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***

2 de noviembre 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 repitan 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.

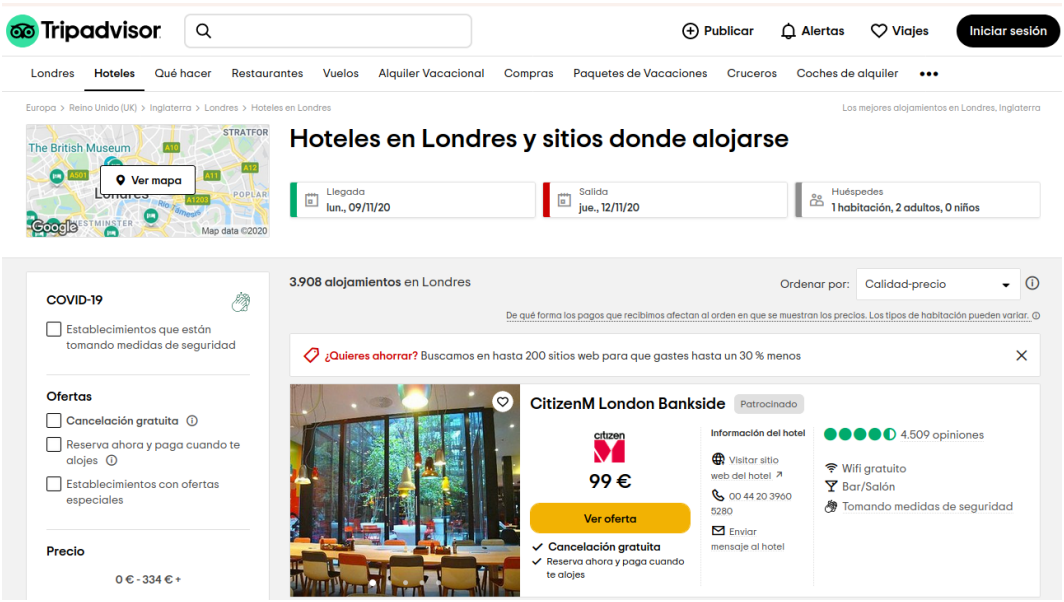

Figura 1: Página de búsqueda de Tripadvisor

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

Figura 2: Ejemplo de 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 llevar a cabo los experimentos se han realizado combinaciones de las distintas 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 modificar exclusivamente una opción del filtro en cada uno de los experimentos, para comprobar cuáles 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 todo el conjunto de datos.

Para cada uno de los experimentos mostrados se deben ejecutar distintos algoritmos de clasificación con el objetivo de averiguar cuál de ellos funciona mejor dada la configuración elegida por el experimento. Los algoritmos de clasificación seleccionados por ser los más ampliamente extendidos son:

- *J48*
- *RandomForest*
- *JRip*
- *IBk*
- *Naive Bayes*

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 en clasificación

La Tabla 1 muestra la relación de experimentos realizados junto con los resultados en cada uno de los algoritmos de clasificación seleccionados. Un vistazo rápido a la misma permite averiguar que el algoritmo de clasificación más prometedor es ***RandomForest***.

Por lo tanto, a continuación se abre una nueva ventana en la experimentación, la cual tratará de afinar la configuración de los distintos parámetros propuestos en los experimentos para *RandomForest*.

2.2.2. Experimentación avanzada

Como en los experimnentos básicos se prueba que los mejores resultados son siempre 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 opción de *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 en clasificación

La Tabla 2 muestra los resultados obtenidos en la experimentación avanzada, siendo el experimento 8 el que obtiene los mejores resultados. La siguiente sección trata de dar sentido a estos resultados.

2.3. Comentario de los resultados obtenidos

La sección experimentación básica permite descubrir qué algoritmo de clasificación tiene los mejores resultados además de dar **indicaciones sobre qué configuración del filtro de minería de datos mejora** los mismos.

Después, **la sección avanzada descubre cómo es posible mejorar el clasificador seleccionado** a través de la afinación de parámetros del filtro, descubriendo, como era de esperar, que una combinación de los mejores parámetros del filtro permite añadir exactitud al clasificador sumando los efectos de las mejoras aplicadas.

Así los resultados arrojados seleccionan al *experimento 8*, y que **limita el ruido** en los datos incluyendo una **frecuencia mínima de aparición** de términos. Además, permite **aumentar la precisión** al establecer un algoritmo de *tokeninzing*, que unido al establecimiento de **métricas sobre la frecuencia y número de ocurrencias** de las palabras permite aumentar y mejorar la calidad de la información extraída de los datos, permitiendo así una generalización mayor, logrando de esta manera cumplir con el objetivo de clasificar las reseñas de *Tripadvisor*.

3. Parte 2: Clustering

En la segunda parte de esta práctica se va a realizar una aproximación mediante ***clustering***, recordar que esta técnica consiste en agrupar instancias sin etiquetar de manera que las pertenecientes a un mismo grupo sean más similares entre sí que con las del resto.

En este caso, se agruparán las instancias procesadas que obtuvieron un mejor resultado en la primera parte. Esta agrupación se hará con el algoritmo ***K-Medias***, que se basa en el valor medio de las distancias de cada grupo, aunque la primera agrupación ha de fijarse, en este caso, con una semilla.

Por último, como el proceso de *clustering* no muestra qué le ha hecho agrupar a unas instancias con otras, para intentar entenderlo, se aplicarán sobre los *clusters* técnicas de ***clasificación***, tanto de árboles de decisión como de reglas.

Todo este proceso se vuelve a realizar en *Weka* y, como resumen, se compone de los siguientes pasos:

- Cargar el archivo *.arff* con los mejores datos generados en la primera parte.
- Generar diferentes modelos de *clustering* con *K-Medias* (ignorando el atributo *class*).
- Comparar los modelos y decidir cuál funciona mejor.
- Ejecutar diversos algoritmos de clasificación que expliquen como se ha formado.

3.1. Experimentación

Los experimentos que se llevan a cabo son los que se muestran en la en la Figura 3. Los parámetros del algoritmo que se modifican son la *seed* (10, 20 y 30), el *números de clusters* (2, 3, 4, 5 y 6) y, el *tipo de distancia* (Euclídea y Manhattan).

ID Experimento	Error	Seed	Número de clusters	Tipo de distancia
0	25436,31171	10	5	Euclidea
1	25382,55604	20	5	Euclidea
2	25218,09711	30	5	Euclidea
3	25703,79957	10	2	Euclidea
4	25704,73815	20	2	Euclidea
5	25704,73815	30	2	Euclidea
6	25545,30221	10	3	Euclidea
7	25684,13735	20	3	Euclidea
8	25643,51841	30	3	Euclidea
9	56512,35917	10	5	Manhattan
10	56666,77425	20	5	Manhattan
11	56680,27896	30	5	Manhattan
12	25509,57983	10	4	Euclidea
13	25670,38789	20	4	Euclidea
14	25449,76246	30	4	Euclidea
15	25250,01674	10	6	Euclidea
16	25342,88323	20	6	Euclidea
17	25160,2531	30	6	Euclidea

Figura 3: Experimentos con *K-Medias*

Como se puede apreciar en la Figura 3, se prueba para un mismo número de *clusters* las 3 *seeds*. Cabe destacar que los experimentos probados con distancia *Manhattan* cometen mucho más error que los que usan la *Euclídea*. Por este motivo, en la gráfica donde se comparan los errores no se añaden, ya que no son relevantes.

Los errores cometidos por los experimentos que usan distancia *Euclídea* varían en el rango de 25160.25, siendo este el mejor, a 25704.73, siendo este el peor. De estos, los mejores resultados son los marcados en verde.

Este error representa la suma de las distancias medias a cada cluster, por ello, cuanto menor sea el error, menor serán estas distancias, y, como consecuencia, las instancias estarán mejor agrupadas.

A continuación se muestra en la Figura 4 la gráfica comparativa de los errores de los diferentes modelos para poder visualizarlos de una manera mas gráfica y visual.

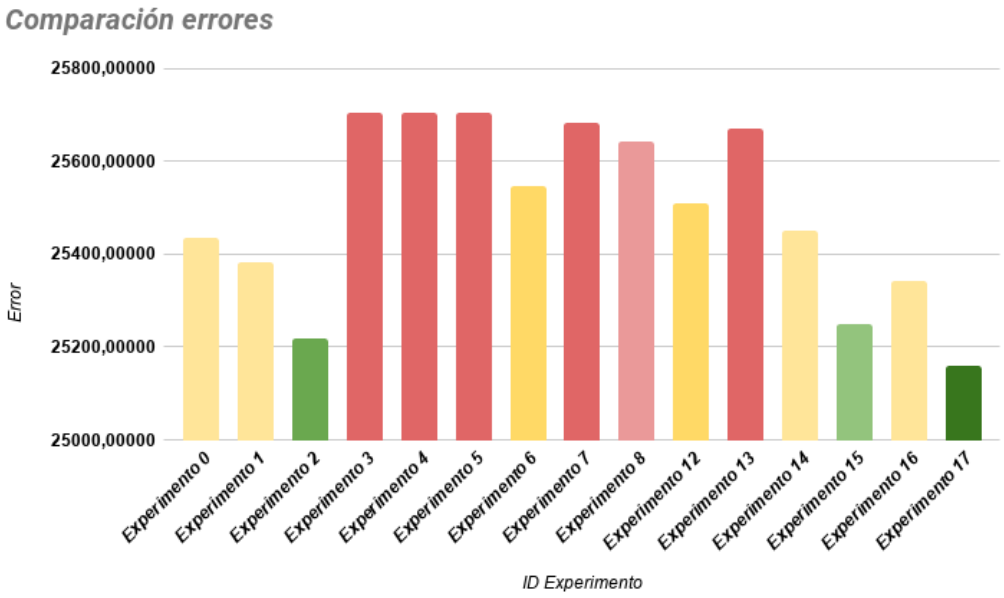

Figura 4: Comparación de los errores con *K-Medias*

Por lo tanto, se concluye con que el mejor resultado es el obtenido en el **Experimento 17**, así que en la siguiente sección se va a analizar en detalle este modelo.

3.2. Mejor modelo

El modelo del Experimento 17 se compone de **6 *clusters***, aunque según la gráfica de la Figura 5 parece que deja 2 *clusters* vacíos (debido a la escala del histograma), mientras que el *cluster* 4 es el que tiene mayor número de elementos, seguido del *cluster* 5. Por otro, lado los clusters 2 y 3 tienen muchos menos elementos.

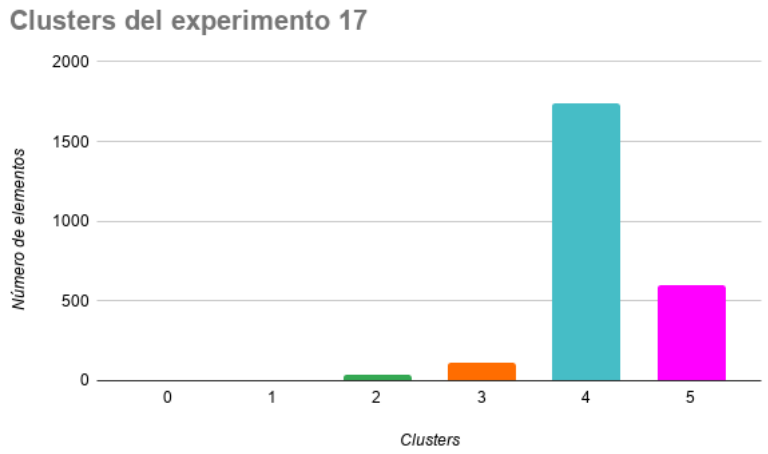

Figura 5: Clusters del Experimento 17

Ya que con la Figura 5 no se puede afirmar que los *clusters* 0 y 1 estén vacíos, se van a visualizar los datos del experimento en la siguiente tabla de la Figura 6.

Cluster	Número de elementos	Porcentaje de elementos
0	1	0
1	6	0
2	39	2
3	116	5
4	1737	69
5	601	24

Figura 6: Distribución de los datos en el Experimento 17

Con esta nueva información a la vista, se puede observar que los *clusters* 0 y 1 no están vacíos pero poseen muy pocos elementos, 1 y 6 respectivamente. Por lo tanto, se llega a la conclusión de que estos elementos son *outliers* y no aportan nada [1].

Las instancias, inicialmente, tenían ya una clase asignada (*excellent, fair, etc.*), las cuales se ignoraron para la agrupación. En la Figura 7 se muestra como han quedado repartidas en los diferentes *clusters* del modelo.

Las clases *excellent, very good, good, fair y poor*, están representadas por los colores azul oscuro, rosa anaranjado, rojo, gris azulado y azul claro, respectivamente.

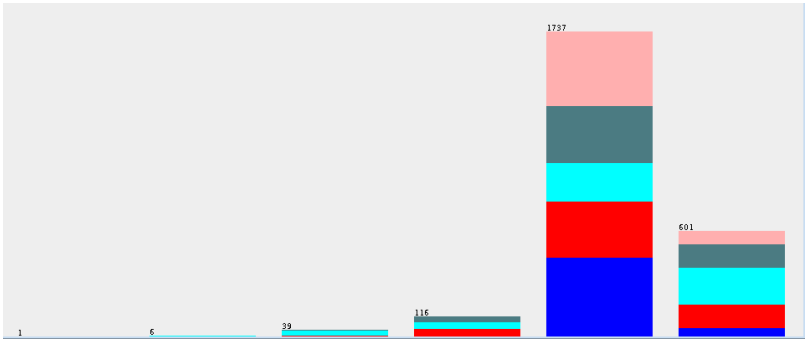

Figura 7: Distribución de las clases en los *clusters* del Experimento 17

En el **tercer *cluster*** (empezando por la izquierda) sólo hay instancias previamente clasificadas como *poor*, mientras que en el **cuarto** se encuentran instancias clasificadas con clases de calidad media-baja (desde *good* hasta *poor*). Después, los ***clusters* 5 y 6** contienen instancias de todo tipo, aunque se aprecia una concentración en el quinto de instancias con clases de calidad alta (*excellent, very good*), así como de instancias con clases de calidad media-baja en el sexto.

Una explicación a esta mezcla de clases (aunque es pura especulación), podría ser que a pesar de que las *reviews* estén clasificadas con una puntuación concreta, el vocabulario utilizado sería semejante en *reviews* con diferentes puntuaciones.

Por ello, las *reviews* del *cluster* 3 tendrían un vocabulario mucho más negativo. Luego, los *clusters* 4 y 6 contendrían *reviews* con un vocabulario mas neutro, siendo más negativas las del 4. Finalmente, las del *cluster* 5 la ocuparían *reviews* con un vocabulario más positivo, aunque incluyendo puntuaciones negativas.

Tras este análisis, se ejecutan varios algoritmos de **clasificación** con este modelo, de manera que se pueda entender cómo se ha realizado el agrupamiento. Los algoritmos probados, con su precisión a la hora de explicar el modelo, son los siguientes:

- *PART: 98.88 %*
- *J48: 97.36 %*
- *RandomForest: 100 %*
- *JRip: 90.28 %*

A pesar de que *RandomForest* sea capaz de explicar de manera perfecta el modelo, no es posible saber cómo lo hace, por lo que no es útil para esta tarea. Si pasamos al siguiente mejor, *PART*, recordar que es un algoritmo de reglas, y aunque sí que se puede ver cómo explica el modelo con reglas, resulta poco intuitivo visualmente.

Por todo esto, y porque el error al explicar es asumible, se ha elegido ***J48***, pues el árbol de decisión que muestra es visualizable y más intuitivo. En la Figura 8 se puede apreciar un camino del árbol cuya clasificación es el quinto *cluster*.

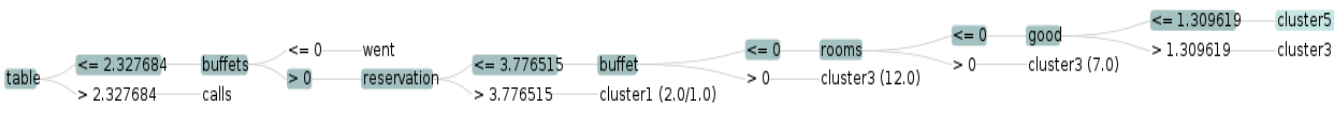

Figura 8: Árbol de decisión representado con prefuseTree [2]

Para cada instancia, el árbol empieza evaluando el valor del atributo *table*, y según sea este, se continúa por una rama distinta, *calls* o *buffets*. Según se va avanzando por el árbol se van

sucediendo este tipo de comparaciones, hasta que se llega a una hoja del árbol, donde muestra a qué *cluster* pertenece la instancia evaluada.

4. Contexto de la práctica

La minería de datos, y más en concreto, la *Minería de Texto* es un área que está sufriendo una gran revolución, gracias a las redes neuronales, similar a la que sufrieron las redes convolucionales desde la aparición de las arquitecturas de *Deep Learning*. Gracias a los avances en técnicas de **NLP** se han conseguido hazañas como la arquitectura de **GPT-3** [3], que permite llevar a otro nivel las tareas que es capaz de resolver la minería de texto, como puede ser *Búsqueda Semántica*, *Generación de Código desde lenguaje natural* o *Resumen de textos entendiendo el contenido* entre otros. Todos estos proyectos son accesibles desde la API de *OpenAI* [4].

Un caso particularmente llamativo es el de **AI Dungeon**, que en sus inicios usaba la arquitectura *GPT-2*, y que se ha actualizado a *GPT-3*. Su funcionamiento consiste en generar historias al estilo *Dragones y Mazmorras*, donde el usuario puede introducir el texto que quiera para interactuar con la historia que le están contando, y esta se adecuará a las reacciones. El juego es capaz de generar contenido potencialemnte infinito, y permitir que los jugadores creen sus propias aventuras.

Dentro del contexto de la minería de opinión, las críticas de huéspedes se están convirtiendo en un factor importante que afecta a las personas a la hora de hacer sus reservas en hoteles, lo que afecta de forma muy directa a la industria hotelera.

Esto ha hecho que se generen nuevas herramientas para obtener información relevante sobre las reseñas de una manera rápida y sencilla. Un ejemplo de esto es [5], donde se utiliza la herramienta de *web scraping* **Octoparse** [6], que es un *web scraper* diseñado para personas sin experiencia previa en data mining. Con esta herramienta se extraen y se preparan los datos para su posterior análisis.

Este análisis tiene el objetivo de descubrir el sentimiento subyacente de cada reseña mediante la utilización de la librería de *Python NLTK* [7], con la cual se asigna a la reseña un valor en cuanto a su negatividad, neutralidad y positividad, con lo que se puede concluir si la reseña es o no favorable.

5. Conclusiones

A lo largo del desarrollo de esta práctica se ha profundizado en la técnica de *Minería de Texto*, la cual presenta un enfoque completamente diferenciado de otras técnicas de Inteligencia Artificial, ya que permite resolver problemas para los cuales no existe una alternativa eficiente a la par que viable.

Por todo ello, esta práctica resulta especialmente útil a la hora de adentrase en este área. Sin embargo, tambien acarrea unos retos que deben ser solventados antes de poder comprobar las utilidades reales de esta técnica.

Uno de los principales inconvenientes viene de la mano de la escasa planificación que se puede llevar a cabo debido al nivel altamente experimental del *text mining*, resultando así una opción poco viable como primer acercamiento a técnicas de IA si se quiere evitar dar palos de ciego antes de conseguir un buen resultado.

Otro obstáculo a superar consiste en el procesado de datos requerido para posibilitar la aplicación de esta técnica, habitualmente es necesario disponer de los datos a usar perfectamente estructurados para posibilitar su tratamiento. Además, estos datos deben ser muy extensos para asegurar el buen funcionamiento del algoritmo.

Pese a todo, el *text mining* resulta una técnica especialmente útil en contraste a otras consideradas como de caja negra, puesto que resulta fácil extraer justificaciones y explicaciones sobre las decisiones tomadas.

En resumen, esta práctica demuestra cómo es posible abordar problemas para los cuales hasta ahora no se disponía de herramientas eficaces, planteando un enfoque completamente nuevo y eficaz a la hora de tratar con las colosales cantidades de texto e información generadas actualmente.

Referencias

[1] Muammad Ryan. How to cluster and detect outlier at the same time, 2018.

[2] Peter Reutemann. prefusetree: A visualization component for displaying trees that uses the prefuse visualization toolkit.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

[4] OpenAI Team. Openai technology, just an https call away.

[5] Ashley Ng. Sentiment analysis for hotel reviews, 2019.

[6] Octoparse Team. Octoparse.

[7] NLTK. nltk.sentiment package.