E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

Aprendizaje multi-instancia aplicado al análisis de sentimientos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor: Ian Recke Campos

Director: Mikel Sesma Sara

Pamplona, junio de 2022

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

RESUMEN:

En este trabajo fin de grado nos proponemos abordar un problema de clasificación multi-instancia de textos. En concreto, partiendo de un conjunto de reseñas etiquetadas como positivas o negativas, el objetivo de este trabajo es separar la clasificación grupal de una reseña a una clasificación de cada una de las frases de la reseña. Conociendo qué partes de la reseña son positivos o negativos se puede realizar un análisis más detallado de la predicción de sentimientos en documentos, evitando clasificar toda una reseña con una etiqueta. Al no disponer de una etiqueta para cada frase, el problema no se puede resolver con algoritmos de aprendizaje supervisado usuales. Por lo tanto, para llevar a cabo este objetivo, se utilizarán técnicas de aprendizaje multi-instancia junto con técnicas de procesamiento de lenguaje natural y minería de texto.

PALABRAS CLAVE:

1. MIL (Multiple-instance Learning)
2. BOW (Bag of Words)
3. NLP
4. Clasificación de textos
5. OWA (Ordered Weighted Average)

ÍNDICE:

Índice de tablas

Índice de figuras

Índice de ecuaciones

1. INTRODUCCION:

La clasificación de textos es un problema de Machine Learning, donde su dificultad proviene de que la información no es estructurada. Esto significa que los datos no tienen un formato u organización predefinidos, lo que hace que sean más difíciles de recopilar, procesar y analizar. Por ello, para poder realizar un tratamiento de los datos, es necesario un proceso de extracción de características previo.

Otra característica común de los problemas de clasificación de texto es que no siempre es fácil obtener etiquetas de las instancias individuales y únicamente disponemos de etiquetas de grupos de instancias [1].

Por ejemplo, en el caso de las reseñas de productos podemos obtener una clasificación positiva o negativa de la propia reseña, si existe una valoración numérica adicional (por ejemplo, con estrellas), pero no disponemos de una etiqueta para cada una de las frases que componen dicha reseña. Esto es un problema porque normalmente dentro de una reseña hay aspectos negativos y positivos, que, dependiendo de cómo se clasifique, no se tienen en cuenta. Por ejemplo, un usuario puede poner una puntuación y un comentario en relación con un producto adquirido en una plataforma, como por ejemplo Amazon. El usuario puede calificar un producto con 1 estrella, pero puede resaltar alguna cualidad del artículo en concreto. En definitiva, disponer de una clasificación positiva o negativa a nivel de frase puede ser muy beneficioso pues cada frase aporta información sobre un aspecto en concreto del producto.

Este problema es fácil de resolver para una persona, pero, el hecho de no tener etiquetas a nivel de frase hace que sea un problema difícil de abordar para que realizar la clasificación automática de cada frase partiendo de la reseña global.

En este trabajo, vamos a plantear este problema como un problema de clasificación multi-instancia, aplicado al análisis de sentimientos de reseñas para poder aprender a clasificar instancias partiendo de las etiquetas de grupos de instancias. Los clasificadores que entrenaremos obtendrán la información de las reseñas (los grupos de instancias), y estos podrán traspasar esa información aprendida a las frases (instancias individuales). Además, esto permitirá poder clasificar futuras instancias individuales.

Para abordar este problema, nos basamos en el algoritmo propuesto en [2], que se basa en minimizar una función de coste adaptada a la clasificación multi-instancia. Esta función tiene en cuenta la similitud entre las frases y, a su vez, tiene en cuenta las etiquetas de cada documento, de manera que penaliza aquellas instancias que sean muy similares y de grupos opuestos.

Por lo tanto, en este trabajo nos plantemos el siguiente objetivo:

- Implementar y entrenar un algoritmo de clasificación multi-instancia para inferir el sentimiento a nivel de frase partiendo de etiquetas grupales.

Para conseguir el objetivo principal, estos son nuestros objetivos específicos:

- Estudiar el algoritmo de clasificación multi-instancia propuesto en [2].
- Estudiar distintas formas de extraer características de texto.

- Implementar el algoritmo de MIL de [2] modificando la forma de construir los vectores de características.
- Experimentar con dicho algoritmo y analizar los resultados de aplicarlo con distintas funciones de agregación.

2. PRELIMINARES

Este trabajo trata de un problema de Machine Learning, más concretamente, un problema de Procesamiento de lenguaje natural (NLP, de sus siglas en inglés). Este es una rama de Machine Learning que permite a un ordenador entender, analizar, manipular e incluso generar lenguaje. El Machine Learning, a pesar de estar ligado a la Inteligencia Artificial no son el mismo término. La diferencia principal estriba en el desarrollo de las metodologías y los algoritmos.

2.1. Inteligencia Artificial, Machine Learning, Deep Learning y su origen:

La inteligencia artificial es la ciencia que se encarga de generar cualquier tipo de conducta humana en las máquinas, generalmente se intenta copiar el razonamiento humano. El objetivo fundamental de la inteligencia artificial es intentar imitar la capacidad de procesamiento del cerebro humano para conseguir crear máquinas inteligentes, máquinas capaces de desarrollar conductas humanas.

La primera mención del término de Inteligencia Artificial se encuentra fechado en 1940 [3] , aunque este no cobra fuerza hasta 1950 mediante la publicación de un trabajo del famoso matemático Alan Turing abriendo una nueva forma de pensar en el ámbito de las ciencias de la información. Alan Turing fue el creador de la famosa máquina de Turing, máquina que era capaz de comprender los primeros algoritmos. Como primer concepto antes de la creación de las máquinas digitales, se ideó la máquina que era capaz de leer instrucciones a través de la perforación de una cinta, e introdujo el concepto de problemas irresolubles (problemas que pueden llegar a ser tan complejos que su solución es imposible de hallar o simplemente no tienen solución). Hasta finales de los años 80, fue un modelo muy importante en el ámbito de la Inteligencia Artificial sirviendo como referencia para diferentes estudios. Hoy en día, se considera a Alan Turing como creador o padre de la Inteligencia Artificial, cuya máquina de Turing sigue siendo un modelo digno de destacar y estudiar.

El aprendizaje automático o Machine Learning es una subcategoría de la inteligencia artificial, que automatiza el proceso de creación de algoritmos que permiten resolver problemas de la vida real, permitiendo a las máquinas que se adapten a nuevas situaciones o diferentes eventos de manera independiente, sin necesidad de ser reprogramadas. La metodología cambia según el modelo que se emplea, pero de manera general, una máquina analiza datos, identifica los patrones presentes en los datos y luego usa los conocimientos aprendidos para completar mejor la tarea asignada. En la Figura 1, podemos ver una representación del proceso de aprendizaje.

Los algoritmos de Machine Learning sirven para modelar una gran cantidad de problemas. En la actualidad hay algoritmos que resultan ser muy efectivos por su robustez y facilidad de aprendizaje como por ejemplo arboles de decisión (por ejemplo, Random Forest), o porque se encuentran muy introducidos en otras ramas científicas como los algoritmos genéticos. A pesar de su alta aplicabilidad, hay problemas más complejos que son más difíciles de resolver mediante Machine Learning, como por ejemplo puede ser el reconocimiento de imágenes y su procesamiento. Esto motivó al desarrollo de soluciones para poder ampliar el abanico de la Inteligencia Artificial, originando técnicas de Deep Learning. El Deep Learning es una subcategoría del Machine Learning, cuya traducción al castellano es *Aprendizaje Profundo* y su base se sostiene sobre las redes neuronales.

2.2. Tipos de Aprendizaje

Los diferentes algoritmos del aprendizaje automáticos pueden funcionar con mayor o menor intervención humana. De esta manera, se pueden destacar los distintos modelos de aprendizaje.

2.2.1. Aprendizaje supervisado

En el entorno se encuentran todos los datos etiquetados, facilitando a la máquina el aprendizaje de la tarea. De modo que cada entrada existe un valor de la salida real.
De esta manera, mediante el conocimiento previo de un problema, la máquina aprende a tomar decisiones acordes a la salida esperada y se ajusta de mejor manera a la salida estimada.

Resulta de gran utilidad en aplicaciones reales como filtros detectores de spam en correos electrónicos, detectores de imágenes en captchas o en aplicaciones de reconocimiento de voz o escritura.

Dentro del aprendizaje supervisado podemos distinguir dos tipos de problemas, problemas de regresión y problemas de clasificación.

Problemas de regresión

Estos son los problemas en los que la salida esperada es una variable cuantitativa, es decir, un valor numérico. Por ello, se realiza un ajuste del modelo según las variables de entrada y la predicción de los modelos se realiza en base a este ajuste. Dentro de la regresión podemos encontrar problemas que trabajan con atributos con valores numéricos o categóricos.

Problemas de clasificación

Estos son los problemas en los que la salida esperada es una variable cualitativa, es decir, una respuesta de estilo categórica. Dentro de los problemas de clasificación, podemos encontrar problemas binarios, donde la salida pertenece a dos posibles clases, como por ejemplo problemas de clasificación de tumores donde la salida es benigna o maligna. También podemos encontrar problemas de clasificación multi clase, donde la salida se comprende entre varias categorías distintas, como por ejemplo problemas de clasificación del tiempo, donde la salida esperada puede pertenecer a una entre un conjunto de clases: *soleado, lluvia, nublado…etc*.

2.2.2. Aprendizaje no supervisado

Se trata de problemas para los cuales no se obtienen los datos etiquetados, y se extraen de los datos patrones previamente desconocidos. En este caso, no tenemos el vector de salidas reales, y la máquina debe de aprender relacionando las características, buscando patrones, etc. Un ejemplo de este método de aprendizaje es el clustering.

Se encuentran aplicaciones en problemas de análisis de comportamiento de la sociedad, o distribución o clasificación de grupos según sus características.

2.2.3. Aprendizaje semisupervisado o híbrido

Son los problemas para los cuales los datos se encuentran parcialmente etiquetados y la máquina aprende de ellos para poder clasificar los que no se encuentran etiquetados. Esta metodología combina el aprendizaje supervisado y no supervisado, en donde el modelo a estudiar solo recibe las variables de salida para algunas de las instancias.

2.2.4. Aprendizaje por refuerzo

En el entorno, la máquina observa los datos del entorno, y trata de buscar la conducta que minimizara el riesgo, es decir, minimizará un coste o maximizará una recompensa. Son técnicas de aprendizaje aplicables en problemas de robótica.

2.3. Pre procesamiento

El pre procesamiento es una etapa por la cual los datos son modificados y restructurados con el fin de que tengan una mejor representación para obtener un conjunto de datos final que estén libres de erratas, siendo catalogados de alta calidad con información útil para que posteriormente sea extraída por los modelos. Es una de las fases más importantes de la minería de datos, y una de las partes que mayor tiempo consume.

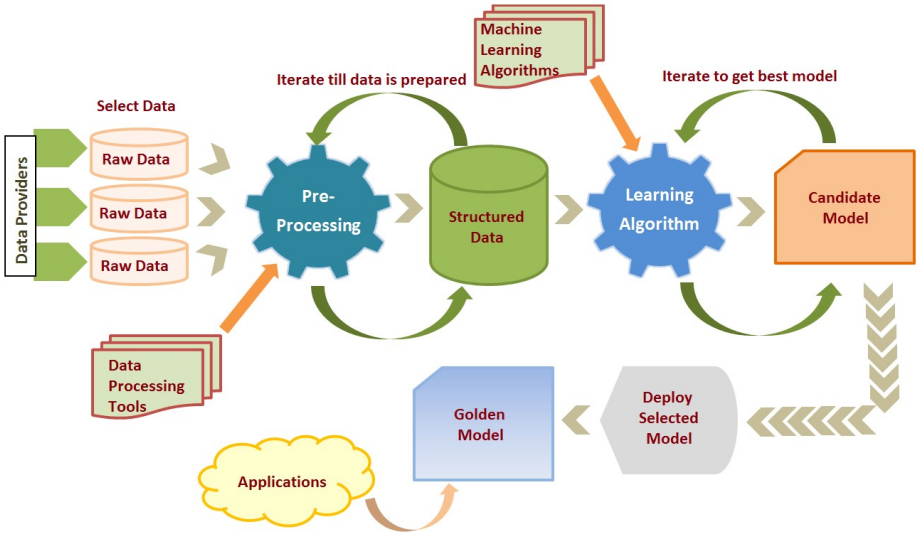

Figura 1: Modelo descriptivo de un proceso de Machine Learning. Origen de la imagen: https://i.pinimg.com/originals/da/d8/b2/dad8b2c2e33e5df1312cbdf08e4835f4.png

2.3.1. Causas y métodos para resolver las impurezas

La creación de diferentes métodos de limpieza de datos se encuentra diseñado para poder arreglar un problema específico. Hay una gran cantidad de técnicas de limpieza de datos que se encargan de contrarrestar las impurezas causadas.

Colección e integración de datos

Es el proceso por el cual se integran los datos de diferentes fuentes mediante una búsqueda exhaustiva, obteniendo un conjunto de datos más grande y consistente. Es una técnica muy empleada en el análisis de datos, con el fin de obtener un dataset más completo para que el modelo realice un aprendizaje más preciso.

Para un buen estudio de modelos, es muy importante que los datos de una misma característica se encuentren relacionados entre sí, e incluso para algunos otros casos, que diferentes características se encuentren correlacionadas.

Hay herramientas muy empleadas para la colección de datos como por ejemplo *Beautiful Soup* [4], una herramienta de Python para analizar documentos HTML.

Además, dentro de esta metodología se incluye la resolución de duplicados e inconsistencias de datos. Estos son casos como por ejemplo datos cuyo significado sea el mismo, muy frecuente en los problemas de análisis de texto donde se tiene la misma palabra, pero mal escrita ("Pepsi" y "Pespi" quiere significar lo mismo, lo cual hace referente al refresco, pero se hace una distinción de las dos palabras como elementos distintos). Por ello se aplican técnicas de filtrado para la resolución de estos errores. También se incluyen problemas de representación de las variables tales como una representación monetaria en euros y dólares o simplemente diferentes formas de asignar valores a variables de estilo binario (por ejemplo, etiquetas del estilo 0,1 o por ejemplo "positive", "negative").

Transformación de datos

Es el proceso por el que los datos son transformados y restructurados, de forma que la técnica de datos a emplear pueda ser aplicada, como por ejemplo en casos del tratamiento de textos (empleo de técnicas de BOW) o que el algoritmo a emplear sea más eficiente.

Limpieza de datos

Es el proceso por el que los datos son corregidos de errores presentes, o simplemente se realiza una reducción de variables con demasiado detalle.

Reducción de datos

Es el proceso de obtención de una representación reducida en volumen de nuestro conjunto de datos, de manera que se produzcan resultados analíticos iguales o similares que si se empleara el conjunto total de datos.

La alta dimensionalidad de los datos viene dada por tener un conjunto de datos demasiado grande. A pesar de que al tener más datos conlleva a tener más información y por lo tanto podría resultar en un mejor aprendizaje, puede llevar a consecuencias inesperadas como un sobre aprendizaje del modelo o incluso un coste computacional inviable.

La finalidad es agilizar el proceso de aprendizaje reduciendo el coste computacional y facilitando la manipulación de los datos.

2.4. Bag of Words

Los problemas para el tratamiento del lenguaje no se pueden tratar de una manera simple, por ello se tiene que emplear algoritmos de transformación especializados en la minería de textos, como Bag of Words (*BOW)*. La minería de textos son técnicas de computación para extraer información de alta calidad a partir de diferentes textos, es decir extracción de información a partir de datos no estructurados. La información no estructurada puede venir de diversas fuentes, como libros, informes, documentos administrativos, noticias o comentarios de redes sociales. La información no estructurada puede suponer una dificultad a la hora del aprendizaje, ya que supone una interpretación sencilla para los humanos, pero no para los ordenadores. Estas dificultades de compresión vienen dadas por términos y frases ambiguas, suposición del conocimiento y contexto del lector o de su sentido común y razonamiento o la influencia de las interacciones como puede ser en casos de redes sociales.

2.4.1. Pre procesamiento de documentos

Por lo tanto, para poder utilizar de forma eficaz y eficiente toda la información no estructurada es necesario emplear métodos de computación para extraer información de forma automática a partir de textos no estructurados, y analizar y resumir la información extraída. Para poder realizar el aprendizaje automático, se necesita un conjunto de entrenamiento donde,

- Cada ejemplo o instancia corresponde a un documento.
- Cada documento está caracterizado por sus palabras.

El modelo de BOW es una técnica para extraer información a partir de texto enmarcado dentro del procesamiento del lenguaje natural. En este algoritmo, se considera el cada documento como una bolsa de palabras, en donde se realizan dos suposiciones para tratar con los datos:

- Las palabras que se aprenden son independientes y no tienen relación entre sí.
- El orden de las palabras que se aprenden es irrelevante.

Cada documento del corpus, es decir, el conjunto de documentos de texto disponibles para aprender el modelo de BOW, se representa como un vector multidimensional. Cada dimensión de este vector es un término único del corpus, siendo este una palabra o conjunto de palabras y además el número de términos aprendidos determina la dimensión del vector (el número de variables del problema). Por lo tanto, cada documento o instancia del problema viene representado por un vector con tantos elementos o variables como términos aprendidos del corpus. Tal como se muestra en Figura 2 podemos observar una representación vectorial de un corpus de tamaño 3.

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Figura 2. En esta tabla se tienen tantas variables como palabras aprendidas por el corpus formado por las 3 reseñas. Si no se presenta dicha palabra en el documento, se le asigna peso 0.

Para poder crear las representaciones vectoriales, hay que tratar los documentos marcando un objetivo claro: reducir el número de términos del corpus. Esto se debe a que no todas las palabras aportan la misma cantidad de información, como por ejemplo los conectores léxicos. Podemos destacar varias etapas previas a la transformación vectorial:

Normalización

La normalización es el proceso por el que transformamos varias formas del mismo término a un formato común. En este caso, se emplean técnicas de supresión de signos de puntuación (puntos, guiones, comas…etc.), se transforma todo el texto a minúsculas con el fin de no aprender la misma palabra y el empleo de diccionarios de sinónimos.

Eliminación de términos con frecuencias muy altas o bajas

En la eliminación de términos con frecuencias muy altas o bajas, los términos que tienen frecuencias muy altas componen una gran proporción del total de palabras, pero no tienen mucha utilidad semántica, así como conectores léxicos o semánticos. Aquellas que tienen frecuencias muy bajas, pueden ser muy ricos semánticamente y aportar gran información, pero su aparición en los textos puede ser tan escasa que no merece la pena estudiarlos. Aquellos términos que no se comprenden entre estos dos valores, son aquellos que mejor representan al corpus y, por lo tanto, son aquellos que deberán ser incluidos en la matriz.

Eliminación de Stop-words

Las stop-words son palabras que probablemente sean irrelevantes para el análisis del corpus, aquellas que no poseen información. Hay una gran variedad de listas de stop-words, por lo que la categorización de estas es variada. No siempre es recomendable la eliminación de las stop-words, ya que se puede perder el significado original de la frase y la estructura del texto.

Reducción de palabras a su raíz

En la reducción de palabras a su raíz, el objetivo principal es reducir la variabilidad de términos reduciéndolos a su forma básica o raíz. Dentro de este proceso podemos distinguir dos técnicas distintas:

Stemming

Stemming trata de cortar las terminaciones de las palabras sin considerar las características lingüísticas de las palabras, es decir, trata de reducir una palabra a su raíz (de ahí proviene su nombre). El empleo del Stemming puede causar un aumento en el recall, causado por la abstracción de algunos términos [5].

Un ejemplo de Stemming sería: las palabras *argue, argued, argues, arguing* producen *argu*. En este caso, del término argue del inglés, nos devolvería su raíz, catalogando las 4 distintas palabras como la misma.

Lematización

La lematización trata de reducir las palabras a su forma básica, teniendo en cuenta el vocabulario y sus características morfológicas. Se obtiene le lema de la palabra. A diferencia de Stemming, sí se tiene en cuenta la identificación de sus palabras y su significado sintáctico. Además, se busca también reconocer sinónimos, como, por ejemplo: *better* tiene como raíz *good*.

Un ejemplo de este sería: las palabras *argue, argued, argues, arguing* producen *argue*. Como se puede comprobar en este ejemplo, se obtiene el lema argue para todas aquellas palabras que difieren de este verbo.

2.4.2. Obtención de los pesos

Una vez se tiene el corpus tratado, se realiza la obtención de la matriz de las palabras o vector de palabras. En cada posición, vendrá asignado un peso, el cual puede variar dependiendo de la técnica que utilicemos.

Pesos Binarios (Binary Weights)

Los pesos binarios representan la presencia o ausencia del término en el documento. Los pesos toman los valores 0 o 1, dependiendo de si se encuentra presente en dicho documento o no. En la Figura 3 podemos ver un ejemplo de asignación de pesos binarios.

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious	in	the
D1	1	1	1	1	0	1	1	1	0	0	0	1	1
D2	1	1	0	0	1	1	1	0	1	0	0	0	0
D3	0	0	0	0	0	1	0	0	0	1	1	0	0

Figura 3: Representación de pesos binarios para un corpus de tamaño 3.

Frecuencia de los términos (TF)

La frecuencia de los términos representa la frecuencia de un término en un documento. Para esta representación, la primera propuesta que hubo era simplemente realizar un conteo con el número de apariciones que tiene un término en el documento, así los términos con más apariciones serán más importantes en dicho documento. Esta representación tiene un problema, ya que los documentos que sean más largos tendrán un conteo mucho mayor que el resto, por lo que se propone la frecuencia de aparición de un término en documento, donde se realiza *Número de apariciones / Número de palabras de un documento*. En la Figura 4 podemos ver un ejemplo de asignación de pesos por frecuencia de aparición.

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious	in	the
D1	2/10	1/10	1/10	1/10	0	1/10	1/10	1/10	0	0	0	1/10	1/10
D2	1/6	1/6	0	0	1/6	1/6	1/6	0	1/6	0	0	0	0
D3	0	0	0	0	0	1/3	0	0	0	1/3	1/3	0	0

Figura 4: Representación de la Frecuencia de los Términos para un corpus de tamaño 3.

Inversa de la frecuencia en los documentos (IDF)

En la inversa de la frecuencia en los documentos, la idea es asignar pesos más grandes a términos no comunes en el corpus, así obteniendo un mayor poder de diferenciación. En esta metodología el peso de cada término se calcula a partir de todo el corpus, ya que este describe la importancia del corpus globalmente, no a nivel individual. Cuando se finaliza el proceso de asignación de pesos, aquellos términos que aparezcan en un documento se le asignará un peso, y para aquellos que no aparezcan se le asignará peso 0. Para calcular el peso, se debe emplear la formula indicada en la Ecuación 1: fórmula para calcular la inversa de la frecuencia en los documentos.

$$IDF(t) = 1 + \log(\frac{N}{df(t)})$$

Ecuación 1: fórmula para calcular la inversa de la frecuencia en los documentos

Dónde:

- **N** es el número de documentos del corpus.
- **Df(t)** es el número de documentos que contienen el término t.

En la Figura 5 se muestra la información referente al estado del corpus y la aparición de los términos en los documentos. En la Figura 6 podemos observar cómo quedaría el resultado final de la asignación de pesos a un dataset.

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious	in	the
N	3	3	3	3	3	3	3	3	3	3	3	3	3
df(t)	2	2	1	1	1	3	2	1	1	1	1	1	1
IFD	1.41	1.41	2.10	2.10	2.10	1	1.41	2.10	2.10	2.10	2.10	2.10	2.10

Figura 5: ejemplo de asignación de pesos. Se muestran el tamaño del corpus, la asignación de peros y el resultado final

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious	in	the
D1	1.41	1.41	2.10	2.10	0	1	1.41	2.10	0	0	0	2.10	2.10
D2	1.41	1.41	0	0	2.10	1	1.41	0	2.10	0	0	0	0
D3	0	0	0	0	0	1	0	0	0	2.10	2.10	0	0

Figura 6: ejemplo de un dataset asignado con los pesos empleando la inversa de la frecuencia de aparición.

TF-IDF

En TF-IDF, la idea principal de este método es fusionar las dos últimas metodologías mencionadas, valorando los términos que no son muy comunes en el corpus, pero que tienen un nivel de frecuencia razonable. Es el método más habitual de asignar pesos en el modelo de BOW. Dentro de esta metodología podemos encontrar dos fórmulas mostradas en la Ecuación 2: Fórmula general de **¡Error! No se encuentra el origen de la referencia.** y Ecuación 3: Fórmula comúnmente utilizada.

$$TF - IDF(t) = TF(t) * IDF(t)$$

Ecuación 2: Fórmula general de TF-IDF

$$TF - IDF(t) = TF(t) * \log \frac{N}{df(t)}$$

Ecuación 3: Fórmula comúnmente utilizada para el método TF-IDF

2.5. Clasificación automática

Cuando se finaliza un proceso de pre procesamiento, se pasa a la fase de selección del modelo, es decir, el algoritmo con el que se va a trabajar. Idealmente, el que mejor se adapte al aprendizaje del problema a tratar.

2.5.1. Regresión logística

El objetivo es encontrar un modelo que represente un conjunto de datos observados, en donde el modelo servirá para predecir valores para nuevos datos. A diferencia de los problemas de regresión mencionados anteriormente, dónde la salida era un valor, en este tipo de regresión la salida es un valor discreto (una clase o etiqueta).

Para los casos binarios, tenemos que modificar la hipótesis presente en la regresión lineal para poder clasificar, de manera que vamos a definir que $0 \leq h(x) \leq 1$. Para ello, se emplea la función sigmoide, la cual logra que los valores queden comprendidos entre [0,1] de la manera que se observa en Figura 7 mediante la fórmula de la Ecuación 4: Fórmula de la hipótesis de la regresión logística..

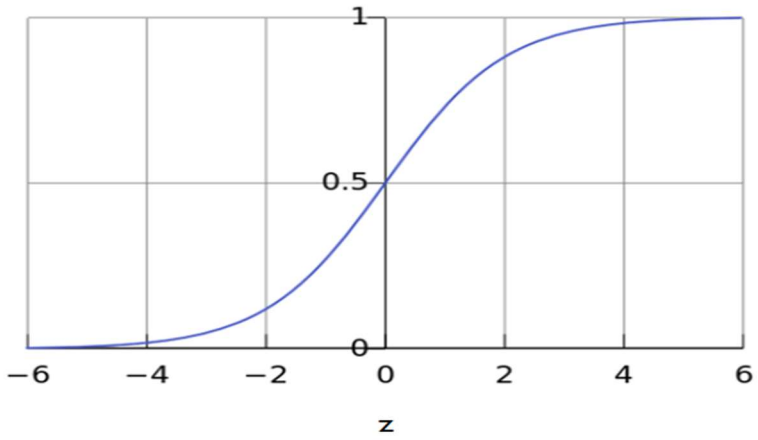

Figura 7: Representación gráfica de la función sigmoide. Origen de la imagen: https://es-academic.com/pictures/eswiki/76/Logistic-curve.svg

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Ecuación 4: Fórmula de la hipótesis de la regresión logística.

Formalmente, se podría decir que la regresión logística es una representación de la probabilidad de pertenecer a la clase positiva ya que si el resultado de $h(x)$ = 0.7 se podría representar como que existe un 70% de probabilidad de que x pertenezca a la clase y = 1. Por lo tanto, hay que obtener una frontera de decisión de manera en la que se puedan separar los valores representados por agrupaciones (se pueden aplicar características lineales como polinomiales si es necesario). Para poder obtener un buen ajuste de la frontera de decisión, hay que obtener primero un buen ajuste de las θ.

Como se toman valores comprendidos en el intervalo [0,1], podemos definir la fórmula de la **¡Error! No se encuentra el origen de la referencia.** A esto se le denomina entropía cruzada, función que se debe minimizar para obtener los pesos de la regresión logística. Esta función, como bien se puede observar si las predicciones son distintas, la función va a devolver valores altos, y valores bajos en el caso contrario. Esto se debe a que cuando toma valores distintos en la predicción significa que el ajuste no es adecuado, y por lo tanto se debe corregir.

$$\boldsymbol{J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))}$$

Ecuación 5: Representación de la entropía cruzada.

Por lo tanto, para minimizar la función de coste se emplea el descenso por gradiente. Dentro del descenso por gradiente, los parámetros se modifican en la dirección de mayor pendiente, y los pasos de modificación de las thetas son en función del valor del gradiente en el punto actual. La dirección del gradiente viene dada por el negativo del gradiente, donde en cada iteración se modifican todos los parámetros de theta simultáneamente aplicando la fórmula de la Ecuación 6.

$$\theta_0 \coloneqq \theta_0 - \alpha \frac{\partial J(\theta)}{\partial \theta_0}$$

Ecuación 6: Fórmula para la modificación de las thetas en cada iteración del descenso por gradiente, en este caso para theta 0.

Dentro de esta fórmula se pueden destacar dos componentes principales, las cuales hacen que se modifique el valor de cada parámetro en la dirección en la que se reduce el error más rápidamente:

- Factor de aprendizaje (α): mide cuanto modificamos el valor del parámetro. Es muy importante seleccionar un buen parámetro de aprendizaje, ya que si se escoge un valor muy pequeño el descenso puede ser muy costoso y lento, y si se escoge un valor muy grande podemos llegar a saltarnos el mínimo y la función puede no converger.
- Derivada: indica el valor de la pendiente con mayor variación en el punto actual.

Se puede emplear el descenso por gradiente por "lotes", en el que en cada iteración se emplean todos los ejemplos, aunque si tenemos muchos ejemplos puede llegar a ser muy costoso computacionalmente. Por ello, también se emplea el descenso por gradiente incremental o incluso otras técnicas de optimización [6].

2.5.2. NAIVE BAYES

Naive Bayes es un modelo perteneciente a los algoritmos de clasificación probabilista que se apoya en el teorema de Bayes. A la hora de realizar la clasificación, Naive Bayes emplea una regla de clasificación denominada máximum a posteriori (MAP), de entre las probabilidades a posteriori de cada clase y escoge la más alta. Por lo tanto, en los modelos generativos lo que se realiza es el cálculo de las probabilidades condicionadas a cada clase y posteriormente se emplea la regla de MAP.

$$C_{NB} = argmax(P(c_j) \prod_{i=1}^{n} P(x_i|c_j))$$

Ecuación 7: Fórmula de máximum a posteriori (MAP)

Existen distintas formas de estimar las probabilidades condicionadas. Una de las más comunes cuando combinamos Naive Bayes con BOW es utilizar distribuciones multinomiales. [7]

2.5.3. REDES NEURONALES

Se trata de un algoritmo de aprendizaje que surge de la idea de intentar imitar el funcionamiento del cerebro humano. Estos modelos son la base del Deep Learning, que en la actualidad son los modelos que mejores resultados han dado en clasificación de imágenes y procesamiento del lenguaje natural.

Las redes neuronales basan su estructura en neuronas artificiales que intentan imitar a una neurona humana (ver Figura 8). Cada neurona funciona como una unidad logística, en la que se combinan las entradas con un conjunto de pesos y, posteriormente, se aplica una función de activación (que puede ser una función sigmoidea u otra [8]). Las redes se construyen combinando neuronas en capas. La idea de su funcionamiento es que, en cada capa, la red aprenda diferentes características, y luego se realiza la combinación de estas características en la salida.

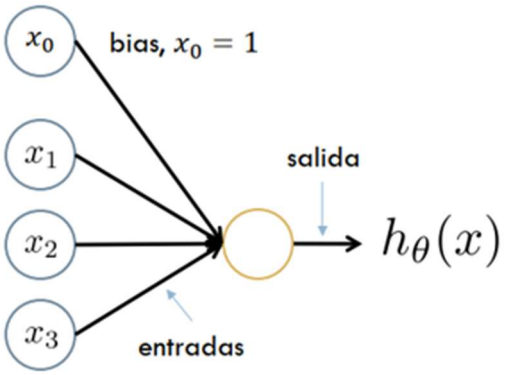

Figura 8: Ejemplo de neurona de una capa de una red neuronal.

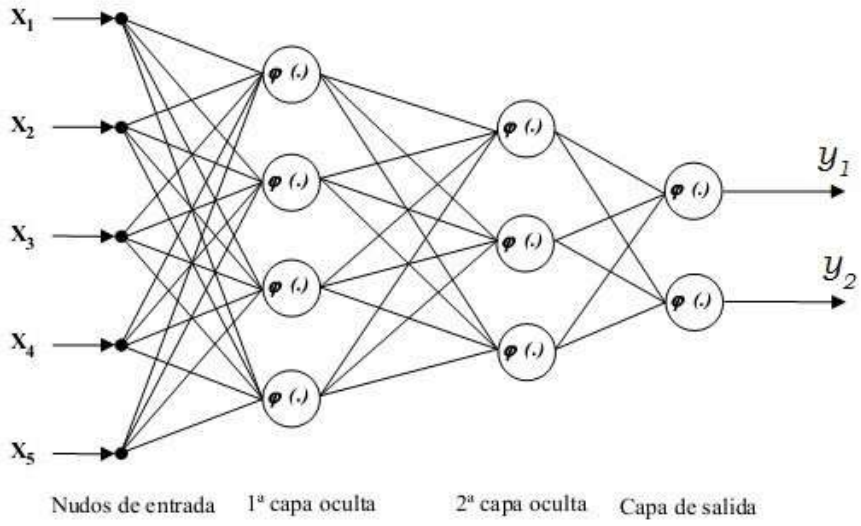

Figura 9: Ejemplo de una red neuronal multicapa con dos capas ocultas. Origen de la imagen: https://s3.amazonaws.com/s3.timetoast.com/public/uploads/photo/11369920/image/fc75040b06b31d566d18b59f5030b7b2

Para poder realizar el aprendizaje de una red neuronal, es decir, aprender los pesos que se asignan a cada capa de la neurona, se emplea un algoritmo denominado backpropagation. La idea del algoritmo es ir recorriendo desde la capa de salida hasta los nodos de entrada modificando los pesos de cada capa en base a los valores de la capa de salida para agilizar el cómputo de las derivadas parciales.

2.6. Validación de modelos, métodos de validación

La validación de un modelo es la comprobación de la capacidad de aprendizaje de la máquina. Esto es, comprobar que el modelo se ajusta de manera correcta al conjunto de datos de entrenamiento, y además se comporta y ajusta de manera esperada al conjunto global de datos.

Para la validación se tiene un conjunto de validación, el cual es un conjunto de datos similar a los de entrenamiento. De esta manera, se tiene un conjunto de datos que no ha sido empleado

para entrenar el modelo, de modo que no son pertenecientes al conjunto de entrenamiento, pero si forman parte del conjunto total de datos.

El método más común es la validación cruzada o *cross validation*. Dentro de este proceso, se entrenan los modelos sobre el conjunto de entrenamiento y de manera seguida se prueba el modelo aprendido sobre el conjunto de prueba o *test*. Lo más habitual en el empleo de estas técnicas es el entrenamiento de múltiples algoritmos en busca del mejor rendimiento en el conjunto de prueba, lo cual conlleva procesos largos de búsqueda y de computación.

La validación cruzada de k particiones consiste en dividir el conjunto total de datos en k subconjuntos y, seguido, se entrena el modelo con las primeras k-1 particiones y se deja la última para la validación del modelo. Este proceso se realiza k veces alternando el conjunto de validación, para comprobar si el modelo de aprendizaje es robusto y comprobar si tiene una buena capacidad de generalización.

2.7. Métricas de evaluación

Para poder realizar un seguimiento de los fallos, se puede realizar un análisis del error, donde se estudian manualmente los ejemplos que se fallan. Para poder realizar este estudio, se modela una matriz de confusión, la cual representa los valores de la clase real frente a los valores de la predicción. (Ver **¡Error! No se encuentra el origen de la referencia.**).

	Predicción		
		Predicción positiva	**Predicción negativa**
Clase Real	**Clase Positiva**	**Verdadero Positivo**	**Falso Negativo**
	Clase Negativa	**Falso Positivo**	**Verdadero Negativo**

Tabla 1: Matriz de confusión. Las celdas coloreadas en azul representan el recall, en cambio las celdas coloreadas en naranja representan la precisión.

Mediante el empleo de esta matriz de confusión podemos calcular de diferentes formas el índice de acierto o fallo de nuestro modelo. El accuracy es una de estas metodologías y mide el porcentaje de ejemplos que se han acertado en el modelo. La fórmula para calcular el accuracy se describe como

$$accuracy = \frac{Verdadero\ Positivo + Verdadero\ Negativo}{Verdadero\ Positivo + \ Verdadero\ Negativo + \ Falso\ Positivo + Falso\ Negativo}$$

Ecuación 8: Fórmula para calcular el accuracy.

La precisión mide la diferencia entre qué porcentaje real es de la clase positiva y se calcula como

$$\boldsymbol{P = \frac{Positivo\ Verdadero}{(Positivo\ Verdadero\ +\ Falso\ Positivo)}}$$

Ecuación 9: Fórmula para calcular la precisión.

En cambio, el recall mide que porcentaje se ha predicho de manera correcta, y se calcula como

$$\boldsymbol{R = \frac{Positivo\ Verdadero}{(Positivo\ Verdadero + Falso\ Negativo)}}$$

Ecuación 10: Fórmula para calcular el recall.

También se pueden emplear técnicas que combinan la precisión y el recall, como por ejemplo el FScore.

$$FScore = 2\frac{P * R}{P + R}$$

Ecuación 11: Fórmula para calcular el FScore.

La precisión y el recall vienen determinados por el umbral que se escoge (únicamente en los casos en los que los modelos probabilistas). Si se selecciona un umbral alto, aumenta la precisión y disminuye el recall, de manera que evitamos en su mayoría la cantidad de falsos positivos, pero por el contrario aumenta el número de falsos negativos. Por el lado contrario, si seleccionamos un umbral bajo, aumenta el recall y disminuye la precisión, en donde se predicen todos los positivos correctamente, pero aumenta el número de falsos positivos.

2.8. MULTI INSTANCE LEARNING

Multi Instance Learning o MIL son tipos de aprendizaje supervisado, en donde se quiere clasificar instancias individuales, pero en vez de tener la salida de las instancias individuales tenemos las salidas de las instancias que le agrupan. Por lo tanto, en nuestro problema el objetivo que tenemos es poder clasificar aquellas instancias individuales (correspondiente a las frases de las reseñas) a partir de un conjunto de instancias (en nuestro caso, un documento entero o reseña).

Tradicionalmente, los algoritmos de MIL incluyen ciertas creencias [9]. Por ejemplo, se asume que todas las instancias individuales pertenecientes a una bolsa pertenecen a una clase y, bajo esta suposición, define que un ejemplo (documento) es positivo únicamente si hay una o más instancias individuales positivas. Por el contrario, un ejemplo se considera negativo si y solo si todas las instancias individuales son negativas. Por lo tanto, también se supone que la etiqueta del grupo viene definida por las etiquetas de las instancias individuales [10]. De hecho, el objetivo principal de estos modelos radicaba en hacer predicciones sobre los grupos de instancias y no en las instancias individuales.

Este tipo de enfoques son bastante restrictivos, y esta metodología de pensamiento puede contrastar con algunos casos más realistas en el mundo cotidiano, como, por ejemplo, en el caso de data sets como el nuestro, en el que, dentro etiquetado como negativo, podemos encontrar instancias individuales positivas, del mismo modo, podemos encontrar instancias negativas en grupos de instancias etiquetados como positivos. Por ello, este estilo de razonamiento no se ajusta a nuestro problema. En nuestro caso, vamos a tolerar que dentro de cada conjunto de instancias podamos encontrar ya sean instancias individuales positivas o negativas, independientemente de la etiqueta que tenga.

3. PROPUESTA DE MULTI-INSTANCE LEARNING

Como se ha mencionado en apartados anteriores, vamos a emplear el algoritmo de MIL para la clasificación de instancias individuales de reseñas [2]. Dentro de nuestro problema, nosotros tenemos para cada grupo de instancias una etiqueta, de modo que podemos establecer nuestro corpus como grupos de instancias etiquetados. Por lo tanto, podemos establecer una definición formal de la siguiente manera:

$$corpus = \{G_k, l_k\}_{k=1..K}$$

Ecuación 12: Definición del conjunto del corpus.

- G_k: Es el grupo de instancias del documento k del corpus.
- L_k: Es la etiqueta asociada al grupo de instancias k, la cual puede tomar el valor 0 o 1.

Para el caso de las instancias individuales que se encuentran en cada grupo vamos a asumir que su etiqueta es desconocida, por lo tanto, son instancias que no han sido etiquetadas y buscamos etiquetar.

Para realizar las predicciones, vamos a emplear la misma hipótesis que en el caso de la regresión logística, donde cada predicción de valores se establece como:

$$\hat{y}_i = \hat{y}_\theta(x_i) = \sigma(\theta^T x_i) = \frac{1}{1 + e^{\theta^T x_i}}$$

Ecuación 13: Desarrollo de las predicciones para un clasificador y entrenado con unos valores θ para cierto valor x_i

A pesar de emplear la misma fórmula que en regresión logística, vamos a emplear una función de coste específica la cual habrá que minimizar, con el fin de cumplir los objetivos especificados. Para poder establecer una buena distinción entre las frases que tenemos, vamos a definir una función de similitud que se encargará de comprobar lo parecidas que son las sentencias entre sí. Además, la función de coste deberá tener en cuenta la predicción del clasificador comparado con su valor real. Por lo tanto, se define la función de coste para nuestro problema:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} K(x_i, x_j)\,\Delta_1(\hat{y}_i, \hat{y}_j) + \frac{\lambda}{K}\sum_{k=1}^{K}\Delta_2(\hat{l}_k, l_k)$$

Ecuación 14: Definición de la función de coste específica sin desarrollar.

- $K(x_i, x_j)$: Es la función que se encarga de medir la similitud entre las instancias x_i y x_j. Esta función toma valores comprendidos en el intervalo [0,1] y cuyos parámetros x_i y x_j corresponderán a las instancias ya transformadas de BoW.
- $\Delta_1(\hat{y}_i, \hat{y}_j)$: Es un parámetro de penalización que toma valores comprendidos entre [0,1]. Este parámetro se encarga de penalizar de manera más alta aquellas instancias cuyas predicciones de etiquetas son diferentes pero la similitud de las frases es muy parecida.
- $\Delta_2(\hat{l}_k, l_k)$: Es un parámetro de penalización que toma valores comprendidos entre [0,1]. Se encarga de medir la diferencia entre la etiqueta real de grupo y la predicción a nivel grupal resultante de hacer la agregación de las instancias individuales.

- λ: es un parámetro positivo mayor que cero, que establece la relevancia del segundo argumento de la función. Cuanto más valor tenga, más relevancia obtiene este segundo argumento en la función.

Es destacable en esta función que encontramos dos argumentos distintos separados por un operador de suma. El primero de los argumentos tiene en cuenta las instancias individuales, y la similitud entre ellas penalizando de manera más costosa aquellas instancias que son similares entre ellas, pero son clasificadas con etiquetas distintas. Además, este primer argumento realiza iteraciones en bucle hasta llegar a N, que representa el número total de instancias. Por otro lado, el segundo de los argumentos itera hasta llegar a K, representando el número total de grupos del corpus. Dentro de este segundo argumento se realiza una comparativa entre grupos y las instancias que contiene.

De esta manera, se puede obtener una buena comparación entre las instancias y los grupos simultáneamente. A pesar de que estos dos parámetros se pueden distinguir y definir por manera separada, no pueden emplearse de manera individual ya que no obtendrían buenos resultados.

Dentro de la función de coste, vamos a escoger para las funciones Δ_1 y Δ_2 como el error cuadrático, y para la función de similitud vamos a emplear la siguiente función Kernel de la Ecuación 17: Ecuación de la norma euclídea empleada en la función de coste. [11].

$$\Delta_1(\hat{y}_i, \hat{y}_j) = (\hat{y}_i - \hat{y}_j)^2$$

Ecuación 15: Ecuación del error cuadrático medio empleado en la función de coste para Δ_1.

$$\Delta_2(\hat{l}_k, l_k) = (\hat{l}_k - l_k)^2$$

Ecuación 16: Ecuación del error cuadrático medio empleado en la función de coste para Δ_2.

$$K(x_i, x_j) = \exp(-||x_i - x_j||^2)$$

Ecuación 17: Ecuación de la norma euclídea empleada en la función de coste.

Para poder asociar los grupos de instancias y añadirlos a la función de coste, vamos a definir que $\hat{l}_k$, es decir la predicción de grupo que se obtiene como resultado de agregar las predicciones de las frases que forman cada grupo. Para ello, necesitamos una función de agregación A. En este caso, utilizaremos la media aritmética. Por lo tanto, de cada grupo se obtiene como media de los elementos de su grupo.

$$\hat{l}_k = A(G_k, \theta) = \frac{1}{|G_k|} \sum_{i \in G_k} \hat{y}_\theta$$

Ecuación 18: Obtención de la etiqueta de un grupo como media de sus elementos.

De esta manera, podemos desarrollar la función de coste definida Ecuación 14: Definición de la función de coste específica sin desarrollar. Añadiendo las funcionalidades que hemos ido mencionando, de modo que podemos definir de manera completa la función de coste.

$$J(\theta) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} e^{(-||x_i - x_j||^2)} (\sigma(\theta^T x_i) - \sigma(\theta^T x_j))^2 + \frac{\lambda}{K} \sum_{k=1}^{K} \frac{1}{|G_k|} \sum_{i \in G_k} (\sigma(\theta^T x_i) - l_k)^2$$

Ecuación 19: Función de coste específica desarrollada.

Por lo tanto, una vez definido el clasificador a emplear y la función de coste, podemos realizar un planteamiento sobre cómo relacionar estos elementos. El primer paso que debemos dar dentro de nuestro trabajo es realizar las representaciones vectoriales x_j mediante algoritmos de BoW. El segundo de los pasos que debemos de realizar es obtener una optimización de los parámetros θ, empleando metodologías de descenso de gradiente. Una vez entrenado el modelo, hay que realizar una optimización de este tratando de obtener unos buenos resultados en conjuntos de prueba y validación, y posteriormente poder predecir las instancias individuales de cada grupo de instancias.

4. IMPLEMENTACIÓN

Nuestro problema, por lo definido y comentado previamente, trata de clasificar frases individuales dentro de una reseña. Por lo tanto, tenemos que seguir el proceso de Machine Learning comentado en la Figura 1: Modelo descriptivo de un proceso de Machine Learning.

4.1. Lenguaje de programación

Antes de comenzar con la resolución del problema, hay que establecer el lenguaje de programación que se va a emplear. Hoy en día existen herramientas que son capaces de implementar algoritmos de clasificación. Podemos encontrar funciones implementadas en C++ [12] o Java [13] que permiten realizar un análisis de los textos o también podemos encontrar programas más potentes como Matlab que contiene extensiones como *Text Analytics Toolbox* que permite tratar y extraer datos a partir de textos[14]

Además de eso, también podemos encontrar distintos servicios que pueden ayudar a la implementación y compilación del código, que permiten ejecutar código en la nube o servicio online sin tener que realizar consumo de nuestra CPU. Ejemplos de este tipo de servicios pueden ser AWS de Amazon [15] que ofrecen tanto servicios de pago como gratuitos. Del mismo modo, Microsoft también ofrece sus servicios con Microsoft Azure, siendo un servicio de computación en la nube implementado con sus centros de datos [16]. Además, también existen herramientas destinadas al empleo de un único lenguaje como el Google colab, herramienta de Google que permite la ejecución de código de Python en el navegador. Esta herramienta está más destinada al aprendizaje automático. [17]

En nuestro caso, el lenguaje que vamos a seleccionar es Python. A pesar de que los otros lenguajes mencionados anteriormente son muy útiles a la hora de emplear los algoritmos, Python resulta mucho más práctico y tiene una gran cantidad de librerías disponibles.

Python es un lenguaje muy popular de programación de alto nivel interpretado cuyo origen consta a finales de los años ochenta. Es programa muy utilizado debido a la fácil implementación del código, lo cual es únicamente una característica extra en la funcionalidad de su contenido. Es un programa que es capaz de soportar la orientación a objetos, programación interpretativa y programación funcional.

Además, hoy en día consta con un gran número de librerías que le aportan funcionalidades adicionales. En el ámbito de Machine Learning es muy empleado, ya que tiene a su disposición muchas librerías que permiten realizar el entrenamiento de los modelos y facilitan los procesos.

Por ello, para poder trabajar con Python vamos a emplear en nuestro entorno Anaconda Navigator, interfaz gráfica de usuario que permite trabajar con Python.

4.1.1. Librerías utilizadas

Para poder realizar el desarrollo de nuestro proyecto, empleamos diversas librerías de Python, que facilitan tanto el tratamiento de datos como la implementación de modelos. A continuación, vamos a describir las librerías implementadas junto a las funcionalidades que empleamos en el proyecto.

Numpy

Numpy es una librería muy práctica en Python la cual se encuentra especializada en el cálculo y análisis de un gran volumen de datos. Es la clase que implementa la clase de tipo *array*, estructura de datos de un mismo tipo distribuido en un espacio N-dimensional, permitiendo la implementación de *arrays* de una dimensión, dos dimensiones e incluso tres dimensiones. [18]

Esta librería la implementamos con la finalidad de poder obtener de manera mucho más sencilla el porcentaje de acierto de cada prueba.

Pandas

De manera similar a Numpy, la librería de Pandas es una herramienta basada en la estructura y organización de los datos, que facilita la manipulación de los datos. Es una librería que define una nueva estructura denominada DataFrame basada en los arrays de Numpy, pero dispone de diferentes funcionalidades [19].

Es una muy buena herramienta para la lectura de ficheros y organización de datos. Es una librería muy práctica que permite leer ficheros csv, Excel, Json o SQL y lo distribuye en una estructura accesible mediante índices. Por lo tanto, de esta manera vamos a poder distinguir las columnas mediante un nombre y poder acceder a los valores de dicha columna mediante ese nombre.

Scipy

Es una librería de software libre que contiene algoritmos matemáticos. Además de contener de herramientas matemáticas contiene herramientas de optimización e interpolación [20].

Sklearn

Es una librería de software libre para el aprendizaje automático diseñada para Python. Presenta compatibilidad con muchas otras librerías e implementa algoritmos de clasificación, clustering,

regresión y algunos otros algoritmos para la reducción de dimensionalidad de datos como PCA [21].

Nltk

Nltk proviene de sus siglas en inglés *Natural Language Toolkit*, y es una librería especializada en el tratamiento del lenguaje natural para el estudio de los algoritmos. Dentro de librería encontramos bibliotecas o diccionarios y distintas funciones para el tratamiento de los datos [22].

Seaborn y MatplotLib

Ambas son librerías de visualización de datos. Seaborn [23] es una librería desarrollada sobre la base de matplotlib [24], la cual es una librería especializada en la generación de gráficos siendo esta compatible con las librerías de Numpy, arrays y listas.

Re

Esta librería se encuentra especializada para el tratamiento de cadenas de texto, comprobando patrones o coincidencias entre distintas cadenas de texto [25].

Time

Esta es una librería de Python especializada en medir el coste computacional de un algoritmo o ejecución. Esta librería es capaz de medir el tiempo de ejecución de una función o el tiempo que está en ejecución nuestro programa [26].

4.2. Datasets

Trabajamos con un total de 5 datasets diferentes, los cuales contienen formatos y contenidos distintos, que recopilan reseñas de diferentes sitios comerciales o de ocio como Amazon, IMDB o Yelp.

Amazon es una compañía estadounidense para la compra y venta online de cualquier tipo de productos mediante su aplicación o distrito web. Dentro de esta aplicación se puede evaluar o catalogar un producto por estrellas (en un rango de 0 a 5 estrellas) y también dispone de una sección en cada producto para publicar reseñas. Como Amazon contiene una gran cantidad de productos y reseñas, se ha obtenido los datos referentes a el apartado *Cell phones and Accessories.*

IMDB proviene de su significado en inglés *Internet Movie Database* y contiene toda la información relevante a las peliculas, incluyendo las reseñas de estas.

Yelp es una aplicación para encontrar reseñas de diferentes lugares de ocio, abarcando desde el sector de ocio hasta zonas pertenecientes al sector sanitario como dentistas.

4.2.1. Datasets con grupos de instancias

IMDB_Dataset

Este dataset proporcionado en [27] contiene la información sin tratar de las reseñas de IMDB. Esto significa que cada reseña es potencialmente un grupo de instancias, ya que dentro de cada reseña puede contener varias frases o no. El formato del fichero es un fichero de estilo csv, ficheros cuyas siglas significan valores separados por comas o *Comma Separated Values* de su terminología en inglés.

En este fichero, en cada línea del documento encontramos cada instancia a excepción de la línea 1 que contiene el nombre de cada variable. Dentro del dataset, encontramos un total de 2 variables *"review"* que contiene el contenido de la reseña y *"sentiment"* que contiene en formato de texto si la instancia es positiva o negativa.

El dataset contiene un gran número de instancias llegando hasta las 50000 instancias.

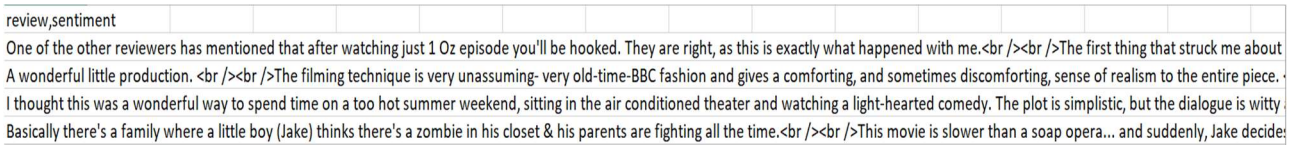

Figura 10: Ejemplo de las instancias del dataset de IMDB_Dataset en formato csv.

Amazon Cell_Phones_and_Accessories_5

Este dataset proporcionado en [28] contiene la información sin tratar de las reseñas de Amazon pertenecientes a la categoría de "Cell Phones and Accessories" que, al igual que el dataset anterior, contiene las reseñas que son potencialmente grupos de instancias. El formato de fichero es distinto al anterior, ya que es un formato de estilo json.

Los json son cadenas de textos, diseñadas con el objetivo de transmitir datos de manera más sencilla y eficaz a través de la red. Los datos se encuentran separados por comas, pero cada dato se escribe en pares, donde el primer elemento es el nombre del atributo o variable (es conocido como *key*) y el segundo es el valor que toma el dato. Cada par de datos se encuentra separado por el símbolo de ":".

Dentro de este dataset, encontramos en cada fila del dataset reseñas con un total de 9 atributos, pero nos vamos a centrar únicamente en *"reviewText"* y *"overall"* que contienen el contenido de la reseña y la calificación de la reseña en una puntuación de 1 a 5 respectivamente.


```
{"reviewerID": "A30TL5EWN6DFXT", "asin": "120401325X", "reviewerName":
"christina", "helpful": [0, 0], "reviewText": "They look good and stick
good! I just don't like the rounded shape because I was always bumping it
and Siri kept popping up and it was irritating. I just won't buy a
product like this again", "overall": 4.0, "summary": "Looks Good",
"unixReviewTime": 1400630400, "reviewTime": "05 21, 2014"}
```


Figura 11: Ejemplo de las instancias del dataset de cell_phones_and_Accessories_5 en formato json.

Este dataset es el más extenso de todos y el que más coste computacional requiere ya que alberga un total de 194.439 de instancias.

4.2.2. Datasets de instancias individuales

Los siguientes son datasets de instancias individuales que se implementan con el propósito de poder realizar comparaciones a la hora de realizar clasificaciones de instancias individuales partiendo de grupos de instancias. De esta manera se puede probar de manera justa el algoritmo proporcionado por [2].

Amazon_cells_labelled

Este dataset proporcionado en [2] contiene la información de las instancias individuales, es decir, de las frases individuales de reseñas recopiladas en Amazon. El formato del fichero es un documento de texto (.txt) que contiene en cada línea una frase con su etiqueta correspondiente. Dentro de cada línea, podemos encontrar la frase y la etiqueta separada por una tabulación. Este fichero contiene un total de mil instancias.

```
2   Good case, Excellent value. 1
3   Great for the jawbone.  1
4   Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!! 0
```

Figura 12: Ejemplo de las instancias del dataset de Amazon_cells_labeled.

Imdb_labelled

Este dataset proporcionado en [2] contiene la información de las instancias individuales, es decir, de las frases individuales de reseñas recopiladas de IMDB. El formato de fichero es similar al anterior, pero sus contenidos son distintos. Este fichero contiene un total de mil instancias.

```
1   A very, very, very slow-moving, aimless movie about a distressed, drifting young man.   0
2   Not sure who was more lost - the flat characters or the audience, nearly half of whom walked out.   0
3   Attempting artiness with black & white and clever camera angles, the movie disappointed - became even more ridiculous - as the
4   Very little music or anything to speak of.      0
```

Figura 13: Ejemplo de las instancias del dataset de imdb_labelled.

Yelp_labelled

Este dataset proporcionado en [2] contiene la información de las instancias individuales, es decir, de las frases individuales de reseñas recopiladas de Yelp. El formato de fichero es similar es similar a los dos anteriores y contiene al igual que los otros un total de mil instancias.

```
Wow... Loved this place.    1
Crust is not good.  0
Not tasty and the texture was just nasty.   0
Stopped by during the late May bank holiday off Rick Steve recommendation and loved it. 1
```

Figura 14: Ejemplo de las instancias del dataset de yelp_labelled.

4.3. Pre procesamiento de datos

4.3.1. Lectura de los datos

Para la lectura de datos, empleamos la librería de Pandas la cual ofrece métodos muy útiles de lectura de datos y la almacena en DataFrames el cual facilita el manejo y tratamiento de los datos (Figura 15: Visualización de los datos obtenidos y almacenados en los DataFrames de Pandas.).

```
Dataset_IMDB_completo.head(10)
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The filming tec...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1
5	Probably my all-time favorite movie, a story o...	1
6	I sure would like to see a resurrection of a u...	1
7	This show was an amazing, fresh & innovative i...	0
8	Encouraged by the positive comments about this...	0
9	If you like original gut wrenching laughter yo...	1

Figura 15: Visualización de los datos obtenidos y almacenados en los DataFrames de Pandas.

4.3.2. Tratamiento de las instancias y variables

Para la estructura de datos proveniente del fichero json dispone de 9 variables, por lo tanto, vamos a modificarlo para quedarnos únicamente con las que son de nuestro interés (las mencionadas anteriormente). Por otro lado, tenemos una inconsistencia de datos entre los diferentes datasets en el caso de las etiquetas.

Los datasets de las instancias individuales [2] contienen las etiquetas evaluadas con los valores 0 o 1, siendo estas negativas o positivas respectivamente. Por otro lado, en el dataset de Amazon cell_phones_and_accessories_5, encontramos las instancias evaluadas en una escala del 1 al 5, siendo 1 la puntuación mínima y 5 la máxima. Por último, encontramos en el IMDB_Dataset las etiquetas clasificadas como “positive” o “negative” en forma de texto.

Para poder realizar un estudio y una colección adecuada de datos, tenemos que establecer unas etiquetas comunes para todos los dataset, en este caso estableceremos 1 como etiqueta negativa y 0 como etiqueta positiva. Para el caso de IMDB, simplemente establecemos los valores de 1 en aquellas instancias donde encontremos “negative” como etiqueta y 0 en caso contrario. Para el caso de Amazon, consideramos que las etiquetas evaluadas con 1 o 2 estrellas son negativas estableciendo el valor 1, y el caso de que se encuentren evaluadas con 3,4 o 5 estrellas establecemos el valor 0.

4.3.3. Distribución de instancias

Una vez finalizado el proceso de colección e integración de datos, visualizamos las etiquetas en los dataset de los grupos de instancias para comprobar el número total de instancias de cada categoría. En IMDB, nos encontramos un dataset perfectamente balanceado, ya que de las 50 mil instancias 25 mil son positivas y las otras 25 mil son negativas. De manera contraria, en el caso de Amazon, nos encontramos con un dataset no balanceado, ya que de las 195 mil instancias (aproximadamente), nos encontramos con 24.343 instancias negativas y 170.096 instancias de la clase positiva. Este estilo de datasets no balanceados puede suponer un problema de cara a futuras predicciones, ya que los modelos van a tener tendencia a asignar futuros ejemplos a la clase positiva.

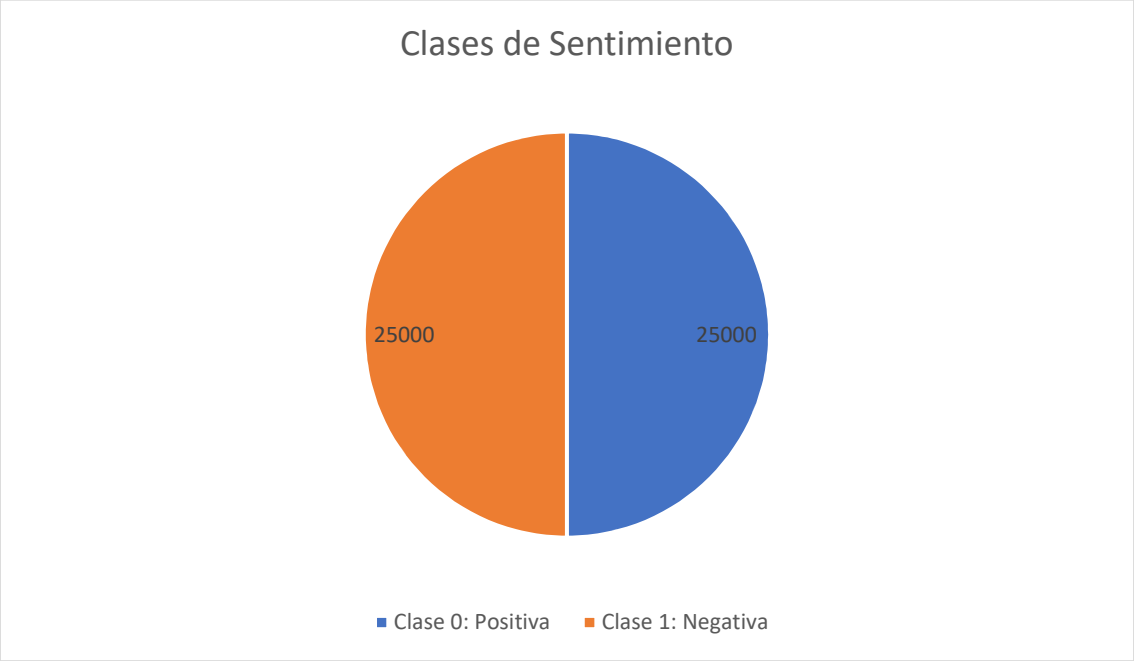

Figura 16: Gráfico que muestra la diferencia entre las clases positivas y negativas del dataset de IMDB

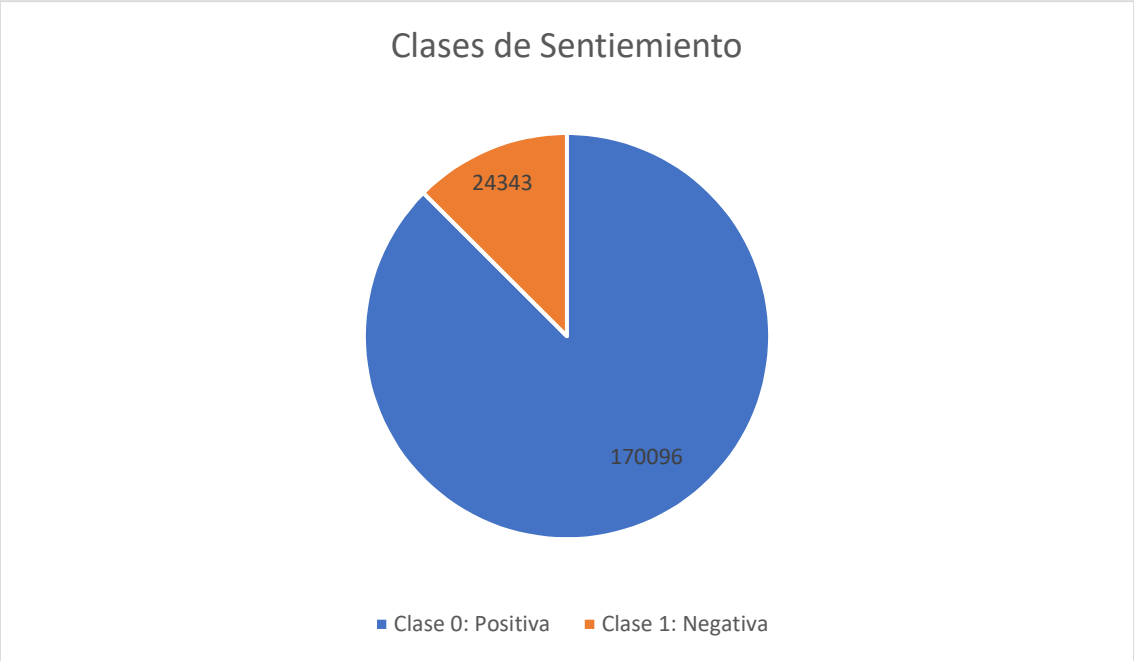

Figura 17: Gráfico que muestra la diferencia entre las clases positivas y negativas del dataset de Amazon.

4.4. De documentos a frases individuales

Implementamos una función que nos va a servir más adelante para conseguir de nuestros datasets las frases individuales de cada documento y agruparlas en una lista.

Para poder dividir los documentos en frases individuales, implementamos una función que sea capaz de reconocer los distintos conectores léxicos que consideramos que marcan el final de una frase. Por lo tanto, esta función recibirá como parámetro de entrada el documento y nos devolverá una lista cuyas posiciones corresponden a las frases encontradas en el documento. Para poder adquirir esta funcionalidad, utilizamos una librería de Python llamada re, la cual dispone una función llamada Split que permite dividir cadenas de texto. Escogemos esta librería, ya que permite dividir de manera más simple las cadenas de texto y además permite seleccionar varios elementos por los que debe dividir el texto.

El final de una frase viene decantado por la presencia de algunos símbolos, como por ejemplo un punto o la presencia de símbolos de exclamación e interrogación. Por lo tanto, nos orientamos según esta metodología, pero teniendo en cuenta también la escritura de los foros web. Dentro de los foros, no todo el mundo emplea correctamente los signos de puntuación o exclamación ya que se suele emplear de una manera más amplia el lenguaje urbano. Para el caso de los símbolos de puntuación, siempre que una frase acaba viene seguida de un punto y un espacio (por ejemplo: “Hola, estoy bien. Hoy hace buen día.”), para el caso de las interrogaciones y exclamaciones incluimos también el espacio en el delimitador, ya que una persona en una reseña puede escribir un mensaje de manera similar a *“what???? How?”*. En este estilo de mensajes sólo nos interesa coger el último símbolo de interrogación. Para estos casos, simplemente tendremos en cuenta los símbolos de cierre, ya que los textos se encuentran en inglés. El último separador que definimos es el salto de línea, ya que es una forma clara de separar diferentes párrafos.

De este modo, a la salida de la función habremos obtenido las instancias divididas por instancias individuales, y se aplica a los datasets de IMDB y de Amazon que contienen los grupos de instancias.

4.5. Extracción de características

Una vez hecho un tratamiento de los datos, utilizamos técnicas de BoW y comprobar que estilo de palabras es capáz de aprender de estos documentos. De esta manera, como salida obtendremos una representación vectorial de las palabras de cada documento como se ha definido previamente. Para ello vamos a emplear la librería de Python de Sklearn.feature_extraction.text, librería especializada en el tratamiento y extracción de información de los textos de Sklearn.

Dentro de esta librería, vamos a emplear la función de CountVectorizer. Esta función permite realizar la representación vectorial de BoW empleando el método de conteo de palabras.

Gracias a este método, podemos obtener la representación vectorial buscada para el estudio del modelo.

4.5.1. Empleo de lematización

Para emplear la lematización vamos a importar de la librería de nltk.stem la función de WordNetLemmatizer y de nltk importamos la función de Word_tokenize.

Para poder realizar la lematización, implementamos una función la cual recorre una frase o documento palabra a palabra, y se encarga de obtener el lema de la palabra. De este modo, como argumento de entrada tenemos la frase que se quiere lematizar y como argumento de salida tenemos la frase lematizada.

4.5.2. Empleo de técnicas BoW

Una vez realizada la limpieza de datos realizamos la transformación de datos alternando un parámetro que permite únicamente aprender palabras ignorando todos aquellos casos donde la frecuencia de aparición es menor que el umbral que se pasa como argumento. De este modo, conseguimos deshacernos de todas aquellas instancias que tienen una única aparición en un documento del corpus, o en su minoría.

Una vez realizada la transformación vectorial al modelo de BoW, realizamos la transformación TF-IDF. Para realizar esta transformación empleamos el método TfidfTransformer() de la librería de sklearn.feature_extraction.

4.5.3. Problemas encontrados

Al visualizar las palabras que ha aprendido empleando el método de CountVectorizer sin pasar ningún parámetro adicional, nos damos cuenta de que había muchos símbolos, números y palabras sin coherencia alguna. Dentro de la Figura 18: Palabras aprendidas por el método CountVectorizer sin parámetros adicionales. podemos observar muchas palabras que se han

fusionado con diferentes letras, pero sintácticamente no parecen representar a ninguna palabra, o simplemente no aportan información útil.

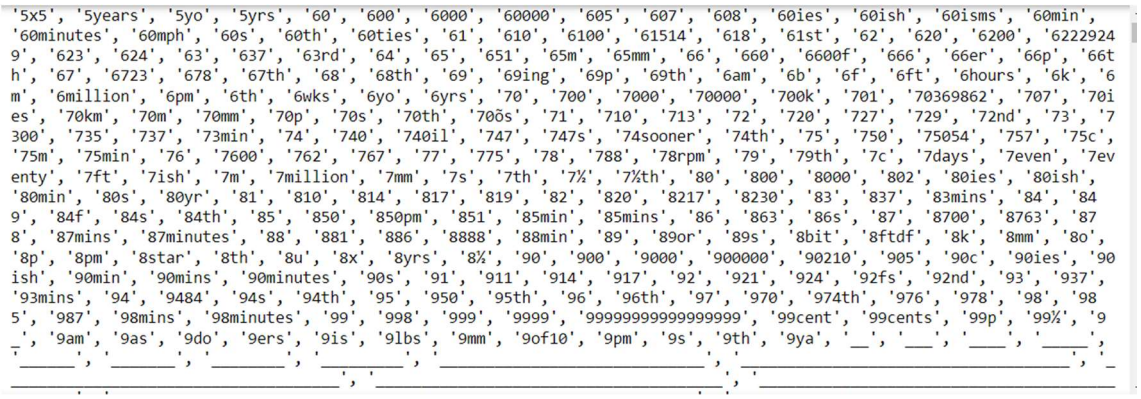

Figura 18: Palabras aprendidas por el método CountVectorizer sin parámetros adicionales.

Al investigar el origen de estas palabras, encontramos que en algunas reseñas el usuario que publicó la reseña adjunto en su mensaje un enlace a una página web externa (enlaces del estilo http o https). Por lo tanto, al realizar el aprendizaje de las diferentes palabras también extrae las características presentes en este estilo de enlaces, y por esa razón algunos de los símbolos indeseados. Cómo estos enlaces no nos aportan información alguna a la hora de clasificar si una frase es positiva o negativa, vamos a eliminar todos los posibles enlaces presentes en los corpus. También nos damos cuenta, de que el dataset de IMDB contiene en la mayor parte de sus instancias la siguiente cadena “
”. Buscando el origen de esta palabra, llegamos a la conclusión de que al ser este dataset obtenido con *BeautifulSoup* que extrae las características de HTML, ha implementado los saltos de línea en formato de HTML. Por lo tanto, vamos a sustituir todas estas apariciones por un salto de línea “\n”. Para eliminar los enlaces, simplemente vamos a sustituir el contenido del enlace por un contenido vacío, “”.

Como se puede observar en la Figura 18: Palabras aprendidas por el método CountVectorizer sin parámetros adicionales., todavía hay características que no queremos en nuestro aprendizaje que según las eliminaciones aplicadas deberían persistir, y la mayor parte de ellas constan de la presencia de un número. Por ello, eliminamos cualquier número presente en el texto, ya que no aportan gran información y del mismo modo podemos eliminar las palabras que se encuentren adheridas a algún número.

4.5.4. Aparición de palabras poco frecuentes

Una vez que realizamos la limpieza del dataset, volvemos a realizar la extracción de características del texto para comprobar los cambios realizados en el aprendizaje, y comprobar si la eliminación de características que empleamos se ha aplicado correctamente. A pesar de la limpieza de los datos que aplicamos, seguimos teniendo ciertas anomalías en algunas de las palabras, pero la gran mayoría se han eliminado.

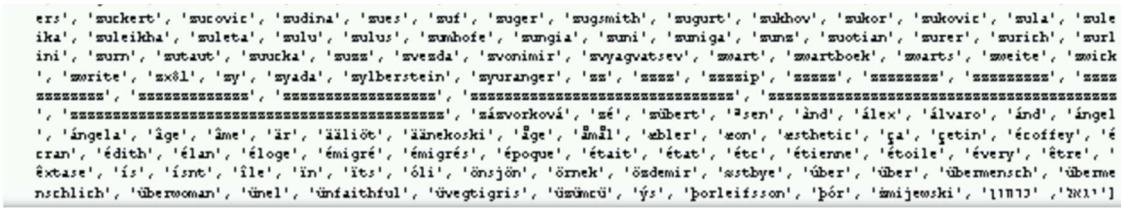

Figura 19: Anomalías detectadas tras haber eliminado los URL y cadenas con contenidos numéricos.

Analizando estas palabras, muchas de ellas contienen acentos, diéresis, o incluso son palabras que tienen otro estilo de escritura porque se encuentran en otro idioma o también encontramos cadenas de texto con caracteres repetitivos. Como estas palabras pueden presentar una minoría de los textos, es decir que únicamente van a aparecer una vez en todo el corpus o simplemente de manera minoritaria, no vamos a eliminarlas, sino que vamos a introducir parámetros en el constructor del corpus para que simplemente omita el estudio de esas palabras.

4.6. Validación de modelos

Para poder realizar una separación de los datasets en conjunto de entrenamiento y test, vamos a emplear la función de train_test_split de la librería Sklearn. Para la división de los datasets, empleamos un 80% de las instancias para el conjunto de entrenamiento y un 20% de los datos han ido al conjunto de prueba. Para realizar esta división empleamos técnicas de estratificación para evitar que los dataset queden desbalanceados.

4.7. Implementación de la función de coste y optimización de los parámetros

Para poder aplicar la función de coste, creamos una función que se encargue de recrear la función definida en el apartado **¡Error! No se encuentra el origen de la referencia.**. El primer paso por dar ha sido identificar los parámetros de entrada que le daremos a nuestra función.

Vamos a pasar un total de cuatro parámetros de entrada, los valores theta, el contenido de los grupos de instancias transformados en BoW, la matriz de instancias individuales correspondientes al parámetro anterior, las etiquetas de cada documento y por último el parámetro de lambda. Una vez definidos los parámetros con los que vamos a trabajar dentro de la función vamos a comenzar con su implementación la cual dividimos en dos subsecciones, el primer argumento y el segundo.

En el primer argumento nos encargamos de aplicar la norma euclídea para cada elemento de los grupos de instancias y dicho resultado multiplicarlo por la diferencia al cuadrado de aplicar la función sigmoide para cada elemento de los grupos de instancias (cada elemento corresponde a un elemento i y j, dónde $i, j \in N$).

En el caso del segundo argumento, el bucle externo se encarga de recorrer toda la matriz de documentos mientras que el bucle interno se encarga de recorrer las instancias individuales presentes en cada documento. Dentro del bucle interno se aplica la función sigmoide y al

finalizar el bucle, se realiza la media de dicho sumatorio y se le resta el valor real de la etiqueta correspondiente a dicho documento y el resultado se eleva al cuadrado. Dicho valor queda envuelto por el sumatorio externo.

Una vez finalizan todos los bucles, aplicamos la suma del primer argumento y el segundo argumento y el resultado de dicha suma corresponderá al coste y se calcula de la siguiente forma

$$coste = \frac{1}{N^2} primerArgumento + \frac{\lambda}{K} segundoArgumento$$

Ecuación 20: Operación realizada al finalizar los dos sumatorios de cada argumento.

Para poder realizar una optimización de la función de coste mediante descenso de gradiente, empleamos la función minimize de la librería de scipy.optimize. A dicha función de coste le pasamos como argumentos las thetas a ser optimizadas y la llamada a nuestra función de coste con sus parámetros correspondientes. También se puede introducir un valor adicional indicando el número de iteraciones de descenso de gradiente que puede realizar en la optimización.

4.8. Modificaciones de la función de coste

Para la realización de algunas de las pruebas vamos a emplear una serie de modificaciones en la función de coste, modificando el valor del segundo argumento.

Una de las modificaciones consiste en aplicar el máximo de las probabilidades que, en este caso, en vez de realizar la media con las probabilidades nos quedamos con la probabilidad máxima.

$$J(\theta) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} e^{(-||x_i - x_j||^2)} (\sigma(\theta^T x_i) - \sigma(\theta^T x_j))^2 + \frac{\lambda}{K}\sum_{k=1}^{K} (\max_{i \in G_k}(\sigma(\theta^T x_i)) - l_k)^2$$

Ecuación 21: Función de coste específica desarrollada modificada para el máximo.

Otra de las modificaciones será la función de voto, donde nos quedamos con la clase que mayor aparición haya tenido dentro de las probabilidades.

$$J(\theta) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} e^{(-||x_i - x_j||^2)} (\sigma(\theta^T x_i) - \sigma(\theta^T x_j))^2 + \frac{\lambda}{K}\sum_{k=1}^{K} (\max\_\text{clase}_{i \in G_k}(\sigma(\theta^T x_i)) - l_k)^2$$

Ecuación 22: Función de coste específica desarrollada modificada para el voto.

Por último, vamos a emplear una función de agregación OWA. Una función OWA es un método donde a cada probabilidad se le asigna un peso, de este modo se multiplica cada probabilidad por dicho peso, donde las probabilidades se encuentran ordenadas de mayor a menor. Sea *w* el conjunto de pesos donde la suma de todos los pesos debe ser equivalente a 1, podemos definir la función OWA como

$$OWA(b_1, \dots, b_n) = \sum_{i=1}^{n} w_i * b_{(i)}$$

Ecuación 23: Función OWA donde w_i equivale a cada peso y b_i equivale a cada probabilidad, donde $b_{(i)}$ indica la i-ésima mayor probabilidad.

La función de agregación aplicada para OWA consiste en multiplicar probabilidades por unos pesos distribuidos, en nuestro caso empleamos una distribución del 25%. Escogemos este valor, ya que cuando tenemos una probabilidad muy baja o una muy alta nos es de gran interés ponderar esas probabilidades, ya que esta o asignando mal la clase o esta asignándola de manera muy certera. Por ello, mediante la distribución del 25% asignamos los pesos a las probabilidades más altas y bajas (por ejemplo: una distribución para un total de 6 probabilidades la asignación de pesos sería de la siguiente manera (1,1,0,0,1,1) donde la asignación de pesos sería equivalente a (0.25,0.25,0,0,0.25,0.25) = 1)

$$J(\theta) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} e^{(-||x_i - x_j||^2)}(\sigma(\theta^T x_i) - \sigma(\theta^T x_j))^2 + \frac{\lambda}{K}\sum_{k=1}^{K}\frac{1}{|G_k|}\sum_{i \in G_k}(\sigma(\theta^T x_{(i)}) * \omega_i - l_k)^2$$

Ecuación 24: Función de coste específica desarrollada modificada para OWA.

Además, la función de agregación que se empleara en las pruebas para calcular las predicciones de grupo a partir de instancias individuales cambiará dependiendo de la función que se vaya a emplear.

5. Experimentación y resultados

5.1. Diseño de la experimentación

En esta sección definimos las pruebas que deseamos hacer, en donde todos los modelos de entrenamiento contienen un total de 100 instancias y ambos modelos contienen un total de palabras aprendidas inferior a 200. Como conjunto de validación tenemos 25 instancias balanceadas para cada dataset, por lo tanto, hemos seleccionado una validación cruzada de un 80%. Esta selección de valores se ha escogido de esta manera, debido al alto coste computacional que hemos ido encontrando.

Para la experimentación, distinguimos 16 configuraciones.

1. Entrenamiento con el dataset de IMDB empleando la media aritmética.

2. Entrenamiento con el dataset de Amazon empleando la media aritmética.

3. Entrenamiento con el dataset de IMDB modificando la función de coste por el máximo de las probabilidades.

4. Entrenamiento con el dataset de Amazon modificando la función de coste por el máximo de las probabilidades.

5. Entrenamiento con el dataset de IMDB modificando la función de coste por el voto de las probabilidades.

6. Entrenamiento con el dataset de Amazon modificando la función de coste por el voto de las probabilidades.

7. Entrenamiento con el dataset de IMDB modificando la función de coste por el aplicando una función OWA a las probabilidades.

8. Entrenamiento con el dataset de Amazon modificando la función de coste por el aplicando una función OWA a las probabilidades.

Además, dentro de cada prueba vamos a realizar una comparativa de la ejecución aplicando la transformación TF-IDF y sin aplicar dicha transformación. A su vez vamos a ir modificando los parámetros de lambda del segundo argumento de la función de coste para comprobar su influencia dentro de la función de coste, comprobando con distintos valores de lambda. Del mismo modo, vamos a ir modificando los diferentes umbrales de predicción. Por último, para realizar las predicciones vamos a emplear tres metodologías distintas, calculando mediante predicción de las instancias individuales de los datasets de Datasets de instancias individuales, calculando las predicciones de los grupos de instancias con los datasets de Datasets con grupos de instancias, y calculando la predicción de los grupos de instancias empleando los datasets de instancias individuales aplicando la función correspondiente (media, máximo, voto, OWA). Para esta última prueba, los grupos de instancias que se emplean corresponden a las instancias individuales de los datasets de grupos de instancias tras aplicar la función de división de frases que definimos anteriormente.

Para clarificar la distribución de estas pruebas, vamos a esquematizar la forma general

- Ejecuciones empleando transformación TF-IDF y sin emplear la transformación
    - Ejecuciones cambiando las lambdas y los umbrales.
        - Predicción de grupos de instancias mediante las instancias individuales aplicando la función correspondiente (media, máximo, voto, OWA).
        - Predicción de instancias individuales mediante los datasets de instancias individuales.

Para seleccionar el mejor umbral y el mejor valor de lambda, en predicciones de instancias individuales vamos a escoger el valor que maximice los tres valores de accuracy encontrados.

5.2. Resultados de las pruebas

Para poder organizar las pruebas y sus resultados vamos a representarlo mediante tablas. En cada tabla vamos a indicar a que prueba corresponden los resultados, junto al accuracy, precisión, recall, fscore y tiempo de ejecución.

5.2.1. Configuración 1: IMDB empleando la media aritmética.

Ejecución sin emplear transformación TF-IDF				
λ = 100 y umbral = 0.6 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	61,23	0,57	0,8	0,67
Instancias Amazon	52,4	0,52	0,72	0,6
Instancias Yelp	49,9	0,5	0,72	0,59
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0,65	0,98	0,78
Validación IMDB	52	0	0	0
Train Amazon	52	0,1	0,88	0,19
Validación Amazon	52	0	0	0
Tiempo de ejecución: 1.72 horas				

Tabla 2: tabla de resultados de IMDB con media aritmética sin emplear la transformación TF-IDF.

Para las instancias individuales se selecciona un umbral alto con una lambda alto, y en este caso podemos observar como los resultados para las instancias individuales ofrecen un accuracy superior a 50, a excepción del dataset de Yelp.

Ejecución empleando transformación TF-IDF				
λ = 10 y umbral = 0.5 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	60,83	0,63	0,46	0,53
Instancias Amazon	51,4	0,54	0,21	0,3
Instancias Yelp	50,6	0,51	0,26	0,34
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0,91	0,43	0,58
Validación IMDB	52	0	0	0
Train Amazon	52	0,14	0,12	0,13
Validación Amazon	52	0	0	0
Tiempo de ejecución: 2.27 horas				

Tabla 3: tabla de resultados de IMDB con media aritmética empleando la trasformación TF-IDF.

Para la selección del máximo accuracy ha resultado en un umbral alto. En este caso podemos observar como algunos accuracys disminuyen y otros aumentan comparándolo con la tabla anterior, pero de manera análoga podemos observar como las precisiones aumentan.

5.2.2. Configuración 2: Amazon empleando la media aritmética.

Ejecución sin emplear transformación TF-IDF				
λ = 0.1 y umbral = 0.6 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	55,35	0,53	0,73	0,61
Instancias Amazon	53,2	0,54	0,49	0,51
Instancias Yelp	50,1	0,5	0,69	0,58
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0,55	0,76	0,64
Validación IMDB	52	0	0	0
Train Amazon	52	0,14	0,25	0,18
Validación Amazon	52	0	0	0
Tiempo de ejecución: 1.96 horas				

Tabla 4: tabla de resultados para Amazon con media aritmética sin emplear la transformación TF-IDF.

Para la selección del máximo accuracy se selecciona un umbral alto. En este caso, a pesar de haberse seleccionado un umbral alto, la lambda seleccionada es más bajo.

Ejecución empleando transformación TF-IDF				
λ = 10 y umbral = 0.5 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	60,83	0,63	0,46	0,53
Instancias Amazon	51,4	0,54	0,21	0,3
Instancias Yelp	50,6	0,51	0,26	0,34
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0,91	0,43	0,58
Test IMDB	52	0	0	0
Train Amazon	52	0,14	0,12	0,13
Test Amazon	52	0	0	0
Tiempo de ejecución: 1.56 horas				

Tabla 5: tabla de resultados para Amazon con media aritmética empleando la transformación TF-IDF.

Para la selección del máximo accuracy se selecciona un umbral alto. En este caso, a pesar de haberse seleccionado un umbral alto, la lambda seleccionada es más bajo. A la hora de aplicar esta transformación TF-IDF, hemos podido observar como el accuracy aumenta frente a no usarlo, y del mismo modo lo hace la precisión.

5.2.3. Configuración 3: IMDB empleando máximo de las probabilidades.

Ejecución sin emplear transformación TF-IDF				
λ = 100 y umbral = 0.6 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	53,88	0,52	0,58	0,55
Instancias Amazon	53	0,52	0,9	0,66
Instancias Yelp	49,2	0,5	0,88	0,64
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	48	0,57	0,53	0,55
Validación IMDB	48	0,48	1	0,65
Train Amazon	52	0,09	0,88	0,16
Validación Amazon	52	0	0	0
Tiempo de ejecución: 1.78 horas				

Tabla 6: tabla de resultados para IMDB empleando máximo de las probabilidades sin emplear la transformación TF-IDF.

Podemos observar que para la selección del máximo accuracy se selecciona un umbral alto. Y un valor de lambda alto, pero no ofrece un buen ajuste para en las predicciones de grupos de instancias.

Ejecución empleando transformación TF-IDF				
λ = 100 y umbral = 0.5 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	52,81	0,54	0,17	0,26
Instancias Amazon	50,4	0,51	0,19	0,27
Instancias Yelp	52,1	0,55	0,23	0,32
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	1	0,04	0,08
Validación IMDB	52	0	0	0
Train Amazon	52	0,12	0,12	0,12
Validación Amazon	52	0	0	0
Tiempo de ejecución: 2.21 horas				

Tabla 7: tabla de resultados para IMDB empleando máximo de las probabilidades empleando la transformación TF-IDF.

Podemos observar que se ha seleccionado un valor alto de lambda. También podemos observar un comportamiento en donde la transformación TF-IDF no resulta efectiva.

5.2.4. Configuración 4: Amazon empleando máximo de las probabilidades.

Ejecución sin emplear transformación TF-IDF				
λ = 10 y umbral = 0.6 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	54,68	0,52	0,72	0,6
Instancias Amazon	53,6	0,54	0,49	0,51
Instancias Yelp	50,3	0,5	0,69	0,58
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0,54	0,78	0,64
Validación IMDB	52	0	0	0
Train Amazon	52	0,06	0,12	0,08
Validación Amazon	52	0	0	0
Tiempo de ejecución: 2.03 horas				

Tabla 8: tabla de resultados para la Amazon empleando máximo de las probabilidades sin emplear la transformación TF-IDF.

Para la selección del máximo accuracy se selecciona un umbral alto. En este caso, podemos observar que el accuracy para Amazon es superior al de IMDB.

Ejecución empleando transformación TF-IDF				
λ = 10 y umbral = 0.6 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	52,94	0,51	0,8	0,62
Instancias Amazon	52,8	0,53	0,58	0,55
Instancias Yelp	49,8	0,5	0,76	0,6
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0,52	0,9	0,66
Validación IMDB	52	0	0	0
Train Amazon	48	0,09	0,38	0,14
Validación Amazon	48	0,48	1	0,65
Tiempo de ejecución: 1.92 horas				

Tabla 9: tabla de resultados para empleando máximo de las probabilidades empleando la transformación TF-IDF.

Para la selección del máximo accuracy se selecciona un umbral alto. También sucede lo mismo que en el conjunto de entrenamiento de IDMB, donde la precisión y el accuracy disminuyen empleando la transformación TF-IDF.

5.2.5. Configuración 5: IMDB empleando el voto de las probabilidades.

Ejecución sin emplear transformación TF-IDF				
λ = 0.1 y umbral = 0.1 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	51,6	0	0	0
Instancias Amazon	50	0	0	0
Instancias Yelp	50	0	0	0
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0	0	0
Validación IMDB	52	0	0	0
Train Amazon	48	0	0	0
Validación Amazon	48	0	0	0
Tiempo de ejecución: 1.93 horas				

Tabla 10: tabla de resultados para IMDB empleando el voto de las probabilidades sin emplear la transformación TF-IDF.

En este caso, podemos observar un comportamiento inusual. Se han seleccionado los mismos umbrales y los mismos valores de lambda, y todos los resultados tienen 0 recall. Debido a este suceso y al comportamiento observado en el entrenamiento de las thetas, hemos podido observar que la función de agregación junto al modelo no ofrece un buen ajuste.

Ejecución empleando transformación TF-IDF				
λ = 0.1 y umbral = 0.1 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	51,6	0	0	0
Instancias Amazon	50	0	0	0
Instancias Yelp	50	0	0	0
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0	0	0
Validación IMDB	52	0	0	0
Train Amazon	48	0	0	0
Validación Amazon	48	0	0	0
Tiempo de ejecución: 1.97 horas				

Tabla 11: tabla de resultados para IMDB empleando el voto de las probabilidades empleando la transformación TF-IDF.

En este caso, los valores del umbral y lambda coinciden con la tabla anterior. Del mismo modo lo hacen los resultados, como el ajuste de las thetas y, por lo tanto, podemos establecer la misma conclusión.

5.2.6. Configuración 6: Amazon empleando el voto de las probabilidades.

Ejecución sin emplear transformación TF-IDF				
λ = 0.1 y umbral = 0.1 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	51,6	0	0	0
Instancias Amazon	50	0	0	0
Instancias Yelp	50	0	0	0
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0	0	0
Validación IMDB	52	0	0	0
Train Amazon	48	0	0	0
Validación Amazon	48	0	0	0
Tiempo de ejecución: 2.03 horas				

Tabla 12: tabla de resultados para Amazon empleando el voto de las probabilidades sin emplear la transformación TF-IDF.

Al igual que ha sucedido para el entrenamiento de IMDB, podemos observar el mismo comportamiento en el entrenamiento de Amazon para la función de agregación de voto.

Ejecución empleando transformación TF-IDF				
λ = 0.1 y umbral = 0.1 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	51,6	0	0	0
Instancias Amazon	50	0	0	0
Instancias Yelp	50	0	0	0
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	52	0	0	0
Validación IMDB	52	0	0	0
Train Amazon	52	0	0	0
Validación Amazon	52	0	0	0
Tiempo de ejecución: 2.01 horas				

Tabla 13: tabla de resultados para Amazon empleando el voto de las probabilidades empleando la transformación TF-IDF.

En este caso, podemos observar que no se ha obtenido un buen ajuste, ya que todas las predicciones los realiza de la clase negativa.

5.2.7. Configuración 7: IMDB empleando función OWA.

Ejecución sin emplear transformación TF-IDF				
λ = 0.1 y umbral = 0.7 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	52,94	0,51	0,51	0,51
Instancias Amazon	53,6	0,52	0,82	0,64
Instancias Yelp	49	0,49	0,77	0,6
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	48	0,49	1	0,66
Validación IMDB	48	0,49	1	0,65
Train Amazon	48	0,08	1	0,15
Validación Amazon	48	0,48	1	0,65
Tiempo de ejecución: 2.12 horas				

Tabla 14: tabla de resultados para IMDB empleando función OWA sin emplear la transformación TF-IDF.

Para la selección del máximo accuracy se selecciona un umbral alto. En este caso, se ha seleccionado un umbral alto y una lambda de valor bajo.

Ejecución empleando transformación TF-IDF				
λ = 300.0 y umbral = 0.5 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	54,68	0,63	0,15	0,25
Instancias Amazon	50,8	0,53	0,14	0,22
Instancias Yelp	50,5	0,52	0,14	0,23
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	48	0,49	1	0,66
Validación IMDB	48	0,48	1	0,65
Train Amazon	48	0,08	1	0,15
Validación Amazon	48	0,48	1	0,65
Tiempo de ejecución: 2.07 horas				

Tabla 15: tabla de resultados para IMDB empleando función OWA empleando la transformación TF-IDF.

Podemos observar que se ha escogido un umbral alto. En la selección del máximo accuracy de instancias individuales, el máximo accuracy encontrado es superior que a no aplicar TF-IDF, al igual que su precisión y accuracy.

5.2.8. Configuración 8: Amazon empleando función OWA.

Ejecución sin emplear transformación TF-IDF				
λ = 10 y umbral = 0.6 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	54,81	0,52	0,84	0,64
Instancias Amazon	53,3	0,53	0,59	0,56
Instancias Yelp	49,5	0,5	0,77	0,6
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	48	0,49	1	0,66
Validación IMDB	48	0,49	1	0,65
Train Amazon	48	0,08	1	0,15
Validación Amazon	48	0,48	1	0,65
Tiempo de ejecución: 2.25 horas				

Tabla 16: tabla de resultados para Amazon empleando función OWA sin emplear la transformación TF-IDF.

En este caso, en la selección del máximo accuracy de instancias individuales podemos observar que los resultados son algo inferiores frente al entrenamiento de IMDB.

Ejecución empleando transformación TF-IDF				
λ = 0.1 y umbral = 0.8 mediante selección del máximo accuracy de instancias individuales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	48,4	0,48	1	0,65
Instancias Amazon	50,5	0,5	0,96	0,66
Instancias Yelp	50	0,5	1	0,67
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	48	0,49	1	0,66
Validación IMDB	48	0,48	1	0,65
Train Amazon	48	0,08	1	0,15
Validación Amazon	48	0,48	1	0,65
Tiempo de ejecución: 2.29 horas				

Tabla 17: tabla de resultados para Amazon empleando función OWA empleando la transformación TF-IDF.

En este caso, podemos observar cómo disminuye de manera general el accuracy y precisión mediante el empleo de técnicas de TF-IDF con respecto a no aplicarlas.

5.3. Pruebas adicionales

Como pruebas adicionales, para poder comprobar el rendimiento de nuestro modelo frente a otros aprendizajes, implementamos unas pruebas que realizan el aprendizaje de un modelo de redes neuronales y otro de Naive Bayes. Ambos modelos los entrenamos mediante GridSearch, función que se encarga de encontrar los mejores parámetros entre los candidatos que establezcamos para el aprendizaje de un modelo mediante el uso de validación cruzada.

Como los mejores resultados observados han sido con el entrenamiento del dataset de IMDB, ya que ha proporcionado un mejor ajuste, vamos a realizar las ejecuciones con dicho conjunto de entrenamiento. Para el estudio de estas técnicas, ya que son más rápidas que el modelo, vamos a emplear todo el dataset de IMDB con una validación cruzada del 80%.

5.3.1. Ejecución con un modelo de Naive Bayes para IMDB

Ejecución empleando metodología de naive bayes				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	53,20	0,64	0,48	0,59
Instancias Amazon	59,76	0,54	0,43	0,52
Instancias Yelp	48,00	0,47	0,36	0,47
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	74,17	0,74	0,73	0,74
Validación IMDB	73,60	0,74	0,73	0,73
Train Amazon	63,99	0,64	0,91	0,52
Validación Amazon	65,74	0,67	0,91	0,53
Tiempo de ejecución: 9.46 segundos				

Tabla 18: Tabla para el entrenamiento de IMDB mediante Naive Bayes

Podemos observar que el aprendizaje es mucho más rápido, pero el mayor de los Fscore para las instancias individuales no supera el alcanzado por nuestro método. Por el contrario, podemos observar como la precisión para las instancias individuales aumenta mientras que el recall disminuye. Los resultados mediante gridsearch aplican TF-IDF.

5.3.2. Ejecución con un modelo de redes neuronales para IMDB

Ejecución empleando metodología de redes neuronales				
	Accuracy	**Precisión**	**Recall**	**Fscore**
PREDICCIONES DE INSTANCIAS INDIVIDUALES				
Instancias IMDB	52,50	0,6	0,71	0,6
Instancias Amazon	61,50	0,51	0,8	0,48
Instancias Yelp	50,30	0,5	0,76	0,46
PREDICCIONES GRUPOS INSTANCIAS MEDIANTE INSTANCIAS INDIVIDUALES				
Train IMDB	75,09	0,76	0,74	0,75
Validación IMDB	74,64	0,76	0,73	0,74
Train Amazon	70,78	0.74	0,9	0,55
Validación Amazon	72,63	0,86	0,90	0,56
Tiempo de ejecución: 30 minutos				

Tabla 19: Tabla para el entrenamiento de IMDB mediante Redes Neuronales

Podemos observar que el aprendizaje es mucho más rápido, y los resultados frente accuracy para las instancias individuales son similares. En la selección de grupos de instancias, podemos observar cómo el accuracy y precisión aumentan.

6. CONCLUSIONES

Una de las principales conclusiones que hemos llegado con este estudio, es que la metodología empleada en [2] es mucho más efectiva comparando los resultados que hemos obtenido con los que se muestran en el documento. Por lo tanto, hemos podido concluir que el método de extracción de características de BoW no resulta tan efectivo como el método embebido.

Por otro lado, comprobando los valores obtenidos para umbrales bajos, hemos podido comprobar que los resultados aportados no son muy buenos, ya que tiende a predecir todos los elementos de una única clase. Del mismo modo sucede con los umbrales más altos.

De manera general, a excepción de las pruebas de Amazon para OWA y máximo, hemos podido comprobar que la transformación TF-IDF hace que los resultados mejoren levemente. Hemos podido analizar que a pesar de que igual el Fscore no sea más alto en estos casos frente a no emplearlo, tanto el accuracy como la precisión si lo son de manera general.

En cuanto a las pruebas, hemos concluido de manera general que la función de agregación de voto de probabilidades no funciona bien, ya que tanto para los resultados como para el ajuste de las thetas no ha ofrecido buenos resultados. En el ajuste de las thetas, para todos los casos de descenso de gradiente, los valores eran 0.

Por otro lado, hemos concluido que las pruebas que mejores resultados han aportado han sido para el dataset de IMDB con media aritmética empleando la transformación IMDB. En este caso, al no aplicar la transformación TF-IDF el máximo accuracy es superior, pero por el contrario el accuracy y la precisión disminuyen de manera general. Además, mediante la selección del máximo accuracy de instancias individuales, hemos podido obtener un mejor ajuste para el dataset de Amazon y Yelp en cuanto al accuracy y precisión.

Además, hemos podido observar que los resultados proporcionados por Naive Bayes y Redes Neuronales son ligeramente inferiores en cuanto al accuracy de instancias individuales, pero por el contrario podemos ver que los resultados de precisión son muy similares a los nuestros. De manera contraria, podemos observar que, en los grupos de instancias, los resultados son bastante mejores que empleando la regresión logística. Esto puede estar causado a que el dataset de estudio es más amplio. A su vez, en ambos casos coincide que la técnica empleada para la extracción de características emplea la transformación TF-IDF.

A pesar de no haber obtenido unos buenos resultados, hemos podido observar el comportamiento y el cambio que tiene el modelo ante los diferentes valores de lambda y los diferentes umbrales. Por lo tanto, podemos observar que dependiendo de cómo se realice el aprendizaje del modelo, puede desviarse a una predicción unilateral de clases, llegando a predecir una única clase.

7. LÍNEAS FUTURAS

Una de las principales líneas futuras que se podrían implementar sería ampliar el número de instancias de entrenamiento y test para realizar el estudio, ya que el coste computacional de este método es bastante alto.

Como se ha podido comprobar, nuestro método no ha resultado de los más efectivos, ya que el procedimiento de [2] aporta mejores resultados. Pero del mismo modo, se podría continuar la investigación sustituyendo por ejemplo el modelo de extracción de características. De este modo, se podrían emplear diferentes modelos de extracción de características como Word2Vec o FastText. [29] [30].

Del mismo modo, una de las posibles líneas futuras sería comprobar el rendimiento de nuestras diferentes funciones de agregación junto a la metodología de extracción de características empleado en [2].

Bibliografía

[1] A. Reshamwala, D. Mishra, and P. Pawar, “REVIEW ON NATURAL LANGUAGE PROCESSING Precision Agriculture View project Image retrieval View project Alpa Reshamwala Narsee Monjee Institute of Management Studies REVIEW ON NATURAL LANGUAGE PROCESSING,” 2013. [Online]. Available: https://www.researchgate.net/publication/235788362

[2] D. Kotzias, M. Denil, N. de Freitas, and P. Smyth, “From group to individual labels using deep features,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2015, vol. 2015-August, pp. 597–606. doi: 10.1145/2783258.2783380.

[3] A. Abeliuk and C. Gutiérrez, “Inteligencia Artificial El primer programa de IA.”

[4] “Beautiful Soup: We called him Tortoise because he taught us.” https://www.crummy.com/software/BeautifulSoup/ (accessed Apr. 24, 2022).

[5] “Stemming-Lematización”, [Online]. Available: http://www.kramirez.net/RI/Material/Presentaciones/Stemming.pdf

[6] F. Schwendinger, B. Grün, and K. Hornik, “A comparison of optimization solvers for log binomial regression including conic programming,” *Computational Statistics*, vol. 36, no. 3, pp. 1721–1754, Sep. 2021, doi: 10.1007/S00180-021-01084-5/FIGURES/5.

[7] A. Mccallum and K. Nigam, “A Comparison of Event Models for Naive Bayes Text Classification”.

[8] “Función de activación - Redes neuronales - Diego Calvo.” https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/ (accessed May 01, 2022).

[9] Nils Weidmann, “Nils Weidmann, 2003,” 2003

[10] J. Foulds and E. Frank, “A Review of Multi-Instance Learning Assumptions”, doi: 10.1017/S000000000000000.

[11] “Máquinas de Vector Soporte (Support Vector Machines, SVMs).” https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines (accessed May 08, 2022).

[12] A. Leclair, Z. Eberhart, and C. Mcmillan, “Adapting Neural Text Classification for Improved Software Categorization.”

[13] Ó. Alberto García Pérez and I. Martínez Fernández, “CLASIFICADOR LINGÜÍSTICO DE TEXTOS EN JAVA”.

[14] “Text Analytics Toolbox - MATLAB.” https://www.mathworks.com/products/text-analytics.html (accessed May 04, 2022).

[15] “¿Qué es AWS?” https://aws.amazon.com/es/what-is-aws/ (accessed May 08, 2022).

[16] “Servicios de informática en la nube | Microsoft Azure.” https://azure.microsoft.com/es-es/ (accessed May 22, 2022).

[17] “Te damos la bienvenida a Colaboratory - Colaboratory.” https://colab.research.google.com/?hl=es (accessed May 22, 2022).

[18] “NumPy.” https://numpy.org/ (accessed May 04, 2022).

[19] “pandas - Python Data Analysis Library.” https://pandas.pydata.org/ (accessed May 04, 2022).

[20] “SciPy.” https://scipy.org/ (accessed May 04, 2022).

[21] “scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation.” https://scikit-learn.org/stable/ (accessed May 04, 2022).

[22] “NLTK :: Natural Language Toolkit.” https://www.nltk.org/ (accessed May 04, 2022).

[23] “seaborn: statistical data visualization — seaborn 0.11.2 documentation.” https://seaborn.pydata.org/ (accessed May 04, 2022).

[24] “Matplotlib documentation — Matplotlib 3.5.2 documentation.” https://matplotlib.org/stable/index.html (accessed May 04, 2022).

[25] “re — Regular expression operations — Python 3.10.4 documentation.” https://docs.python.org/3/library/re.html (accessed May 04, 2022).

[26] “time — Time access and conversions — Python 3.10.4 documentation.” https://docs.python.org/3/library/time.html (accessed May 04, 2022).

[27] “IMDB Dataset of 50K Movie Reviews | Kaggle.” https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews (accessed May 04, 2022).

[28] “Amazon review data.” http://jmcauley.ucsd.edu/data/amazon/index_2014.html (accessed May 04, 2022).

[29] “Word2Vec.” https://radimrehurek.com/gensim/models/word2vec.html (accessed June 06, 2022).

[30] “FastText.” https://fasttext.cc/docs/en/python-module.html (accessed June 06, 2022).