Práctica 2
Análisis de las reseñas de Tripadvisor

Inteligencia Artificial en las Organizaciones

Grupo 83-1

Miguel Gutiérrez Pérez
100383537@alumnos.uc3m.es

Mario Lozano Cortés
100383511@alumnos.uc3m.es

Alba Reinders Sánchez
100383444@alumnos.uc3m.es

Alejandro Valverde Mahou
100383383@alumnos.uc3m.es

GitHub: ***InteligenciaArtificialOrganizaciones***

31 de octubre de 2020

Índice

1. Introducción

La **Minería de Texto** es una técnica de minería de datos que busca extraer información útil y relevante de documentos de texto de diferentes fuentes diferentes, como puede ser páginas web, correos electrónicos, periódicos o redes sociales. Para ello se hace una identificación de patrones en los datos, como puede ser la repetición de palabras o conjuntos de palabras, estructuras sintácticas que se repiton a lo largo de los datos, etc.

Esta minería de texto tiene numerosas aplicaciones, y en esta práctica se van a desarrollar una clasificación en función a unas categorías predefinidas y un agrupamiento sin tener en cuenta estas categorías.

La colección de textos que se va a usar en la práctica consiste en un conjunto de reseñas de la página web *Tripadvisor*, donde cada una tiene una clasificación de 1 a 5 estrellas.

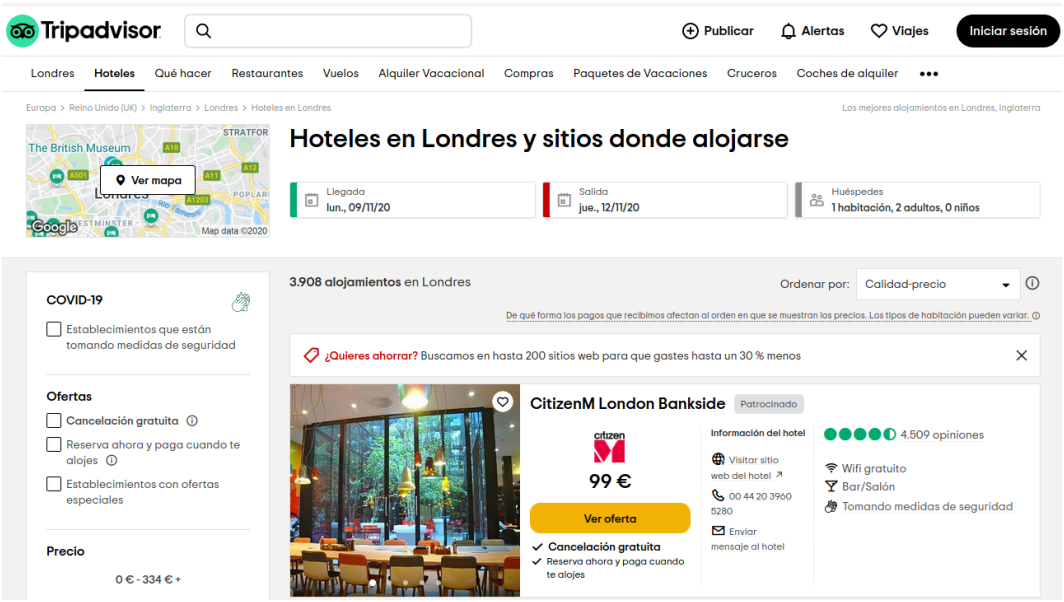

Figura 1: Página de búsqueda de Tripadvisor

Las reseñas están en inglés, y en texto en plano, por lo tanto, para poder ser tratadas, tendran que pasar por un proceso de preparación de datos.

Figura 2: Ejemplode reseña de Tripadvisor en *español*

En la práctica se plantean dos problemas, uno de **aprendizaje supervisado** (clasificación), donde el objetivo es determinar la puntuación que le da un usuario a un hotel, en base a lo que escribe en su reseña; y otro de **aprendizaje no supervisado** (agrupamiento), cuyo objetivo es agrupar las diferentes reseñas en función a su contenido, sin tener en cuenta su puntuación.

2. Parte 1: Clasificación

2.1. Análisis y preprocesado de datos

El primer paso que debe ser abordado en el desarrollo de esta práctica es el del procesado de los datos con el objetivo de **posibilitar la aplicación de minería de texto** gracias a la herramienta *Weka*. Los datos sin tratar se encuentran contenidos en un archivo de formato *csv* donde la primera columna se corresponde de con las reseñas en formato de texto y la segunda con el número de estrellas (valoración del 1 al 5) correspondientes a dicha reseña. De esta manera, la secuencia de operaciones lógicas a seguir para posibilitar el análisis con la herramienta propuesta son:

- Transformación del fichero desde el formato ***csv*** al formato ***arff***
- Procesamiento propio de las técnicas de minería de texto en la herramienta *Weka*

2.1.1. De .csv a .arff

Como se ha comentado anteriormente el material proporcionado consta de un archivo en formato *csv*. La mejor manera de convertirlo al formato *arff* utilizado por *Weka* es generar una estructura de directorios en función de la clasificación de la reseña para una vez dentro de cada uno de los directorios, encontrar un fichero por cada una de las reseñas. Dicha estructura se puede generar gracias a la herramienta de Macros contenida dentro del programa *Excel*. Cabe destacar que esta estructura es óptima para el propósito seguido puesto que se proporciona un comando que en la opción *CLI* de *Weka* produce directamente un fichero *arff* a partir de la estructura descrita.

2.1.2. Procesamiento específico de minería de texto

Una vez dentro de la herramienta *Weka* se deben considerar técnicas de procesamiento específicas de la minería de texto, concretamente el filtro no supervisado ***StringToWordVector***, el cual permite convertir atributos basados en cadenas de caracteres en un conjunto numérico que representa la ocurrencia de las distintas palabras contenidas en el archivo seleccionado. Este filtro contiene **multitud de parámetros interesantes** que merece la pena reseñar de cara a la experimentación que se llevará a cabo para obtener el mejor modelo posible.

- **StopWords**: Formadas por palabras sin significado, es decir, aquellas que **no aportan información útil** al proceso de minería de texto que se pretende realizar. Se aporta una lista para la realización de la práctica, sin embargo **es importante reseñar que se añaden algunas más a ella para mejorar los resultados del análisis**. Algunas de las palabras añadidas incluyen números y palabras sin sentido consecuencia posiblemente de faltas de ortografía a la hora de escribir las reseñas.
- **TFTransform**: Establece si la frecuencia de las palabras debe transformarse a $log(1+f_{ij})$ siendo f_{ij} la frecuencia de la palabra i en el documento j.

- **IDFTransform**: Establece su la frecuencia de palabras en un documento debe trasnformarse a $f_{ij} * log(\frac{numeroDocumentos}{numeroDocumentos\_con\_i})$ siendo f_{ij} la frecuencia de la palabra i en el documento j.
- **outputWordCounts**: Número exacto de ocurrencias de una palabra en vez de indicar únicamente presencia.
- **stemmer**: Algoritmo de *stemming* a usar. Conviene recordar que un algoritmo de *stemming* trata de reducir las palabras a sus raíces.
- **normalizeDocLength**: Establece si se normalizan las frecuencias de palabras de un documento.
- **minTermFreq**: Determina el número mínimo de ocurrencias que debe tener una palabra para se tenida en cuenta.
- **tokenizer**: Algoritmo de *tokenizing* que se aplica. Conviene recordar que un algoritmo de *tokeninzing* divide una secuencia de caracteres en tokens que pueden ser desde palabras a frases completas.

2.2. Experimentación

Para realizar los experimentos, se han realizado combinaciones de los distintos opciones que habilita la herramienta de *Weka* con el filtro *'StringToWordVector'*, para buscar la combinación que permita obtener los mejores resultados para este conjunto de datos. Este filtro, tal y como su nombre indica, transforma un atributo compuesto por texto, en un conjunto de atributos que representa la información de ese texto.

2.2.1. Experimentación básica

Los primeros experimentos que se han realizado consisten en modifcar exclusivamente una opción del filtro en cada uno de los experimentos, para comprobar cuales son los que generan los mejores resultados por separado. Los experimentos realizados son los siguientes:

- **Experimento 0**: Es el experimento base, respecto al que se van a comparar el resto.
- **Experimento 1**: En este experimento se prueba las combinaciones que se pueden hacer entre la opción *IDFTransform* y *TFTransform*, por lo que está compuesto de 3 subexperimentos.
  - **Experimento 1-1**: *IDFTransform* **True** y *TFTransform* **False**.
  - **Experimento 1-2**: *IDFTransform* **False** y *TFTransform* **True**.
  - **Experimento 1-3**: *IDFTransform* **True** y *TFTransform* **True**.
- **Experimento 2**: Este experimento prueba a activar la opción *outputWordCounts*
- **Experimento 3**: En este experimento se prueba el uso de diferentes *stemmer*, con dos subexperimentos.
  - **Experimento 3-1**:Usando el *LovinsStemmer*
  - **Experimento 3-2**:Usando el *IterativeLovinsStemmer*
- **Experimento 4**: En este experimento se prueba a cambiar el valor por defecto de la opción de *minTermFreq*, que es 1. Tiene 7 subexperimentos.
  - **Experimento 4-1**: El valor de *minTermFreq* es 2

- **Experimento 4-2**: El valor de *minTermFreq* es 5
- **Experimento 4-3**: El valor de *minTermFreq* es 10
- **Experimento 4-4**: El valor de *minTermFreq* es 25
- **Experimento 4-5**: El valor de *minTermFreq* es 125
- **Experimento 4-6**: El valor de *minTermFreq* es 250
- **Experimento 4-7**: El valor de *minTermFreq* es 625

- **Experimento 5**: Este experimento prueba la eficacia de la opción *normalizeDocLength* sobre tod el conjunto de datos.

ID Experimento	J48	RandomForest	JRip	IBk	Naive Bayes
0	38.88 %	51.80 %	35.28 %	26.12 %	49.76 %
1-1	38.88 %	52.64 %	33.32 %	26.12 %	49.76 %
1-2	38.88 %	52.04 %	35.64 %	26.12 %	49.76 %
1-3	38.88 %	51.84 %	33.76 %	26.12 %	49.76 %
2	44.92 %	59.92 %	40.36 %	27.76 %	49.28 %
3-1	38.88 %	51.56 %	34.92 %	26.12 %	49.76 %
3-2	38.44 %	51.68 %	34.96 %	26.36 %	49.76 %
4-1	38.88 %	51.56 %	33.32 %	26.12 %	49.76 %
4-2	38.88 %	51.56 %	33.32 %	26.12 %	49.76 %
4-3	38.56 %	52.32 %	34.16 %	26.64 %	49.80 %
4-4	39.16 %	52.44 %	36.20 %	28.72 %	50.32 %
4-5	38.68 %	49.16 %	34.96 %	33.48 %	47.32 %
4-6	34.64 %	34.64 %	40.96 %	32.48 %	40.44 %
4-7	27.04 %	27.04 %	25.80 %	25.56 %	25.12 %
5	39.20 %	51.68 %	34.04 %	20.00 %	48.8 %

Tabla 1: Experimentos realizados

2.2.2. Experimentación avanzada

Como en los experimnentos básicos se prueba que los mejores resultados son siempr obtenidos con el algoritmo de **RandomForest**, en esta experimentación avanzada, tan solo se va a evaluar con él.

La experimentación avanzada consiste en realizar combinaciones entre las opciones del filtro que generan mejores resultados en el apartado anterior, para intentar encontrar una transformación que maximice el resultado obtenido.

- **Experimento 6**: En este útimo experimento se analizan los resultados que se obtienen al cambiar la opciónd e *tokenizer*. Tiene 2 subexperimentos.
  - **Experimento 6-1**: Se utiliza el *tokenizer* de *NGramTokenizer*
  - **Experimento 6-2**: Se utiliza el *tokenizer* de *AlphabeticTokenizer*
- **Experimento 7**: Se prueba una combinación de *IDFTransform*, *TFTransform* y *outputWordCounts*.
- **Experimento 8**: Se seleccionan las siguientes opciones: *IDFTransform*, *TFTransform*, *outputWordCounts*, como *tokenizer* el *NGramTokenizer* y un valor de *minTermFreq* de 25.

ID Experimento	RandomForest
6-1	52.12 %
6-2	51.40 %
7	59.00 %
8	60.28 %

Tabla 2: Experimentos avanzados

2.3. Comentario de los resultados obtenidos

Mucho text

3. Parte 2: Clustering

3.1. Experimentación

3.2. Mejor resultado

10 modelos de K Medias variando n Clústeres, semillas y funciones de distancia
Analizar al menos 1 (en detalle)
clasificación sobre el modelo y sacar el árbol

4. Conclusiones

5. Referencias

1. Introduction to Neurons in Neural Networks. Medium. Consultado en Octubre 2020. Url: https://medium.com/artificial-neural-networks

6. Anexos

1. Perceptron Multicapa usando '*K Fold*'
   perceptron_kfold.py
2. Perceptron Multicapa usando '*split percentage*'
   perceptron_split.py
3. Programa para realizar la predicción de los modelos
   predict.py
4. Tabla de resultados de los experimentos de la primera parte
   valores_reales_vs_predicciones_&_errores_absolutos_parte1.xlsx
5. Tabla de resultados de los experimentos de la segunda parte
   valores_reales_vs_predicciones_&_errores_absolutos_parte2.xlsx