TÉCNICAS DE MINERÍA DE TEXTO Y SUS PRINCIPALES APLICACIONES.

TRABAJO FIN DE MÁSTER (TFM).

FACULTAD DE CIENCIAS ECONÓMICAS Y EMPRESARIALES.

MÁSTER EN EMPRESAS Y TÉCNOLOGÍAS DE LA INFROMACIÓN.

AUTOR: ANDREA RODRÍGUEZ GUTIÉRREZ

DIRECTORES DEL TFM: ÁNGEL COBO ORTEGA
ELIANA ROCÍO ROCHA BLANCO

PRESENTADO EN SEPTIEMBRE DEL 2023

ÍNDICE DE CONTENIDO.

1. INTRODUCCIÓN. ---- 6

2. DESCRIPCIÓN DE LOS DATOS Y BIBLIOTECAS PRINCIPALES. ---- 7

3. CONTEXTO HISTÓRICO. ---- 8

4. CONSIDERACIONES DE CALIDAD Y VALIDEZ. ---- 10

4.1. MATRIZ DE CONFUSIÓN. ---- 10
4.1.1. Precisión. ---- 11
4.1.2. Sensibilidad o Recall. ---- 11
4.1.3. F1-Score. ---- 11
4.2. CURVA DE ROC. ---- 12
4.3. CALIDAD - APLICADA AL *CORPUS*. ---- 12

5. PREPROCESAMIENTO DEL LENGUAJE NATURAL. ---- 13

5.1. PREPARACIÓN DEL *CORPUS* - LIMPIEZA ---- 13
5.2. TOKENIZACIÓN. ---- 14
5.2.1. Manejo de los signos de puntuación. ---- 14
5.2.2. Stop words. ---- 14
5.2.3. Minúsculas. ---- 15
5.3. LEMATIZACIÓN. ---- 15
5.4. STEMMING. ---- 15
5.5. EJEMPLO PRÁCTICO - PREPROCESAMIENTO DEL *CORPUS*. ---- 16

6. MODELOS DE TEMAS - *LATENT DIRICHLET ALLOCATION.* ---- 17

7. EJEMPLO DE VISUALIZACIÓN – NUBES DE PALABRAS. ---- 18

8. PRINCIPALES APLICACIONES DE LA MINERÍA DE TEXTO. ---- 19

8.1. SISTEMAS DE CLASIFICACIÓN DE DOCUMENTOS. ---- 20
8.1.1. Problemas de clasificación binaria. ---- 20
8.1.2. Problemas de clasificación multi-clase (Multiclass Clasification, MCC). ---- 22
8.1.3. Problemas de clasificación multi-etiqueta (Multilabel Clasification, MLC). ---- 23
8.1.4. Problemas de clasificación difusa (Fuzzy). ---- 24
8.1.5. EJEMPLO DE CLASIFICACIÓN BINARIA – Películas Románticas / No románticas. ---- 26
8.2. ELABORACIÓN DE RESUMENES. ---- 27
8.2.1. Resúmenes basados en extracción de contenido. ---- 28
8.2.2. Resúmenes basados en abstracción de contenido. ---- 29
8.3. MINERÍA DE OPINIONES / ANÁLISIS DE SENTIMIENTOS. ---- 30
8.3.1. Sentimiento construido por herramientas de machine learning. ---- 30
8.3.2. Algoritmo de redes neuronales – Perceptrón multicapa ---- 31
8.3.3. Sentimiento construido por medio de diccionarios lexicográficos - SentiWordNet ---- 32
8.3.4. EJEMPLO 1. ANÁLISIS DE SENTIMIENTOS – Asignación automática de polaridad ---- 33
8.3.5. EJEMPLO 2. ANÁLSIS DE SENTIEMINTOS – Redes neuronales – Perceptrón. ---- 34

9. IMPLICACIONES ÉTICAS Y DE PRIVACIDAD. ---- 36

10. CONCLUSIONES. ---- 37

11. BIBLIOGRAFÍA. ---- 39

ANEXO I: ---- 49

ANEXO II ---- 54

ANEXO III ---- 55

ANEXO IV ---- 57

ANEXO V ---- 58

ÍNDICE DE ILUSTRACIONES.

Ilustración 1. Principales fases del preprocesamiento de textos. 13
Ilustración 2. Modelo gráfico para la asignación latente de Dirichlet (LDA) 17
Ilustración 3. Representación del *corpus* "Bag of Words" 18
Ilustración 4. Diagrama de Venn. Aplicaciones de la minería de texto. 19
Ilustración 5. SVM: Los hiperplanos de separación y el hiperplano óptimo. 21
Ilustración 6. Diferencia entre clasificación y clustering. 24
Ilustración 7. Informe de clasificación. 26
Ilustración 8. Representación de la curva de ROC. 27
Ilustración 9. Árbol de dependencias. 28
Ilustración 10. Arquitectura interna del modelo Seq2Seq 29
Ilustración 11. Rueda de las emociones de Plutchik. 30
Ilustración 12. Ilustración de un perceptrón red neuronal. 31
Ilustración 13. Matriz de confusión. 35

ÍNDICE DE TABLAS.

Tabla 1. Matriz de confusión. 10
Tabla 2.Ejemplo de problema multi-etiqueta. 23
Tabla 3. Transformación BR del problema multi-etiqueta. 23
Tabla 4. Algoritmos para la elaboración de resúmenes mediante la extracción de contenidos. 28
Tabla 5. Resultados de la asignación de polaridad con la biblioteca NLTK. 33
Tabla 6. Resultados de la asignación de polaridades conocidas. 34
Tabla 7. Informe de clasificación. 34

ÍNDICE DE ECUACIONES.

Ecuación 1. Fórmula para el cálculo de la precisión. 11
Ecuación 2. Fórmula para el cálculo del recall. 11
Ecuación 3. Fórmula para el cálculo del F1-Score. 11
Ecuación 4. Función de densidad de probabilidad de la Distribución de Dirichlet. 17
Ecuación 5. Función del clasificador. 20
Ecuación 6. Hiperplano representado mediante una ecuación lineal. 21
Ecuación 7. Fórmula del clasificador de Naive Bayes. 22
Ecuación 8. Fórmula para calcular el algoritmo K-medias. 25
Ecuación 9. Fórmula para el cálculo de la polaridad. 31
Ecuación 10. Función Sigmoide. 32
Ecuación 11. Función ReLU. 32
Ecuación 12. Función tangente hiperbólica. 32

RESUMEN.

El campo de la minería de texto se ha convertido en una herramienta poderosa a la hora de analizar grandes volúmenes de datos no estructurados en formato de texto. Este trabajo de fin de máster tiene como objetivo proporcionar una revisión teórica sobre el campo de la minería de texto y sus aplicaciones acompañada de ejemplos prácticos. Para ello, se realizará un análisis de las técnicas y metodologías utilizadas en la minería de texto, explicando la importancia del preprocesamiento y la selección de características en el análisis de texto.

Posteriormente, se revisarán las diversas aplicaciones de la minería de texto en diferentes campos, como la clasificación de documentos, la elaboración de resúmenes y el análisis de sentimientos. Se destacarán los beneficios y desafíos de la aplicación de la minería de texto en cada uno de estos contextos, y se mencionarán los resultados y avances más relevantes en la literatura científica. Estas aplicaciones teóricas irán acompañadas de un ejemplo práctico para su mejor comprensión. El objetivo final de este TFM es contribuir al conocimiento en el área y proporcionar una base sólida para futuras investigaciones y aplicaciones de la minería de texto.

ABSTRACT.

Text mining field has become a powerful tool for analyzing large volumes of unstructured data in text format. This master's thesis aims to provide a theoretical review of the field of text mining and its applications accompanied by practical examples. For this purpose, an analysis of the techniques and methodologies used in text mining will be carried out, explaining the importance of preprocessing and feature selection in text analysis.

Subsequently, several applications of text mining in different fields, such as document classification, summarization and sentiment analysis, will be reviewed. The benefits and challenges of applying text mining in each of these contexts will be highlighted, and the most relevant results and advances in the scientific literature will be mentioned. All these theoretical applications will be followed by a practical example for better appreciation. The goal of this TFM is to contribute to the knowledge in the area and provide a solid framework for future research and applications of text mining.

1. INTRODUCCIÓN.

Con el auge de la información digital, cada día se generan más datos y documentos en formato de texto, aumentando la necesidad de aplicar técnicas computacionales para analizarlos y obtener nueva información relevante para su estudio. En la actualidad, la cantidad de información disponible online ha experimentado un crecimiento exponencial. La información no estructurada como los correos electrónicos, los artículos en las noticias, los blogs o las publicaciones en las redes sociales presentan una buena oportunidad para las empresas de obtener información valiosa para así, poder ampliar su capacidad de tomar decisiones. Recopilar información es uno de los principios clave de las empresas a la hora de cosechar su éxito, necesitan conocer a sus clientes, sus deseos, opiniones y valoraciones para poder garantizar el éxito comercial. El método más natural de almacenar información y conocimiento es en formato de texto, debido a esto se presume que las técnicas de minería de texto *(Text Mining, TM)* tienen un potencial comercial superior al de las técnicas de minería de datos *(Data Mining, DT)*.

La minería de texto es un campo interdisciplinario que se enfoca en el procesamiento y análisis de grandes cantidades de datos para conseguir extraer información y conocimiento. La información que se puede obtener de los datos de texto es muy amplia, desde patrones de comportamiento hasta sentimientos u opiniones. La minería de texto se enfoca en extraer esta información mediante técnicas y algoritmos de procesamiento del lenguaje, cuyo objetivo es analizar grandes cantidades de texto. Este campo ha experimentado un crecimiento notable gracias a los avances de software y hardware para la web y redes sociales, permitiendo el almacenamiento de grandes cantidades de información en forma de texto y provocando un aumento del interés por descubrir patrones y analizar la información recopilada.

Mander J. (2014) como parte de la empresa GlobalWebIndex realizó un estudio donde afirma que aproximadamente el 85% de la información en línea está disponible en formato de texto. A su vez, IBM indica que, ya que el 80% de los datos mundiales se presentan en formato no estructurado, la minería de texto es una práctica de gran valor en las organizaciones. Según Mena Roa, M. (2022) en su informe "*El Big Bang del Big Data*" publicado en Statista indica que actualmente cada persona genera alrededor de 1,75MB de datos por segundo. El peso de todos los datos que han sido creados, copiados y consumidos globalmente para el año 2022 es de 97 Zettabytes. Pero, aun así, estas cifras no paran de incrementarse, estimándose un crecimiento de estos datos hasta los 181 ZB en 2025. Podemos apreciar el notable crecimiento en los últimos 7 años, durante los cuales, se ha generado 12 veces el volumen de datos que se generó en 2015 (8 ZB). Si bien es cierto que, una gran parte de este volumen de datos utiliza formatos de video o foto (Netflix, YouTube, Instagram, etc.) los datos con formato de texto ocupan una parte importante (WhatsApp, e-mails, artículos publicados en Google, Twitter, etc.).

Algunas de las aplicaciones más comunes de la minería de texto incluyen la clasificación automática de documentos, la identificación de patrones de comportamiento en el análisis de sentimiento y opiniones, la extracción de la información en documentos y la creación de resúmenes. Es un campo emocionante y de constante evolución, que debido a la creciente cantidad de información disponible se está convirtiendo en una herramienta invaluable para el análisis de datos y la toma de decisiones informadas en una amplia gama de campos.

En el marco de este trabajo de investigación, se explorarán las principales aplicaciones y herramientas de la minería de texto, así como algunos de los algoritmos utilizados para su implementación. La minería de texto promete seguir avanzando y ofreciendo nuevas oportunidades para aprovechar el potencial de los datos de texto y obtener información valiosa a partir de ellos.

2. DESCRIPCIÓN DE LOS DATOS Y BIBLIOTECAS PRINCIPALES.

La muestra de datos de texto utilizada en los ejemplos prácticos de este Trabajo de Fin de Máster ha sido obtenida de la base de datos de reseñas de películas del módulo NLTK (*Natural Language Toolkit*) para Python. En concreto, este *corpus* de reseñas está escrito en inglés y consta de 50.000 archivos de texto sin preprocesar divididos en conjuntos de entrenamiento y pruebas, almacenados en los directorios "train/" y "test/".

Cada uno de estos directorios cuenta con 25.000 archivos previamente etiquetados según su polaridad y separados en carpetas con etiquetas positivas "pos/" y negativas "neg/". El nombre de cada archivo sigue un formato especifico "[id]_[rating].txt", donde el campo "[id]" representa el identificador único que se ha asignado a cada reseña mientras que "[rating]" indica la clasificación asignada a cada una de las películas en escala de estrellas, donde una estrella es el mínimo y 10 es el máximo. Además de esto, se incluyen otros 50.000 documentos sin etiquetar que se pueden utilizar para el aprendizaje no supervisado.

Para simplificar los análisis, se ha decidido trabajar con 1.000 archivos obtenidos de la carpeta "train/" de los cuales, 500 pertenecen a la categoría "pos/" y los otros 500 pertenecen a la categoría "neg/". De este modo, conseguimos aplicar las técnicas de minería de texto mediante ejemplos sencillos.

A continuación, se presenta un ejemplo del formato de una de las reseñas:

> "2 deathly unfunny girls stays a their deathly unfunny Uncle Benny's beach house. Uncle Beeny doesn't like party. But guess what? the deathly unfunny girls have a, yup you guessed it, a deathly unfunny beach party. If you didn't catch the not so subliminal message that I'm trying to convey. First off, you're a moron. I would rather watch a nude jello tag team watching match between Bea Aurther and Cameryn Manhiem VS. Rosie O'Donnell and Jessica Tandy. This movie, and I lose the term loosely is just THAT bad.

My Grade: F

Eye Candy: Kristin Novak and Charity Rahmer go topless, Iva Singer shows breasts and buns"

Para llevar a cabo el análisis, se utilizará el lenguaje de programación Python y sus diversas librerías especializadas en procesamiento de texto. Estas librerías proporcionan funcionalidades para poder realizar diversas tareas mediante una amplia gama de funciones y algoritmos que facilitan el análisis y extracción de información útil. Algunas de las librerías que se utilizaran en los ejemplos prácticos son las siguientes:

- NLTK (Natural Language Toolkit): Se centra en el preprocesamiento del lenguaje natural. Sus diversos módulos ofrecen herramientas para trabajar la tokenización, etiquetado, análisis sintáctico, etc.

- Sklearn (Scikit-Learn): Esta biblioteca se centra en la parte del aprendizaje automático y modelado de datos. Ofrece algoritmos de aprendizaje supervisado y no supervisado además de herramientas para la selección de características o evaluación de modelos.

- SpaCy: Esta biblioteca ofrece módulos destinados al preprocesamiento del lenguaje natural, al igual que "nltk". Por otro lado, proporciona modelos pre-entrenados en diversos lenguajes.

3. CONTEXTO HISTÓRICO.

Cuando hablamos de *Minería de Texto (Text mining, TM)* hablamos de un problema que está relacionado los siguientes términos y conceptos que se han venido utilizando en los últimos años: *Arqueología de Datos (Data Archaeology), Extracción de Conocimiento en Texto (Knowledge Discovery in Text), Gestión del Conocimiento (Knowledge Management), Minería de documentos (Document Mining), Minería de Documentos Web (Web Document Mining), Minería de la Información (Information Mining), o Pesca de datos (Data Fishing).*

Debido al gran incremento de la información recogida en formato de texto en el último siglo, surgió la necesidad de desarrollar procedimientos avanzados para poder analizarla. Uno de los requisitos esenciales era el de poder resumir los documentos para reducir su cuerpo a un tamaño en el que fueran más fáciles de tratar, ordenar o clasificar. El primer autor en desarrollar un procedimiento para la catalogación de documentos fue el bibliotecario Thomas Hyde en 1677. Pero no fue hasta 1958 cuando Luhn en su artículo "*A Business Intelligence System"* consiguió desarrollar un sistema automático para resumir y codificar documentos y crear perfiles de interés para las organizaciones, además de adaptar el primer ordenador que fuese capaz de aplicarlo. En 1967, se desarrolló la disciplina de la lingüística del *corpus*, la cual suele asociarse con la lingüística computacional, tras la publicación del artículo *"Computational Analysis of Present-Day American English"* de Kucera y Francis en 1967. El *corpus* puede definirse como un conjunto de textos que se integran en un sistema de almacenamiento y recuperación para realizar un tratamiento de datos mediante técnicas de *Text Mining*. En este artículo, se desarrolla el primer *corpus* denominado *BROWN* que contiene un millón de palabras seleccionadas de una amplia variedad de fuentes.

Con el desarrollo del software y hardware surgieron mejoras en el campo del procesamiento de lenguaje natural *(Nature language preocessing, NPL)*. Los orígenes del NPL se sitúan en la década de 1950 cuando Turing publicó su libro "*Computing machinery and intelligence"* donde propuso el criterio de inteligencia que hoy conocemos como "Test de Turing". Entre 1950 y 1980 estos sistemas estaban basados en un conjunto de reglas complejas diseñadas manualmente. A finales de esta década, fue cuando se comenzaron a introducir algoritmos de aprendizaje automático para el procesamiento del lenguaje. Esta evolución permitió el desarrollo de la minería de texto actual y la creación nuevas aplicaciones.

Entre los años 1980 y 2000 se desarrollaron una gran parte de los algoritmos de Aprendizaje Automático *(Machine Learning, ML)* que se aplican actualmente, como la categorización de documentos con redes neuronales. También se comenzaron a aplicar varios de los diferentes enfoques estadísticos que se utilizaban en otros campos, como *los Modelos Ocultos de Markov (Hidden Markov Model, HMM),* en el reconocimiento de patrones. Otra de las grandes técnicas aplicadas fue el Indexado Semántico Latente *(Latent Semantic Indexing, LSI).* Deerwester *et al.* (1990**)** publicaron el artículo *"Indexing by latent semantic análisis"* donde introducían el *LSI* en el *ML*. Este método utiliza la técnica de valores singulares (*SVD*) para analizar las relaciones entre los documentos y las palabras que hay en ellos, creando un conjunto de conceptos que guarden relación con estos. Laundauer *et al.* (1998), presentan el Análisis Semántico Latente (*Latent Semantic Analysis, LSA)* que emplea las mismas técnicas que el *LSI* con la única diferencia de utilizar este método exclusivamente para determinar la similitud entre los elementos analizados fijándose en las veces que estos son repetidos en determinados contextos verbales.

Un año después, Hofmann (1999) combinó los modelos *LSI* y la variante probabilística creando así el modelo Indexado Probabilístico de la Semántica Latente (*Probabilistic Latent Semantic Indexing, PLSI*) o Análisis Probabilístico (*Probabilistic Latent Semantic Analysis, PLSA*)**.** Lo cual supuso grandes avances en la modelización probabilística del texto. Pero tras esto, surgieron nuevos problemas, como el gran número de parámetros utilizados frente al *corpus.* Para solucionar este problema, Blei D. M. *et al*. presentaron el modelo de categorías denominado Asignación Latente de Dirichlet *(Latent Dirichlet Allocation, LDA)* en el año 2002, el cual, es similar al modelo *PLSA* exceptuando que el modelo *LDA* asume que la distribución de las categorías es a priori una distribución Dirichlet.

En la actualidad además de utilizar las técnicas propias del Machine Learning, los investigadores de la minería de texto están comenzando a implementar técnicas de Aprendizaje Profundo (*Deep Learning, DL*) como las Redes Neuronales Recurrentes (*RNN*). Sin embargo, aún queda mucho por avanzar en el campo de la minería de texto como, por ejemplo, la gestión de los grandes *corpus* de texto multitemático.

4. CONSIDERACIONES DE CALIDAD Y VALIDEZ.

Para que las conclusiones obtenidas de las investigaciones sean relevantes y se puedan tomar decisiones informadas y validas es primordial considerar la calidad de los datos seleccionados para el análisis y la validez de los resultados obtenidos. La calidad de los datos utilizados para el estudio puede verse afectada por diferentes factores, autores como Kao, A. & Poteet, R. S. (2007) abordan la importancia de escoger fuentes adecuadas y fiables. Los datos recopilados de las redes sociales tienden a ser menos fiables que aquellos que se obtienen de bases de datos gubernamentales o encuestas oficiales.

Por otro lado, autores como Reyes-López, O. & Olvera, R. (2019) hacen énfasis en la importancia del tamaño muestral y su representatividad teniendo en cuenta la naturaleza de los datos disponibles y el propósito de la investigación. Elegir una muestra demasiado pequeña o no representativa puede truncar e incluso invalidar los resultados obtenidos.

En lo que se refiere a los criterios para seleccionar el tamaño muestral, Reyes-López, O. & Olvera, R en su artículo *"Criterios para determinar el tamaño de muestra en estudios descriptivos"* presentan una distinción entre las investigaciones cualitativas y cuantitativas: "*Los criterios para la determinación del tamaño de muestra, serán diferentes en caso de tratarse de una investigación cualitativa a diferencia de trabajar con énfasis cuantitativo, incluso este último dependerá del propósito del proyecto por si requiere la estimación de medias o de proporciones y si el valor de "N" es indeterminado o conocido; por otra parte los criterios para la determinación de "n" como tamaño de muestra variaran en el caso de ensayar con pruebas piloto, o tratándose de comunidades muy dispersas, así también cuando se busca muestrear cada localidad en un amplio universo de estudio*." Para considerar la validez de nuestros resultados es frecuente realizar rigurosas evaluaciones mediante diferentes medidas cuantitativas. A continuación, describiremos brevemente algunas de las más comunes:

4.1. MATRIZ DE CONFUSIÓN.

La matriz de confusión es una herramienta que permite al investigador contabilizar las veces que el algoritmo clasifica de forma correcta o incorrecta las instancias de cada una de las clases seleccionadas en el estudio. Es una medida de validez común en estudios de aprendizaje supervisado válida tanto en clasificaciones binarias como no binarias. La *Tabla 1* presenta la representación gráfica de una matriz de confusión tipo binaria:

Tabla 1. Matriz de confusión.
Fuente: Elaboración propia.

		valores predichos	
		positivo	negativo
valores reales	positivo	true positive (tp)	false negative (fn)
	negativo	false positive (fp)	true negative (tn)

La matriz de confusión se divide en cuatro secciones: verdaderos positivos (*tp*), verdaderos negativos (*tn*), falsos positivos (*fp*) y falsos negativos (*fn*). Estas secciones representan el número de instancias que son clasificadas correcta o incorrectamente por el modelo en cada clase. En muchas situaciones el rendimiento del estimador se analiza sin tener en cuenta las puntuaciones asignadas a cada clase, si no, comparando la clase estimada con a clase real para cada instancia del conjunto de datos. (Luque, A. *et al*, 2022).

4.1.1. Precisión.

La precisión es una métrica capaz de evaluar la proporción de predicciones correctas que ha realizado el modelo en función de todas las predicciones positivas realizadas. Su cálculo se realiza mediante la división de los verdaderos positivos entre la suma de los verdaderos positivos y los falsos positivos. Matemáticamente se expresa con la siguiente fórmula:

Ecuación 1. Fórmula para el cálculo de la precisión.

$$P = \frac{VP}{VP + FP}$$

A la hora de interpretar los resultados debemos de tener en cuenta una serie de consideraciones. González, S. G. (2022) menciona la dispersión en el número de muestras como una de estas consideraciones. Podemos verlo a través de un ejemplo, suponiendo que contamos con 495 muestras para la clase 0 y 5 muestras para la clase 1, obtendremos un clasificador sesgado hacia la clase 0. Si dicho clasificador clasifica todas las muestras en la clase 0, la precisión obtenida será del 99%. Sin embargo, su rango de error será del 100% para la clase 1. Aunque la precisión obtenida sea del 99%, está muy lejos de ser un buen clasificador.

4.1.2. Sensibilidad o Recall.

El recall también conocido como sensibilidad o tasa de verdaderos positivos trata de medir la capacidad que tiene un modelo de identificar y clasificar correctamente todos los ejemplos positivos. Su cálculo se realiza dividiendo la cantidad de verdaderos positivos entre la suma de los verdaderos positivos y los falsos negativos (Pío-Álvarez, S. 2018). Representada matemáticamente del siguiente modo:

Ecuación 2. Fórmula para el cálculo del recall.

$$R = \frac{TP}{TP + FN}$$

Esta métrica es importante en aquellas investigaciones donde la identificación de los casos positivos es crítica. Por ejemplo, en pruebas métricas para detectar enfermedades, detección de fraude, etc.

4.1.3. F1-Score.

El F1-Score según Li, X. *et al*., (2008) es una medida que combina el recall y la precisión. Se utiliza cuando se quiere conocer una medida balanceada entre estas dos métricas. Su cálculo se realiza mediante la media armónica entre la precisión y el recall.

Ecuación 3. Fórmula para el cálculo del F1-Score.

$$F1 = 2\,\frac{P \cdot R}{P + R}$$

El F1-Score es adecuado para aquellas investigaciones en las que una clase es mucho más frecuente que la otra, puesto que esta métrica considera el equilibrio entre la capacidad del modelo de clasificar correcta e incorrectamente las instancias positivas y negativas.

4.2. CURVA DE ROC.

La curva de ROC (*Receiver Operating Characteristic*) es un gráfico bidimensional que representa la relación entre la sensibilidad o la tasa de verdaderos positivos, (representada en el eje de ordenadas) y la tasa de falsos positivos (representada en el eje de abscisas). Spackman, K. A. (1989) fue uno de los primeros en utilizar las curvas de ROC en el aprendizaje automático.

La mejor predicción del clasificador nos situaría en el punto de la esquina superior izquierda (0,1) denominado clasificación perfecta, en el cual, no existirá ningún falso positivo. El caso contrario, una clasificación completamente aleatoria, nos situaría en algún punto de la línea diagonal y = x, la cual, representa una estrategia de clasificación totalmente aleatoria. Según Fawcett, T. (2006), un ejemplo que se suele utilizar para medir las clasificaciones es comparar los resultados obtenidos por el clasificador con los obtenidos a partir de lanzar una moneda al aire. Si lanzamos una moneda al aire nos situaremos sobre la diagonal que divide la curva de ROC (0.5,0.5). Esta diagonal indica que los puntos situados por encima de ella representan unos resultados mejores que el azar y los puntos situados por debajo unos resultados peores que el azar.

Para su interpretación se suele utilizar el área de bajo la curva de (AUC). Podemos guiarnos a partir de las indicaciones de Fawcett para interpretar los resultados del AUC de nuestra curva de ROC:

- Si el AUC = 0.5, la clasificación es completamente aleatoria (Como lanzar una moneda al aire).
- Si el AUC está entre 0.5 y 0.6, la clasificación es mejor que aleatoria pero no es buena.
- Si el AUC está entre 0.6 y 0.75, la clasificación es regular.
- Si el AUC está entre 0.75 y 0.9 la clasificación es buena.
- Si el AUC está entre 0.9 y 0.97 la clasificación es muy buena.
- Si el AUC está entre 0.97 y 1 la clasificación es excelente o perfecta.

4.3. CALIDAD - Aplicada al *corpus*.

Debemos de tener presentes ciertas consideraciones de calidad antes de comenzar cualquier análisis. Centrándonos en el *corpus* seleccionado, debemos de evaluar la calidad de las reseñas en términos de su contenido y coherencia. Algunas de las reseñas contienen información irrelevante, errores gramaticales o sesgos, lo cual, puede afectar a la interpretación y precisión de nuestro modelo. Además, es importante comprobar la representatividad del *corpus*. Es decir, que exista una diversidad de géneros de películas. En el caso de que todas o la mayoría de las reseñas de películas evaluasen un género en concreto (no es nuestra situación), no deberíamos de generalizar los resultados a un contexto más amplio.

Debemos tener en consideración que el análisis de un *corpus* con mayor tamaño y equilibrio entre clases positivas y negativas, como la muestra completa etiquetada de 50.000 documentos, proporcionará unos resultados más confiables y robustos en términos de clasificación de sentimientos y análisis de textos.

En último lugar, es recomendable tener en cuenta que el *corpus* de películas de NLTK se centra en reseñas en inglés, lo cual limita su aplicabilidad a otros idiomas. Hemos de considerar las diferencias lingüísticas y complementar el análisis con otras fuentes de datos y técnicas de validación para obtener los mejores resultados posibles.

5. PREPROCESAMIENTO DEL LENGUAJE NATURAL.

El procesamiento del lenguaje natural *(NLP)* es una rama de la inteligencia artificial que permite, mediante el uso de herramientas de computación, la manipulación de los lenguajes naturales (Vásquez, M. A. C*, et al.,* 2020). Se pretende transformar los datos brutos en datos que cuenten con un formato más sencillo de interpretar por las máquinas. Si no realizásemos este paso, los resultados obtenidos en el análisis serían de muy baja calidad. El objetivo principal del NLP es desarrollar sistemas informáticos que puedan procesar el lenguaje humano de manera efectiva y precisa. Para esto, es necesario considerar el orden, estructura y significado de las palabras que lo componen. Este enfoque está centrado en el lenguaje en sí, su estructura nace en los años 50 con Chomsky e incluye las siguientes técnicas:

Ilustración 1. Principales fases del preprocesamiento de textos.
Fuente: Gobierno de España – Secretaría de estado de digitalización e inteligencia artificial.

5.1. PREPARACIÓN DEL *CORPUS* - Limpieza

La eliminación de ruido es una tarea de normalización que suele producirse antes de la "tokenización". Este paso se refiere al proceso de recopilación, limpieza y organización de los datos textuales. Se tratará de eliminar el ruido de los documentos para poder realizar los análisis posteriores. Eliminar el ruido de nuestro *corpus* consiste en eliminar los encabezados de los archivos de texto, la numeración y los pies de página, eliminar el formato original HTML, XML, JPG… el marcado, los metadatos y extraer los datos útiles de otros formatos. En este TFM se han utilizado el módulo "re" y "nltk" de Python para llevar a cabo estas operaciones.

Según Álvarez, D. P. (2021) puede ser necesario realizar un mapeo de las contracciones a su forma extendida que nos permita aumentar la coherencia en el estudio. Sobre todo, es necesario si alguno de los textos está escrito en otro idioma, especialmente en inglés (En ocasiones en español también sucede). Para ello debemos reemplazar las contracciones del lenguaje por sus expansiones (didn't → did not), este paso no es un requisito obligatorio, pero si puede ser muy beneficioso a la hora de analizar los resultados.

5.2. TOKENIZACIÓN.

La tokenización es el proceso de dividir las cadenas de texto más extensas en otras más pequeñas (palabras, frases, símbolos u otros elementos significativos) que se denominan tokens (Jurafsky, D., & Martin, J. H. 2019). Dentro de la tokenización también podemos considerar el manejo de los signos de puntuación, la transformación de las palabras de mayúsculas a minúsculas y la eliminación de palabras no relevantes para la comprensión del texto como son los "stop words". La tokenización es un paso esencial en el NLP porque permite a las máquinas entender y procesar el texto de manera más eficiente.

Uno de los mayores desafíos que presenta la tokenización de textos es el idioma. Según Kannan, S. *et al.,* (2014) los idiomas como el español, inglés y francés se consideran "lenguajes espaciados", ya que cada palabra está separada por espacios, lo que facilita la tokenización segmentada. Sin embargo, hay otros idiomas como el chino mandarín o el tailandés en los cuales las palabras no tienen límites claros, lo que nos obliga a realizar un proceso de tokenización no segmentada. Este último enfoque requiere de una mayor información léxica y morfológica para llevar a cabo la tokenización de manera adecuada.

5.2.1. Manejo de los signos de puntuación.

En el proceso de tokenización uno de los dilemas frecuentes al que se enfrentan los investigadores es el manejo de los signos de puntuación (comas, puntos, puntos suspensivos, etc.). Se pueden aplicar dos enfoques diferentes: Considerar cada uno de ellos como un token separado o eliminarlo durante el proceso (Addati, A. & Roger, S. 2017). La elección de un método u otro dependerá del objetivo específico del análisis.

Mantener los signos de puntuación como tokens separados puede resultar útil en tareas como el análisis de sentimientos, el análisis de opiniones o el análisis de emociones. En estos casos, los signos de puntuación pueden tener un impacto significativo en la interpretación del texto, por lo que conservarlos permite capturar esa información adicional. Por otro lado, en tareas de clasificación de textos, donde el objetivo principal es identificar la categoría a la que pertenece un texto, es común eliminar los signos de puntuación contribuyendo a reducir la complejidad del análisis y simplificar el procesamiento de los datos.

5.2.2. Stop words.

Los "stop words" o "palabras vacías" son términos comunes y frecuentes en un idioma que suelen ser eliminados durante la etapa de preprocesamiento del texto debido a su escaso valor para el análisis. Estas palabras incluyen conectores, preposiciones, pronombres, adverbios y otros términos propios del lenguaje que no aportan un significado sustancial al texto. Existen diversas librerías, como por ejemplo NLTK o spaCy que incorporan una lista bastante amplia de "stop words". Kannan, S. *et al.,* (2014) estiman que aproximadamente un 20 - 30% de las palabras presentes en un documento son "stop words".

No obstante, es importante tener en cuenta que no es necesario eliminar los "stop words" en todas las aplicaciones del NLP. Existe una vertiente de autores a la que pertenecen Popova, S. & Skitalinskaya, G. (2017) quienes sostienen que, para determinados tipos de análisis, como el de sentimientos u opiniones, estas palabras pueden ser relevantes para capturar la intención o el tono del texto, especialmente cuando el *corpus* objeto de estudio cuenta con textos de gran extensión. Además, en tareas específicas de procesamiento del lenguaje natural, como el análisis gramatical o la traducción, los "stop words" pueden desempeñar un papel fundamental para mantener la estructura y la gramática del texto.

5.2.3. Minúsculas.

En el proceso de análisis de texto, a menudo resulta beneficioso convertir todas las letras a minúsculas con el fin de reducir la variabilidad causada por las letras mayúsculas. Esto nos permite tratar las palabras en mayúsculas y minúsculas como iguales, evitando posibles discrepancias en el conteo o la comparación de palabras y aumentar el rendimiento del clasificador (Widyassari, A. P.*, et al.,* 2022). No obstante, es importante tener en cuenta que en algunos análisis las letras mayúsculas desempeñan un papel relevante en la identificación de ciertas características, como los nombres propios. En estos casos, conservar las mayúsculas puede ser esencial para garantizar la correcta identificación y comprensión de dichas entidades.

5.3. LEMATIZACIÓN.

El objetivo de la lematización es convertir las palabras en un lema, es decir, en su forma de vocabulario original. Se pretende normalizar la forma de una palabra (flexionada) en la forma utilizada como palabra principal en un diccionario (Merriam, C. & G. 2007). Su propósito es disminuir la variabilidad de las palabras y agrupar aquellas que comparten la misma raíz en una forma estándar.

Para nuestros ejemplos prácticos, al contar con un *corpus* escrito en inglés, debemos de ser conscientes de que no se trata de un lenguaje de gran complejidad como otras lenguas, por ejemplo, el griego moderno, árabe o chino mandarín. Guía, P. & Atkinson-Abutridy, J. (2010) explican el proceso de lematización para palabras en inglés del siguiente modo:

En la mayoría de los casos, la raíz de la palabra no se altera durante el proceso de lematización (por ejemplo, play → play-s → play-ing → play-ed) y la mayoría de los tiempos verbales ingleses se forman utilizando grupos de palabras (por ejemplo, I play → I will play, I played → I had played). Sin embargo, en inglés existen algunos casos muy irregulares en los que incluso la raíz cambia radicalmente. Un ejemplo típico de verbo muy irregular en inglés es el caso del verbo "go" (es decir, go → went).

En cuanto a los algoritmos utilizados para la lematización, encontramos el algoritmo de Porter, el cual, ha resistido la prueba del tiempo y sigue siendo un algoritmo de lematización popular. Propuesto originalmente en 1980, elimina de forma iterativa los sufijos de una palabra, asignándola a su raíz.

5.4. STEMMING.

El “stemming” al igual que la lematización consiste en reducir las palabras a su raíz o lema. Por ejemplo, un algoritmo de stemming reduce las palabras "likes" y "likely" a la raíz "like". El stemming es muy utilizado en los motores de búsqueda puesto que en diversas ocasiones puede ser necesario buscar variantes de las palabras que estén relacionadas con el tema de interés para poder ofrecer los mejores resultados (Colmenares, E. & Wu, H. 2021).

Smelyakov, K., *et al.* (2020) diferencian dos enfoques básicos para los algoritmos de stemming modernos. El primero de ellos consiste en utilizar tablas o diccionarios especiales con palabras y sus raíces y el segundo, consiste en recortar los sufijos y prefijos siguiendo ciertas reglas. Este último enfoque, es mucho más rápido y se puede utilizar con palabras conocidas y desconocidas a diferencia del primero, pero comete errores con bastante frecuencia.

Es importante destacar que el stemming es un enfoque basado en reglas en comparación más simple que la lematización. Por lo tanto, aunque el stemming puede ser más rápido, también puede generar resultados menos precisos. No obstante, en ciertos casos donde la simplicidad es prioritaria, el stemming puede ser una opción adecuada.

5.5. EJEMPLO PRÁCTICO - Preprocesamiento del *corpus*.

El objetivo de este ejemplo es preparar nuestro *corpus* para un posterior análisis binario y de sentimientos. Para realizar el preprocesamiento utilizaremos las bibliotecas: "os", "nltk", "re", "html", "spacy" y varios módulos específicos de "nltk" como "PlaintextCorpusReader", "word_tokenize", "WordNetLemmatizer", "stopwords" y "wordnet". En el **Anexo I** se recoge el código utilizado para realizar el preprocesamiento.

En primer lugar, realizamos un mapeo de las etiquetas POS (Part-of-Speech) del conjunto de datos TreeBank a etiquetas POS de WordNet para posteriormente poder realizar la lematización. Reemplazamos aquellos caracteres que están repetidos por un solo caracter (con el objetivo de solventar uno de los errores gramaticales frecuentes localizados en las reseñas) y unificamos la nomenclatura.

Tras esto definimos nuestra función de preprocesamiento denominada **preprocess_text[]** la cual realizará una limpieza completa del *corpus:* elimina el formato HTML, reemplaza los signos de puntuación por espacios, elimina contracciones, elimina los saltos de página del formato HTML (los cuales se representan como <br/>), convierte las mayúsculas en minúsculas, elimina los "stop words" contenidos en la lista de "stop words" de la biblioteca "nltk" y una serie de palabras comunes predefinidas que no aportan ningún valor al análisis. Tras esta limpieza inicial, nuestra función divide el texto en tokens, lematiza las palabras y aplica una normalización lingüística para finalmente unificar la nomenclatura, reducir la redundancia y devolver el texto preprocesado como una cadena de texto.

Tras definir la función de preprocesamiento, definimos un array denominado **processed_corpus []** que recogerá los 1.000 archivos (500 con polaridad negativa y 500 con polaridad positiva) ya preprocesados.

A modo de demostración del funcionamiento de nuestra función **preprocess_text[]** elegimos una reseña al azar y la preprocesamos obteniendo el siguiente resultado:

"If you had asked me how the movie was throughout the film, I would have told you it was great! However, I left the theatre feeling unsatisfied. After thinking a little about it, I believe the problem was the pace of the ending. I feel that the majority of the movie moved kind of slow, and then the ending developed very fast. So, I would say the ending left me disappointed.

I thought that the characters were well developed. Costner and Kutcher both portrayed their roles very well. Yes! Ashton Kutcher can act! Also, the different relationships between the characters seemed very real. Furthermore, I thought that the different plot lines were well developed. Overall, it was a good movie and I would recommend seeing it.

In conclusion: Good Characters, Great Plot, Poorly Written/Edited Ending. Still, Go See It!!!."

"ask film throughout film would told great however left theatre feel unsatisfied think little believe problem pace end feel majority film move kind slow end developed fast would say end left disappointed thought character well developed costner kutcher portrayed role well yes ashton kutcher act different relationship character seem real furthermore thought different story line well developed overall good film would recommend see conclusion good character great story poorly write edit end go see"

6. MODELOS DE TEMAS - *Latent Dirichlet Allocation.*

Los modelos de representación son técnicas que se utilizan para transformar los datos no estructurados en datos estructurados, permitiendo que puedan ser procesados por los algoritmos de aprendizaje automático. Dentro de los modelos de representación destacan los modelos de temas, los cuales, proporciona métodos de organización, comprensión, búsqueda y resumen de grandes archivos de forma automática con el objetivo de descubrir los temas ocultos en ellos (Bhutada, S., *et al*., 2014). Cabe destacar el funcionamiento del modelo de temas *Latent Dirichlet Allocation* (*LDA*), propuesto por Blei, D. M. en 2002, este modelo probabilístico para conjuntos de datos discretos realiza una breve descripción de los documentos, conservando la información estadística esencial y contribuyendo a manejar eficientemente conjuntos de documentos a grande escala.

El objetivo del *LDA* es modelar los "*tópicos*". Un "*tópico*" es un tema o idea que se relaciona con un conjunto de palabras que aparecen en el *corpus* mediante la asignación de una distribución de probabilidad a cada documento y una distribución de palabras a cada tópico. El modelo *LDA* se basa en la distribución de Dirichlet, que es una distribución de probabilidad multivariante, la cual, describe la distribución de probabilidad de un conjunto de variables aleatorias que representan las proporciones de múltiples categorías que se observan en la muestra. (Liu *et al,* 2011). Esta distribución se puede actualizar a posteriori a medida que se obtengan nuevas muestras. La función de densidad de probabilidad de la Distribución de Dirichlet tiene la siguiente forma:

Ecuación 4. Función de densidad de probabilidad de la Distribución de Dirichlet.

$$f(x_1, x_2, \dots, x_k) = \frac{1}{B(\alpha)\prod\left(x_i^{(\alpha_i - 1)}\right)}$$

Donde $x_1, x_2, \dots, x_k$ son las variables aleatorias que representan las k dimensiones de la distribución, $B(\alpha)$ es la función Beta multivariada de normalización, y $\alpha_i - 1$ es el exponente al que se eleva cada x_i en la función de densidad de probabilidad.

El modelo asume que los tópicos y las palabras en cada documento se generan a partir de distribuciones de Dirichlet y utiliza un algoritmo de inferencia bayesiana de tres capas para estimar estas distribuciones a partir de los datos observados (Blei, D.M., *et al.,* 2012).

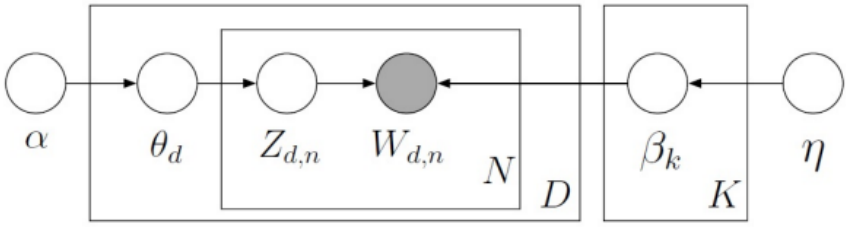

Ilustración 2. Modelo gráfico para la asignación latente de Dirichlet (LDA)
Fuente: Blei, D.M., et al., (2012).

Cada uno de los nodos de la *Ilustración 2* representa una variable aleatoria y se etiqueta acorde con su papel en el proceso generativo del *corpus*. Los nodos ocultos no están sombreados, los nodos observados en cambio sí lo están. El rectángulo *N* representa la colección de palabras en el documento, el rectángulo *D* la colección de documentos en el *corpus* y el rectángulo *K* el conjunto de tópicos que atraviesan el *corpus*, α, β y θ representan los hiper-parámetros del modelo.

7. EJEMPLO DE VISUALIZACIÓN – Nubes de palabras.

Las nubes de palabras o "bag of words" (BoW) son una técnica derivada del NPL. Consiste en la construcción de un diccionario que nos permite representar de forma visual los términos más comunes de un *corpus* de texto. Según Shekhar, R. & Jawahar, C. V. (2012) en este modelo, cada documento está representado por un conjunto de palabras independientes de la gramática y que no sigue ningún orden.

Habitualmente, cada documento se representa como un vector de pesos de las palabras. Sin embargo, De la Fuente, O. S. (2019) establece dos limitaciones principales a este tipo de representaciones La primera de ellas se caracteriza por dividir los grupos de palabras en sus palabras constituyentes. Es decir, si tenemos el conjunto de palabras "I have been playing" → se dividirá en las palabras "I" "have" "been" y "playing". En segundo lugar, este método trata los sinónimos como dos palabras independientes sin asociación semántica, por ejemplo "film" y "movie". Para tratar de solventar la última de estas limitaciones, antes de aplicar la función de preprocesamiento, se ha generado la función **unify_nomenclature** (), que incluye reemplazos en la nomenclatura más común relacionada con las películas. En el **Anexo II** se recoge el código empleado para generar la nube de palabras.

A continuación, importaremos las bibliotecas necesarias para realizar la nube de palabras que nos permitirá la visualización del *corpus*. Estas bibliotecas incluyen "matplotlib.pyplot", "collections.Counter" y "wordcloud.WordCloud".

Crearemos un array denominado **words []**, donde se registrarán las palabras y su frecuencia y seleccionaremos las 50 palabras más frecuentes. Tras esto se creará un diccionario que sea capaz de mapear cada palabra a su frecuencia para finalmente generar la nube de palabras presentada en la *Ilustración 3:*

Ilustración 3. Representación del *corpus* "Bag of Words"
Fuente: Elaboración propia.

A simple vista, fijándonos exclusivamente en la *Ilustración 3* somos capaces de identificar las palabras más presentes en el *corpus*: "film", "show", "see", "story" and "character". Estas palabras nos indican los temas y elementos más relevantes del *corpus* sin necesidad de conocer el contenido completo de los documentos. En nuestro caso, el tema principal son las descripciones o reseñas de películas, el cual, está estrechamente relacionado con las palabras obtenidas en el BoW. Destaca la palabra "film", como aquella que posee una mayor frecuencia. Asimismo, palabras como "character", "story", "scene" y "show" sugieren que la actuación, los personajes y las escenas son elementos clave en las opiniones expresadas.

8. PRINCIPALES APLICACIONES DE LA MINERÍA DE TEXTO.

La minería de texto es un campo que se encuentra en la intersección de la lingüística computacional, la información textual, el aprendizaje automático *(Machine Learning, ML*), la inteligencia artificial y la minería de datos. Estas disciplinas se relacionan y se complementan entre sí para analizar y procesar grandes cantidades de datos textuales. La *Ilustración 4*, presenta un diagrama de Venn donde se representan de manera simplificada las relaciones entre estas disciplinas. Talib, R. *et al*, (2016) exponen que el círculo central hace referencia al área de superposición que abarca a todas las disciplinas, demostrando que la minería de texto se basa en los fundamentos de todas estas disciplinas.

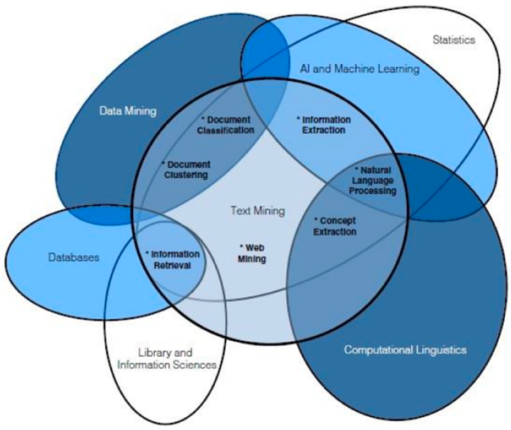

Ilustración 4. Diagrama de Venn. Aplicaciones de la minería de texto.
Fuente: Talib, R. et al, (2016)- "Text Mining: Techniques, Applications and Issues".

La aplicación de la minería de texto es una tarea compleja que requiere considerar factores como la longitud de los textos, la cantidad de documentos y las colecciones utilizadas para el análisis. Sin embargo, los beneficios son significativos, ya que permite desarrollar aplicaciones de gestión del conocimiento que ayudan a las personas en tareas de análisis, recopilación y procesamiento de grandes volúmenes de información de manera eficiente. Esto ahorra tiempo y permite tomar decisiones más informadas, contribuyendo al avance de la sociedad hacia una base de conocimiento sólida. Dentro del ML podemos distinguir dos tipos de modelos, los modelos de aprendizaje supervisado y los modelos de aprendizaje no supervisado.

Los modelos de aprendizaje supervisado son algoritmos de ML que utilizan datos etiquetados para aprender a realizar predicciones o clasificar nuevos datos en función de características de entrada. Estos modelos son entrenados con un conjunto de datos de entrenamiento que contiene etiquetas o respuestas conocidas (Delua, J. 2021).

- Modelo de regresión lineal.
- Modelo de regresión logística.
- Árboles de decisión.
- Máquinas de vector de soporte.
- Redes neuronales.

Por otro lado, los modelos de aprendizaje no supervisado son aquellos en los que los datos no están etiquetados, lo que significa que el modelo no tiene conocimiento previo de las respuestas correctas (Delua, J. 2021).

- Clustering o agrupamiento.
- Análisis de los componentes principales.
- Reglas de asociación.

8.1. SISTEMAS DE CLASIFICACIÓN DE DOCUMENTOS.

En la actualidad, gran parte de la información es etiquetada para ofrecer a los usuarios una mayor rapidez en sus búsquedas de contenido. Esto incluye la información de texto, video, música o películas. Las etiquetas de datos o clasificadores son mecanismos que nos permiten agrupar los diferentes atributos que contienen características similares. Es decir, estas etiquetas que asignamos a los atributos tienen como objetivo dar una descripción rápida del contenido y pertenecen a conjuntos que han sido previamente definidos.

Basándonos en el artículo "*Machine learning in automated text categorization*" de Sebastiani, F. (2002) detallaremos brevemente el funcionamiento de los sistemas de clasificación de textos. La clasificación de textos consiste en asignar a cada uno de los documentos (d_j) de nuestro *corpus* una etiqueta λ_j que designe a que clase pertenece. El elemento que realiza esta clasificación se denomina clasificador. Podemos definir un clasificador como una función cuyo objetivo es asignar las diferentes etiquetas a los documentos:

Ecuación 5. Función del clasificador.

$$\gamma : (d_1, \dots, d_j) \rightarrow (1, 2, \dots, n), \qquad \gamma = \lambda_1, \lambda_2, \dots, \lambda_k$$

Lo más común es utilizar algoritmos de aprendizaje supervisado a través del etiquetado de texto, cuyo principal objetivo es realizar una predicción de la etiqueta que le otorgaremos a los nuevos documentos (etiqueta es desconocida) a partir de la relación entre las etiquetas que conoce y otras series de características *(x_p)*. Para realizar un modelo de aprendizaje supervisado que pueda utilizarse en una clasificación de los documentos es necesario contar con una muestra de datos etiquetada que permita la organización y estructuración del texto.

Para la implementación de un campo de entrenamiento (*Training, Test, Validation*) debemos de contar con un *corpus* inicial (*D*) donde cada uno de los documentos de nuestra muestra $D = \{d_1, \dots, d_j\}$, esté asignado a una de las clases o categorías que previamente se han determinado, $C = \{c_1, \dots, c_i\}$, para así, poder crear un conjunto de entrenamiento Φ: $D \times C = \{(d_1, c_1), \dots, (d_j, c_i)\}$ con el cual, tras aplicar el algoritmo adecuado podamos obtener un sistema de clasificación que, tras el aprendizaje, pueda llegar a ser capaz de predecir la clasificación de los futuros documentos. Podemos encontrar diferentes tipos de clasificaciones de los documentos, entre ellos:

8.1.1. Problemas de clasificación binaria.

El etiquetado de spam del correo electrónico o la detección de fraude son un ejemplo de este tipo de clasificación (Rawal, S. *et al*., 2017). En la clasificación binaria, se consideran únicamente dos categorías (dos etiquetas) con un valor para cada una de ellas, por ejemplo 1 y -1. Podemos definir la clasificación o categorización de textos binaria como la aproximación de una función de categoría desconocida mediante la asignación de un valor Booleano (verdadero o falso) Ω: $D \times C \rightarrow \{1,-1\}$. Dentro del conjunto de entrenamiento (Φ) los valores de la función son conocidos para cada par d_j, c_i. Un documento d_j sería un ejemplo positivo de c_i si Φ $(d_j, c_i) = 1$, en cambio se produce un ejemplo negativo de c_i cuando Φ $(d_j, c_i) = -1$.

Existen numerosos algoritmos de clasificación binaria, en este ensayo nos centraremos en las máquinas de vector de soporte o *SVM*. Este es un algoritmo de aprendizaje supervisado creado por Vapnik, V. en 1963 para casos perfectamente separables linealmente, y posteriormente extendida por Cortés, C. & Vapnik, V. en 1995 para casos en los que las clases no son linealmente separables. Inicialmente fue diseñado para problemas de clasificación binaria, pero actualmente se emplea tanto en problemas de clasificación múltiple como regresión.

El funcionamiento de los *SVM* se basa en el concepto de hiperplano, el cual se define como una extensión del plano en un subespacio p-dimensional. Cada una de las características (x) que escogemos para nuestra clasificación representa una dimensión (p). Si el espacio que queremos clasificar tiene p dimensiones, un hiperplano será el subespacio de $p-1$ dimensiones que es capaz de dividirlo en dos mitades. A continuación, se representa el concepto de hiperplano mediante una ecuación lineal:

Ecuación 6. Hiperplano representado mediante una ecuación lineal.

$$f(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

Si los casos fueran perfectamente separables, para conocer a que clase pertenece el documento objetivo solo sería necesario calcular el signo de la ecuación y ver en qué parte del plano se sitúa. Cuando $f(x)$ es positiva el documento pertenecerá a la clase A, en el caso de que $f(x)$ sea negativa pertenecerá a la clase B.

$$Clase\ A: \quad (\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p) > 0$$
$$Clase\ B: \quad (\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p) < 0$$

Cuando $f(x) \rightarrow \{1,-1\}$.

Una de las primeras dificultades que nos encontramos es que existen infinitos hiperplanos capaces de dividir correctamente el espacio p-dimensional en dos categorías. Es por esto por lo que, debemos encontrar el hiperplano óptimo. Este será aquel que se encuentra más alejado de todas las observaciones consiguiendo así maximizar el margen (distancia) entre las observaciones y el hiperplano. Esto se puede conseguir mediante métodos de optimización de las ecuaciones anteriormente descritas.

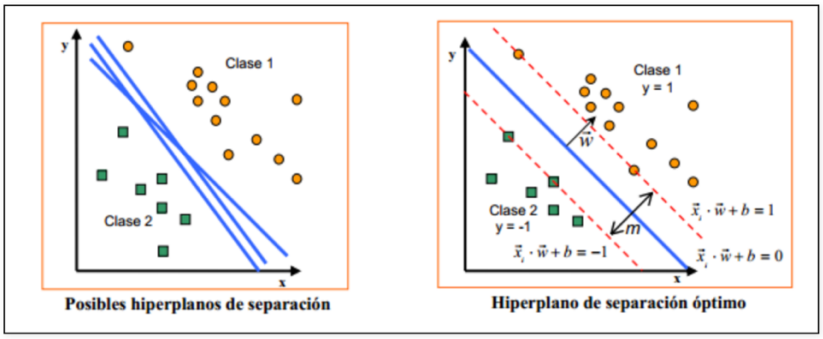

Ilustración 5. SVM: Los hiperplanos de separación y el hiperplano óptimo.
Fuente: Rpubs – Jig Badouin (2022)

Cuando los casos no son perfectamente separables, es mejor crear un clasificador basado en un hiperplano denominado clasificador de vector de soporte (*Support Vector Classifiers)*. El cual, a pesar de no separar las dos clases de una forma perfecta, tenga un pequeño margen de error, pero una mayor robustez y capacidad predictiva. Kowalczyk, A. (2018) en su libro *"Support Vector Machines Succinctly"* presenta una explicación más detallada sobre el funcionamiento de las SVM y su desarrollo matemático. El algoritmo de las SVM se puede considerar como una extensión de los clasificadores de vectores de soporte que nos permite obtener el hiperplano óptimo al aumentar las dimensiones de los datos, *"usando un hiperplano como frontera de decisión entre dos clases"* (Aggarwal, C. C. 2015).

8.1.2. Problemas de clasificación multi-clase (Multiclass Clasification, MCC).

Denominamos multi-clase al problema de clasificar un documento entre dos o más posibles clases. Los clasificadores multi-clase se caracterizan por clasificar nuestro documento en una sola clase dentro de estas *i* clases, es decir, el documento d_j solo puede pertenecer a $c_{i,}$ por lo tanto, solo puede estar asignado a una etiqueta λ_k (Yukinawa, N. *et al*., 2007). Es posible que dentro de este problema de clasificación nos encontremos con documentos a los que se les puede asignar dos o más etiquetas de clase, a este suceso se le denomina problema de clasificación multi-etiqueta.

Para poder realizar la clasificación multiclase primero debemos adaptar el problema generando *n* problemas de clasificación binaria. Existen numerosas estrategias para conseguirlo, pero en este apartado presentaremos el método Uno vs Uno (*One vs One, OvO*). Este método fue popularizado por los investigadores Platt C. J., Cristianini, N & Shawe-Taylor, J. (2000) en su trabajo sobre la clasificación de imágenes. Consiste en entrenar $\frac{(i-1)}{2}$ clasificadores binarios donde cada uno de ellos recibe un par de muestras del conjunto de entrenamiento (Φ) original. Para la predicción de la clase de nuestro nuevo documento se utiliza un sistema de votación, donde todos los clasificadores realizan una votación de las clases. El clasificador combinado escogerá la clase con mayor número de votos como clase predicha. El clasificador Naive Bayes multinomial tiene un carácter multi-clase siendo capaz de observar más de dos clases, es por esto, por lo que es un buen algoritmo que utilizar en este tipo de clasificaciones. A continuación, procederemos a explicar brevemente el funcionamiento de este clasificador.

El clasificador Naive Bayes utiliza la máxima verosimilitud, es decir, se puede utilizar el modelo sin tener que aceptar la probabilidad Bayesiana. Baeza-Yates, R. & Ribeiro-Neto, B. (1995) categoriza este método como la base de la clasificación de textos. Estos clasificadores se caracterizan por su simplicidad, no requieren calcular la matriz de covarianza completa, en su lugar, se calcula exclusivamente la varianza de las variables puesto que se asumen variables independientes, por lo que, solo requieren una pequeña cantidad de datos de entrenamiento para poder predecir la clase de los futuros documentos que se van a clasificar.

El clasificador de Naive Bayes se basa en la Regla de Bayes para calcular la probabilidad condicional de que nuestro documento d_j se encuentre en una clase $c_i \rightarrow P\{c_i|d_j\}$, mediante el cálculo de la probabilidad de que los documentos pertenezcan a esa clase $P\{dj|c_i\}$ y la probabilidad de cada clase obtenida en el conjunto de entrenamiento $P(c_i)$ dividido entre la probabilidad de cada documento.

Ecuación 7. Fórmula del clasificador de Naive Bayes.

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)}$$

Los pasos que realiza en clasificador Naive Bayes son los siguientes: en primer lugar, se calcula la probabilidad de las etiquetas, tras esto, calcula la probabilidad para cada una de las características (atributos), una vez calculada la probabilidad de los atributos, calcula la probabilidad posterior con el Teorema de Bayes y finalmente otorga la clase del documento a aquella que tiene una probabilidad más alta. La principal desventaja de este clasificador es que asume que todas las variables son independientes entre sí por lo tanto no tiene en cuenta la relación existente entre ellas.

8.1.3. Problemas de clasificación multi-etiqueta (Multilabel Clasification, MLC).

La clasificación multi-etiqueta se caracteriza principalmente por la existencia de *i* clases, las cuales no son excluyentes entre sí, permitiendo así que un documento d_j pueda estar asociado a más de una etiqueta de forma simultánea (Alfaro, R. A. & Allende O. 2020). El Aprendizaje Multi-etiqueta *(Multilabel Learning, MLL)* es un ejemplo del aprendizaje supervisado que nos permite analizar las relaciones que presentan las etiquetas, mostrándonos problemas como el desbalance entre etiquetas, donde algunas están muy presentes mientras que otras no tanto. En la *Tabla 2* se presenta un ejemplo de una muestra con un problema de clasificación multi-etiqueta.

Tabla 2.Ejemplo de problema multi-etiqueta.
Fuente: Elaboración propia.

DOCUMENTO (Observación)	ETIQUETA (λq)			
	CIENCIA	POLÍTICA	DEPORTE	VIAJES
Documento 1	X		X	
Documento 2		X		
Documento 3	X	X		
Documento 4			X	X

El método más común para resolver el problema multi-etiqueta es el denominado método de Relevancia Binaria *(Binary Relevance, BR)* que fue introducida por primera vez por el matemático y estadístico británico Austin Bradford Hill en 1920. Este método consiste en transformar el problema multi-etiqueta en *n* problemas binarios, es decir, a cada problema se le asigna un valor de pertenencia y un valor de no pertenencia, el cual, indica si el documento está asociado o no a dicha etiqueta. Para implementar este método se deberán entrenar *n* clasificadores binarios en un mismo conjunto de entrenamiento. La transformación del problema multi-etiqueta se realiza de forma independiente al uso del algoritmo, es por esto, por lo que podemos implementar el método *BR* utilizando algoritmos ya existentes. La *Tabla 2* describe nuestra muestra de ejemplo tras utilizar el método de transformación *BR*.

Tabla 3. Transformación BR del problema multi-etiqueta.
Fuente: Elaboración propia.

DOCUMENTO (Observación)	ETIQUETA (λq)	
	CIENCIA	NO CIENCIA
Documento 1	X	
Documento 2		X
Documento 3	X	
Documento 4		X

DOCUMENTO (Observación)	ETIQUETA (λq)	
	POLÍTICA	NO POLÍTICA
Documento 1		X
Documento 2	X	
Documento 3	X	
Documento 4		X

DOCUMENTO (Observación)	ETIQUETA (λq)	
	DEPORTE	NO DEPORTE
Documento 1	X	
Documento 2		X
Documento 3		X
Documento 4	X	

DOCUMENTO (Observación)	ETIQUETA (λq)	
	VIAJES	NO VIAJES
Documento 1		X
Documento 2		X
Documento 3		X
Documento 4	X	

Este método es efectivo para resolver problemas en los que nuestra muestra pertenezca a un conjunto pequeño de etiquetas puesto que en la medida que *n* aumente, el número de clasificadores aumentara también, pero debemos de ser conscientes de que este método no tiene en cuenta la relación entre las etiquetas.

Para poder hacer frente a esta complejidad se presenta el algoritmo RAkEL (Random k-Labelsets) desarrollado por los investigadores Read, J. *et al*., (2011). Este algoritmo realiza una codificación de las variables del siguiente modo, cogiendo como referencia el ejemplo anterior, si nuestro documento habla de viajes y deporte estas etiquetas se recodificarían pasando a crear una nueva clase llamada viajes_deportes. Tras esto se seleccionan *k* etiquetas para entrenar un clasificador repitiendo esta operación hasta generar *m* clasificadores binarios. Este algoritmo resuelve los problemas anteriores y su mayor dificultad es elegir de forma correcta el número de etiquetas (k) y el número de clasificadores a entrenar (m).

8.1.4. Problemas de clasificación difusa (Fuzzy).

La clasificación difusa (*Fuzzy*) utiliza las técnicas de *ML* para modelos de aprendizaje supervisado y no supervisado. Centrándonos en el aprendizaje no supervisado, por lo general se realiza una categorización difusa a partir de los datos o grupos que obtenemos después de realizar un análisis clúster a nuestros documentos (Porras, J. C.C. 2008). El análisis clúster o clustering en minería de texto consiste en agrupar nuestros documentos no etiquetados del *corpus* en grupos que poseen características similares denominados clústeres.

El algoritmo utilizado más simple y computacionalmente eficiente para resolver el problema de agrupamiento que se presenta en el análisis clúster es el algoritmo de k-medias propuesto por el matemático y estadístico estadounidense Stuart Lloyd en el año 1957. Este algoritmo clasificara nuestros datos en *K* grupos, los cuales tendrán *K* centroides representados el centro de cada grupo.

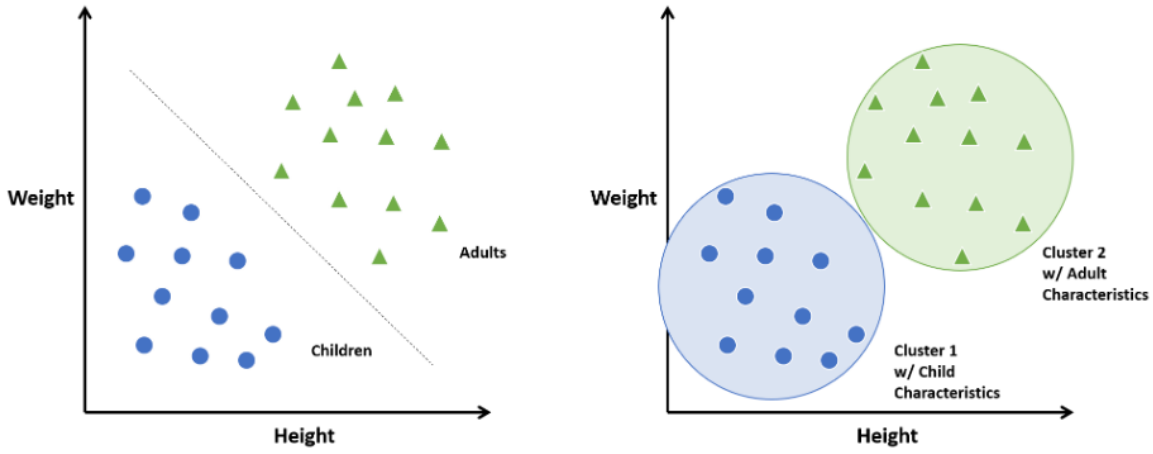

Ilustración 6. Diferencia entre clasificación y clustering.
Fuente: Comunidad Huawei Entrerprise (2022)

El algoritmo K-medias utiliza habitualmente la distancia cuadrática, se caracteriza principalmente por ser un algoritmo de carácter iterativo, es decir, su funcionamiento se basa en colocar aleatoriamente cada centroide en el espacio vectorial y calcula la distancia entre los centroides y los puntos cercanos, consiguiendo así que el centroide se mueva al centro de cada punto más cercano. Se realizan estos pasos iterativamente hasta que la posición de los grupos ya no cambia más. El algoritmo K-medias minimiza las distancias cuadráticas de cada documento al centroide de su clúster, minimizando la siguiente función objetivo (f):

Ecuación 8. Fórmula para calcular el algoritmo K-medias.

$$f = \sum_{m=1}^{k} \sum_{j=1}^{d} \left\| x_p^{(m)} - s_m \right\|^2$$

Donde $k_m$es el número de clúster, d_j representa el número de documentos, x_p el número de casos y s_m el centroide del clúster m.

La lógica difusa nos permite obtener conclusiones a partir de información cualitativa imprecisa, ambigua o incompleta definiéndose en grados variables de pertenencia a las clases. Lo que diferencia a la lógica difusa de los otros problemas de clasificación es que, en este caso, un documento d_j pertenece a una clase c_i con un grado de veracidad entre 0 y 1 (Pérez Pueyo, R. 2005). Esta medida se denomina grado de pertenencia (∂) y significa que documento puede no ser únicamente un ejemplo verdadero de c_i si Φ (d_j, c_i) = 1 o un ejemplo negativo de c_i cuando Φ (d_j, c_i) = -1, si no que puede ser también difuso cuando sea igual a un valor intermedio. Por ejemplo, si queremos realizar una clasificación de la temperatura donde el calor toma el valor de 1 y el frio toma el valor de -1 diremos que:

- Si hay 40 ºC hace calor con valor de 1.
- Si hay 30 ºC hace calor con valor 0,5.
- Si hay 0 ºC grados hace calor con valor 0,1.

Es por esto, por lo que la clasificación difusa nos permite describir aspectos semánticos, permitiendo la clasificación múltiple mediante el grado de pertenencia del documento a cada clase. El criterio más utilizado para esta clasificación es el cálculo del índice *TF-IDF (Term frequency – Inverse document frequency)* de Salton, G. & Buckley, C. (1989), esta es una medida numérica que expresa la relevancia de un documento dentro de una colección.

8.1.5. EJEMPLO DE CLASIFICACIÓN BINARIA – Películas Románticas / No románticas.

El objetivo es realizar una clasificación binaria a partir del *corpus* preprocesado obtenido en el apartado 5.5. El código empleado para la elaboración de este ejemplo práctico está recogido en el **Anexo III**. Para esta clasificación, asignaremos la etiqueta 1 a aquellas reseñas que contengan una crítica a películas de género romántico y la etiqueta 0, en el caso de que el género de la película sea otro. Para etiquetar el corpus utilizaremos el siguiente listado de palabras:

- "love", "affection", "amorous", "passion", "tender", "girlfriend", "boyfriend", "sentimental", "sensual", "sweetheart", "heart".

Debemos de tener precaución con la palabra "love" puesto que, en función de su contexto, puede generar ambigüedad en la clasificación.

Comenzamos importado las bibliotecas necesarias: "numpy", "sklearn" y "tabulate" y "matplotlib" así como varios módulos específicos de estas bibliotecas. Una vez importados todos los recursos necesarios, definimos las etiquetas, convertimos el *corpus* preprocesado en una matriz de características y posteriormente dividimos el conjunto de datos en: datos de entrenamiento (500) y datos de prueba (500).

Tras esto, realizaremos la predicción del conjunto de prueba mediante SVM. Comenzamos buscando los mejores hiperparámetros mediante validación cruzada considerando los diferentes tipos de kernel: "linear", "rbf", "poly" y "sigmoid" junto con los valores de "C "y "gamma". Como resultado obtenemos los valores óptimos para los parámetros que constituirán el mejor modelo de clasificación SVM:

"C" → 0,5, "gamma" → 0,1 y "kernel" → "linear".

Por último, se imprime el informe de clasificación, el cual, nos brinda métricas como la precisión, recall, puntuación F1 poder evaluar cada una de las clases individualmente y el modelo de forma general.

	Precision	**Recall**	**F1-score**	**Support**
Género Romántico (1)	0.88	0.60	0.71	189
Género No Romántico (0)	0.80	0.95	0.87	311
Accuracy			0.82	500
Macro Avg	0.84	0.77	0.79	500
Weighted Avg	0.83	0.82	0.81	500

Ilustración 7. Informe de clasificación.
Fuente: Elaboración propia.

El modelo presenta unos buenos resultados generales con 189 documentos clasificados en la clase de genero romántico y 311 documentos en la clase de genero no romántico, con una precisión ponderada del 82%. La clase de género romántico presenta una precisión del 0.88, lo que indica que el 88% de las clasificaciones positivas son clasificadas de forma correcta, el recall indica que se han identificado de forma correcta el 60% de los documentos de genero romántico y por último el F1-score muestra un buen equilibrio entre la precisión y el recall, exactamente de 0.71.

En el caso de la clase de género no romántico, obtenemos unos resultados más favorables. La precisión es de 0.80, por lo tanto, el 80% de las clasificaciones negativas son correctas, en este caso, el recall muestra que se han identificado correctamente un 95% de los documentos y un equilibrio entre estas métricas razonablemente mejor que para la clase positiva, de 0.87.

Para una evaluación más completa de nuestra clasificación se ha calculado la Curva de ROC a la cual, se ha agregado una línea diagonal que representa la clasificación aleatoria:

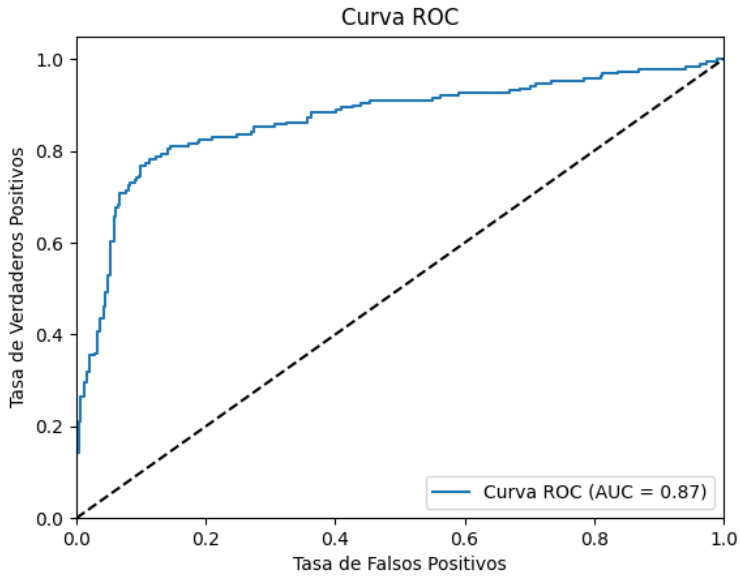

Ilustración 8. Representación de la curva de ROC.
Fuente: Elaboración propia.

Se ha calculado el umbral óptimo, es decir, la frontera que delimita la clasificación positiva y la negativa obteniendo un valor de – 0.2903. Aquellos documentos que superen este umbral serán clasificados como románticos, mientras que aquellos que no lo superen serán clasificados como no románticos.

En cuanto a la interpretación de la curva de ROC, obtenemos un AUC de 0.86. Esto nos indica un buen rendimiento del modelo según Fawcett, T. (2006), pero mejorable. En términos de interpretación, la curva de ROC muestra una tendencia favorable, aproximándose al límite superior izquierdo (alta tasa de verdaderos positivos – baja tasa de falsos positivos) en muchos puntos de la curva, por lo tanto, podemos concluir que el modelo es capaz de distinguir entre la clase positiva y la negativa en un porcentaje significativo de documentos.

8.2. ELABORACIÓN DE RESUMENES.

Reunir todos los datos disponibles y resumirlos es una tarea tediosa, por esto, han surgido aplicaciones o IA capaces de generar resúmenes automáticamente. Estas aplicaciones tienen como objetivo principal descubrir patrones y relaciones intrínsecas con un enfoque lingüístico y semántico para conseguir elaborar un resumen que contenga los aspectos relevantes del texto y a su vez sea coherente. Su funcionamiento consiste en condensar el documento en una versión más corta que ofrezca un resumen claro y conciso, ayudando a realizar las tareas de recuperación de archivos de textos largos y ahorrando tiempo (D. Singhal *et al.*, 2020).

8.2.1. Resúmenes basados en extracción de contenido.

Si el objetivo de nuestro análisis es obtener una comprensión general y rápida de uno o varios textos de amplia extensión, la elaboración de un resumen mediante la extracción de contenido es una buena opción. Esta técnica se fundamenta en identificar las frases y palabras claves que son capaces de transmitir la esencia del contenido. El proceso de extracción se lleva a cabo mediante algoritmos de lenguaje automático y el análisis del lenguaje natural.

Es importante tener en cuenta que estas técnicas se basan en la extracción de subconjuntos, por lo tanto, ignoran la estructura del discurso subyacente en el documento. Para poder generar un resumen coherente es importante el análisis del lenguaje natural y la estructura, la cual, determina las relaciones entre las unidades textuales y su relevancia. Según Hirao, T. *et al*., (2015) esta estructura puede representarse como un árbol cuyas aristas representan la relación de dependencia entre unidades textuales (cabeza – modificador o padre – hijo) y su profundidad de nodo representa la visibilidad de las unidades textuales.

El lingüista francés Lucien Tesnière fue pionero al utilizar un árbol de dependencias para analizar la estructura sintáctica de una oración en 1969. Tesnière propuso el primer modelo de análisis sintáctico inicialmente denominado "stemma", actualmente conocido como "gramática de dependencia" en su libro "Éléments de syntaxe structurale". Donde representaba las relaciones sintácticas entre palabras en una oración mediante un árbol de dependencias.

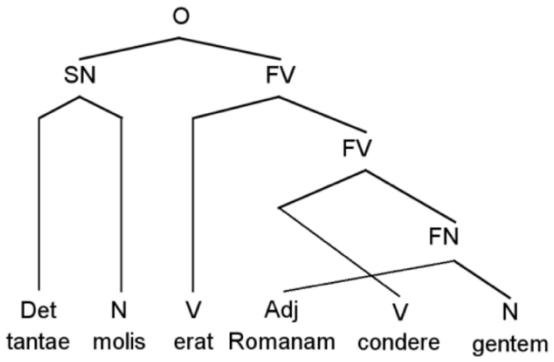

Ilustración 9. Árbol de dependencias.
Fuente: "Éléments de syntaxe structurale" - Luciene Tesniére

En cuanto a los algoritmos de lenguaje automático más frecuentes, se ha definido la *Tabla 4*. donde se detalla brevemente su funcionamiento:

	Extracción de frases	**Extracción de oraciones**	**Extracción de párrafos**	**Extracción de términos clave**
1ª Fase	Dividir el texto en frases (Tokenización) utilizando la puntuación	Dividir el texto en frases (Tokenización) utilizando la puntuación	Técnicas de segmentación de párrafos	Preprocesamiento del texto y tokenización
2ª Fase	Análisis y etiquetas sintácticas	Agregar los tokens a una oración temporal	Agregar cada párrafo a una lista	Calcular TF - IDF y análisis semántico
3ª Fase	Calcular el peso de cada frase	Almacenar las oraciones en una lista	Validar los párrafos	Seleccionar las palabras / Frases clave con mayor peso
4ª Fase	Seleccionar las frases con mayor peso	Elegir las oraciones mas adecuadas	Almacenar los párrafos válidos para su análisis	Utilizar técnicas de ML para generar el resumen
		RESUMEN		

Tabla 4. Algoritmos para la elaboración de resúmenes mediante la extracción de contenidos.
Fuente: Elaboración propia.

8.2.2. Resúmenes basados en abstracción de contenido.

Si nuestro objetivo es elaborar un resumen completamente nuevo, similar al que podría elaborar una persona, generando nuevas frases capaces de resumir de forma eficiente y coherente las ideas principales del texto, necesitamos utilizar técnicas de procesamiento del lenguaje natural. Para generar un resumen basado en la abstracción es necesario identificar las ideas principales y las relaciones existentes entre ellas para sintetizar un resumen conciso y coherente que capture la esencia original. La calidad final del resumen depende en gran medida de escoger un algoritmo que pueda comprender y sintetizar de manera precisa el contenido del texto original.

Uno de los modelos que nos encontramos es el modelo Secuencia a Secuencia (Seq2Seq) utilizado por Google en 2016 con mucho éxito en tareas de traducción (google translate), resumen de texto y subtítulos de videos. Este modelo se basa en redes neuronales recurrentes. Este algoritmo se pre-entrena con grandes cantidades de texto sin etiquetar mediante tareas de auto-regresión en las que se le presenta un fragmento de texto y se entrena para predecir la siguiente palabra, es decir, predecir una secuencia X a partir de una secuencia Z (Phuong, M & Hutter, M 2022).

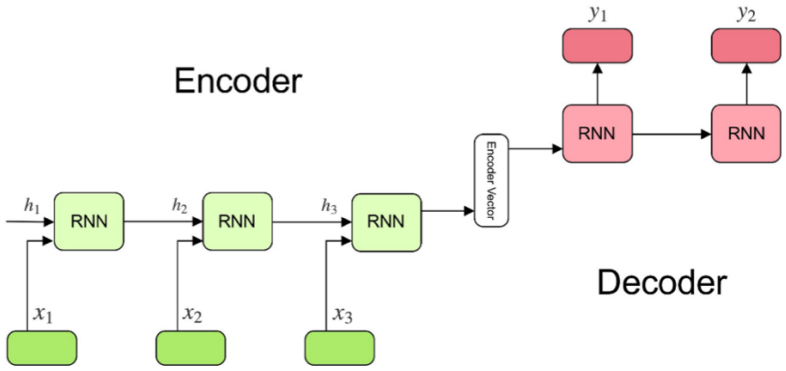

Ilustración 10. Arquitectura interna del modelo Seq2Seq
Fuente: "Análisis del estado del arte de la generación de texto con redes neuronales mediante modelos de Transformer" - Gozalo, J. S. (2020).

Uno de los modelos más polémicos actualmente es el modelo Generative Pre-Trained Transformer o más conocido como GPT (modelo base de la aplicación ChatGPT introducida por OpenIA en 2019) que ha destacado por su capacidad para generar resúmenes / texto de forma coherente y de alta calidad. Este modelo del lenguaje se basa en la arquitectura Transformer, la cual se diferencia del resto, por el uso de capas de atención en lugar de redes neuronales recurrentes. López-Brea, D. R. (2022) explica su funcionamiento del siguiente modo:

- Mecanismo de atención: Permite capturar las relaciones de dependencia entre las palabras. Se calcula mediante tres matrices de pesos: consultas (queries), claves (keys) y valores (values).

- Capas de atención múltiples y cabezas de atención: Trabajan de forma conjunta para captar dependencias relaciones entre las palabras. Cada capa de atención no se basa en una sola función de atención, si no, en múltiples funciones de atención paralelas denominadas "cabezas".

- Codificador y decodificador: El codificador (encoder) es utilizado para leer y comprender la entrada (input) y el decodificador (decoder) es utilizado para generar las salidas (output), un output por input introducido.

8.3. MINERÍA DE OPINIONES / ANÁLISIS DE SENTIMIENTOS.

El análisis de sentimientos es el estudio de los sentimientos de las personas, opiniones o emociones en determinados textos, esto también se conoce como minería de opiniones. El análisis de sentimientos se ocupa de la detección y clasificación automática de opiniones expresadas en textos escritos en lenguaje natural (Wiegand, M. *et al*., 2010). Para los consumidores y empresarios, conocer las opiniones acerca de sus productos es de gran utilidad. El análisis de sentimientos ha sido objeto de una activa investigación con estudios y publicaciones académicas que se remontan a los años 60-70. El objetivo principal de este análisis es extraer automáticamente las opiniones relacionadas con el contenido de interés generadas por los usuarios.

Liu, B. (2012) establece tres categorizaciones generales para la minería de sentimientos: A nivel de documento, a nivel de oración y a nivel de concepto. Además de la categorización, Liu, B. (2012) indica que las tareas a nivel documento se definen como un problema de clasificación en el que, en primer lugar, el documento se debe definir como documento objetivo o subjetivo para después poder aplicar el análisis de sentimientos sobre los documentos subjetivos clasificándolos como: positivos, negativos o neutros. En el nivel de frase y el nivel de concepto, cada frase / concepto se consideran una entidad y a través del método suma se calcula el resultado global del documento.

8.3.1. Sentimiento construido por herramientas de machine learning.

A lo largo de la historia se han desarrollado numerosos métodos de clasificación de emociones. Sin embargo, una de las herramientas más utilizadas es la rueda de las emociones de Plutchik R. (1980). Esta rueda permite a los investigadores identificar las emociones primarias y secundarias presentes en los textos. Se presentan ocho emociones primarias: alegría, tristeza, ira, miedo, disgusto, sorpresa, anticipación y confianza. Cada una de ellas ostenta su opuesto y, al combinarse, dan lugar a emociones secundarias como el amor, la culpabilidad, la envidia, el orgullo, la vergüenza y la preocupación, tal como se detalla en la *Ilustración 10.*

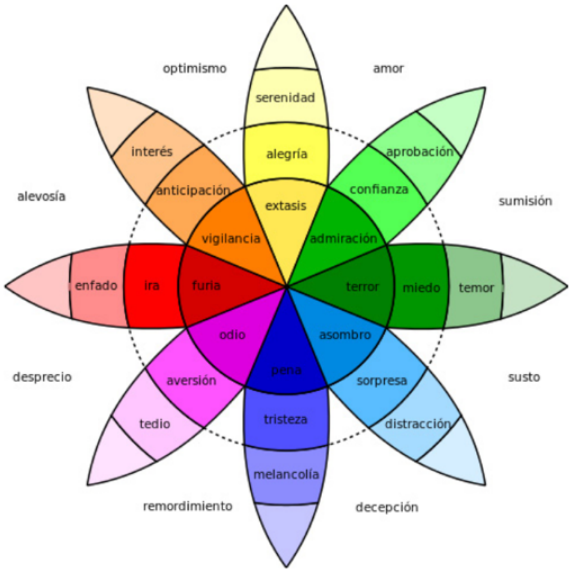

Ilustración 11. Rueda de las emociones de Plutchik.
Fuente: Plutchik R. (1980).

Por otro lado, encontramos el análisis de polaridad, el cual consiste en identificar las frases o palabras que expresan emociones u opiniones y se clasifican según su polaridad: positiva, negativa o neutra (Murillo, E. C. & Leoni, J. A. 2016). Existen diversas técnicas para realizar el análisis de polaridad, y una de las más utilizadas es el análisis de diccionario (análisis léxico). Este método implica la creación de diccionarios de palabras positivas y negativas, las cuales se utilizan para identificar la polaridad del texto. A cada palabra del diccionario se le asigna una puntuación de polaridad (positiva, negativa o neutra), y estas puntuaciones se utilizan para calcular la polaridad general del texto mediante una fórmula específica:

Ecuación 9. Fórmula para el cálculo de la polaridad.

$$Polaridad = \frac{\sum PP - \sum PN}{\sum PP + \sum PN + \sum PO}$$

Donde *PP* se refiere a las puntuaciones de palabras positivas, *PN* hace referencia a las puntuaciones de palabras negativas y *PO* se refiere a las puntuaciones de palabras neutrales.

8.3.2. Algoritmo de redes neuronales – Perceptrón multicapa

Las redes neuronales artificiales *(RNA)* son un modelo que se inspira en el funcionamiento del cerebro humano. Se componen de un conjunto de nodos conectados donde cada neurona (nodo), realiza una operación matemática en los datos de entrada y produce una salida que se transmite a las neuronas de la siguiente capa (Guan, S. 2023). Una de las funciones de las redes neuronales artificiales es aprender patrones con los datos de entrenamiento y clasificar los textos en función de su polaridad.

El perceptrón simple, es un modelo capaz de resolver exclusivamente problemas lineales. Rumelhart, D. E. *et al*., (1986) demostraron que aumentando el nivel de capas del perceptrón y propagando los errores hacia las capas ocultas el perceptrón es capaz de trabajar con múltiples capas y funciones de activación no lineales.

Según Mishra, K. M. & Huhtala, K. J. (2019) el perceptrón multicapa *(MLP)* es una *RNA* en la que todas las neuronas son capaces de reaccionarse entre sí, incluyendo las capas ocultas. La estructura del *MLP* contiene las siguientes capas:

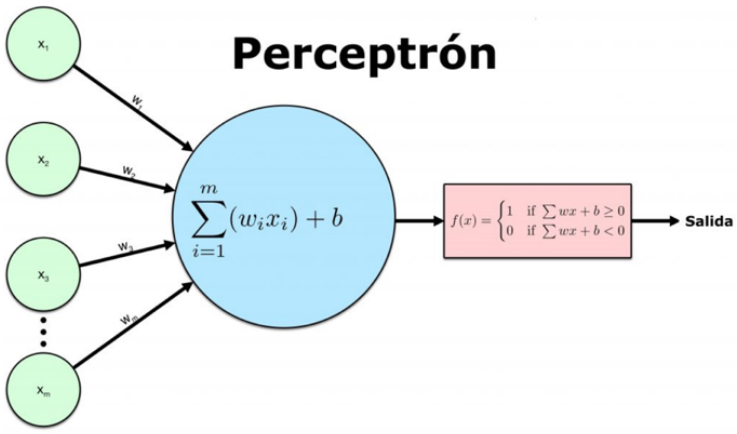

Ilustración 12. Ilustración de un perceptrón red neuronal.
Fuente: José Mariano Álvarez (2018)

- Capa de entrada. Esta capa recibe los datos de entrada, cada palabra del texto se puede representar como un vector numérico mediante distintas técnicas como la TF-IDF.
- Capas ocultas. Estas capas procesan la información de entrada generando una representación numérica de la polaridad del texto.
- Capa de salida. En esta capa se genera el modelo, en nuestro caso, la polaridad del texto.

Durante el entrenamiento de la *RNA*, se ajustan los pesos y las funciones de activación con el objetivo de minimizar el error entre la salida producida por la red y la salida deseada. Tras el entrenamiento, la *RNA* puede usarse para realizar predicciones acerca de nuevos datos de entrada. En cuanto a la función de activación, se aplica a la salida de cada neurona para determinar si esta debe activarse o no. La elección de la función de activación depende del problema al que se enfrente el investigador. Calvo, D. (2018) explica detalladamente el funcionamiento de la función de activación y el funcionamiento de las más comunes: sigmoide, ReLU y tangente hiperbólica. Representadas matemáticamente del siguiente modo:

Ecuación 10. Función Sigmoide.

$$f(x) = \frac{1}{(1 + e^{(-x)})}$$

La función sigmoide devuelve un valor entre 0 y 1.

Ecuación 11. Función ReLU.

$$f(x) = \max(0, x)$$

La función ReLU devuelve un valor cualquiera si es positivo y el valor 0 si es negativo.

Ecuación 12. Función tangente hiperbólica.

$$f(x) = \frac{e^{x} - e^{(-x)}}{e^{x} + e^{(-x)}}$$

Por último, la función tangente hiperbólica devuelve un valor entre -1 y 1.

8.3.3. Sentimiento construido por medio de diccionarios lexicográficos - SentiWordNet

El sentimiento construido por diccionarios lexicográficos pretende determinar la polaridad de un texto en función de la presencia y frecuencia de palabras positivas y negativas recogidas previamente en un diccionario. Este diccionario debe de recoger un listado de palabras con una polaridad ya asignada, esta polaridad generalmente oscila entre los valores -1 y 1 (donde los valores negativos se asignan a polaridades negativas y los positivos a polaridades positivas, siendo 0 el valor que indica una polaridad neutra). Para determinar la polaridad final del texto el procedimiento es sencillo, en primer lugar, sumar los valores de polaridad obtenidos y tras esto, se divide esta suma entre la cantidad total de palabras del texto. Si obtenemos un valor positivo, la polaridad del texto será positiva, en caso contrario tendremos una polaridad negativa.

En este TFM, nos centraremos en uno de los diccionarios léxicos de polaridad más utilizados: SentiwordNet. La primera versión de este diccionario se lanzó en el año 2006 por Esuli, A. & Sebastiani, F. como un recurso léxico a disposición de todo el público para fines de investigación.

En la actualidad SentiWordNet 3.0 de Pang & Lee (2008) mejora el diseño del 1.0, y está concebido explícitamente para dar soporte a aplicaciones de clasificación de sentimientos y minería de opiniones. SentiwordNet es una base de datos que contiene palabras con su polaridad, manejando las palabras de negación como "no", "nunca" etc., las cuales afectan a la polaridad de las frases. Baccianella, S. *et a*l., (2010) explica el funcionamiento de este diccionario mediante cuatro pasos:

- Expansión del conjunto "semilla". La unidad básica de WordNet son los synsets (conjuntos de sinónimos), estos representan un concepto de forma léxica y están formados por siete términos que otorgan una polaridad especifica. (Vázquez, Y. G. 2012). Existen dos pequeños conjuntos "semilla". Uno de los conjuntos contiene todos los synsets con términos "paradigmáticamente positivos" y el otro contiene en todos los synsets con términos "paradigmáticamente negativos" (Turney, P. D., & Littman, M. L. 2003). Estos conjuntos se expanden automáticamente, recorriendo una serie de relaciones binarias de WordNet, las cuales, pueden mantener o invertir las propiedades positivas y negativas.

- Entrenamiento del clasificador. Durante este paso, los dos conjuntos generados anteriormente se unen a otro conjunto de sinónimos (objetivo) para entrenar un clasificador.

- Clasificación de sinónimos. Todos los synsets de WORDNET se clasifican en "positivo", "negativo" u "objetivo" con el clasificador generado en el paso 2.

- Combinación del clasificador. Para que la clasificación sea más precisa (como explican en detalle Esuli, A. & Sebastiani, F. 2006), es más eficiente generar varios clasificadores con diferentes parámetros, los cuales se combinan en este paso para obtener los resultados de clasificación finales.

8.3.4. EJEMPLO 1. ANÁLISIS DE SENTIMIENTOS – Asignación automática de polaridad

A modo de demostración de la asignación de polaridad automática de Python, se ha etiquetado el *corpus* preprocesado (**processed_corpus []**) utilizando el módulo "SentimentIntensityAnalizer" de la biblioteca "nltk". Este módulo, asigna una puntuación a cada uno de los documentos y en base a esto, les otorga una polaridad negativa si la puntuación es menor de - 0,05, positiva si es mayor de 0,05 y neutra si se encuentra entre -0,05 y 0,05. El código empleado está disponible en el **Anexo IV.**

Tabla 5. Resultados de la asignación de polaridad con la biblioteca NLTK.
Fuente: Elaboración propia.

Etiqueta	N° Documentos
Positivo	688
Negativo	307
Neutro	4
N° Total de documentos	**1000**

Al contrastar los resultados de la asignación de polaridad automática con las polaridades previamente conocidas, observamos que "SentimentIntensityAnalizer" genera 192 clasificaciones erróneas. En otras palabras, aproximadamente al 80,8% de los documentos se les ha otorgado una polaridad correcta, pero existe un rango de error de aproximadamente un 19,2%.

8.3.5. EJEMPLO 2. ANÁLSIS DE SENTIEMINTOS – Redes neuronales – Perceptrón.

El código empleado para el análisis de sentimientos está disponible en el **Anexo V**. Antes de comenzar el análisis de sentimientos generamos un array denominado **processed_corpus_with_polarity[]** el cual, contiene el *corpus* preprocesado y etiquetado según su polaridad conocida:

Tabla 6. Resultados de la asignación de polaridades conocidas.
Fuente: Elaboración propia.

Etiqueta	N° Documentos
Positivo	500
Negativo	500
N° Total de documentos	**1000**

Una vez asignadas las polaridades, procedemos con el análisis de sentimientos utilizando las bibliotecas "sklearn", "random" y "seaborn" y diversos módulos de la biblioteca "sklearn" como "MLPClassifier" o "Perceptrón". Una vez importadas las bibliotecas, dividimos nuestro *corpus* en características (X) e (Y) en función de su polaridad y creamos los conjuntos de entrenamiento (400) y prueba (600). Tras esto, entrenamos nuestro modelo perceptrón y realizamos las predicciones.

Para interpretar los resultados, imprimimos el informe de clasificación que nos presenta la precisión, el Recall y el F-1 Score:

Tabla 7. Informe de clasificación.
Fuente: Elaboración propia.

	Precision	Recall	F1-score	Support
Negativa	0.80	0.74	0.77	314
Positiva	0.73	0.80	0.76	286
Accuracy			0.77	600
Macro Avg	0.77	0.77	0.76	600
Weighted Avg	0.77	0.77	0.77	600

Nuestro modelo perceptrón presenta buenos resultados globales, 314 de los documentos han sido clasificados como documentos negativos, mientras 286 han sido clasificados como positivos con una precisión general del 77%. El modelo es capaz de clasificar de forma correcta el 80% de los documentos negativos y el 73% de los documentos positivos con un equilibrio del 77%.

Tras una primera interpretación de los resultados del modelo, calculamos la matriz de confusión con el módulo de "sklearn.metrics" y la presentamos en forma de gráfico, obteniendo los siguientes resultados:

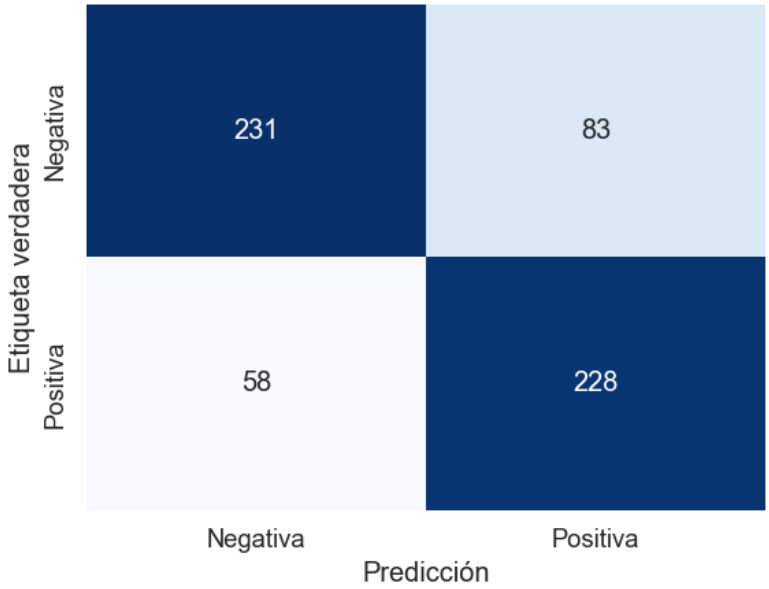

Ilustración 13. Matriz de confusión.
Fuente: Elaboración propia.

Observamos que se han clasificado de forma correcta la mayoría de los documentos negativos (TN = 231) y de los documentos positivos (TP = 228). En cuanto a los falsos positivos, los documentos que en realidad son "negativos" pero han sido clasificados como "positivos" encontramos una cifra significativa (FP = 83). En cambio, los falsos negativos, aquellos documentos "positivos" que se han clasificado como "negativos" la cantidad disminuye (FN = 58).

9. IMPLICACIONES ÉTICAS Y DE PRIVACIDAD.

Establecer un código ético en todos los campos de investigación es necesario. En el campo de la inteligencia artificial, muchos investigadores son capaces de describir la conducta de un individuo basándose exclusivamente en sus registros digitales, lo cual, en ciertas ocasiones puede dar lugar a una discriminación. Para explicarlo mejor, pondremos un ejemplo a través de un estudio de Car Vertical descrito por Lois, A. (2022). En este estudio se han utilizado aproximadamente 10.000 informes obtenidos de sus propios mercados europeos durante el periodo de mayo 2002 a mayo 2022. La principal conclusión obtenida es que los individuos que tienen un coche rojo son más propensos a sufrir accidentes que, aquellos individuos que tienen un coche de otro color. Estos resultados pueden modificar (de hecho, ya los modifican, basándose en este y otros estudios) los precios prestados por las aseguradoras a los vehículos de color rojo. Desde un punto de vista ético, esto no sería totalmente correcto, puesto que esta generalización ha perjudicado a aquellos individuos que tienen un coche rojo y no han tenido ningún accidente.

Para evitar este tipo de generalizaciones, los investigadores necesitan garantizar que los datos recolectados y utilizados se basan y cumplen las normas aplicables, evitando sesgos y discriminación. Estos sesgos pueden surgir, por ejemplo, debido a la falta de representatividad de la muestra, provocada por una exclusión de ciertos grupos o la presencia de variables confusas (Crawford, K. 2013). Otro de los puntos que se deben de tener en cuenta, es la interpretabilidad de los modelos y algoritmos utilizados, proporcionando una explicación clara de los métodos, criterios de selección de características, parámetros y limitaciones del modelo con el fin de evitar consecuencias negativas en la sociedad (O'Neil, C. 2016).

Por otro lado, encontramos uno de los aspectos más críticos de la minería de texto y datos: la privacidad de los datos personales. Al realizar un análisis de investigación es frecuente encontrarse con información sensible que afecta a la privacidad de los usuarios y propietarios de estos datos. La privacidad puede definirse como el derecho de un individuo a controlar la divulgación y el uso de su información personal (Westin, A. F. 1967). En cuanto a los datos personales, por otro lado, se refieren a cualquier información relacionada con una persona identificada o identificable: el nombre, la dirección, el número de teléfono, las preferencias de compra, los datos biométricos, etc. (European Parlament and Council, 2016). La Directiva 95/96/CE del Parlamento Europeo y del Consejo y la Ley Orgánica 15/1999 de protección de datos de carácter personal establece que cualquier individuo es titular de los datos que le afectan personalmente.

Actualmente esta recopilación de información (incluyendo los datos personales) se realiza de un modo “silencioso” y muchos de los usuarios no conocen la gran cantidad de información que generan y los peligros que pueden causar (Garriga-Domínguez, A. 2009). Es decir, todas las compras que realizamos electrónica o presencialmente, llamadas telefónicas, consultas de internet, etc., quedan registradas y permiten a la empresa u otros usuarios hacer uso de ellos si el titular no indica lo contrario.

Para evitar comprometer estos datos personales y vulnerar la privacidad de los usuarios, es esencial cumplir con las normas de protección de datos aplicables. En nuestro caso, el reglamento General de Protección de Datos (GDPR). Además de este reglamento la Comisión europea ha presentado la iniciativa de IA ética, que busca desarrollar un marco legal para la IA y el ML (European Commission, 2021). Nosotros como investigadores, debemos asegurar la anonimización de estos datos eliminando o enmascarando toda información sensible que pueda identificar a los individuos / entidades involucradas. Si no fuera posible, utilizaremos enfoques de agregación de datos y técnicas de preservación de la privacidad como el aprendizaje federado y la privacidad diferencial (Kairouz, P. *et al*., 2021).

10. CONCLUSIONES.

El presente Trabajo de Fin de Máster tiene como objetivo explorar el espectro de aplicaciones de la minería de texto. Específicamente se centra en áreas como la clasificación binaria, múltiple, multi-clase y difusa, así como en la generación automática de resúmenes y análisis de sentimientos u opiniones. Además de esto, se abordan las etapas precedentes a estas aplicaciones como son el preprocesamiento, vectorización o modelado y la representación o visualización de datos de texto. Este TFM pretende presentar los procedimientos y ejemplos con un nivel de detalle riguroso, pero a su vez de un modo sencillo de entender para aquellos lectores que sean nuevos en la materia, se pretende facilitar la comprensión de las metodologías y ejemplos desarrollados.

En cuanto a la aplicación de la minería de texto en el ámbito empresarial, en el contexto de clasificación, las empresas son capaces de gestionar y organizar de forma eficiente los contenidos de sus bases de datos y páginas web. Mediante la elaboración de resúmenes, se pueden resumir grandes volúmenes de información para una mejor toma de decisiones. Por último, el análisis de sentimientos permite comprender la percepción de los consumidores acerca de su producto y ejercer un control preciso sobre la reputación de su marca.

- Conclusiones específicas sobre los resultados prácticos del TFM:

En cuanto al lenguaje de programación utilizado en los ejemplos prácticos, cabe destacar que Python es una herramienta sumamente eficaz y versátil para el análisis de datos de texto. Cuenta con una amplia variedad de bibliotecas y módulos especializados para este objetivo como: NLTK, spaCy o Sickit-learn. Aunque el lenguaje Python resulta sumamente eficaz para desempeñar estas tareas, existen otros lenguajes igualmente aptos como R, RapidMiner, Weka, o Matlab. La elección de uno u otro dependerá del investigador y los objetivos de la investigación.

En la parte del preprocesamiento el objetivo principal ha sido mostrar de la forma más completa posible las técnicas de preprocesado de textos. Los resultados obtenidos han sido satisfactorios, si bien es cierto que, se observa una carencia de estructura semántica. Una de las recomendaciones usuales para solventar este tipo de carencias es realizar una corrección ortográfica inicial de la muestra. Por otro lado, se podía haber considerado una eliminación de los números (tanto numéricos como escritos) puesto que para nuestros análisis posteriores carecen de relevancia. Finalmente, en el ámbito del análisis de sentimientos, se podría considerar el manejo de las negaciones. Estas tienen la capacidad de invertir el sentido de una oración y pueden provocar problemas a la hora de interpretarlas.

En relación con los resultados obtenidos en la tarea de clasificación, el modelo SVM presenta una buena capacidad para diferenciar entre clases en un número significativo de los documentos analizados. No obstante, estos resultados están lejos de ser impecables, es por esto por lo que se considera que la ambigüedad inherente a la palabra “love” influye en el sesgo de los resultados obtenidos. Cabe destacar que, se ha recogido un recall significativamente bajo en la clase positiva (genero romántico), lo cual significa que los documentos pertenecientes a esta clase no están siendo identificados de forma precisa, quizás se deba al excesivo preprocesamiento, que provoca una limitación en el modelo para capturar las características del género romántico o simplemente el algoritmo utilizado no era el más adecuado.

Por otro lado, el análisis de sentimientos presenta unos resultados satisfactorios, a pesar de que incluye errores significativos en la clasificación. Tanto en la clasificación como en el análisis de sentimientos es posible obtener unos mejores resultados al emplear algoritmos alternativos, los algoritmos utilizados en este TFM se han empleado con un enfoque demostrativo, pero existe una amplia gama de algoritmos disponibles.

La restricción de recursos computacionales ha influido tanto en la elección de estos algoritmos como en la reducción de la muestra analizada a 1.000 documentos. En una perspectiva alternativa donde se pudiera disponer de la muestra completa (50.000 documentos) los resultados obtenidos hubieran tenido un alcance y precisión mayor.

- Conclusiones generales acerca de la minería de texto y nuevos desafíos:

La minería de texto, a pesar de sus notables avances en los últimos años, se enfrenta aun a un número considerable de desafíos y limitaciones. A continuación, examinaremos algunos de los ámbitos de investigación que confrontan estos desafíos.

En numerosas ocasiones, nos encontramos con datos de texto que se caracterizan por su ambigüedad. En este sentido, sería interesante investigar el desarrollo de métodos y técnicas específicas que sean capaces de abordar es desafío. Además de investigar en técnicas de preprocesamiento de datos que sean sensibles a estas características.

Otra de las limitaciones de la minería de texto con la que nos encontramos es la diversidad lingüística presente en los textos. Ser capaces de interpretar un *corpus* que contenga documentos redactados en diferentes idiomas es una tarea compleja y costosa. Un ámbito de investigación emergente y de relevancia es el análisis del texto multilingüe, en el que, se busca aplicar las técnicas de minería de texto mediante un enfoque global, considerando un conjunto de documentos de diversas lenguas.

Por otra parte, otro de los desafíos que presenta la minería de texto es la adaptación de las técnicas para el análisis de los datos multimedia. En numerosas ocasiones nos encontramos textos presentes en otros tipos de datos como imágenes o videos. Una línea de investigación interesante consistiría en la extracción y análisis de los componentes textuales inherentes a estos formatos multimedia.

En último lugar, pero no menos relevante, encontramos el ámbito de la ética y la privacidad en el contexto de la minería de texto. Investigaciones dedicadas al desarrollo de técnicas de anonimización de datos y la implementación de prácticas éticas en la recolección, almacenamiento y uso de la información son otro de los ámbitos interesantes de estudio.

11. BIBLIOGRAFÍA.

Addati, A. & Roger, S. (2017). *"Agentes Inteligentes y Web Semántica: preprocesamiento de Texto de redes Sociales". Grupo de Investigación en Lenguajes e Inteligencia Artificial. Departamento de Teoría de la Computación – Facultad de Informática. Universidad Nacional del Comahue.* Disponible en: https://core.ac.uk/download/pdf/86407004.pdf

Aggarwal, C. C. (2015). "*Data Mining: The Textbook*." Springer, London. https://doi.org/10.1007/978-3-319-14142-8

Aggarwal, C.C. & Cheng-Xiang, Z. (2012). "*Mining Text Data".*

Akerkar, R. & Akerkar, P.W. (2010) *" Text Mining: Techniques, Applications and Issues" ISBN0763776475, 9780763776473*

Alfaro, R. A. & Allende O. (2020). *"Clasificación de textos Multi-etiquetados con Modelo Bernoulli Multi-Variado y Representación Dependiente de la etiqueta"*. Revista signos. Estudios de lingüística. ISSN 0718-0934, 2020, PUCV, Chile. pp. 549-567.

Álvarez, D. P. (2021). *"Estudio de la percepción pública de la vacuna contra la COVID-19 mediante técnicas de PNL y de Aprendizaje Automático"*. Disponible en: https://docta.ucm.es/rest/api/core/bitstreams/6a8985dc-6767-4cf4-9a41-d0580d7204d4/content

Álvarez, J. M. (2018). "*El perceptrón como neurona artificial*". Disponible en: http://blog.josemarianoalvarez.com/2018/06/10/el-perceptron-como-neurona-artificial/

Armstrong, S. *et al*., (1999). *"Natural Language Processing Using Very Large Corpora"*

Baccianella, S. *et al*., (2010). *"SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining".* Istituto di Scienza e Tecnologie dell'Informazione Consiglio Nazionale delle Ricerche Via Giuseppe Moruzzi 1, 56124 Pisa, Italy.

Baeza-Yates, R. & Ribeiro-Neto, B. (1995). "*Modern information retrieval*". ACM Press/ Addison Wesley.

Berry, M. J. A. & Linoff, M. (1997). "*Data Mining Techniques for Marketing, Sales, and Customer support*".

Berry, M. & Kogan, J. (2010). "*Text Mining: Applications and Theory*". Print ISBN:9780470749821.

Bhutada, S., *et al*., (2014). *"Latent Dirichlet Allocation based multilevel classification*". *International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, Kanyakumari, India, 2014, pp. 1020-1024, doi: 10.1109/ICCICCT.2014.6993109.

Blei, D.M. *et al*., (2002). *"Latent Dirichlet Allocation"*. Journal of Machine Learning. Research 3. pp. 993-1022. https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

Bothma, T. J. D. *et al*., (2008). *"From tags to topic maps: using marked-up Hebrew text to discover linguistic patterns". Disponible en:* https://repository.up.ac.za/handle/2263/5778

Breiman, L. *et al*., (1984). "*Classification and Regression Trees"* Editorial: Chapman and Hall/CRC, ISBN 10: 0412048418.

Brun, E. R. & Senso, J. A. (2004). *"Minería textual"*. v. 13, n. 1, pp. 11-27. https://www.researchgate.net/profile/Jose-Senso/publication/28157651_Mineria_textual/links/0fcfd509a259d09f98000000/Mineria-textual.pdf

Calvo, D. (2018). *"Función de activación – Redes neuronales"*. Disponible en: https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/

Carrillo, R. M. (2021). *"Text mining: Principios básicos, aplicaciones, técnicas y casos prácticos"*. Universidad de Sevilla. Disponible en: https://idus.us.es/bitstream/handle/11441/134449/DGME%20CARRILLO%20GARCIA%2c%20ROSA%20MARÍA.pdf?sequence=1&isAllowed=y

Chen, J., Zhang, Y. & Du, Z. (2018) "*Recurrent convolutional neural network for sentiment analysis of short texts*". Information Sciences, 449, 94-102.

Colmenares, E. & Wu, H. (2021). *"Accelerating Workload Processing with MPI for Porter's Stemming Algorithm,". International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, 2021, pp. 1783-1787, doi: 10.1109/CSCI54926.2021.00337.

Cortés, C. & Vapnik, V. (1995). "*Redes de vectores de soporte." Mach Learn* 20, 273–297 (1995). https://doi.org/10.1007/BF00994018

Crawford, K. (2013). *"The hidden biases in big data"*. Harvard Business Review, 91(4), 56-62. https://doi.org/10.1016/j.imavis.2016.03.005

De la Fuente, O. S. (2019). *"Análisis y aplicación de técnicas de aprendizaje automático para clasificación de reseñas en redes sociales*". *Universidad Carlos III de Madrid. TFG para acceder al grado de Ingeniería Informática.*
Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/29605/TFG_Oscar_Martin-De-La-Fuente_Sanz.pdf?sequence=1&isAllowed=y

Deerwester, S. *et al.* (1990). *"Indexing by latent semantic analysis"*. Journal of the American Society for Information Science, v. 41, n. 6, pp. 391- 407. https://asistdl.onlinelibrary.wiley.com/doi/10.1002/(SICI)1097-4571(199009)41:6%3C391::AID-ASI1%3E3.0.CO;2-9

Del Tío, M. (2019). "*Punto de referencia de compilaciones paralelas de Maven*". https://mikeldeltio.com/2019/04/11/maven-parallel-builds-benchmark/

Delua, J. (2021). *"Supervised vs. unsupervised learning: What's the difference?"*. Disponible en: https://www.ibm.com/blog/supervised-vs-unsupervised-learning/

DOMO (2022). "Data Never Sleeps 10.0". https://www.domo.com/data-never-sleeps

Esuli, A. & Sebastiani, F. (2006). *"SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining"*. In *Conference Paper May 2006.* Doi: https://www.researchgate.net/publication/200044289

European Commission. (2021). "*Ethics guidelines for trustworthy AI".* Retrieved from https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy- ai

European Parlament and Council. (2016). General Data Protection Regulation (GDPR). Official Journal of the European Union. https://doi.org/10.3000/19770677.L_2016.119.eng

Fawcett, T. (2006). *"An introduction to ROC analysis"*. Disponible en: https://web.archive.org/web/20090306204700/http://www.csee.usf.edu/~candamo/site/papers/ROCintro.pdf

Fayyad, U. M., Piatetsky, S. G. & Smyth, P. (1996). "*From Data Mining to Knowledge Discovery: An Overview. Advances in Knowledge Discovery and Data Mining".* AAAI/MIT Press. pp. 1-34.

Feldman, R. & Sanger, J. (2006). *"The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data".* Cambridge University Press. https://doi.org/10.1017/CBO9780511546914

García-Zorita, C., *et al.,* (2018). "*Aplicación de minería de texto para el análisis métrico sobre ahorro energético en proyectos del Séptimo Programa Marco*". Bibliotecas. Canales de Investigación; 14(2), 149-163.

Garriga-Domínguez, A. (2009) "*Tratamiento de datos personales y derechos fundamentales*". 2º Edición, Dykinson, Madrid, 2009, p.53.

Gobierno de España. Secretaría de estado de digitalización e inteligencia artificial. *"Como crear nubes de palabras a partir de un texto utilizando técnicas de PNL"*. Disponible en: https://datos.gob.es/es/documentacion/como-crear-nubes-de-palabras-partir-de-un-texto-utilizando-tecnicas-de-pln

Goebel, M., y Gruenwald, L. (1999). *"A survey of data mining and knowledge discovery software tools".* ACM SIGKDD Explorations Newsletter, v. 1, n. 1, pp. 20-33.

González, S. G. (2022). *"Sistema de minería de opiniones para la detección de sesgos sociales". Universidad de Salamanca. TFG para acceder al grado de Ingeniería Informática.*
Disponible en:
https://gredos.usal.es/bitstream/handle/10366/151341/MEMORIA%20TFG__%28memoria_tfg%29.pdf?sequence=1&isAllowed=y

Gozalo, J. S. (2020). *"Análisis del estado del arte de la generación de texto con redes neuronales mediante modelos de Transformer". Universidad de Valladolid. Escuela de ingeniería informática de Segovia. TFG para acceder al grado en Ingeniería Informática de Servicios y Aplicaciones.*
Disponible en: https://uvadoc.uva.es/bitstream/handle/10324/43030/TFG-B.%201562.pdf?sequence=1&isAllowed=y

Guan, S. (2023). *"Performance Analysis of Convolutional Neural Networks and Multilayer Perceptron in Generative Adversarial Networks". IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA)*, Shenyang, China, 2023, pp. 817-821, doi: 10.1109/ICPECA56706.2023.10076041.

Guía, P. & Atkinson-Abutridy, J. (2010). "Evaluación de Reglas de Asociación en Text Mining Utilizando Métricas Semánticas y Estructurales".

Hearst, M. A. (1999). *"Untangling text data mining"*. Universidad de California, Berkeley. https://people.ischool.berkeley.edu/~hearst/papers/acl99/acl99-tdm.html

Hirao, T. *et al*., (2015). "*Summarizing a Document by Trimming the Discourse Tree".* in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 11, pp. 2081-2092, Nov. 2015, doi: 10.1109/TASLP.2015.2465150.

Hoffman, T. (1999). *"Probabilistic Latent Semantic Indexing".* International Computer Science Institute, Berkeley.
https://dl.acm.org/doi/pdf/10.1145/312624.312649

IBM. Artículo: *"¿Qué es la minería de texto?". Disponible en:* https://www.ibm.com/es-es/topics/text-mining

Jurafsky, D., & Martin, J. H. (2019). *"Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition". Disponible en:* https://doi.org/10.5555/3314103

Justicia de la Torre, C. (2017). *"Nuevas Técnicas de Minería de Textos: Aplicaciones".* Universidad de Granada.
https://digibug.ugr.es/bitstream/handle/10481/46975/26606203.pdf?sequence=1&isAllowed=y

Kairouz, P. *et al.,* (2021). *"Advances and open problems in federated learning. Foundations and Trends in Machine Learning".* 14(1-2), 1-174. https://doi.org/10.1561/2200000075

Kandukuri, M. & Vajjha, H. (2021). "*Topic Modelling Extraction of "Mann Ki Baat*"". European Journal of Mathematics and Statistics. 2. 1-12.

Kannan, S. *et al.*, (2014). "Preprocessing Techniques for Text Mining," *Int. J. Comput. Sci. Commun. Networks*, vol. 5, no. 1, pp. 7–16, 2015.

Kao, A. & Poteet, R. S. (2007). "*Natural Language Processing and Text Mining*", pág. 171–192. Springer London.

Kowalczyk, A. (2018). *"Support Vector Machines Succinctly".* Publicado por Syncfusion Inc.
https://timofey.pro/static/pdfdocs/AI_024_support_vector_machines_succinctly.pdf

Kucera, H., *et al*. (1967). *"Computational Analysis of Present-Day American English"*. Brown University Press.

Landauer, T. K., Foltz, P.W. & Laham, D. (1998). "*An Introduction to Latent Semantic Analysis"*. pp. 259 -284.
http://ryanrossi.com/teaching/search/papers/intro_to_LSA.pdf

Li, X. *et al*., (2008*). "Learning query intent from regularized click graphs".* In Proceedings of 31st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 339-346. ACM.

Li, Y. & Shen, B. (2017) "*Research on sentiment analysis of microblogging based on LSA and TF-IDF". 3rd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2017, pp. 2584-2588, doi: 10.1109/CompComm.2017.8323002.

Liu, B. (2012), *'Sentiment Analysis and Opinion Mining'*, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers

Liu, Z. *et al.,* (2011). *"Performance evaluation of Latent Dirichlet Allocation in text mining". Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Shanghai, China, 2011, pp. 2695-2698, doi: 10.1109/FSKD.2011.6020066.

Lois, A. (2022). *"Los colores de coches más populares y cuales son más seguros y más peligrosos para tener accidentes"*. Noticias – Autopista. Disponible en: https://www.autopista.es/noticias-motor/colores-coches-mas-populares-cuales-son-mas-seguros-mas-peligrosos-tener-accidentes_262736_102.html

López-Brea, D. R. (2022). *"No todo lo que necesitas es atención"*. Universidad de Cantabria – Facultad de ciencias. TFG para acceder al grado de Matemáticas. Disponible en: https://repositorio.unican.es/xmlui/bitstream/handle/10902/26249/RiveraLópez-BreaDiego-TFG-Matemáticas.pdf?sequence=1

Luhn, H. P. (1958). *"A Bussines Intelligence System"*. IBM Journal. http://altaplana.com/ibm-luhn58-BusinessIntelligence.pdf

Luque, A. *et al*, (2022). *"Visualizing Classification Results: Confusion Star and Confusion Gear,"* in *IEEE Access*, vol. 10, pp. 1659-1677, 2022, doi: 10.1109/ACCESS.2021.3137630.

Lyras, D. P., *et al.,* (2007). *"Using the Levenshtein Edit distance for Automatic Lemmatization: A case Study for modern Greek and English". 19th IEEE International Conference Tools with Artificial Intelligence,* Patras, Greece. Pp. 428-435, doi:10.1109/ICTAI.2007.41.

Mander, J. (2014). "*GWI Audience report summary. Placing a spotlight on the digital behaviors of the teen demographic*". Globalwebindex. Disponible en: https://insight.gwi.com/hs-fs/hub/304927/file-667503056-pdf/Content_Marketing/GWI_Teens_Audience_Report_-_Q1_2014_Summary.pdf?submissionGuid=a80f3f1b-9f25-4627-8387-4120412c70cf

Manning, C. D., et al., (2008). "*Introduction to Information Retrieval*". Cambridge University Press.

Mena Roa, M. (2022). *"El Big Bang del Big Data"*. Statista 2022. Disponible en: https://es.statista.com/grafico/26031/volumen-estimado-de-datos-digitales-creados-o-replicados-en-todo-el-mundo/

Merriam, C. & G. (2007). "*Webster's Revised Unabridged Dictionary*". Disponible en: http://www.thefreedictionary.com/lemmatise

Miner, G., *et al.* (2012). *"Practical Text Mining and Statistical Analysis for non-structured text data applications"*. Academic Press. https://www.sciencedirect.com/book/9780123869791/practical-text-mining-and-statistical-analysis-for-non-structured-text-data-applications

Mishra, K. M. & Huhtala, K. J. (2019). "Fault Detection of Elevator Systems Using Multilayer Perceptron Neural Network". 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 2019, pp. 904-909, doi: 10.1109/ETFA.2019.8869230.

Mitchell T. M., (1997). "*Machine Learning*". McGraw-Hill.

Murillo, E. C. & Leoni, J. A. (2016). "Transferencia de la polaridad semántica de frases idiomáticas a comentarios de opinión". Universidad de Costa Rica, Costa Rica. Doi: https://doi.org/10.15517/rk.v40i3.28813

Ni, J., Li, J., & McAuley, J. (2019). "*Empirical Methods in Natural Language Processing (EMNLP".* Disponible en: pdf

O'Neil, C. (2016). *"Weapons of math destruction: How big data increases inequality and threatens democracy".*

Pang, B., & Lee, L. (2008). *"Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval"*, 2(1–2), 1-135.

Paralic, J. (2001). *"Knowledge discovery in databases*". In 2nd Internal Scientific conference of the Faculty of Electrical Engineering and Informatics, pp. 33-34.

Pérez Pueyo, R. (2005). *"Procesado y optimización de espectros Raman mediante técnicas de lógica difusa: Aplicación a la identificación de materiales pictóricos".* (Tesis inédita, Departament de Teoria del Senyal i Comunicacions, Universitat Politècnica de Catalunya, Barcelona, España).

Phuong, M & Hutter, M (2022). "Formal Algorithms *for Transformers".*

Pío-Álvarez, S. (2018). *"Minería de calidad de datos: Aplicación de técnicas de minería de datos para la evaluación de la calidad de los datos".* Instituto de la computación – Facultad de ingeniería. Universidad de la República. Montevideo, Uruguay. Disponible en: https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/25468/1/PIO18.pdf

Platt, C. J. *et al*., (2000). *"Large Margin DAGs for Multiclass Classification"*. Computer science Department, UK, Bristol. Disponible en: https://proceedings.neurips.cc/paper/1999/file/4abe17a1c80cbdd2aa241b70840879de-Paper.pdf

Plutchik R. (1980). "*Emotion: A Psychoevolutionary Synthesis*". Harper & Row, Nueva York

Popova, S. & Skitalinskaya, G. (2017). *"Extended list of stop words: Does it work for key phrase extraction from short texts?".*12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2017, pp. 401-404, doi: 10.1109/STC-CSIT.2017.8098815.

Porras, J. C. C. (2008). *"Técnicas de lógica difusa aplicadas a la minería de datos".* Scientia et Technica Año XIII, No x, Mes de 200x. Universidad Tecnológica de Pereira. ISSN 0122-1701

Prateek, J. (2022). "*Un enfoque de PNL para la minería de reseñas en línea mediante el modelado de temas (con códigos de Python)".*
https://www.analyticsvidhya.com/blog/2018/10/mining-online-reviews-topic-modeling-lda/

Quispe Poccohuanca, O. E. (2018). "*Integración de técnicas de Deep Learning y algoritmos de aprendizaje multi etiqueta para la clasificación de textos*".
http://repositorio.unsa.edu.pe/bitstream/handle/UNSA/6422/ISMqupooe.pdf?sequence=1&isAllowed=y

Rawal, S. *et al.,* (2017). *"Phishing Detection in E-mails using Machine Learning".* International Journal of Applied Information Systems, vol. 12, no. 7, pp. 21–24, 2017.

Read, J. *et al.* (2011). *"Classifier chains for multi-label classification"*. Machine Learning. 85. 333-359. Doi: https://doi.org/10.1007/s10994-011-5256-5

Reyes-López, O. & Olvera, R (2019). *"Criterios para determinar el tamaño de muestra en estudios descriptivos"*. Disponible en: https://www.researchgate.net/profile/Octavio-Lopez/publication/331687597_Criterios_para_determinar_el_Tamano_de_Muestra_en_Estudios_Descriptivos/links/5c880965299bf14e7e7820d9/Criterios-para-determinar-el-Tamano-de-Muestra-en-Estudios-Descriptivos.pdf

Rodríguez, V. (2018). "*Decision Trees / Árboles de decisión para clasificar en python*".https://vincentblog.xyz/posts/decision-trees-arboles-de-decision-para-clasificar-en-python

RPubs por RStudio. Disponible en: https://rpubs.com/

- Gil, C. (2020). "*Reglas de asociación*". https://rpubs.com/Cristina_Gil/Reglas_Asociacion

- Bosco-Mendoza, J. (2016). "*Introducción a la minería de textos con R*". https://rpubs.com/jboscomendoza/mineria-de-textos-con-r

- Baudouin, J. (2022). "*SVM – Máquinas de soporte vectorial*". https://rpubs.com/jigbadouin/SoporteVectorial

Rumelhart, D. E. *et al.,* (1986). *"Learning Internal Representations by Error Propagation"*. International Journal of Communications, Network and system Sciences. Procesamiento distribuido en paralelo: exploraciones en la microestructura de la cognición, vol. 1: Fundaciones, MIT Press, Cambridge, MA, 318-362.

Salton, G. & McGill, M.J. (1983). "*Introduction to Modern Information Retrieval*", New York: McGraw–Hill, 1983.

Singhal, D*., et al.,* (2020). "Abstractive Summarization of Meeting Conversations". *IEEE International Conference for Innovation in Technology (INOCON),* Bangluru, India, 2020, pp. 1-4, doi: 10.1109/INOCON50539.2020.9298305

Sebastiani. F. (2002). "Machine learning in automated text categorization". ACM Comput. Surv. 34, 1 (March 2002), 1–47. https://doi.org/10.1145/505282.505283

Sharma, A. & Sharma, K.K. (2015). "*Text Mining: A Brief Survey of Techniques and Applications*" publicado en el International Journal of Computer Applications.

Shekhar, R. & Jawahar, C. V. (2012). *"Word Image Retrieval Using Bag of Visual Words". 10th IAPR International Workshop on Document Analysis Systems*, Gold Coast, QLD, Australia, 2012, pp. 297-301, doi: 10.1109/DAS.2012.96

Smelyakov, K., *et al.* (2020). "*Effectiveness of Preprocessing Algorithms for Natural Language Processing Applications". IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2020, pp. 187-191, doi: 10.1109/PICST51311.2020.9467919.

Spackman, K. A. (1989). *"Signal Detection Theory: Valuable tolos for evaluating inductive learning". Pp. 160-163. ISBN 9781558600362. Doi: https://doi.org/10.1016/B978-1-55860-036-2.50047-3.*

Sullivan, D. (2001). *"Document warehousing and text mining"*. Wiley Computer Publishing, pp. 324.

Talib, R. *et al.,* (2016). *"Text Mining: Techniques, Applications and Issues".* Department of Computer science. Government College University, Faisalabad, Pakistan. International Journal of Advanced Computer Science and Applications, Vol. 7 No. 11, 2016

Tan, P. N., *et al.,* (2019). "*Introducción a la minería de datos*". Edición global (2ª ed.).

Tesnière, T. (1969). *"Les Éléments de syntaxe structurale"*, n°1, 1969. La syntaxe. pp. 36-40; doi: https://doi.org/10.3406/lfr.1969.5395

Torres, J., & Vaca, C. G. (2015). "*Caracterización de líderes políticos de Ecuador en Twitter usando aprendizaje de máquina no supervisado*"

Turing, A. M. (1950). "*Computing machinery and intelligence"*. Mind. v. LIX, n. 236, pp. 433 – 460. Disponible en: https://doi.org/10.1093/mind/LIX.236.433

Turney, P. D., & Littman, M. L. (2003). *"Measuring praise and criticism: Inference of semantic orientation from association". acm Transactions on Information Systems (tois)*, *21*(4), 315-346.

Vapnik, V. (1963). *"Pattern Recognition Using Generalized Portrait Method".* Automation and Remote Control, 774-780.

Vásquez, M. A. C, et al., (2020). *"Procesamiento de lenguaje natural".*

Vázquez, Y. G. (2012). "*Análisis semántico multidimensional aplicado a la desambiguación del lenguaje natural"*. Doctoral dissertation, Universitat d'Alacant/Universidad de Alicante.

Westin A. F. (1967). "*Social and political dimensions of privacy".* Journal of social issues, 2003. Wiley Online Library.

Widyassari, A. P., *et al.,* (2022). *"The 7-Phases Preprocessing Based On Extractive Text Summarization". Seventh International Conference on Informatics and Computing (ICIC)*, Denpasar, Bali, Indonesia, 2022, pp. 1-8, doi: 10.1109/ICIC56845.2022.10006998.

Wiegand, M. *et al.,* (2010). *"A Survey on the Role of Negation in Sentiment Analysis".* ACM, Proceedings of the Workshop on Negation and Speculation in Natural Language Processing, pp. 60–68, July 2010.

Yukinawa, N. *et al.,* (2007). "*Optimal aggregation of binary classifiers for multi-class cancer diagnosis using gene expression profiles*". IEEE/ACM Transactions on Computacional Biology and Bioinformatics, 6(2):333–343.

Zhang, Y. & Gu, H. (2011). *"Text Mining with Application to Academic Libraries. En Computer Science for Environmental Engineering and EcoInformatics".* Springer Berlin Heidelberg. pp. 200 - 205.

Zhong, N. *et al.,* (2010). *"Effective pattern discovery for text mining"*. IEEE Transactions on Knowledge and Data Engineering https://core.ac.uk/download/pdf/10904077.pdf

Zhu, D. & Porter, A. L. (2002*). "Automated extraction and visualization of information for technological intelligence and forecasting"*. v. 69, n. 5, pp. 495-506. https://www.sciencedirect.com/science/article/pii/S0040162501001573

Žižka, J., Darena, F. & Svoboda, A. (2019). *"Text Mining with Machine Learning: Principles and Techniques"*. CRC Press. https://doi.org/10.1201/9780429469275

ANEXO I:

#PREPROCESAMIENTO:

```
#Importar las bibliotecas necesarias:
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
import os
import nltk
import re
import html
import spacy
from nltk.corpus import PlaintextCorpusReader
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.corpus import wordnet

# Descargar recursos adicionales de NLTK:
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')

#Direccion del corpus:
base_path                                                                    =
'/Users/andrearodriguezgutierrez/Desktop/movie/test/analizar'
# Mapeo de etiquetas POS de TreeBank a etiquetas POS de WordNet:
def get_wordnet_pos(token):
    tag = nltk.pos_tag([token])[0][1][0].upper()
    tag_dict = {
        "J": wordnet.ADJ,
        "N": wordnet.NOUN,
        "V": wordnet.VERB,
        "R": wordnet.ADV
    }
    return tag_dict.get(tag, wordnet.NOUN)

# Reemplazar múltiples caracteres repetidos por uno solo
def unify_nomenclature(token):
    token = re.sub(r'(.)\1+', r'\1\1', token)
    return token

# Definir reemplazos de nomenclatura:
def unify_nomenclature(token):
    replacements = {
        "movie": "film",
        "movies": "films",
        "actor": "performer",
        "actress": "performer",
        "actors": "performers",
        "actresses": "performers",
        "television": "TV",
```

```
        "series": "show",
        "episode": "show",
        "episodes": "shows",
        "director": "filmmaker",
        "directors": "filmmakers",
        "screenplay": "script",
        "cinema": "film",
        "theater": "theatre",
        "plot": "story",
        "soundtrack": "music",
        "directed by": "directed",
        "awesome": "amazing",
        "awful": "terrible",
        "unforgettable": "memorable",
        "don't like": "dislike",
        "can't stand": "hate",
        "not good": "bad",
        "didn't enjoy": "disappointing",
        "disappointed with": "unhappy with",
        "terrible experience": "awful",
        "poorly made": "low-quality",
        "waste of time": "disastrous",
        "disgusting": "repulsive",
        "pathetic": "pitiful",
        "regret watching": "lamentable",
        "lousy": "mediocre",
    }
    return replacements.get(token.lower(), token)

#Definir la función de preprocesamiento:

def preprocess_text(text):

    # Eliminar formato HTML
    text = html.unescape(text)
    # Reemplazar símbolos de puntuación por espacios
    text = re.sub(r'[^\w\s]', ' ', text)
    # Reemplazar múltiples espacios en blanco por uno solo
    text = re.sub(r'\s+', ' ', text)
    # Eliminar contracciones
    contractions = {
        "ain't": "am not",
        "aren't": "are not",
        "can't": "cannot",
        "could've": "could have",
        "couldn't": "could not",
        "didn't": "did not",
        "doesn't": "does not",
        "don't": "do not",
        "hadn't": "had not",
        "hasn't": "has not",
        "haven't": "have not",
        "he'd": "he had / he would",
        "he'll": "he will / he shall",
```

```
        "he's": "he is / he has",
        "I'd": "I had / I would",
        "I'll": "I will / I shall",
        "I'm": "I am",
        "I've": "I have",
        "isn't": "is not",
        "it'd": "it had / it would",
        "it'll": "it will / it shall",
        "it's": "it is / it has",
        "let's": "let us",
        "mustn't": "must not",
        "she'd": "she had / she would",
        "she'll": "she will / she shall",
        "she's": "she is / she has",
        "should've": "should have",
        "shouldn't": "should not",
        "that's": "that is / that has",
        "there's": "there is / there has",
        "they'd": "they had / they would",
        "they'll": "they will / they shall",
        "they're": "they are",
        "they've": "they have",
        "we'd": "we had / we would",
        "we're": "we are",
        "we've": "we have",
        "weren't": "were not",
        "what'll": "what will / what shall",
        "what're": "what are",
        "what's": "what is / what has",
        "what've": "what have",
        "where's": "where is / where has",
        "who'd": "who had / who would",
        "who'll": "who will / who shall",
        "who're": "who are",
        "who's": "who is / who has",
        "who've": "who have",
        "won't": "will not",
        "wouldn't": "would not",
        "you'd": "you had / you would",
        "you'll": "you will / you shall",
        "you're": "you are",
        "you've": "you have"
    }
    text = re.sub(r"\b(" + "|".join(contractions.keys()) + r")\b",
lambda match: contractions[match.group(0)], text)
    # Eliminar los br:
    text = re.sub(r'<br\s*\/?>', ' ', text)
    # Eliminar la palabra "br"
    text = re.sub(r'\bbr\b', '', text)
    # Eliminar palabras comunes que no aportan valor – Fuera de la lista
“stop words” de NLTK.
    common_words = ['in', 'on', 'at', 'and', 'but', 'or', 'for', 'from',
'to', 'with', 'one', 'two', 'also', 'ever', 'many', 'much',
'still','could', 'would', 'even','make','lot']
```

```
    text = ' '.join([word for word in text.split() if word.lower() not
in common_words])
    # Tokenizar el texto en palabras
    tokens = word_tokenize(text)
    # Convertir a minúsculas
    tokens = [token.lower() for token in tokens]
    # Inicializar lematizador de WordNet
    lemmatizer = WordNetLemmatizer()
    # Filtrar palabras no deseadas (stop words)
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token.lower() not in
stop_words]
    # Lematizar las palabras y aplicar normalización lingüística
    normalized_tokens = []
    for token in tokens:
        pos = get_wordnet_pos(token)
        lemma = lemmatizer.lemmatize(token, pos=pos)
        normalized_tokens.append(lemma)
    # Unificar la nomenclatura y reducir redundancia
    normalized_tokens = [unify_nomenclature(token) for token in
normalized_tokens]
    # Devolver el texto preprocesado como una cadena de texto
    return ' '.join(normalized_tokens)

texto = "If you had asked me how the movie was throughout the film, I
would have told you it was great! However, I left the theatre feeling
unsatisfied. After thinking a little about it, I believe the problem
was the pace of the ending. I feel that the majority of the movie moved
kind of slow, and then the ending developed very fast. So, I would say
the ending left me disappointed.<br /><br />I thought that the
characters were well developed. Costner and Kutcher both portrayed their
roles very well. Yes! Ashton Kutcher can act! Also, the different
relationships between the characters seemed very real. Furthermore,I
thought that the different plot lines were well developed. Overall, it
was a good movie and I would recommend seeing it.<br /><br />In
conclusion: Good Characters, Great Plot, Poorly Written/Edited Ending.
Still, Go See It!!!"

texto_preprocesado = preprocess_text(texto)
print (texto_preprocesado)

# Procesar los archivos de reseñas:
processed_corpus = []

neg_folder_path = os.path.join(base_path, 'neg')
neg_files = [file for file in os.listdir(neg_folder_path) if
file.endswith('.txt')][:500]

for neg_file in neg_files:
    neg_file_path = os.path.join(neg_folder_path, neg_file)
    with open(neg_file_path, 'r', encoding='latin1') as file:
        neg_text = file.read()
        processed_text = preprocess_text(neg_text)
        processed_corpus.append(processed_text)
```

```
pos_folder_path = os.path.join(base_path, 'pos')
pos_files = [file for file in os.listdir(pos_folder_path) if
file.endswith('.txt')][:500]

for pos_file in pos_files:
    pos_file_path = os.path.join(pos_folder_path, pos_file)
    with open(pos_file_path, 'r', encoding='latin1') as file:
        pos_text = file.read()
        processed_text = preprocess_text(pos_text)
        processed_corpus.append(processed_text)

# Imprimir los primeros 2 documentos preprocesados:
for i in range(2):
    print(f"Documento {i+1}: {processed_corpus[i]}")

# Imprimir el total de documentos preprocesados:
print(f"Total de documentos preprocesados: {len(processed_corpus)}")
```

#CREAR UN BAG OF WORDS:

```
#Importar las bibliotecas necesarias:
import matplotlib.pyplot as plt
from collections import Counter
from wordcloud import WordCloud

# Obtener las palabras del corpus preprocesado:
words = []
for document in processed_corpus:
    words.extend(document.split())

# Contar la frecuencia de las palabras:
word_counts = Counter(words)

# Obtener las palabras más frecuentes:
top_words = word_counts.most_common(50)

# Crear un diccionario con las palabras y su frecuencia:
word_dict = {word: count for word, count in top_words}

# Crear la nube de palabras:
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate_from_frequencies(word_dict)

# Guardar la nube de palabras en un archivo de imagen:
wordcloud.to_file('wordcloud.png')

# Abrir el archivo de imagen para visualizarlo:
img = plt.imread('wordcloud.png')
plt.imshow(img, interpolation='bilinear')
plt.axis('off')
plt.show()
```

#CLASIFICACION BINARIA:

```
#Importar las bibliotecas necesarias:

import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV
from tabulate import tabulate
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

# Definir las etiquetas (1 para películas románticas, 0 para el
resto):

labels = [1 if 'romantic' in text.lower() or 'love' in text.lower() or
'affection' in text.lower() or 'amorous' in text.lower() or
'passionate' in text.lower() or 'tender' in text.lower() or
'heartfelt' in text.lower() or 'sensual' in text.lower() else 0 for
text in processed_corpus[:1000]]
# Convertir el corpus procesado en una matriz de características:

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(processed_corpus[:1000]).toarray()

# Dividir el conjunto de datos en entrenamiento y prueba:

X_train, X_test = X[:500], X[500:]
y_train, y_test = labels[:500], labels[500:]

# Definir los parámetros a probar en la búsqueda de hiperparámetros:

param_grid = {
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'C': [0.1, 1, 10],
    'gamma': [0.1, 1, 10]
}

# Realizar la búsqueda de hiperparámetros con validación cruzada:

svm = SVC()
grid_search = GridSearchCV(svm, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Obtener el mejor modelo y los mejores parámetros:

best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
print("Mejores parámetros:", best_params)
```

```
# Realizar la predicción en el conjunto de prueba usando el mejor
modelo:

y_pred = best_model.predict(X_test)

# Obtener el informe de clasificación
classification_report = classification_report(y_test, y_pred)

# Imprimir el informe de clasificación
print(classification_report)

# Obtener las probabilidades de la clase positiva del modelo
probas = best_model.decision_function(X_test)

# Crear las etiquetas binarias para la curva ROC
y_bin = np.where(np.array(y_test) == 1, 1, 0)

# Calcular la tasa de falsos positivos, la tasa de verdaderos
positivos y los umbrales
fpr, tpr, thresholds = roc_curve(y_bin, probas)

# Calcular el área bajo la curva ROC
roc_auc = auc(fpr, tpr)

# Graficar la curva ROC
plt.figure()
plt.plot(fpr, tpr, label='Curva ROC (AUC = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')  # Línea diagonal (clasificación
aleatoria)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()

optimal_threshold = thresholds[np.argmax(tpr - fpr)]
print("Umbral óptimo:", optimal_threshold)
```

#ASIGNACIÓN AUTOMÁTICA DE POLARIDAD:

```
#Importar las bibliotecas necesarias:

from nltk.sentiment import SentimentIntensityAnalyzer
import pandas as pd

#Descargar los paquetes necesarios:

nltk.download('vader_lexicon')

#Crear las etiquetas:

sid = SentimentIntensityAnalyzer()

labels = []

for text in processed_corpus:
    sentiment_scores = sid.polarity_scores(text)
    if sentiment_scores['compound'] >= 0.05:
        labels.append(1)  # Emociones positivas (alegría, amor, etc.)
    elif sentiment_scores['compound'] <= -0.05:
        labels.append(-1)  # Emociones negativas (ira, tristeza, etc.)
    else:
        labels.append(0)  # Etiqueta neutra

#Contar las etiquetas:

positive_count = labels.count(1)
negative_count = labels.count(-1)
neutral_count = labels.count(0)

data = {
    'Etiqueta': ['Positivo', 'Negativo', 'Neutro'],
    'Cantidad': [positive_count, negative_count, neutral_count]
}
df = pd.DataFrame(data)

print(df)
```

```
#ANÁLISIS DE SENTIMIENTOS CON POLARIDADES CONOCIDAS.

#Cargamos los documentos con la polaridad asignada.

processed_corpus_with_polarity = []

neg_folder_path = os.path.join(base_path, 'neg')
neg_files = [file for file in os.listdir(neg_folder_path) if
file.endswith('.txt')][:500]

for neg_file in neg_files:
    neg_file_path = os.path.join(neg_folder_path, neg_file)
    with open(neg_file_path, 'r', encoding='latin1') as file:
        neg_text = file.read()
        processed_text = preprocess_text(neg_text)
        processed_corpus_with_polarity.append((processed_text,
'negativa'))
        negative_count += 1

pos_folder_path = os.path.join(base_path, 'pos')
pos_files = [file for file in os.listdir(pos_folder_path) if
file.endswith('.txt')][:500]

for pos_file in pos_files:
    pos_file_path = os.path.join(pos_folder_path, pos_file)
    with open(pos_file_path, 'r', encoding='latin1') as file:
        pos_text = file.read()
        processed_text = preprocess_text(pos_text)
        processed_corpus_with_polarity.append((processed_text,
'positiva'))
        positive_count += 1

positive_count = 0
negative_count = 0

#Recorrer el corpus y contar las etiquetas
for _, label in processed_corpus_with_polarity:
    if label == 'positiva':
        positive_count += 1
    elif label == 'negativa':
        negative_count += 1

#Presentar los resultados en forma de tabla
data = {'Polaridad': ['Positiva', 'Negativa'],
        'Cantidad': [positive_count, negative_count]}
df = pd.DataFrame(data)

print(df)
```

```
#Comprobar que el texto esta preprocesado y asignado a una polaridad

for i in range(1):
    print(f"Documento {i+1}: {processed_corpus_with_polarity[i]}")

from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, precision_score,
accuracy_score
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import random
import seaborn as sns

#Dividir el corpus en características (X) y etiquetas (y):

X = [document[0] for document in processed_corpus_with_polarity]
y = [document[1] for document in processed_corpus_with_polarity]

#Dividir los datos en conjuntos de entrenamiento y prueba, asignando
600 documentos al conjunto de prueba.

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=600, random_state=42)

#Crear una instancia del vectorizador de conteo y ajustarlo a los
datos de entrenamiento.

vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)

#Crear una instancia del modelo de perceptrón y entrenarlo con los
datos de entrenamiento vectorizados.

perceptron = Perceptron()
perceptron.fit(X_train_vectorized, y_train)

#Transformar los datos de prueba utilizando el vectorizador ajustado.
X_test_vectorized = vectorizer.transform(X_test)

#Realizar predicciones en los datos de prueba.

predictions_sentiment = perceptron.predict(X_test_vectorized)

#Calcular la precisión y mostrar el informe de clasificación.

report_sentiment = classification_report(y_test,
predictions_sentiment)
print(report_sentiment)

# Calcular la matriz de confusión
confusion = confusion_matrix(y_test, predictions)
```

```
# Crear un DataFrame con la matriz de confusión
confusion_df = pd.DataFrame(confusion, index=['Negativa', 'Positiva'],
columns=['Negativa', 'Positiva'])

# Dar formato a los ejes de la matriz de confusión
sns.set(font_scale=1.2)
sns.heatmap(confusion_df, annot=True, fmt='d', cmap='Blues',
cbar=False)

# Añadir etiquetas a los ejes
plt.xlabel('Predicción')
plt.ylabel('Etiqueta verdadera')

# Mostrar la matriz de confusión
plt.show()
```