UNIVERSIDAD CARLOS III

INTELIGENCIA ARTIFICIAL EN LAS ORGANIZACIONES

2023-2024 GRADO EN INGENIERÍA INFORMÁTICA

DATA MINING

GRUPO 81

Diego Caballero García-Alcaide NIA:100451177

100451177@alumnos.uc3m.es

Diego Calvo Engelmo NIA:100451091

100451091@alumnos.uc3m.es

Álvaro Bernal Torregrosa NIA:100451179

100451179@alumnos.uc3m.es

Raúl Ágreda García NIA:100451269

100451269@alumnos.uc3m.es

ÍNDICE

PARTE 1: CLASIFICADOR DE OPINIONES

1. Preprocesamiento de datos

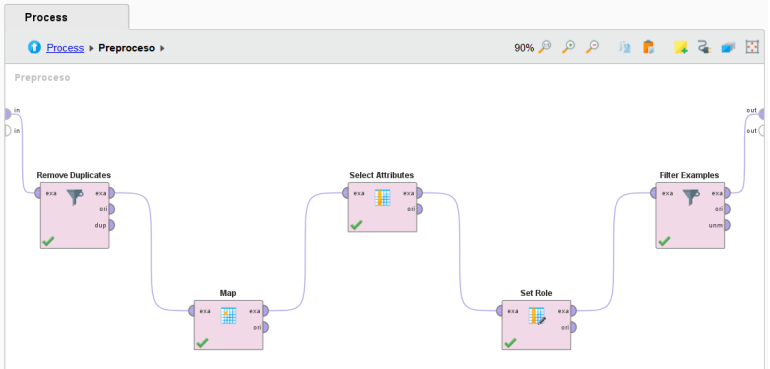
Para empezar con el preprocesamiento de datos, antes de introducirlos en RapidMiner hicimos una visualización de estos a través de Excel. En ella, nos dimos cuenta que desde la fila 4148 hasta la 4945 encontrábamos missing values en las dos últimas columnas de las filas impares y en la fila siguiente encontrábamos dichos valores faltantes de la fila anterior. Nos dimos cuenta que esto era debido a un mal formato del fichero csv, que provocaba que se partieran las líneas. Para eliminar esto, hemos realizado el siguiente script que elimina el salto de línea erróneo.

```python
lines = []
with open('./data/Balanced_AHR.csv', 'r') as f:
  # Read lines
  lines = f.readlines()
  for i in range(len(lines)):
    # If the line starts with ",
    if lines[i].startswith('",'):
      # Remove \n from the previous line
      lines[i-1] = lines[i-1].rstrip('\n')

# Write the lines back to the file
with open('./data/Balanced_AHR.csv', 'w') as f:
  f.writelines(lines)
```

A la hora de cargar los datos utilizando el componente 'Read CSV' debemos cambiar la codificación por defecto del componente. En concreto, cambiando la codificación a UTF-8, conseguimos recuperar las vocales con tilde y con 'ñ' y así recuperamos correctamente la información para su posterior procesado. Por otro lado, tras explorar los datos en la vista nos dimos cuenta de que los campos de texto los establecía con tipo '*polynominal*', lo cual provocará problemas más adelante. Por ello, para cambiarlo debemos acceder a los parámetros avanzados a la opción '*data set meta data information*' y establecer el tipo de los atributos a tipo '*text*'. Además, el atributo 'label' le estableceremos el tipo a 'nominal', ya que sin este, dará problemas al realizar la validación cruzada.

Tras esto hemos creado un componente de proceso, el cual nos ha permitido agrupar todos los componentes encargados del preprocesado de datos. En este preproceso hemos incluido los



siguientes componentes explicados a continuación:

- ***Remove duplicates***: Nos hemos dado cuenta que hay datos que tienen información repetida salvo el id de la reseña. Esto no es bueno, ya que a la hora de procesar los datos se asignará el doble de pesos al mismo patrón. Por ello, hemos usado este componente para eliminar los datos repetidos, que serán aquellos que tengan los valores de todos los atributos iguales salvo los del atributo de la id. Además, marcando la casilla '*thread missing values as duplicates*' conseguimos que cuando dos reseñas sean iguales y tengan missing values se elimine la duplicada.
- ***Map***: Tras realizar el análisis de los datos, observamos que había reseñas que tenían un valor de label de 3 y no había ninguna con valor de 2. Esto es un error, ya que el rango de valores definidos para este atributo es entre 0 y 2. Por tanto, utilizamos este componente para reemplazar los valores de 3 con los valores de 2, obteniendo ya el rango de label correcto.
- ***Select attributes***: De todos los atributos del dataset de entrada, solo nos interesan dos, la '*review_text*' con el propio contenido de la reseña y la valoración del atributo '*label*', que representa la valoración de dicha reseña. Por tanto, usamos este componente para filtrar los atributos y quedarnos con esos dos únicamente.
- ***Set Role***: Para los pasos anteriores del preproceso, hemos tenido que considerar el atributo '*label*' como un atributo y no como la columna de clasificación. Por ello, con este componente establecemos que este atributo tenga el rol de '*label*' y pueda ser interpretado así por los componentes posteriores.
- ***Filter examples***: Explorando los datos obtenidos hasta este momento, nos hemos dado cuenta que había una reseña que no tenía label. Dado que es una cantidad insignificante en comparación con el tamaño del dataset, hemos utilizado este componente estableciendo como clase de condición '*no_missing_labels*' para eliminar esta entrada.

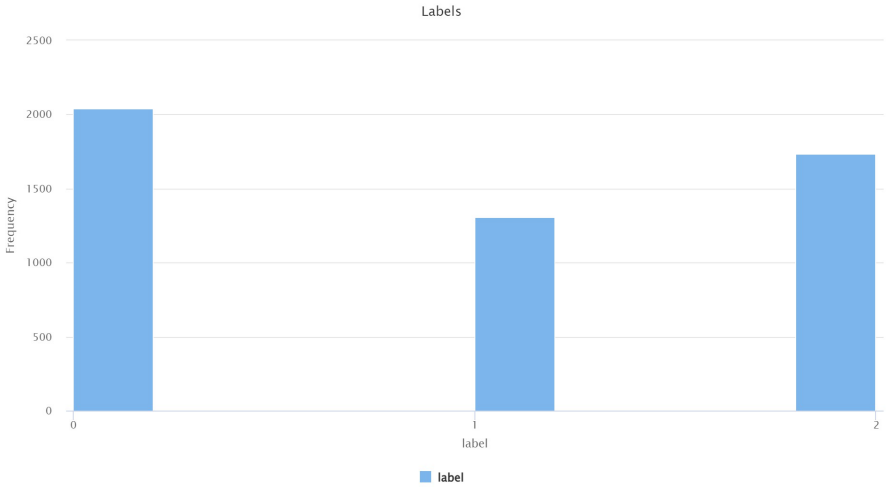
Con esto, terminamos con el componente de proceso y con toda la parte de preprocesado de los datos, dejando los datos listos para el procesamiento de los mismos.

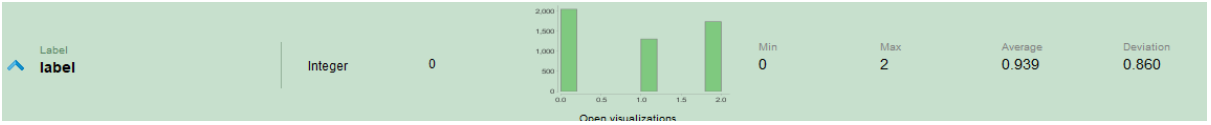
2. Análisis descriptivo de los datos

Teniendo los datos en RapidMiner es necesario hacer un análisis de los datos que tenemos, para ello nos enfocaremos en la distribución de las clases y las palabras más frecuentes.

2.1. Distribución de las clases

Para analizar los datos hemos comprobado si el conjunto está balanceado, es decir, los datos de reseñas positivas, neutras y negativas tienen la misma proporción. Para ello utilizaremos la herramienta de "Visualizations" en el apartado de "Results". Tras observar los resultados, obtenemos que el número total de reseñas que encontramos en nuestro dataset es de 5083 reseñas, estando distribuidas en 1735 reseñas neutras (label: 2), 1305 positivas (label: 1) y 2043 negativas (label: 0).





En el histograma anterior podemos ver dicha distribución, donde apreciamos que el número de reseñas negativas es el grupo más grande. No obstante, en las estadísticas generales podemos ver que la media se sitúa en un 0.939 y que no hay gran variación entre las cantidades de datos en cada subconjunto, lo que nos da a entender que el modelo está bastante equilibrado.

2.1.2. Palabras más frecuentes

Para ver las palabras que más se repetían primero hemos procesado los datos, y posteriormente dividido en palabras para ver sus frecuencias y pesos.

Palabra	Contador
general	2811
limpia	2107
bastante	1524
paredes	1499
días	1438
buffet	1356
recomiendo	1138
piscina	1083
muchas	908
creo	895
tipo	871
toallas	870
personal	801
atento	792
vez	781
instalaciones	761
edificio	692
justo	635
casa	628
llegar	616
reserva	590
pues	583
agradable	566
aire	559
nadie	554
dos	547

Como podemos ver en la tabla, las palabras que más se repiten son “general”, “limpia”, “bastante” y “paredes”. Probablemente, nuestro modelo dará menos peso a estas palabras al repetirse mucho, algo que para algunas de ellas nos beneficiaría. Así pues, como “bastante”, que en un primer momento podríamos pensar que puede ser importante, se repite con cierta frecuencia, suponemos que en nuestro modelo final no tendrá mucha relevancia.

Cabe destacar que en esta lista no aparecen preposiciones o conjunciones porque hemos aplicado un diccionario de stopwords con anterioridad; para eliminar este tipo de palabras que no aportan nada al modelo.

3. Limpieza de datos

Para este paso, no ha hecho falta utilizar un diccionario de reemplazo de símbolos, ya que como indicamos anteriormente, seleccionando UTF-8 en el apartado "File encoding" del componente 'Read CSV'. De esta forma, nos aseguramos que todos los caracteres del español como las tildes o la letra "ñ" se interpreten correctamente en los pasos posteriores.

4. Generación de la matriz de términos por documento

Para el análisis de las distintas palabras a partir de las cadenas que forman el cuerpo de las reviews hemos usado el operador "Process documents form data" con el que RapidMiner convierte los datos que tiene del texto proporcionado en un formato que posteriormente va a poder analizar.

Dentro de este operador hemos considerado usar un filtro de stop words con el que obviamos palabras que no añaden ningún significado ni importancia a las frases lo que nos permite obviarlas haciendo más rápido nuestro modelo e incluso mejorándolo.

El diccionario de palabras que hemos usado se encuentra en este link, que contiene una gran cantidad de artículos, preposiciones, pronombres o palabras vacías.

Hemos decidido incluir el operador de stemming, ya que este mejora a rasgos generales los resultados del modelo utilizado, en torno a un 3% más de accuracy.

PerformanceVector:
accuracy: 71.97% +/- 2.25% (micro average: 71.97%)
ConfusionMatrix:

True:	0	2	1
0:	1561	473	54
2:	423	1079	233
1:	59	183	1018

Sin stemming

PerformanceVector:
accuracy: 74.68% +/- 1.59% (micro average: 74.68%)
ConfusionMatrix:

True:	0	2	1
0:	1600	449	56
2:	420	1174	227
1:	23	112	1022

Con stemming

Como vector creation, el parámetro que mejores resultados daba respecto al accuracy fue el "TF", aunque todas las opciones daban resultados muy parecidos, y el elegido no distaba demasiado como mejor parámetro, al final term frequency (TF) y term occurrences, para los datos que estamos manejando no son muy diferentes. Tampoco son muy diferentes del resultado que da con TF-IDF, ya que más o menos los usuarios expresan las opiniones con palabras parecidas.

Por otro lado, en el parámetro de "prune method" hemos elegido coger el rango del 1% al 80%, ya que, las palabras que aparecen muy pocas veces son irrelevantes y como hemos visto antes, palabras con muchas repeticiones como hotel o habitación solo empeoran nuestro modelo.

También hemos probado a añadir a la lista de stop-words palabras muy utilizadas en el dataset pero que consideramos que no aportan significado, como por ejemplo "hotel"; sin embargo, como utilizamos prune above de 80% estas palabras son ignoradas igualmente por el propio algoritmo aunque no las añadamos a la lista de stop words.

Finalmente, tras hacer varias pruebas concluimos que sin el filtro de stop words, el modelo da mejores resultados, en torno a un 5% más de accuracy, por lo que decidimos quitarlo.

Al utilizar el filtro de stopwords observamos que tanto la clase 0 como la clase 2 son predichas de forma parecida; sin embargo la diferencia ocurre en la clase 1 donde sin el filtro de stopwords es capaz de predecir muchas reseñas correctamente mientras que con el filtro de stopwords muchas de las reseñas con etiqueta 1 son mal clasificadas como clase 2 por el modelo.

Esto significa, que con el filtro de stop words, el modelo clasifica muchas de las reseñas positivas como neutras, como podemos ver en los siguientes resultados.

```
PerformanceVector:
accuracy: 75.05% +/- 1.89% (micro average: 75.05%)
ConfusionMatrix:
True:   0       2       1
0:      1597    427     48
2:      423     1197    236
1:      23      111     1021
```

Sin filtro de stopwords

```
PerformanceVector:
accuracy: 70.89% +/- 3.62% (micro average: 70.88%)
ConfusionMatrix:
True:   0       2       1
0:      1597    473     79
2:      412     1148    368
1:      34      114     858
```

Con filtro de stopwords

Por otro lado, probamos a utilizar, en lugar del atributo 'label', utilizar 'rating' como rol. Sin embargo, al utilizar este último atributo, los valores en la matriz de confusión empeoran bastante, el accuracy baja sobre un 10%, probablemente sea debido a la calidad de los datos.

5. Evaluación de los clasificadores

Para esta práctica hemos elegido el modelo de red neuronal convolucional. Tras varias pruebas observamos que aumentar el número de capas o el número de epochs no beneficia al rendimiento del modelo; sin embargo, reducir el número de nodos o neuronas por capa si lo hace.

Por tanto, ajustamos los parámetros de las redes de acuerdo a la configuración que mejor resultado obtuvo, y utilizamos esta arquitectura para todos los modelos probados a continuación. Esta arquitectura está formada por:

- 10 epochs
- 2 capas ocultas con 10 neuronas cada una

Las pruebas mostradas a continuación son algunos de los resultados durante la realización de la práctica, y hemos priorizado medir distintos modelos con diferentes parámetros de procesado de texto.

Nombre	Prune above	Stop Words	Stemming	Vector creation	n-grams	Accuracy
DL-01	100	Sí	Sí	Term Ocurrences	Sí	**74.92%**
DL-02	80	Sí	Sí	Term Ocurrences	Sí	**75.51%**
DL-03	80	Sí	Sí	Term Ocurrences	No	**75.82%**
DL-04	80	No	Sí	Term Ocurrences	No	**75.53%**
DL-05	80	No	No	Term Ocurrences	No	**73.30%**
DL-06	80	No	Sí	TF-IDF	No	**76.16%**
DL-07	80	No	Sí	TF	No	**76.24%**

A partir de los resultados podemos llegar a las siguientes conclusiones:

- Hay un cierto grado de aleatoriedad por lo que los resultados pueden variar en cada ejecución.
- Disminuir la cota superior de **poda**, en este caso de 100% a 80%, mejora la clasificación, ya que se eliminan palabras que al ser tan comunes no tienen demasiado significado.
- Incluir **bigramas** no mejora ni empeora la clasificación, se debe a que selecciona muy pocos bigramas y varios de ellos no aportan mucho significado a la hora de predecir si la nota del hotel será positiva o negativa . Algunos ejemplos de los bigramas generados son: “hotel_bien”, “personal_amable”, “fin_semana”, “calidad_precio”, “bien_situado” y “aire_acondicionado”.
- El uso o no del **filtro de stop-words** no refleja prácticamente cambio en el resultado, pero durante las pruebas hemos detectado que, en general, mejora sin el filtro, probablemente en el filtro haya palabras que se estén eliminando pero que sí aporten información, como puede ser bueno o buena.
- El uso de **stemming** mejora en gran medida el rendimiento del clasificador, ya que varias palabras como por ejemplo, habitaciones y habitación, se juntarían como ‘habita’, y puede ayudar a la hora de tomar muestras.
- Por último el valor de **Vector Creation** que mejor funciona es el Term Frequency (TF), al final el TF es un valor relativo al resto de datos que aporta más información, en este caso, que el número de ocurrencias absoluto

A continuación mostramos la matriz de confusión obtenida a partir del mejor modelo obtenido.

accuracy: 76.24% +/- 1.86% (micro average: 76.23%)

	true 0	true 2	true 1	class precision
pred. 0	1609	411	36	78.26%
pred. 2	408	1206	209	66.15%
pred. 1	26	118	1060	88.04%
class recall	78.76%	69.51%	81.23%	

Como podemos observar tanto la clase 0 (reseña negativa) como la clase 1 (reseña positiva) tienen bastante buenos valores de precisión y recall. Esto indica que el modelo ha aprendido a predecir significativamente bien las reseñas positivas y las negativas.

Sin embargo, la clase 2 (reseñas neutras) tiene peores resultados de estas métricas, lo cual indica que este tipo de reseñas son peor clasificadas por el modelo.
Esto es lógico, ya que las reseñas neutras son las más **ambiguas**, ya que no contienen palabras que identifiquen claramente si la opinión del hotel es positiva o negativa, y por tanto al modelo le cuesta más saber de qué tipo de reseña se trata.

De hecho, esto también ocurre en los humanos. Si a un humano le enseñas reseñas positivas, neutras y negativas, donde más se equivocará es las neutras puesto que son las más difíciles de clasificar, ya que hay ciertas reseñas que dicen aspectos positivos o aspectos negativos del hotel pero luego aportan una nota neutra, lo cual puede llevar a confusión, puesto que las opiniones humanas no son una ciencia exacta.

Además, por lo que podemos observar en la matriz, el principal error del modelo es que clasifica como reseñas negativas bastantes de las neutras, y clasifica como reseñas neutras bastantes de las negativas.
Que nuestro modelo elija la clase negativa cuando no sabe con exactitud si es una reseña neutra, tiene su explicación. Esto se debe principalmente a que la clase mayoritaria de nuestros datos es la clase 0 (reseñas negativas), y por tanto, el modelo ha aprendido a decantarse más por esta clase cuando se encuentra con una reseña neutra que no sabe clasificar con exactitud.

Por último, concluir que los resultados obtenidos son bastante buenos; sin embargo, consideramos que con un conjunto de datos más grande, se conseguirían mucho mejores resultados, ya que como hemos podido ver en la clase negativa (clase 0), con pocos más datos que las otras clases se obtenía bastante mejores resultados.

PARTE 2: CLUSTERING

1. Preprocesamiento de datos

Para esta parte hemos realizado un preprocesado de los datos diferente a la primera, pero utilizando el csv resultante de aplicar el script de python comentado en la parte 1.

Para ello hemos aplicado los siguientes operadores:

- ***Remove duplicates***: volvemos a eliminar los duplicados para que no interfieran en los resultados de los modelos, dando más importancia a las palabras de estas reseñas duplicadas.
- ***Select attributes***: en este caso solo nos quedamos con el atributo 'review text' puesto que el objetivo es extraer temas agrupando las palabras de las reseñas, por tanto no es necesario añadir la clase (label) porque la valoración del hotel no tiene importancia para este propósito.

En consecuencia, al no utilizar el atributo de salida 'label' no es necesario utilizar los operadores ***Map***, ***Set Role*** y ***Filter Examples*** como hicimos en la primera parte.

2. Procesado del texto

Para el procesado del texto de las descripciones de las reseñas volvemos a aplicar los operadores utilizados en la primera parte pero con las siguientes diferencias.

- ***Filter stopwords***: en primer lugar empezamos sin este operador pero rápidamente nos dimos cuenta observando los resultados que palabras como 'hotel', 'habitaciones', 'habitación' aparecían en la mayoría de los temas, siendo además de las palabras con más importancia debido al gran número de veces que aparecen en los documentos. Por tanto, decidimos añadir estas palabras a la lista de stopwords y aplicar el filtrado de las mismas.
- ***Stemming***: en este caso **no** utilizamos el operador Stem puesto que, de nuevo, como el objetivo es realizar el agrupamiento de temas, utilizar las palabras reducidas a su raíz dificulta la interpretación de los temas obtenidos. Por tanto, a diferencia de la parte 1, decidimos no realizar stemming en el procesado del texto.

El resto de operadores como son ***Tokenize*** para dividir las descripciones de las reseñas en términos, ***Transform Cases*** para convertir todo el texto en minúsculas y ***Filter Tokens*** para filtrar términos de menos de 2 caracteres de longitud, si que son utilizados al igual que en la parte 1 para la creación de la matriz término-documento.

3. Configuraciones del LDA

Para la construcción del mejor modelo hemos probado numerosas configuraciones del operador LDA, variando el número de topics/temas, número de iteraciones y número de palabras por topic.
Tras ello, hemos sacado las siguientes conclusiones:

- Al aumentar el número de topics, se formaban algunos temas más específicos que con un número de topics más bajo, pero esto también ocasiona en otros casos que se diera importancia a términos que no deberían formar parte del tema en el que aparecían.

- Con pocos topics (menos de 5) no se obtenían temas diferenciados. Muchas palabras aparecían en varios temas a la vez, y no se veía una clara temática en algunos de los cluster/agrupamientos.

- Al aumentar el número de iteraciones (1000, 2000, 3000) se obtenían mejores resultados en cuanto a coherencia de los temas y relación de las palabras obtenidas en cada uno, pero implica un aumento considerable del tiempo de cómputo para el entrenamiento del modelo y obtención de los temas.

- Utilizar más palabras por tema favorece la interpretación de estos, permitiendo identificar con mayor facilidad el sentido y temática de cada cluster. Sin embargo, utilizar pocas palabras resulta más complicado diferenciar de qué trata el tema debido a que las reseñas son un dominio el cual utiliza términos muy distintos, a diferencia del ejemplo visto en teoría sobre la clusterización de competencias que debía tener un puesto de Industria 4.0, las cuales eran términos muy concretos y conceptos definidos con muy pocas palabras.
  Por ello, hemos decidido utilizar 15 palabras por tema. Pero cabe mencionar que al obtener las palabras ordenadas por importancia siempre teníamos más en cuenta las primeras palabras de cada tema.

4. Métricas de evaluación de los modelos

Hemos investigado que hay diferentes tipos de métricas para la evaluación de modelos de clustering. En concreto, el operador LDA proporciona varias de ellas, de las cuales nos hemos fijado en las siguientes:

- **Perplexity**: complejidad o dificultad para interpretar los resultados del algoritmo de clustering. Cuanto menor es esta medida, mejor es el modelo.

- **Avg coherence**: coherencia de los temas obtenidos por el algoritmo. También mide la interpretabilidad de los topics. Cuanto mayor es esta medida, mejor es el modelo.

Sin embargo, mencionar que no solo nos hemos guiado por las métricas sino que hemos analizado uno a uno los temas obtenidos y las palabras que lo forman para poder interpretar qué tema pueden tratar cada uno de ellos y la relación y coherencia entre sus palabras.

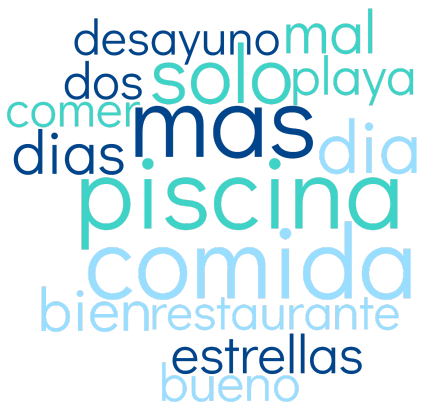
5. Modelo final y temas obtenidos

El modelo que mejor resultado ha obtenido se caracteriza por:

Topics	Iteraciones	Palabras por topic
6	3000	15

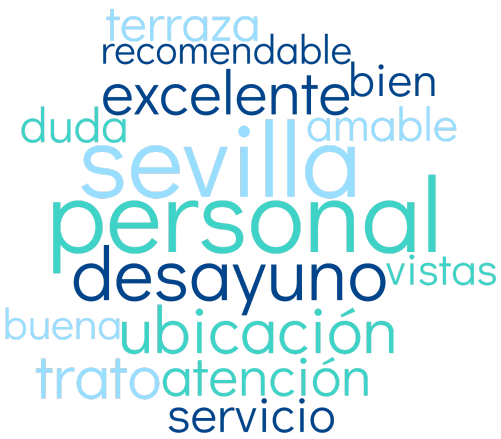
Para visualizar los resultados de cada topic/tema, vamos a mostrar la importancia de las palabras dentro de cada tema mediante word clouds, donde las palabras con un mayor peso aparecen más grandes y las de menor peso o importancia con menor tamaño.

Tema 0:

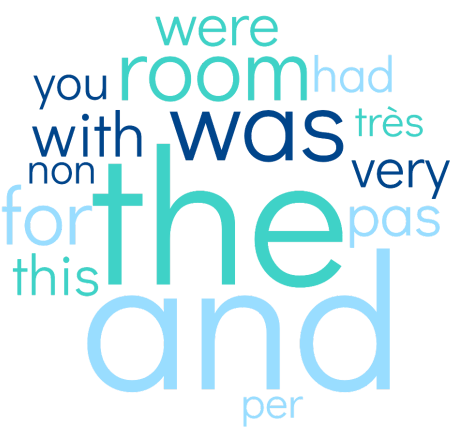


Tema 1:

Tema 2:

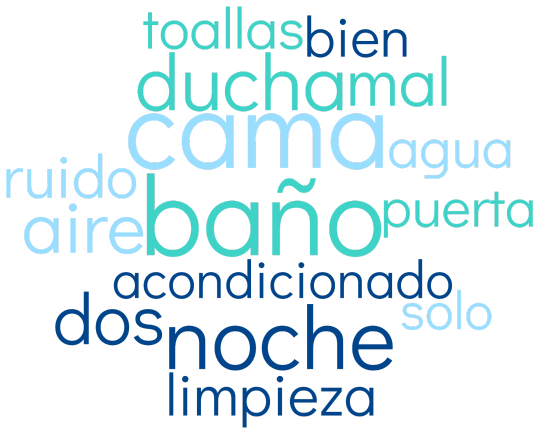


Tema 3:



Tema 4:

Tema 5:



A continuación vamos a analizar cada uno de estos temas para entender cómo el modelo a creado dichos temas:

- **Tema 0**: Podemos entender que en este tema se han agrupado los términos relacionados con la comida principalmente; con términos con bastante peso como "comida", "restaurante", "comer" o "desayuno".
- **Tema 1**: En este tema, vemos como los términos tienen que ver con el momento previo a la llegada al hotel, con términos como "recepción", "mañana", "reserva" o "nadie"
- **Tema 2**: Este segundo tema se refiere a características del personal del hotel y la ubicación del mismo, siendo los términos con más peso "personal", "desayuno", o "sevilla". Además, podemos intuir que las reseñas con respecto a estos aspectos eran positivas ya que encontramos términos positivos como "excelente", "recomendable", "amable" o "bien" dentro de los más comunes.
- **Tema 3**: Cabe destacar las características de este tema, en el cual se han agrupado las palabras que no pertenecen al castellano. Las más comunes son en inglés, aunque también encontramos alguna palabra en otro idioma como "très", "pas" (francés).
- **Tema 4**: En este cuarto tema vemos como se muestran palabras relacionadas con los servicios del hotel, encontrando claros ejemplos como "desayuno", "personal", "parking" o "zona". Además, podemos deducir por el alto peso de la palabra "bien", que la mayoría de las reseñas que hablan sobre estos aspectos
- **Tema 5**: En este último tema, encontramos términos que hacen referencia a características de la habitación del hotel. En concreto, las palabras más repetidas en este tema son "baño", "cama", "ducha" o "noche"; conceptos claramente representativos de este tema.

Tras observar los temas anteriores, podemos llegar a la conclusión de que aunque los temas no son súper específicos y cerrados, podemos ver como para cada uno se engloban ciertos ámbitos de la experiencia de residir en un hotel, diferentes entre sí. Por tanto, podemos afirmar que se ha hecho una correcta clasificación de los temas de las reseñas.

CONTEXTO

IA utilizada por Amazon (Garcia, 2023)

Realizando una investigación acerca del uso de inteligencias artificiales para el procesado e interpretación de reseñas, hemos encontrado un artículo muy interesante referente a la empresa Amazon. En él, se especifica como Amazon aprovecha el uso de la inteligencia artificial generativa para procesar el conjunto de reseñas de un producto y ofrecer a los usuarios un resumen con las opiniones más destacadas.

Con este método, Amazon pretende aprovechar el valor que aportan las reseñas y reducir el tiempo de consulta a los usuarios cuando quieren comprobar la calidad de los productos; pudiendo dar una vista precisa de los aspectos claves de los mismos.

Por otro lado, en relación con las reseñas nos encontramos con el problema de las reseñas falsas. Este tipo de reseñas, generalmente, son utilizadas por atacantes o personas con malas intenciones con el objetivo de perjudicar la imagen de un producto o servicio. En este ámbito, podemos entender que la inteligencia artificial juega un papel clave, pudiendo detectar características sospechosas en el uso del lenguaje y la similitud de contraseñas para determinar si estas son fraudulentas o no.

Sin embargo, en la actualidad los esfuerzos de Amazon por utilizar un método de IA para este propósito no han sido del todo satisfactorios; y aunque se han conseguido identificar ciertas contraseñas fraudulentas, lo cierto es que un grán número de reseñas no son detectadas por estos algoritmos.

Potencial del Web Mining para las empresas

En esta investigación se muestra el gran potencial que tiene el web mining para el análisis predictivo empresarial con el que se extraer información tal como el pensamiento de los clientes hacia ciertos productos o servicios, las tendencias del mercado o analizar a los competidores.

Para ese caso en concreto el artículo nos dice que se dependiendo del caso específico se debería usar uno u otro modelo, dependerá del problema empresarial que queramos abarcar y de los datos disponibles, aunque sí que se recomiendan usar modelos tales como los árboles de decisión, máquinas de vectores de soporte o redes neuronales.

Gracias a este tipo de tecnologías las empresas pueden ganar un conocimiento más profundo del mercado de la gran cantidad de datos que podemos encontrar en internet. Con técnicas avanzadas las empresas pueden utilizar técnicas avanzadas para mejorar sus planes de decisiones para ganar una ventaja competitiva.

BIBLIOGRAFÍA

Garcia, F. (2023, June 20). *Amazon utiliza Inteligencia Artificial para resumir las reseñas*. Brainy Commerce. Retrieved October 26, 2023, from https://www.brainycommerce.com/es/info/amazon-inteligencia-artificial-resumen-resenas/

Khan, S. (2023, September 28). *(PDF) THE POTENTIAL OF WEB MINING IN BUSINESS PREDICTIVE ANALYSIS FROM RAW DATA TO ACTIONABLE PREDICTIONS*. ResearchGate. Retrieved October 26, 2023, from https://www.researchgate.net/publication/374193252_THE_POTENTIAL_OF_WEB_MINING_IN_BUSINESS_PREDICTIVE_ANALYSIS_FROM_RAW_DATA_TO_ACTIONABLE_PREDICTIONS