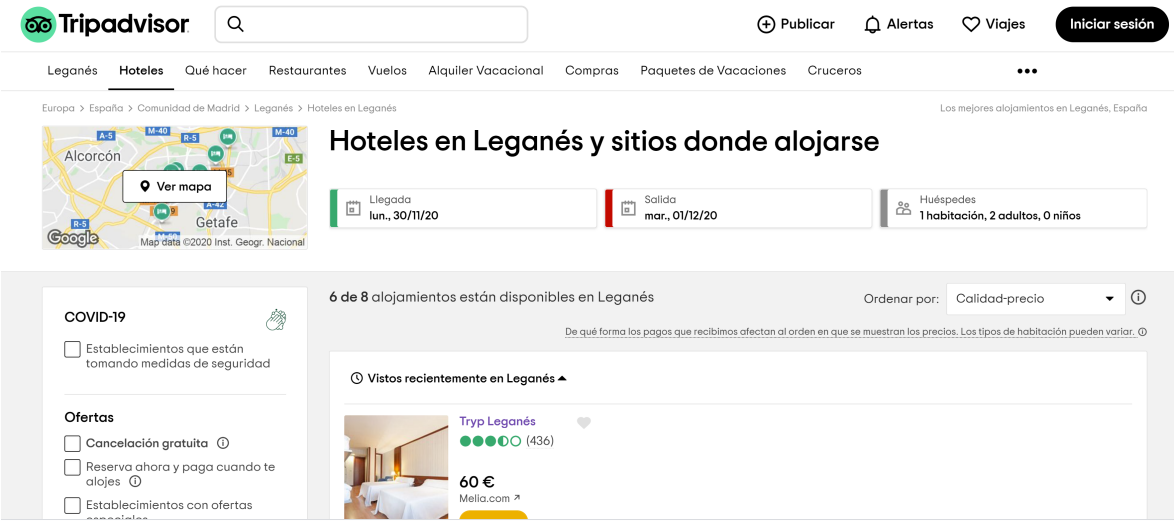

Figura 1: Página web de Tripadvisor

Para el tratamiento de los datos de esta práctica se han asignado 5 categorías que han sido definidas según el número de estrellas de la reseña (Tabla 1)

Tabla 1: Asignación de categorías en función de la puntuación.

Puntuación en Tripadvisor	Categoría
★ ☆ ☆ ☆ ☆	*Poor*
★ ★ ☆ ☆ ☆	*Fair*
★ ★ ★ ☆ ☆	*Good*
★ ★ ★ ★ ☆	*VeryGood*
★ ★ ★ ★ ★	*Excellent*

Los datos se pueden descargar desde Aula Global. El objetivo de esta práctica es realizar análisis de reseñas en función de estas cinco categorías.

Sesión I: DESARROLLO

Lo primero que debemos hacer es descargar los datos de Aula Global contenidos en el archivo de extensión .csv de nombre "*Tripadvisor_Reviews*". Este archivo consta de 2 columnas: la reseña (Columna A) y la puntuación (Columna B). Vamos a crear 5 carpetas correspondientes a las 5 categorías dependiendo de la puntuación: Poor, Fair, Good, VeryGood y Excellent. Filtramos en el .csv por cada una de las categorías y guardamos el resultado como un archivo Excel con Macros Habilitadas en su carpeta correspondiente. Desde este directorio, abrimos el Excel y en la pestaña "Programación", creamos una nueva macro y copiamos el código del archivo "*MacroFiletoText.txt*" de la carpeta "*Recursos*" de Aula Global. Al ejecutar, este programa generará 500 archivos de texto (correspondientes a las 500 primeras reseñas) que se guardarán en dicho directorio. Repetimos el proceso por cada una de las categorías obteniendo 5 carpetas (una por cada categoría: Poor, Fair, Good, VeryGood y Excellent) con 500 reseñas formato .txt cada una[2].

Una vez tengamos las 2500 reseñas que vamos a utilizar para esta práctica, procedemos a obtener es el fichero WEKA (con extensión .arff) que contenga todos los documentos como String. El fichero deberá contener un atributo (texto) de tipo String y si es necesario, la clase a la que corresponde cada texto. Para generarlo, deberemos tener almacenados los datos de la siguiente forma:

- **Cinco (una por categoría) carpetas con 500 documentos de texto cada una**: *Poor, Fair, Good, VeryGood y Excellent.*

Cada documento a tratar será una instancia diferente y por lo tanto el texto de cada documento deberá ir entre comillas simples en una única fila.

Sin embargo, este proceso (paso de ficheros de texto a fichero .arff) se puede hacer utilizando desde la opción SimpleCLI de Weka (el último botón de la interfaz inicial de Weka). Así, desde esta interfaz, deberemos introducir el siguiente comando:

```
java weka.core.converters.TextDirectoryLoader -dir C:\MiPath\fodelsData > C:\MiPath\data8Categories.arff
```

[2] Las carpetas deben contener únicamente archivos de texto. No olvidar quitar los archivos de formato Excel utilizados para la generación de documentos del directorio.

donde el primer comando indicará el directorio en el que se encuentran las 5 carpetas y la salida será el fichero (arff) donde se almacenen todos los textos (string). Además, dado que los ficheros están divididos en diferentes carpetas, se añadirá al fichero arff de salida el atributo clase. Puede verse como ejemplo la siguiente figura:

```
@relation C__Users_asuva_Desktop_IAO_Practicas_Tripadvisor_Data

@attribute text string
@attribute @@class@@ {Excellent,Fair,Good,Poor,VeryGood}

@data

'unique, great stay, wonderful time hotel monaco, location excellent short stroll main downt
'ok nothing special charge diamond member hilton decided chain shot 20th anniversary seattle
lp bags.prior arrival emailed hotel inform 20th anniversary half really picky wanted make su
'nice rooms not 4* experience hotel monaco seattle good hotel n\'t 4* level.positives large
try cover face night bright blue light kept, got room night no, 1st drop desk, called mainta
'horrible customer service hotel stay february 3rd 4th 2007my friend picked hotel monaco app
ed saying check 12 remind package, finally packed things went downstairs check, quickly sign
'nice hotel expensive parking got good deal stay hotel anniversary, arrived late evening too
'excellent stay, delightful surprise stay monaco, thoroughly enjoyed stay, room comfortable
'seattle crown plaza not worth money got late hotel 1230pm gave away reservation king sized
'ace not place husband stayed ace hotel seattle nights excited book room style hotel far all
cceptable solution did wake early deal car, night moved smaller interior room considerably q
'terrible hotel approximately 2 weeks ago april 25 2007 reservation hotel night 19 2007, wro
'good choice seattle stayed night business booked company, overall satisfactory arrived late
'great value clean modern style, reception staff friendly gave complimentary upgrade traditi
```

Figura 2: Ejemplo de archivo.arff generado

Una vez que tenemos el fichero en el formato necesario para utilizar Weka, éste puede ser abierto en el EXPLORER de WEKA. Como podemos ver, puede tener uno o varios atributos (mínimo un atributo de texto). Para obtener el corpus de palabras de los textos, así como información relevante sobre cada una estas palabras, es necesario transformar las cadenas de texto en atributos (términos).

Para ello, aplicaremos el filtro **StringToWordVector** (filters -> unsupervised -> attribute) que nos permite convertir una cadena de caracteres en un vector de ocurrencia binaria de cada término (palabra) que aparece. Este filtro tiene varias opciones y se **podrán generar distintos conjuntos de datos para luego aplicar distintos algoritmos de aprendizaje y realizar un estudio comparativo**.

A continuación, se muestra información (obtenida de la aplicación Weka – *More)* sobre las distintas opciones configurables de este filtro. En negrita, se marcan aquellas que son muy relevantes y que el alumno debe considerar en el desarrollo de la práctica.

weka.filters.unsupervised.attribute.StringToWordVector

SYNOPSIS

Converts string attributes into a set of numeric attributes representing word occurrence information from the text contained in the strings. The dictionary is determined from the first batch of data filtered (typically training data). Note that this filter is not strictly unsupervised when a class attribute is set because it creates a separate dictionary for each class and then merges them.

OPTIONS

attributeNamePrefix -- Prefix for the created attribute names. (default: "")

stopwordsHandler -- The stopwords handler to use (Null means no stopwords are used).

wordsToKeep -- The number of words (per class if there is a class attribute assigned) to attempt to keep.

debug -- If set to true, filter may output additional info to the console.

outputWordCounts -- Output word counts rather than boolean 0 or 1(indicating presence or absence of a word).

lowerCaseTokens -- If set then all the word tokens are converted to lower case before being added to the dictionary.

tokenizer -- The tokenizing algorithm to use on the strings.

doNotCheckCapabilities -- If set, the filter's capabilities are not checked before it is built. (Use with caution to reduce runtime.)

doNotOperateOnPerClassBasis -- If this is set, the maximum number of words and the minimum term frequency is not enforced on a per-class basis but based on the documents in all the classes (even if a class attribute is set).

attributeIndices -- Specify range of attributes to act on. This is a comma separated list of attribute indices, with "first" and "last" valid values. Specify an inclusive range with "-". E.g: "first-3,5,6-10,last".

normalizeDocLength -- Sets whether if the word frequencies for a document (instance) should be normalized or not.

saveDictionaryInBinaryForm -- Save the dictionary as a binary serialized java object instead of in plain text form.

invertSelection -- Set attribute selection mode. If false, only selected attributes in the range will be worked on; if true, only non-selected attributes will be processed.

minTermFreq -- Sets the minimum term frequency. This is enforced on a per-class basis.

TFTransform -- Sets whether if the word frequencies should be transformed into log(1+fij) where fij is the frequency of word i in document (instance) j.

periodicPruning -- Specify the rate (x% of the input dataset) at which to periodically prune the dictionary. wordsToKeep prunes after creating a full dictionary. You may not have enough memory for this approach.

stemmer -- The stemming algorithm to use on the words.

dictionaryFileToSaveTo -- The path to save the dictionary file to - an empty path or a path '-- set me --' means do not save the dictionary.

IDFTransform -- Sets whether if the word frequencies in a document should be transformed into:

fij*log(num of Docs/num of Docs with word i)

where fij is the frequency of word i in document (instance) j.

En estas opciones, es de especial importancia el filtro es Stemmer, que nos permite eliminar del corpus, plurales, derivaciones…

Por otra parte, en la opción *stopwordsHandler* deberemos seleccionar un fichero con las *stopwords*[3] que consideremos para lengua inglesa (puede obtenerse de internet).

La opción *tokenizer*, también es muy relevante, ya nos permite seleccionar no sólo palabras, sino subsecuencias de n palabras (*n-gramas*). Además, podremos seleccionar el número de palabras de la subsecuencia.

Weka nos permite calcular también el valor *TF-IDF* de cada término, para ello, utilizaremos las siguientes opciones:

IDFTransform → True
TFTransform → True
outputWordCounts → True

Para entender qué realiza este filtro, se puede pulsar el botón "*Edit*" que editará el nuevo fichero generado. Una vez aplicado el filtro, podremos obtener la lista de términos y su la frecuencia en cada documento.

En la pestaña "*Select Attributes*" podremos seleccionar determinados atributos. A continuación, en la ventana de preprocesado podremos eliminar los atributos que consideremos que no son relevantes o calcular otros atributos utilizando los filtros disponibles. Cada vez que realicemos un cambio, podemos guardarlos en distintos ficheros con la finalidad de realizar varias pruebas.

[3] Cuando hablamos de stop words o palabras vacías nos referimos a todas aquellas palabras que carecen de un significado por si solas. Las palabras vacías suelen ser artículos, preposiciones, conjunciones, pronombres, etc…

Una vez preprocesados los datos, cada documento será una instancia con gran cantidad de atributos. Teniendo esto en cuenta, podremos realizar operaciones de clasificación y agrupamiento. Para esta práctica, se pide mostrar y analizar los resultados obtenidos al aplicar algoritmos de clasificación. Se utilizará el Experimenter de Weka para hacer las comparaciones.

Sesión II: CLUSTERING

En esta segunda parte de la práctica, vamos a pasar del problema de clasificación presentado en la anterior sesión a un problema de clustering. Seguiremos utilizando Weka y los datos de la primera sesión.

Para llevar a cabo el proceso de clustering se deberá:

- Utilizar y comparar, al menos, diez modelos creados por el algoritmo K-medias (SimpleKMeans) variando el número de clústeres, las semillas y/o la función de distancia. Describir los resultados.
- Analizar, en detalle, al menos uno[4] de los modelos creados.
- Complementar el análisis del modelo seleccionado mediante la descripción de los clústeres. Para ello, se utilizará un algoritmo de generación de reglas y/o árboles de decisión.
    - Para llevar a cabo este análisis, hay que etiquetar las instancias del conjunto de datos utilizando Weka. Esto se lleva a cabo seleccionando el modelo que se desea analizar en la ventana de "Result list" y, en el menú contextual, seleccionar "Visualize clusters assigments". Por último, hay que seleccionar el botón de "Save".
    - Abrir el nuevo fichero arff y eliminar el atributo "Instance_number".
    - Aplicar algoritmos de generación de reglas y/o árboles de decisión (PART, j48… etc.).
    - Describir los clústeres en función del modelo generado.

[4] Se recomienda utilizar alguna medida de evaluación para determinar el modelo que se analizará. https://www.sciencedirect.com/book/9780128042915/data-mining

Evaluación de la práctica

Aspectos a evaluar en la corrección de la práctica:

- Planteamiento y desarrollo del problema: 25%.
- Resultados del problema: 25%.
- Análisis de resultados y conclusiones: 25%.
- Presentación: 15%.
- Contexto de la práctica (Información complementaria sobre el desarrollo de la práctica): 10%.

Debe darse importancia a la presentación para mostrar los resultados, el análisis de los mismos, conclusiones, etc. Es importante tener en cuenta que el contexto de la práctica en la que se incluirá cualquier información relevante conocida por el estudiante, casos similares, noticias al respecto o cualquier otra información de interés y relacionada con la práctica.

Entrega de la práctica

- En esta parte de la práctica se requiere la realización de un documento explicando del conjunto de datos:
  - ✓ La descripción de los datos utilizados.
  - ✓ La descripción detallada de los diferentes filtros aplicados y su justificación.
  - ✓ El proceso de entrenamiento: Los diferentes clasificadores probados, los problemas encontrados, etc. Además, se debe incluir algún clasificador que pueda ser "interpretable".
  - ✓ Descripción y análisis detallado de los resultados
  - ✓ Conclusiones de la práctica.
  - ✓ Contexto de la práctica.
- La práctica deberá realizarse en grupos de 3/4 personas (y entregarse únicamente por uno de los integrantes del grupo).

- Esta práctica está dividida en dos partes. En este documento se detalla la primera de las partes (1/2). Sin embargo, la entrega de la práctica será un único documento en el que se detallen y analicen y relacionen las dos partes de la Práctica 2. Dicha entrega será un documento y todos los ficheros que se consideren relevantes para la evaluación de la práctica. Todos los ficheros que se entregan deberán describirse brevemente en un fichero de texto.
- La entrega de esta Práctica 2 (1/2 y 2/2) se realizará con fecha máxima de entrega (por Aula Global):
  - Grupo 83: 3 de noviembre de 2020 – 12:00h
  - Grupo 80 y 84: 5 noviembre de 2020 – 12:00h
- No hay un formato de entrega establecido. Sin embargo, es importante que toda la información se muestre de forma clara y su presentación también es considerada para la nota de la práctica.