ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Máster Universitario en Big Data: Tecnología y Analítica Avanzada

ANÁLISIS DE SENTIMIENTOS FINANCIEROS EN NOTICIAS

Autor
Marta Simón Pinacho

Dirigido por
Filippo Giulio Frezza

Madrid
Mayo 2025

Marta Simón Pinacho, declara bajo su responsabilidad, que el Proyecto con título **ANÁLISIS DE SENTIMIENTOS FINANCIEROS EN NOTICIAS** presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2024/25 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: ..... Fecha: 15 / 05 / 2025

Autoriza la entrega:

El Director del Proyecto

Filippo Giulio Frezza

Fdo.: ..... Fecha: 15 / 05 / 2025

V. B. del Coordinador de Proyectos

Carlos Morrás Ruiz-Falcó

Fdo.: ..... Fecha: ..... / ..... / .....

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Marta Simón Pinacho **DECLARA** ser el titular de los derechos de propiedad intelectual de la obra: "Análisis de sentimientos financieros en noticias", que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, los derechos de digitalización, de archivo, de reproducción, de distribución y de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

(a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

(b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.

(c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.

(d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.

(e) Asignar por defecto a estos trabajos una licencia Creative Commons.

(f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente*).

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

(a) Que la Universidad identifique claramente su nombre como autor de la misma

(b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

(c) Solicitar la retirada de la obra del repositorio por causa justificada.

(d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

(a) El autor se compromete a:

(b) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

(c) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

(d) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

(e) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.

- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.

- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a ... 15 ... de ... Mayo ..... de ... 2025 ...

ACEPTA

Fdo.: .....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

Máster en Big Data: Tecnología y Analítica Avanzada

ANÁLISIS DE SENTIMIENTOS FINANCIEROS EN NOTICIAS

Autor
Marta Simón Pinacho

Dirigido por
Filippo Giulio Frezza

Madrid
Mayo 2025

Resumen

Este Trabajo de Fin de Máster se centra en el análisis de sentimientos financieros en noticias. El objetivo principal ha sido desarrollar un sistema que permita obtener noticias financieras a través de una API, puntuarlas en función de su sentimiento y comparar diferentes modelos de clasificación enfocados en el análisis de sentimientos financieros para encontrar el que mejor se ajusta al problema.

En primer lugar, se realizó la obtención de noticias financieras mediante una API, asegurando que se recopilase información relevante y actualizada. Posteriormente, se procedió al desarrollo y despliegue de una aplicación web que permitiera puntuar las noticias en función de su sentimiento, de cara a tener un dataset etiquetado a la hora de entrenar y validar los modelos.

Más adelante, se llevó a cabo el entrenamiento de modelos de análisis de sentimientos desde cero utilizando un dataset profesional. Se aplicaron técnicas de procesamiento de lenguaje natural y aprendizaje automático para entrenar modelos capaces de clasificar el sentimiento de las noticias financieras.

A continuación, se realizó una validación exhaustiva de los modelos, tanto aquellos entrenados desde cero como los modelos preentrenados, utilizando un dataset específico generado para este estudio. Se evaluaron diferentes métricas de rendimiento y se compararon los resultados obtenidos por cada modelo.

Finalmente, se llevó a cabo una comparación detallada de los resultados de los diferentes modelos para determinar cuál de ellos ofrecía la mejor precisión en el análisis de sentimientos financieros. Esta comparación permitió identificar el modelo más eficaz y brindó información valiosa sobre las fortalezas y debilidades de cada enfoque.

En resumen, este Trabajo de Fin de Máster ha abordado de manera integral el análisis de sentimientos financieros en noticias, desde la obtención de datos hasta la comparación de modelos.

Abstract

This Master Thesis focuses on the analysis of financial sentiment in news. The main objective has been to develop a system that allows retrieving financial news through an API, scoring them based on their sentiment, and comparing different sentiment analysis models to find the most effective one.

First, financial news was obtained through an API, ensuring that relevant and updated information was collected. Subsequently, a web application was developed and deployed to score the news based on sentiment, in order to have a labeled dataset to train and validate the models.

Furthermore, training of sentiment analysis models from scratch was carried out using a professional dataset. Natural language processing and machine learning techniques were applied to train models capable of classifying financial news sentiment.

Then, a thorough validation of the models, both those trained from scratch and the pre-trained models, was performed using a custom dataset generated specifically for this study. Different performance metrics were evaluated, and the results obtained by each model were compared.

Finally, a detailed comparison of the results of the different models was carried out to determine which model provided the best accuracy in financial sentiment analysis. This comparison identified the most effective model and provided valuable information on the strengths and weaknesses of each approach.

In summary, this Master's thesis has comprehensively addressed financial sentiment analysis in news, from data collection to model comparison.

A mi familia, en especial a mis padres y mi hermana.
Por apoyarme siempre en todo lo que hago y
acompañarme siempre en el camino.
Gracias por estar siempre ahí y confiar en mi.
No lo habría conseguido sin vosotros.

A Andrei, por estar siempre a mi lado y
animarme a conseguir todo lo que me proponga.

A mis amigas de siempre, por ser un apoyo fundamental
y acompañarme en cada uno de los pasos que doy.

Índice general

Índice de figuras

Índice de cuadros

Acrónimos

ICAI	Instituto Católico de Artes e Industrias
TFM	Trabajo de Fin de Máster
NLP	Natural Languaje Processing
GCP	Google Cloud Platform
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-Trained Transformer

Capítulo 1

Introducción

1.1. Motivación

En la era de la información digital, hoy en día nos enfrentamos a múltiples fuentes de noticias financieras que proporcionan constantemente una avalancha de información relacionada con el mercado. Esta gran cantidad de datos financieros presenta tanto desafíos como oportunidades para los inversores y profesionales del sector. Para aprovechar al máximo esta información, es crucial tener la capacidad de procesarla de manera eficiente y extraer conocimientos significativos.

En este contexto, el análisis de sentimientos financieros a través de técnicas de Procesamiento del Lenguaje Natural (NLP) se presenta como una herramienta muy útil y valiosa. No solo permite analizar la información cuantitativa y los informes financieros de una empresa, sino que también aporta una visión diferente a través de la exploración de la polaridad de los sentimientos asociada a las noticias financieras. Esto ofrece una perspectiva más completa sobre el estado de las empresas y su impacto en el mercado.[28]

Actualmente, el auge del procesamiento de lenguaje natural ha impulsado significativamente el desarrollo de técnicas y algoritmos avanzados que permiten analizar de manera eficiente grandes volúmenes de texto en tiempo real. Estos avances tecnológicos brindan una gran oportunidad para aplicar técnicas de NLP para analizar sentimientos financieros en noticias, ya que ahora es posible procesar rápidamente noticias de diferentes fuentes y extraer información relevante.

Además, el análisis de sentimientos financieros se está convirtiendo en un recurso cada vez más útil como complemento del análisis financiero tradicional de una empresa. No solo proporciona una perspectiva emocional sobre la percepción del mercado hacia una empresa o industria en particular, sino que también puede utilizarse para evaluar el riesgo asociado a una empresa al seleccionarla o no como cliente. Esta combinación de análisis financiero y análisis de sentimientos ofrece una visión más completa y ayuda a los inversores y profesionales financieros a tomar decisiones más informadas y fundamentadas.

Por lo tanto, en este Trabajo de Fin de Máster (TFM) se pretende explorar el análisis de sentimientos financieros en noticias mediante técnicas de NLP. El obje-

tivo es desarrollar enfoques innovadores que permitan extraer información relevante y evaluar el impacto emocional de las noticias financieras en el comportamiento del mercado. Se espera que los resultados del proyecto proporcionen una valiosa perspectiva para la selección de clientes en el ámbito financiero.

1.2. Objetivos

El principal objetivo de este proyecto es analizar los sentimientos financieros de las noticias. En concreto, esta investigación se centrará en analizar el sentimiento de las noticias financieras relacionadas con los clientes de Ebury. Esta es una empresa *fintech* que ofrece servicios de pagos internacionales y soluciones financieras a empresas y pymes. Se utilizarán los sentimientos mencionados como base para predecir si el cliente evaluado posee una posición financiera favorable que justifique llevar a cabo transacciones comerciales con el cliente en cuestión. En caso contrario, es decir, si se encuentra en una situación económica inestable o en declive, se evitará entablar negocios con dicho cliente.

La investigación se llevará a cabo utilizando diversas técnicas de NLP y algoritmos de aprendizaje automático.

Para el análisis de los sentimientos financieros se emplearán tanto modelos preentrenados, como por ejemplo el modelo FinancialBERT y GPT3, como también se entrenarán modelos desde cero para tratar de comparar los resultados y analizar los casos en los que cada modelo aporte mejores resultados. Se emplearán modelos como Gradient Boosting o Random Forest,, entre otros.

La finalidad de emplear varios modelos es tratar de comparar las soluciones que proporcionan cada uno de ellos de forma independiente al entrenarlos sobre el mismo dataset. Esta comparación se realizará generando *embeddings* con diferentes algoritmos como TF-IDF, Word2Vec y GloVe, descritos en la sección 2.3.1.2 Estado del Arte de este documento.

A día de hoy, en Ebury existe un equipo de riesgos que se dedica a analizar la posición financiera de sus clientes actuales y de los posibles futuros clientes. El objetivo de estos análisis es evitar posibles problemas en el futuro, como, por ejemplo, impagos o utilización ilegítima de capital.

La idea principal de este proyecto es desarrollar las bases de una herramienta que, según los sentimientos que se identifiquen en las noticias, ayude a los analistas de riesgos a tomar la decisión de involucrar o no al cliente en los negocios de Ebury. En concreto, en este proyecto se va a tratar de encontrar la mejor combinación de algoritmos de Machine Learning que permitan asegurar con la máxima certeza el sentimiento de una noticia financiera. Esto permitirá caracterizar el impacto que esa noticia puede tener sobre la actividad económica de una empresa.

1.3. Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estudio del estado del arte.** En este capítulo se detallan diferentes técnicas de *Natural Languaje Processing* (NLP), tanto de procesamiento como de preprocesamiento. Además, se introducirán diferentes modelos de Machine Learning (ML) actuales y tradicionales, los cuales se usarán para tratar de analizar los sentimientos financieros.

- **Diseño.** En esta sección se exponen las decisiones de diseño que se han ido tomando a lo largo del proyecto. Esto incluye la elección de fuentes de información para la obtención de las noticias, técnicas de preprocesamiento de los datos, elección de hiperparámetros para los modelos de ML, etc.

- **Desarrollo.** En este capítulo se detalla el flujo de trabajo que se ha seguido para llevar a cabo el proyecto. Desde la obtención de las noticias para generar un dataset propio, pasando por el despliegue de una aplicación web dedicada al etiquetado de noticias, hasta el entrenamiento de los modelos.

- **Análisis de resultados.** En este apartado se exponen los resultados obtenidos tras aplicar diferentes técnicas de NLP. Se analizarán los resultados de los sentimientos detectados y se compararán las diferentes técnicas de generación de *embeddings*, para tratar de encontrar la mejor combinación de algoritmo de generación de *embeddings* y el mejor modelo de ML que sean capaces de predecir con mayor exactitud los sentimientos de las noticias.

- **Conclusiones y trabajo futuro.** En este último capítulo, se exponen las conclusiones del trabajo, y se proponen próximos pasos para continuar mejorando y ampliando la funcionalidad del proyecto.

Capítulo 2

Estado del arte

2.1. Introducción

En este capítulo se van a introducir los conceptos y herramientas empleados para la realización de este TFM. En primer lugar, se introduce el término "sentimiento financiero" y cómo afecta a las decisiones que pueden tomarse en el ámbito financiero. A continuación, se explican diferentes técnicas y métricas utilizadas en el análisis financiero de una empresa, introduciendo como una de ellas el análisis de sentimientos financieros en texto. Además, se explican diferentes algoritmos para codificar la información mediante vectorización de los textos, para poder introducirlos como entrada de los modelos. Por último, se introduce el concepto de "*Transformers*", se explica su arquitectura y se analiza su funcionamiento, ya que son las bases de los modelos preentrenados que se van a describir a continuación. Se describen las características de FinancialBert y GPT3.

2.2. Definición de sentimientos financieros

El sentimiento financiero se refiere al estado de ánimo o actitud general de los inversores, comerciantes, consumidores y empresas hacia los mercados financieros, la economía y los instrumentos financieros específicos, como acciones, bonos y divisas. Puede ser positivo, negativo o neutral y puede verse influenciado por varios factores, como indicadores económicos, eventos políticos, artículos de noticias, redes sociales y tendencias del mercado. Comprender el sentimiento financiero puede ayudar a los inversores, comerciantes y empresas a tomar mejores decisiones y predecir los movimientos del mercado. El análisis del sentimiento financiero implica analizar datos de diversas fuentes para evaluar el sentimiento y proporcionar conocimientos a clientes e inversores.

En concreto, los sentimientos financieros en las noticias están relacionados con la situación financiera actual de la empresa, la economía y el valor de los activos financieros, y pueden variar ampliamente dependiendo de la perspectiva y la posición de cada individuo o grupo.

Los sentimientos financieros abarcan una amplia gama de emociones, que van desde la confianza y la satisfacción hasta la ansiedad y el miedo. La confianza surge

cuando una persona se siente segura y satisfecha con su situación financiera, mientras que la ansiedad y el miedo pueden surgir cuando se enfrenta a incertidumbre o dificultades económicas. Estos sentimientos pueden ser influenciados por diversos factores, como el nivel de ingresos, la estabilidad laboral, la capacidad de ahorro, la inversión y la deuda.[28]

Además de las emociones básicas, los sentimientos financieros también pueden estar influenciados por sesgos cognitivos y comportamientos irracionales que afectan la toma de decisiones financieras. Por ejemplo, el miedo a perder dinero puede llevar a una aversión al riesgo excesiva, mientras que la euforia por ganancias pasadas puede conducir a una sobrevaloración de las oportunidades de inversión. Estos sentimientos pueden afectar tanto las decisiones a corto plazo, como el consumo y el ahorro, como las decisiones a largo plazo, como la planificación financiera y la inversión.

Cuando se habla de sentimientos financieros, es importante entender el contexto y el significado de las palabras dentro de la oración. Palabras como "feliz", "triste", "contento", "emocionado", etc. tienen connotaciones específicas, pero cuando se trata de términos financieros, sus connotaciones pueden variar dependiendo del contexto. Por ejemplo, en la oración "La inflación ha subido un 2 % este trimestre", el hecho de que algo esté subiendo podría parecer una buena señal, pero si se trata de la inflación, normalmente se considera algo negativo. Es por eso que es importante comprender el contexto para interpretar correctamente los sentimientos financieros. Utilizando NLP se va a tratar de detectar las palabras relacionadas con finanzas que aporten alguna polaridad a las oraciones, por ejemplo, se analizarán palabras positivas como "ganancias", "recuperación" o "confianza", y negativas como "crisis", "pérdidas" o "declive".

Es importante tener en cuenta los sentimientos financieros transmitidos por los medios de comunicación al analizar el estado financiero de una compañía porque los mercados financieros no se mueven únicamente por números y datos objetivos, sino también por las percepciones y emociones de los inversores y participantes del mercado.

Los medios de comunicación desempeñan un papel fundamental en la formación de la opinión pública en torno a las empresas y los mercados. A través de sus mensajes, pueden influir en los sentimientos de los inversores, generando optimismo o pesimismo, confianza o incertidumbre. Estos sentimientos pueden afectar directamente la demanda y oferta de acciones, bonos u otros activos financieros, lo que a su vez impacta en los precios y en la valoración de una compañía.

Los sentimientos financieros pueden generar movimientos bruscos y exagerados en los mercados, incluso sin una base sólida en los fundamentos económicos de una empresa. Por lo tanto, ignorar estos sentimientos y basarse únicamente en los números puede llevar a una visión limitada y potencialmente incorrecta de la situación financiera de una compañía.[12]

Al considerar los sentimientos financieros transmitidos por los medios de comu-

nicación, se pueden obtener perspectivas más amplias y equilibradas sobre la percepción del mercado hacia una empresa. Esto permite a los analistas y los inversores evaluar no solo los fundamentos económicos, sino también el panorama emocional que rodea a la compañía. Comprender cómo perciben los medios de comunicación la salud financiera de una empresa puede ayudar a tomar decisiones más informadas y a anticipar posibles fluctuaciones en los precios de los activos.

Las noticias financieras suelen ser claras y específicas, ya que su objetivo es evitar confusiones y expresar las ideas de forma clara. Sin embargo, debemos asegurarnos de que provengan de fuentes financieras respetables para garantizar su fiabilidad. Aún así, detectar sentimientos financieros puede resultar complejo, debido a que es un término poco común en el mundo de los sentimientos y depende mucho del contexto de las palabras dentro de las oraciones.

En conclusión, es importante considerar los sentimientos financieros transmitidos por los medios de comunicación al analizar el estado financiero de una compañía, ya que reflejan las percepciones y emociones de los inversores y pueden afectar significativamente los precios y la valoración de los activos. Incorporar estos sentimientos en el análisis financiero ayuda a obtener una visión más completa y precisa de la situación de una empresa en el mercado.

2.2.1. Análisis financiero de una empresa

El análisis financiero de una empresa ha sido objeto de atención constante en la literatura académica y en la práctica empresarial. A lo largo de los años, se han desarrollado y refinado numerosas herramientas y enfoques para evaluar la salud financiera, el rendimiento y la viabilidad de una empresa.

Una de las herramientas más utilizadas en el análisis financiero es el análisis de estados financieros. Este enfoque se centra en el examen detallado de los estados financieros básicos, como el balance general, el estado de resultados y el flujo de efectivo. Se utilizan diversas técnicas, como el análisis vertical y horizontal, los índices financieros y los ratios, para evaluar la solvencia, la rentabilidad, la eficiencia y la liquidez de la empresa.[33]

Otro enfoque relevante en el análisis financiero es el análisis de la estructura de capital. Este análisis se centra en la composición y el equilibrio entre la deuda y el capital propio de una empresa. Los investigadores y los analistas han desarrollado modelos y teorías, como la teoría de la estructura de capital y el modelo de valoración de activos financieros, para comprender cómo las decisiones de financiamiento afectan el costo del capital y el valor de la empresa. Además, el análisis de la estructura de capital también examina el perfil de vencimiento de la deuda y la capacidad de la empresa para hacer frente a sus obligaciones financieras a corto y largo plazo.[35]

En los últimos años, el análisis financiero ha evolucionado para incluir enfoques más avanzados y tecnológicos. Por ejemplo, el análisis de datos masivos (Big Data) y el aprendizaje automático se han utilizado para analizar grandes conjuntos de datos

financieros y extraer patrones, tendencias y relaciones no evidentes a simple vista. Esto ha permitido a los analistas y a las empresas tomar decisiones más informadas y basadas en datos.[27]

Además, el análisis financiero también ha incorporado aspectos no financieros en la evaluación de una empresa. Esto incluye el análisis de riesgos y la consideración de factores ambientales, sociales y de gobernanza, que abordan el impacto y la sostenibilidad a largo plazo de las prácticas empresariales. Dentro del análisis de riesgos, podemos encontrar el análisis de sentimientos en las noticias financieras.

En conclusión, el análisis financiero de una empresa ha experimentado avances significativos a lo largo del tiempo. Desde el análisis de estados financieros tradicionales hasta enfoques más avanzados basados en Big Data y aprendizaje automático, el análisis financiero se ha vuelto más sofisticado y completo. Además, la incorporación de aspectos no financieros y el análisis de riesgos han enriquecido aún más este campo.

2.3. Procesamiento de lenguaje natural: NLP

En los últimos años, el procesamiento de lenguaje natural ha experimentado un avance significativo gracias a los modelos basados en Transformers[36]. Estos modelos han revolucionado la forma en que comprendemos y generamos texto, y han permitido el desarrollo de aplicaciones más sofisticadas en diversas industrias. En esta sección, exploraremos tres aspectos clave del NLP: generación de *embeddings*, modelos de NLP basados en Transformers y su aplicación en el ámbito financiero.

2.3.1. Embeddings

Los *embeddings* son una forma de representar palabras y secuencias de palabras de manera numérica para que puedan ser utilizadas como entrada en algoritmos de aprendizaje automático. En lugar de tratar las palabras como símbolos aislados, los *embeddings* capturan información semántica y relacional entre las palabras en función de su contexto.[31]

2.3.1.1. Preparación de los datos

Al igual que en el resto de escenarios y problemas a los que se aplican técnicas de Machine Learning, antes de pasar a entrenar los modelos es necesario preparar y limpiar los datos para ser capaces de extraer la mayor cantidad de información de calidad posible, y así permitir a los modelos aprender de forma rápida y eficiente. Técnicas como *tokenización* o la eliminación de *stopwords* son algunas de las que se analizarán en el transcurso de este proyecto. Todas ellas se detallan en la sección de 3Diseño de este documento. Además de las técnicas de limpieza que se deben aplicar a los datos, es necesario codificarlos de una forma que el modelo sea capaz de entender. Para ello, surge el concepto de *embedding*, que hace referencia a la forma de codificar el texto con valores numéricos de cara a que puedan servir como entrada de los modelos de Machine Learning tradicionales.

2.3.1.2. Generación de embeddings

Los *embeddings* se generan utilizando técnicas de aprendizaje automático, como el entrenamiento de modelos de lenguaje o la aplicación de métodos estadísticos sobre grandes corpus de texto. Estos modelos o métodos aprenden a asignar una representación vectorial a cada palabra o secuencia de palabras en función de su contexto en el corpus. [8]

Existen múltiples algoritmos que permiten generar embeddings a partir de un texto dado. Algunos de los más comunes y los que se estudiarán en este TFM son TF-IDF, Word2Vec y GLOVE:

- **TF-IDF**: Este algoritmo asigna un peso a cada palabra en un documento en función de la frecuencia con la que aparece en ese documento y la frecuencia inversa de la palabra en todo el corpus. Este enfoque permite destacar palabras que son relevantes para un documento en particular y menos comunes en el corpus en general. Sin embargo, los *embeddings* generados por TF-IDF son dispersos y no capturan relaciones contextuales entre las palabras.[29][10][22]

- **Word2Vec**: Es un algoritmo para extraer vectores matemáticos a partir de palabras[26]. Está basado en el principio de que otras palabras alrededor de la palabra objetivo, representan a dicha palabra (contexto). Se tiene en cuenta el tamaño de la palabra y se convierte la relación entre los vecinos de dicha palabra de tamaño similar en valores numéricos. Esto se acaba convirtiendo en una matriz con cientos de dimensiones. Cada columna especifica una propiedad. Este algoritmo no es un algoritmo individual, se trata de dos modelos de Machine Learning diferentes: [11]

  - **Skip-Gram**: el objetivo de este algoritmo es predecir el contexto, es decir, las palabras vecinas, dado una palabra objetivo en un corpus de texto. El proceso se basa en una ventana deslizante que se mueve por todo el texto de entrenamiento. Para cada palabra objetivo, el modelo intenta predecir las palabras que la rodean en el contexto.
  - **Continuous Bag of Words** (CBOW): Al contrario que el anterior, el objetivo es predecir la palabra objetivo dada una ventana de contexto de palabras vecinas. El modelo toma una ventana de palabras de contexto y trata de predecir la palabra objetivo en el centro de la ventana.

  Los *embeddings* generados mediante este algoritmo son densos, y capturan, además de la posición de las palabras dentro de la oración, el contexto y la información semántica de las mismas. [13]

- **GLOVE** (**Spherical text embedding**): El método GLOVE factoriza una matriz logarítmica palabra-contexto en la que se pondera la función de pérdida de la factorización. Por lo tanto, los errores resultantes de asociaciones frecuentes se penalizan más severamente que los errores infrecuentes. Además, GLOVE se entrena a partir de asociaciones distintas de cero e ignora los ejemplos negativos (pares palabra-contexto de valor cero). Esta es la diferencia entre el entrenamiento en GLOVE y los modelos Skip-gram.

2.3.2. Transformers

En el ámbito de los modelos de Machine Learning enfocados al procesamiento de lenguaje natural, se debe hablar de los Transformers, un tipo de red neuronal desarrollada por Google para mejorar el procesamiento de información no estructurada, que ha sentado precedente sobre todas las investigaciones existentes y futuras de NLP.

El enfoque central del Transformer se basa en la atención, un mecanismo que permite a la red neuronal capturar relaciones y dependencias a largo plazo entre las palabras en una secuencia de texto. A diferencia de los enfoques tradicionales de NLP que se basaban en el uso de capas recurrentes o convolucionales, el Transformer utiliza exclusivamente la atención para modelar estas relaciones, lo que lo hace más eficiente y fácil de entrenar.

El proceso de atención en el Transformer se divide en dos etapas: el cálculo de la atención propia y la atención de codificador-decodificador. En la primera etapa, la red calcula la atención propia de la entrada, lo que significa que la red se enfoca en partes relevantes de la entrada para generar una salida. En la segunda etapa, la red calcula la atención del codificador-decodificador, lo que significa que la red se enfoca en partes relevantes de la entrada y la salida para generar una salida.

El Transformer ha demostrado ser particularmente exitoso en tareas como la traducción automática, el resumen de texto, la generación de texto y la clasificación de sentimientos. Al eliminar las restricciones temporales de las capas recurrentes, el Transformer puede procesar simultáneamente todas las palabras de una oración, capturando las relaciones globales de cada una de las palabras de manera más efectiva.

Además de su efectividad, el Transformer ha impulsado el desarrollo de modelos de lenguaje preentrenados de gran escala, como BERT (Bidirectional Encoder Representations from Transformers) y GPT (Generative Pre-trained Transformer). Estos modelos serán los que se emplearán más adelante en este proyecto para tratar de predecir sentimientos financieros y comparar sus resultados.

2.3.2.1. Arquitectura de los Transformers

Para llegar a comprender mejor cómo funcionan los modelos que se van a utilizar en este proyecto, concretamente BERT y GPT, es necesario estudiar la arquitectura de los Transformers, pues son la base de estos modelos.

Los Transformers pueden dividirse en dos bloques diferenciados: [18]

- **Encoder**: El encoder en un Transformer se encarga de procesar la secuencia de entrada. Para ello, descompone la secuencia en unidades más pequeñas, como palabras o frases, y las representa mediante vectores de alta dimensionalidad llamados *embeddings*. Estos *embeddings* capturan información semántica y contextual de las unidades de la secuencia, permitiendo una mejor comprensión de la información contenida en ella. El Encoder procesa las unidades

de la secuencia en paralelo y genera una representación completa y rica en información de la secuencia de entrada.

- **Decoder**: Una vez que el Encoder ha generado la representación de la secuencia de entrada, esta información se pasa al Decoder. El Decoder, a su vez, utiliza la información del Encoder para generar una secuencia de salida. Comienza generando una representación inicial basada en la información recibida del Encoder. Luego, utiliza un mecanismo de atención para ponderar la relevancia de diferentes partes de la representación del Encoder durante la generación de cada elemento de la secuencia de salida. A medida que se generan los elementos, se van incorporando al contexto y se utilizan para generar el siguiente elemento de la secuencia. Este proceso se repite hasta que se ha generado la secuencia de salida completa.

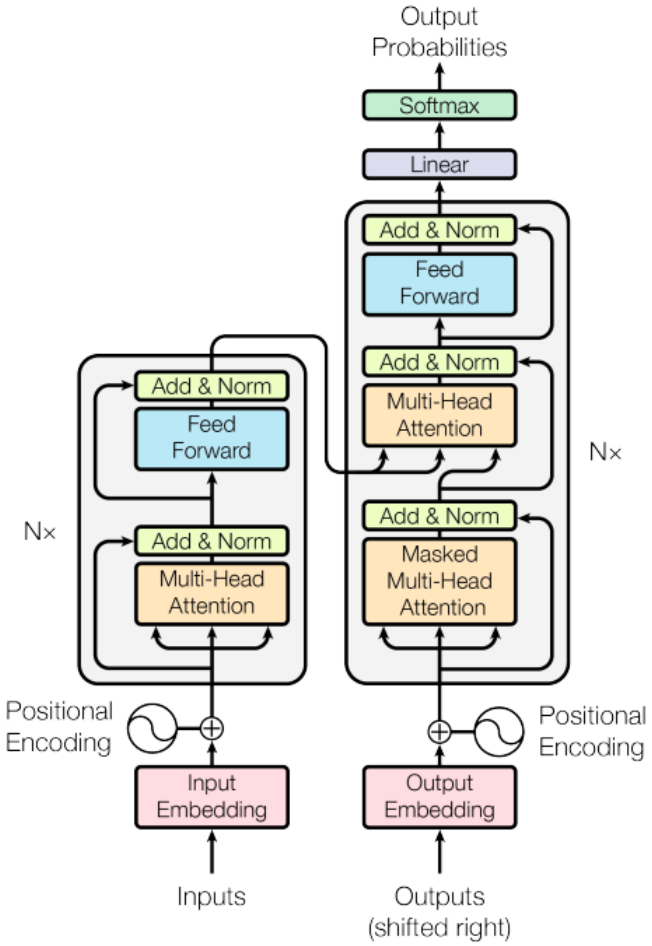

Figura 1: Arquitectura de un Transformer. Encoder y Decoder.

2.3.3. BERT

BERT (Bidirectional Encoder Representations from Transformers)[34] es un modelo de lenguaje basado en Transformers desarrollado por Google. Se trata de uno de los enfoques más destacados en el campo del procesamiento del lenguaje natural

y ha demostrado un rendimiento muy bueno en una amplia variedad de tareas relacionadas con el lenguaje.

La arquitectura de BERT se basa en Transformers, que son modelos de aprendizaje automático diseñados para capturar relaciones de largo alcance en secuencias de datos, como el texto. La principal diferencia entre BERT y los Transformers tradicionales radica en la forma en que se entrenan y cómo utilizan la información del contexto. Mientras que los Transformers tradicionales se entrenan en tareas de lenguaje supervisado, donde se les enseña a predecir la siguiente palabra en una secuencia, BERT se entrena en forma no supervisada utilizando tareas de modelado del lenguaje enmascarado (MLM) y predicción de la siguiente oración (NSP).[17]

Una característica distintiva de BERT es su capacidad para realizar un modelado bidireccional del contexto. A diferencia de los modelos de lenguaje anteriores, que se basaban en contextos unidireccionales (solo consideraban las palabras anteriores), BERT es capaz de aprovechar información tanto de las palabras anteriores como de las palabras posteriores en una secuencia. Esto se logra mediante la utilización de dos estrategias:

- Masked Language Modeling (Modelado de Lenguaje Enmascarado): El objetivo del MLM es que el modelo de BERT aprenda a predecir palabras ocultas en una oración. Durante el proceso de entrenamiento, se toma una oración de entrada y se enmascaran aleatoriamente algunas de las palabras en ella. El modelo entonces intenta predecir las palabras enmascaradas en función del contexto de las palabras vecinas. Esta técnica hace que BERT capture el significado contextual de las palabras y comprenda cómo se relacionan entre sí en el texto. Al enmascarar y predecir palabras aleatorias, BERT desarrolla una representación rica de cada palabra que tiene en cuenta su contexto más amplio.

- Next Sentence Prediction (Predicción de la Siguiente Oración): El objetivo de NSP es permitir que BERT comprenda las relaciones entre las oraciones en un texto. Durante el entrenamiento, se presentan al modelo pares de oraciones consecutivas y se le pide que determine si la segunda oración sigue a la primera en el texto original o no. Esta técnica es importante para tareas como el procesamiento de diálogos o la comprensión de textos que contienen múltiples oraciones. BERT aprende a capturar la relación y coherencia entre las oraciones al realizar esta tarea, lo que permite un mejor entendimiento del flujo del texto.

BERT se entrena en grandes cantidades de texto sin etiquetar, lo que le permite aprender representaciones de palabras y frases de manera no supervisada. Una vez entrenado, el modelo se puede utilizar en una variedad de tareas de procesamiento de lenguaje natural, como clasificación de texto, extracción de información, respuesta a preguntas, análisis de sentimientos, entre otros. Para adaptar BERT a tareas específicas, se suele realizar un proceso de *fine-tuning* en conjuntos de datos etiquetados para afinar los parámetros del modelo y mejorar su rendimiento en la tarea objetivo.

2.3.3.1. FinancialBert: Aplicación de BERT en análisis de sentimientos financieros

El modelo FinancialBert[1] es una adaptación del modelo de lenguaje natural preentrenado BERT (Bidirectional Encoder Representations from Transformers) que ha sido entrenado específicamente en el lenguaje financiero.

Al igual que BERT, FinancialBert es un modelo de lenguaje natural basado en la técnica de aprendizaje profundo de redes neuronales transformadoras, que ha demostrado ser muy eficaz en diversas tareas de procesamiento de lenguaje natural, como la clasificación de texto.

FinancialBert ha sido entrenado en una gran cantidad de datos financieros, incluyendo informes de ganancias, noticias financieras, informes del mercado de valores, entre otros. Esto le permite tener un conocimiento especializado y contextualizado del lenguaje financiero, lo que le permite realizar tareas financieras específicas con gran precisión.

Entre los datasets sobre los que se ha hecho *fine tunning* al modelo BERT para obtener FinancialBert[30] destaca el dataset **Financial PhraseBank**[19]. Este es un dataset etiquetado por expertos en finanzas que contiene más de 4000 oraciones financieras en inglés etiquetadas con las categorías "Positivo", "Negativo" y "Neutro". Este será el dataset que se empleará en este proyecto a la hora de entrenar modelos desde cero.

Entre las aplicaciones más comunes de FinancialBert se encuentran la predicción de precios de acciones, la evaluación de riesgos crediticios y la clasificación de noticias financieras. Esta última será la acción que se pondrá a prueba en este proyecto.

2.3.4. GPT

GPT-3[23] es un modelo de lenguaje avanzado desarrollado por OpenAI, que utiliza el aprendizaje profundo para generar textos lo más parecidos a la lógica humana. La primera versión de GPT tenía 110 millones de parámetros de aprendizaje, mientras que GPT-2 tenía 1500 millones de parámetros de aprendizaje. En la actualidad, se utilizan 175 mil millones de parámetros en GPT-3 como una característica superior a todos los demás modelos. GPT-3 tiene un gran rendimiento en una variedad de tareas como pregunta-respuesta, análisis de sentimientos, decodificación, traducción y el uso de una palabra nueva en un texto.[16]

GPT-3 se basa en la arquitectura de Transformers, al igual que BERT[39]. Una característica que diferencia al modelo GPT3 de los Transformers es que se entrena utilizando un enfoque **generativo** llamado preentrenamiento de lenguaje. Esto significa que el modelo se entrena en grandes cantidades de texto sin etiquetar y aprende a generar texto coherente y relevante basado en la estructura y el contexto del lenguaje.

Una de las características destacadas de GPT-3 es su capacidad de generación de texto de calidad, que se debe en gran parte a la arquitectura de Transformers

en la que se basa. Los Transformers permiten que el modelo capture información contextual a largo plazo, lo que resulta en una mejor comprensión del texto y una generación de lenguaje más precisa y coherente.

Capítulo 3

Diseño

3.1. Introducción

En esta sección del documento se van a detallar las decisiones de diseño llevadas a cabo para realizar el proyecto.

En primer lugar, se muestra un diagrama de Gantt con la planificación del proyecto, en él se muestran las tres fases diferenciadas en las que se ha dividido el proyecto. A continuación, se muestra el diagrama de flujo del prototipo de aplicación que se podrá construir una vez finalizado este proyecto. En él se detallan los diferentes pasos y procesos que se ejecutarán a medida que se interactúa con la aplicación. Una vez descrito el diagrama, se explicarán paso a paso más en profundidad las diferentes partes del proyecto, desde la obtención de noticias, el diseño de la aplicación de etiquetado de noticias, el preprocesado de los datos, la generación de *embeddings*, el entrenamiento y comparación de los diferentes modelos de Machine Learning utilizados durante el proceso de análisis de sentimientos y, por último, la visualización de los resultados obtenidos.

La planificación de la construcción del prototipo para la herramienta final se detallará en la sección de 6 Próximos Pasos de este documento. Ya que este proyecto se ha centrado en la investigación exhaustiva de modelos de Machine Learning para el análisis de sentimientos y en la parte más técnica del NLP.

3.2. Planificación del proyecto

El proyecto se ha dividido en tres fases. Al ser un proyecto transversal, es decir, que cubre todas las fases de un proyecto de Big Data, podemos dividirlo en obtención y etiquetado de las noticias, entrenamiento de modelos de Machine Learning y, por último, análisis y visualización de los resultados de los modelos.

En el siguiente diagrama se muestra la planificación que se ha seguido de cara a desarrollar este proyecto.

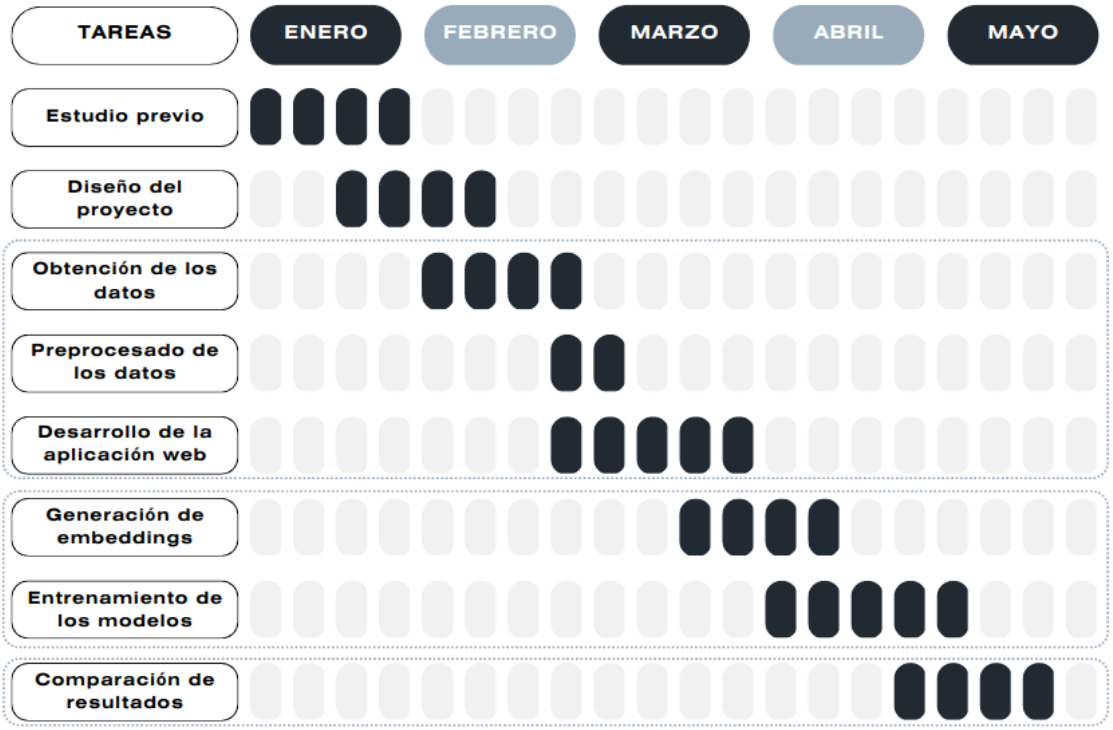

Figura 2: Diagrama de Gantt. Planificación del proyecto

Como se puede observar en el diagrama, el proyecto se divide en cuatro fases.

- En la fase inicial se realiza un estudio previo del proyecto y se traza una idea de diseño inicial.
    - El estudio inicial se realiza para poder tener contexto en el estado del arte del problema y entender concretamente cuáles son los pasos a seguir para poder realizar el proyecto correctamente.
    - Una vez estudiado el estado del arte del problema en concreto, se realiza el diseño del proyecto. En este diseño se incluyen todos los pasos necesarios para poder llevar a cabo el proyecto. Desde el objetivo, pasando por la obtención de datos, el preprocesado de los mismos, el entrenamiento de modelos de Machine Learning apropiados para la tarea de detección de sentimientos y la comparación de los resultados. A partir de esta sección, se puede comenzar a desarrollar el proyecto.
- La primera fase puede dividirse en tres bloques. Esta fase comprende los pasos necesarios para obtener y preprocesar los datos correctamente de cara a que puedan ser utilizados en las fases posteriores.
    - En el primero de ellos, "Obtención de los datos", se especifica cómo se obtendrán los diferentes conjuntos de datos que alimentarán a los modelos utilizados.
    - En el segundo, "Preprocesado de los datos", se indica cómo se aplicarán diferentes técnicas de preprocesado de los datos para poder introducirlos correctamente al modelo. Este paso será crucial para que los modelos puedan aprender correctamente de los datos.

    - Por último, el tercer bloque de esta fase, "Desarrollo de la aplicación web", hace referencia a la creación de una aplicación web, que servirá para obtener *feedback* por parte del equipo de riesgos acerca de los sentimientos que les producen las diferentes noticias.

- La segunda fase, a su vez, puede dividirse en dos bloques principales. Esta fase del proyecto trata de la aplicación de modelos de Machine Learning sobre los datos que se han recabado en la fase anterior.

    - En el primero de los bloques, "Generación de embeddings", se aplicarán diferentes algoritmos de generación de *embeddings* sobre los datos obtenidos, de cara a que podamos introducirlos como entrada de los modelos. La idea de aplicar diferentes algoritmos de generación de *embeddings* es para poder comparar cuál de ellos nos aporta un mejor resultado a la hora de clasificar sentimientos.
    - En el segundo y último bloque de esta fase, "Entrenamiento de los modelos", se emplearán los *embeddings* generados en el bloque anterior para entrenar diferentes modelos de Machine Learning como SVM, Random Forest o LSTM. A su vez, en esta fase se validarán los resultados que devuelven los modelos preentrenados que se van a utilizar, FinancialBert y GPT-3, al aplicarlos sobre el dataset que se ha generado para el proyecto.

- La tercera y última fase consiste en la comparación y visualización de los resultados de los distintos modelos que han sido utilizados en el proyecto.

A partir de este diagrama de Gantt, se ha desarrollado un diagrama de flujo que ilustra la secuencia de acciones que un usuario deberá tomar a la hora de interactuar con el prototipo de la aplicación final. Este diagrama permite ver de forma diferenciada las tres fases en las que se divide el proyecto que se han detallado en la sección anterior.

3.2.1. Diagrama de flujo de la aplicación final

El enfoque principal del proyecto ha sido la evaluación y análisis exhaustivo de diferentes modelos. Se han realizado pruebas rigurosas y se han recopilado datos relevantes para determinar el rendimiento y las capacidades de cada modelo. Estos resultados son de vital importancia para la toma de decisiones informadas en el futuro.

Como paso siguiente a este proyecto, se tiene previsto la generación de una aplicación que permita visualizar los resultados de los modelos de manera clara y concisa. Esta aplicación será diseñada con el objetivo de brindar a todos los miembros de la compañía una herramienta accesible y comprensible para analizar y comprender los resultados obtenidos.

Aunque el desarrollo completo de la aplicación no se ha llevado a cabo en este TFM, el enfoque en la comparación de modelos sienta las bases para futuros trabajos y proporciona una sólida base de conocimientos para la implementación de la aplicación final en etapas posteriores.

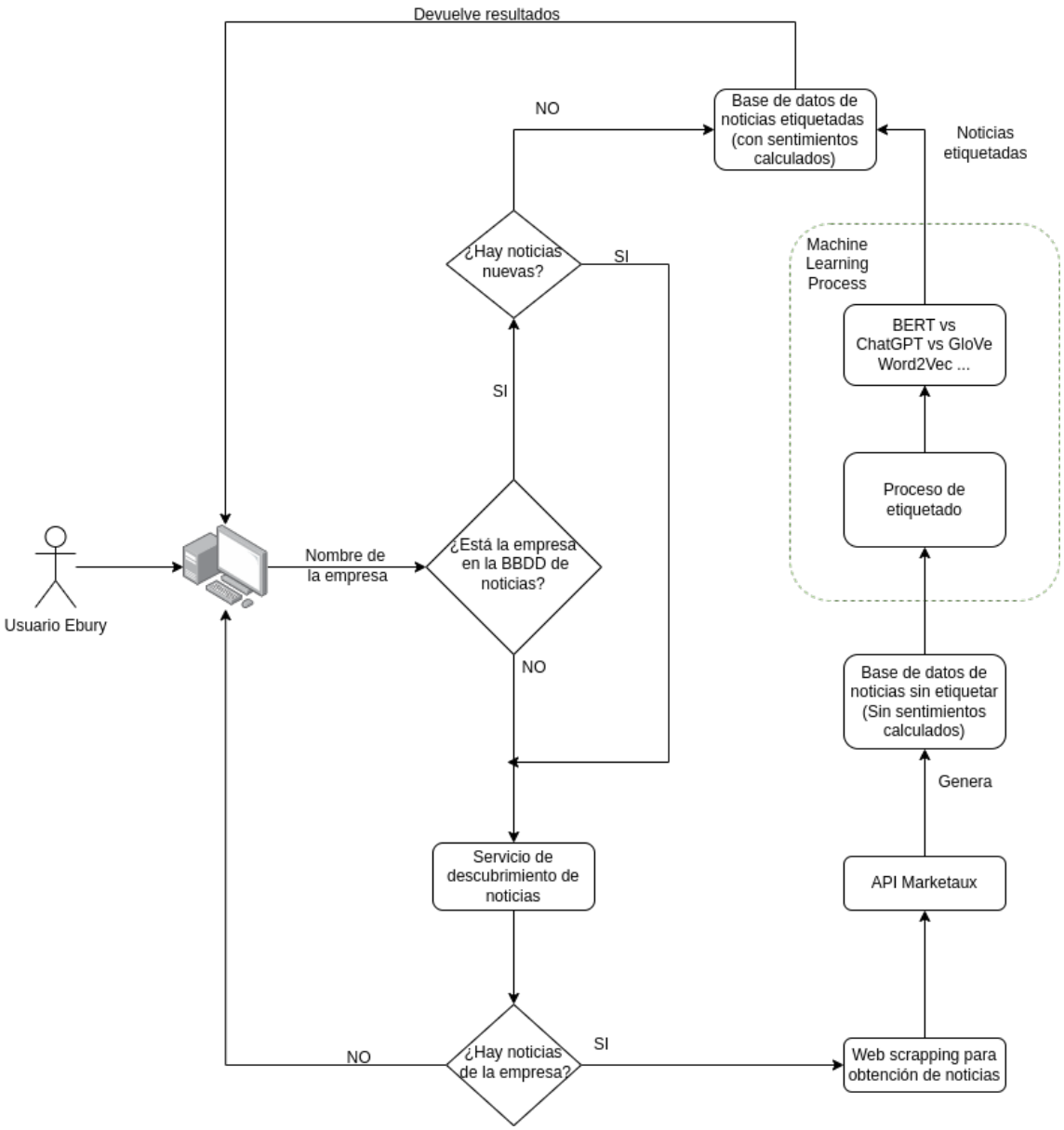

Figura 3: Diagrama de flujo detallado

El funcionamiento de la aplicación final será el siguiente:

- En primer lugar, el usuario accede a la aplicación web y busca el nombre de una empresa que desee analizar.
- En el momento en el que busque el nombre de la empresa, el sistema comprobará si existen noticias de esta empresa en la base de datos.
    - Si existen, comprueba la antigüedad de las mismas, para saber si puede haber o no nuevas noticias sobre la empresa. En caso de que no haya nuevas noticias, accederá a la base de datos de las noticias etiquetadas y devolverá las noticias de esta empresa con sus respectivos sentimientos asociados. Además, mostrará una puntuación global de la noticia asociada a los sentimientos de todas sus noticias.

- En caso de que no existan noticias para el cliente solicitado, o de que las noticias sean demasiado antiguas como para tener una visión actualizada, se lanza el servicio de descubrimiento de noticias. Si el sistema no es capaz de encontrar noticias del cliente seleccionado, devolverá un mensaje de error. Esto puede darse en el caso en que se busque un cliente demasiado pequeño, ya que puede darse el caso de que no se hayan escrito noticias de este cliente en medios fiables y, por tanto, el sistema no encontrará información.

  En el caso de que sí encuentre noticias, se comenzará con las fases definidas en el diagrama de Gantt de la sección anterior[3.2].

  - Se comenzará con la búsqueda de noticias referentes a la empresa solicitada.
  - Una vez encontradas las noticias, se limpiarán los textos encontrados así como se generarán los *embeddings* necesarios para obtener los vectores que puedan servir de entrada a los diferentes modelos.
  - Una vez preparados los datos, se almacenarán en una base de datos de noticias sin etiquetar, de la que se alimentarán los diferentes modelos. Estos modelos se encargarán de clasificar las noticias en función de la polaridad de sus sentimientos, dando como resultado datos etiquetados, que se almacenarán en otra tabla de la base de datos.
  - Una vez clasificadas las noticias, el servicio las devolverá a la aplicación web, que se encargará de mostrar los sentimientos obtenidos a partir de los modelos para cada una de las noticias.

3.3. Datasets

Para la realización de este proyecto se han empleado dos datasets diferentes. Uno de ellos es el dataset público Financial-Phrasebank[19] y el otro es un dataset propio, generado específicamente para este proyecto.

3.3.1. Dataset Financial Phrasebank

Este primer dataset es un dataset etiquetado por 16 expertos en finanzas, 3 investigadores y 13 estudiantes de un máster de negocios en la universidad de Aalto en Finlandia. Este dataset, a su vez, está dividido en 4 subconjuntos de datos, dependiendo de las puntuaciones que diera cada uno de los integrantes del equipo encargado de etiquetar las noticias, se dividieron en cuatro grupos:

- Todos de acuerdo
- 50 % de acuerdo
- 65 % de acuerdo
- 75 % de acuerdo

Para este proyecto, únicamente se va a utilizar el subconjunto de datos en el que todos los integrantes estaban de acuerdo, para poder tener una mejor percepción de los sentimientos de las noticias y tratar de eliminar la mayor subjetividad posible.

Este dataset está compuesto por dos columnas, la primera de ellas contiene fragmentos de noticias financieras, y la segunda contiene la puntuación final que el equipo asignó a esa noticia, pudiendo contener los valores "positivo", "negativo" o "neutro".

3.3.2. Dataset específico

El segundo dataset utilizado ha sido un dataset específico, que se ha generado exclusivamente para este proyecto. La necesidad de generar este dataset surge debido a dos motivos principales.

- En primer lugar, el objetivo principal del proyecto es detectar sentimientos en noticias financieras de los clientes de una empresa. Es por esto que, para que fuesen noticias de clientes reales y no una serie de oraciones financieras, como es el caso del primer dataset, se dio la necesidad de generar un dataset propio para este proyecto.

- Además, uno de los modelos que más adelante se empleará en la detección y análisis de sentimientos es FinancialBert. Este es un modelo Bert al que se ha realizado *fine tunning* sobre el primer dataset mencionado, Financial Phrasebank. Es por esto que, para poder medir la precisión de este modelo y compararla con el resto en igualdad de condiciones, se necesita un dataset que FinancialBert no haya visto nunca.

3.3.2.1. Obtención de noticias

Para la generación del dataset específico, se ha optado por utilizar la API de noticias financieras llamada Marketaux[21]. Esta es una API que ofrece de forma gratuita 100 noticias financieras diarias y que, además, nos permite filtrar por país o incluso tipo de industria del que queremos obtener las noticias.

Como se ha mencionado previamente, la idea inicial del proyecto era analizar noticias financieras de clientes de la empresa, pero, puesto que Ebury actualmente no tiene demasiados clientes grandes, se hacía muy complicado el proceso de búsqueda de noticias financieras de todos los clientes. Por lo que, tras analizar el listado de clientes actuales, se tomó la decisión de recabar noticias financieras de compañías del Reino Unido, sin tener en cuenta si son o no clientes de Ebury. Puesto que el objetivo principal del proyecto es detectar sentimientos financieros, sean o no clientes es algo secundario. Actualmente en Ebury hay un proyecto paralelo en el que se están analizando diferentes fuentes de información para poder obtener noticias financieras de todos los clientes de la cartera.

El proceso de obtención de noticias de la API de Marketaux se ejecuta diariamente, para poder tener una base de datos actualizada, y consta de diferentes pasos:

1. En primer lugar, se hace una llamada a la API indicando como parámetros la fecha del día anterior, así como el idioma en el que queremos recibir las noticias. Para simplificar la tarea de traducción en este proyecto y dado que la mayoría de los modelos están entrenados con datasets en inglés, se ha decidido recabar únicamente noticias en inglés.

2. La API devuelve una serie de atributos en la respuesta para cada una de las noticias. Entre ellos, destacan la URL, la fecha de publicación, un id, la fuente en la que han sido publicadas, y algún otro atributo más. Dado que no se recibe el texto completo de la noticia en la respuesta, es necesario realizar un paso adicional para obtenerlo.

3. Con la URL recibida en el paso anterior, se puede realizar una llamada a la web en la que está publicada la noticia para obtener todo el texto. Esta operación se realiza mediante la librería de Python Newspaper3k, en concreto mediante el módulo Article. Esta librería está desarrollada específicamente para tratar con noticias, y permite extraer toda la información disponible de una noticia simplemente proporcionando la URL de la misma.

4. Tras descargar el artículo con la librería Newspaper3k de Python, es posible parsear la noticia y extraer la información más relevante para el proyecto. En este caso, se trata del id, la URL, la fecha de publicación, el título de la noticia, el *summary* de la noticia y el cuerpo completo, así como las palabras clave de la noticia. Esta librería realiza un pequeño proceso de NLP al descargar el artículo cuando se le proporciona la URL de la noticia, es por esto que se dispone de esta información nada más descargar el artículo.

5. Una vez obtenidas las noticias de la API de Marketaux, y procesadas mediante la librería Newspaper3k de Python, son almacenadas en ficheros JSON que a su vez se almacenan en Buckets de Google Cloud. De esta forma, están disponibles desde cualquier dispositivo que tenga acceso a la instancia de Google Cloud de la empresa.

3.3.2.2. Herramienta de etiquetado de noticias

Para poder validar los modelos de Machine Learning que se van a emplear en el proyecto para la detección y análisis de sentimientos, es necesario un dataset etiquetado, ya que se quieren utilizar modelos de aprendizaje supervisado. De otra forma, no se podría saber qué tan bueno es el modelo, solamente se podría hacer una clusterización de las noticias según su intención o polaridad. Por ello, para llevar a cabo el etiquetado de las noticias que se están almacenando diariamente en los Buckets de GCP (Google Cloud Platform), se ha desarrollado una aplicación web.

Para facilitar este proceso de etiquetado, se ha contado con la colaboración del equipo de riesgos, quienes serán los encargados de asignar una etiqueta de sentimiento a cada noticia. Con este objetivo, se ha desarrollado una aplicación web utilizando la librería Streamlit [32] de Python. Esta herramienta permite al equipo acceder a las noticias almacenadas en formato JSON en los Buckets de GCP, y visualizar un subconjunto de cinco noticias diarias, seleccionadas aleatoriamente del dataset completo. Cada noticia se presenta junto con un menú desplegable en el que

los usuarios pueden seleccionar el sentimiento que les transmite. Una vez completado este proceso, las etiquetas asignadas se almacenan en una base de datos junto con el ID correspondiente de cada noticia. De este modo, se construye progresivamente un dataset etiquetado que servirá para entrenar y validar los modelos supervisados del proyecto.

En la siguiente imagen se muestra una visualización de la aplicación web.

Figura 4: Muestra de la aplicación de puntuación de noticias.

Cada uno de los usuarios del equipo de riesgos necesita identificarse con su correo corporativo para poder acceder a la aplicación, de esta forma se puede saber quién ha puntuado cada noticia.

Una vez un usuario ha puntuado una tanda de noticias, envía los resultados mediante un botón de *submit*. Cuando se detecta la acción en este botón, se realiza una llamada a Big Query desde el *script* de Streamlit de Python para almacenar los resultados en una tabla en la base de datos. Esta tabla contendrá la información del ID de la noticia, el usuario puntuador de la noticia, los sentimientos que el usuario ha asignado a cada una de las noticias y, por ultimo, un *timestamp* indicando la fecha en la que la noticia ha sido puntuada. A continuación se muestra cómo se ve la información dentro de la base de datos.

user_id	new_id	rating	rating_date
[redacted]	308609	Neutral	2023-05-18 07:45:35.461222 U...
[redacted]	308609	Neutral	2023-05-18 08:47:26.319652 U...
[redacted]	308609	Positive	2023-05-16 15:49:36.239588 U...
[redacted]	896525	Neutral	2023-05-16 15:45:45.166822 U...
[redacted]	896525	Negative	2023-05-12 09:07:48.815408 U...
[redacted]	923803	Positive	2023-05-16 15:47:42.398810 U...
[redacted]	923803	Positive	2023-05-16 15:47:00.133125 U...
[redacted]	1650157	Neutral	2023-05-12 08:52:41.861317 U...

Figura 5: Noticias del dataset específico etiquetadas por el equipo de riesgos.

Este procedimiento de puntuación de noticias es similar al que se empleó en la universidad de Aalto a la hora de etiquetar el dataset de Financial Phrasebank. Cada uno de los integrantes del equipo de riesgos va a puntuar noticias, que serán seleccionadas de forma aleatoria, de forma que puede darse el caso de que una noticia haya sido puntuada por varios miembros del equipo. En esta situación pueden darse dos escenarios:

- El primero de ellos, en el que todos los miembros del equipo que han puntuado la noticia le han asignado la misma puntuación, en cuyo caso, se le asignaría directamente dicha puntuación a la noticia.

- Y el segundo, en el que puede haber variaciones en las etiquetas según la persona que las haya etiquetado. En este segundo caso, se ha decidido realizar una media ponderada de las etiquetas, de forma que al final la noticia pertenezca únicamente a una categoría.

  Esta media se realiza asignando valores numéricos a cada una de las etiquetas. En concreto, "Positivo" se corresponde con un 1, "Negativo" con un -1 y "Neutro" con un 0. Una vez convertidas las etiquetas a valores numéricos, se realiza una media de dichos valores.

  Para volver a convertir las etiquetas a sus categorías originales, se ha establecido una regla, entendiendo que si la media de las puntuaciones es mayor que cero, la noticia se considera positiva, si es menor, se considera negativa, y si tiene un valor muy cercano a 0, se considera neutra.

  De esta forma se consigue unificar las votaciones de todos los miembros del equipo en una única etiqueta por noticia.

Aunque se debe hacer el cálculo explicado previamente, las puntuaciones se almacenan una por una, según el usuario que haya puntuado la noticia.

3.4. Análisis exploratorio de los datos

A continuación, se realiza un análisis exploratorio de la información que contienen ambos datasets, para poder tener una visión global de los datos con los que vamos a entrenar y validar nuestros modelos.

3.4.1. Dataset Financial Phrasebank

Este dataset será el empleado para entrenar los modelos desde cero. En concreto, el dataset contiene un total de 2264 filas, cada una de ellas organizadas en dos columnas: el texto financiero y el sentimiento con el que ha sido etiquetado. A continuación, se muestra un ejemplo de información y estructuración del dataset.

	text	sentiment
0	According to Gran , the company has no plans to move all production to Russia , although that is where the company is growing .	neutral
1	For the last quarter of 2010 , Componenta 's net sales doubled to EUR131m from EUR76m for the same period a year earlier , while it moved to a zer...	positive
2	In the third quarter of 2010 , net sales increased by 5.2 % to EUR 205.5 mn , and operating profit by 34.9 % to EUR 23.5 mn .	positive
3	Operating profit rose to EUR 13.1 mn from EUR 8.7 mn in the corresponding period in 2007 representing 7.7 % of net sales .	positive
4	Operating profit totalled EUR 21.1 mn , up from EUR 18.6 mn in 2007 , representing 9.7 % of net sales .	positive
...	...	...
2259	Operating result for the 12-month period decreased from the profit of EUR0 .4 m while turnover decreased from EUR5 .6 m , as compared to 2004 .	negative
2260	HELSINKI Thomson Financial - Shares in Cargotec fell sharply in early afternoon trade after the cargo handling group posted a surprise drop in Apr...	negative
2261	LONDON MarketWatch -- Share prices ended lower in London Monday as a rebound in bank stocks failed to offset broader weakness for the FTSE 100 .	negative
2262	Operating profit fell to EUR 35.4 mn from EUR 68.8 mn in 2007 , including vessel sales gain of EUR 12.3 mn .	negative
2263	Sales in Finland decreased by 10.5 % in January , while sales outside Finland dropped by 17 % .	negative

2264 rows × 2 columns

Figura 6: Muestra del dataset Financial PhraseBank

Como se puede ver, el dataset únicamente contiene una columna con un texto financiero, y otra con el sentimiento que se le ha asignado. Este dataset será empleado para la validación de los modelos desde cero. Además, se ha optado por emplear únicamente el dataset en el que todas las personas encargadas de etiquetar estaban de acuerdo en el sentimiento. De esta forma, se puede tener mayor certeza de que el texto produce el sentimiento con el que ha sido etiquetado. También se puede explorar la proporción de noticias positivas, negativas y neutras que hay dentro del dataset.

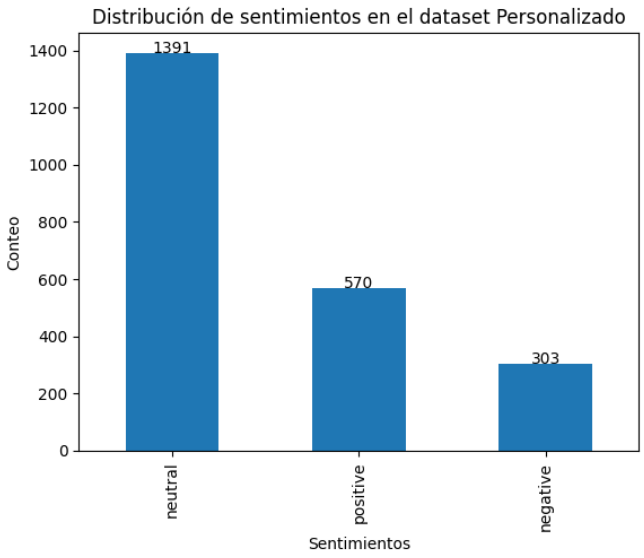

Figura 7: Muestra del dataset Financial Phrase Bank

Se observa que es un dataset claramente desbalanceado, conteniendo 1391 muestras de textos neutros, frente a las 570 de positivos y 303 de negativos. Esta apreciación deberá tenerse en cuenta más adelante, de cara al entrenamiento de los modelos

con este dataset. Al tener las clases desbalanceadas, se deberán emplear métricas que tengan en cuenta este tipo de factores.

A continuación, se muestran las nubes de palabras obtenidas a partir de los textos, a los que se les ha aplicado un preprocesamiento que se detalla en la siguiente sección. Los textos han sido divididos en positivos, negativos y neutros, para poder observar cuáles son las palabras más influyentes para cada uno de los sentimientos.

Figura 8: Nube de palabras de los textos positivos en el dataset Financial Phrase Bank

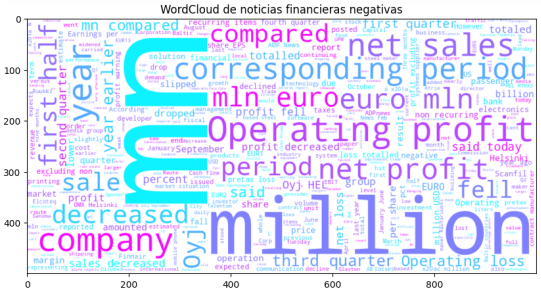

Figura 9: Nube de palabras de los textos negativos en el dataset Financial Phrase Bank

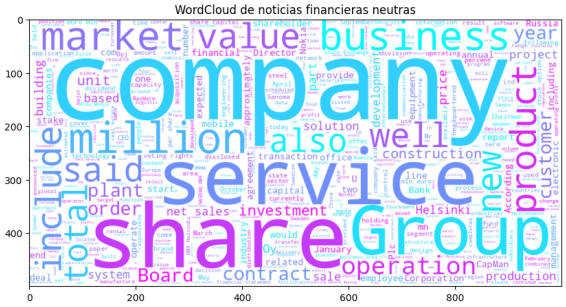

Figura 10: Nube de palabras de los textos neutros en el dataset Financial Phrase Bank

Se puede ver que las palabras más destacadas que aportan sentimientos dentro de cada nube de palabras son:

- Entre las palabras positivas se encuentran “increase”, “significant”, “net profit”, “operating proffit”, “million” ...
- Entre las palabras negativas se encuentran “decreased”, “fell”, “first half”...
- Entre las palabras neutras se encuentran “company”, “share”, “service”...

3.4.2. Dataset específico

A continuación, se va a realizar el mismo análisis para el dataset específico, generado específicamente para la validación de todos los modelos estudiados en este TFM, tanto los preentrenados como los entrenados desde cero con el dataset anterior.

Como se ha explicado antes, en la base de datos que contiene este dataset puede haber noticias que hayan sido puntuadas varias veces por usuarios diferentes. En

concreto, en la base de datos se almacenan un total de 200 noticias, de las cuales, tras agruparlas por id, se obtienen un total de 92 noticias puntuadas.

	id	link	date	media	title	article	summary	key_words	ratings	val_rating	rating_global
0	51188339	https://www...	2023-04-19 0...	https://www...	Why are ASX...	The big ASX...	The big ASX...	[shares, as...	Neutral,Neu...	0.0	Neutral
1	95222513	https://med...	2023-04-19 0...	https://mir...	Clear Visio...	Clear Visio...	Whether you...	[players, m...	Neutral,Pos...	0.8	Positive
2	73147445	https://247...	2023-04-19 0...	https://247...	The US Oil ...	The US Oil ...	The US Oil ...	[record, en...	Neutral,Neu...	-0.6	Negative
3	20573155	https://www...	2023-04-19 0...	https://ml....	STANLEY BLA...	NEW ORLEANS...	This action...	[share, lou...	Neutral,Neu...	-0.6	Negative
4	27742604	https://www...	2023-04-19 0...	https://www...	Own CBA sha...	CBA has rep...	CBA has rep...	[shares, he...	Neutral,Neu...	0.0	Neutral
...	...	...	...	...	...	...	...	...	...	...	...
87	76644572	https://www...	2023-04-11 0...	https://www...	LUMN ALERT:...	NEW YORK, A...	NEW YORK, A...	[klein, sha...	Negative	-1.0	Negative
88	91206702	https://www...	2023-04-11 0...	https://www...	FIS ALERT: ...	NEW YORK, A...	NEW YORK, A...	[klein, wor...	Negative	-1.0	Negative
89	23707923	https://see...	2023-04-11 0...	https://sta...	The Future ...	Javascript ...	Javascript ...	[bearish, c...	Negative	-1.0	Negative
90	9959924	https://www...	2023-04-11 0...	https://d.i...	KKR Buys St...	Private equ...	Private equ...	[kkr, fgs, ...	Neutral	0.0	Neutral
91	53172158	https://www...	2023-04-11 0...	https://ima...	Singapore O...	Singapore b...	Singapore b...	[trading, s...	Neutral	0.0	Neutral

92 rows × 11 columns

Figura 11: Muestra de noticias del dataset específico

En este dataset se almacena más información, que puede ser de utilidad para posibles análisis posteriores. En la imagen se muestra el resultado de unir dos fuentes de datos diferentes. En primer lugar, se obtiene toda la información que se almacena de las noticias en los buckets de GCP. Esta información está almacenada en ficheros JSON pero no está etiquetada. Para obtener los ratings asociados a cada noticia, se debe realizar una query para obtener todas las noticias etiquetadas. Como se ve en la imagen, el dataset resultante contiene varias columnas.

- La columna ID almacena el id de cada noticia. Esta columna es la que permite unir ambas fuentes de datos, e identificar qué puntuación corresponde con cada noticia.

- La columna *article* contiene el texto del artículo completo, que será el que se usará para validar los modelos. Esta columna es equivalente a la columna *text* del dataset Financial Phrasebank.

- La columna *ratings* contiene una lista con todos los sentimientos con los que el equipo de riesgos ha puntuado la noticia

- Por último, la columna rating_global se corresponde con el *rating* final, calculado con el método de las medias explicado anteriormente.

Como se puede apreciar, este dataset no es muy extenso, lo que hará la tarea de validación algo complicada, pues un fallo en la clasificación de una de las noticias penalizará mucho más que si se contase con un dataset más extenso. A continuación se muestra la distribución de las noticias del dataset según el sentimiento.

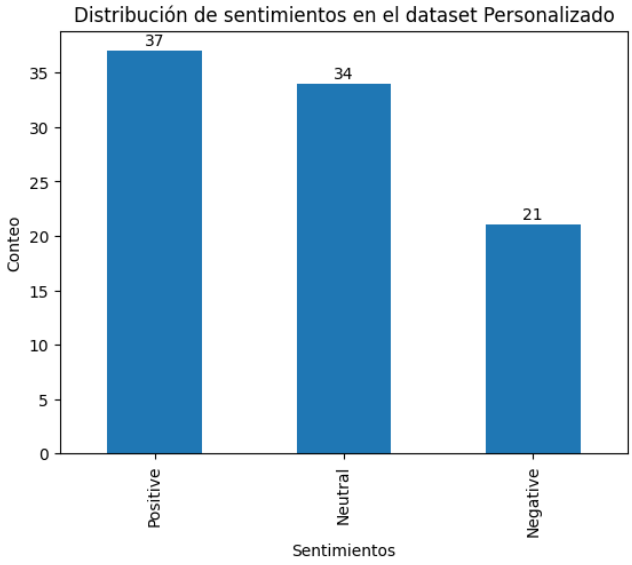

Figura 12: Muestra del dataset específico

En el caso de este dataset, se puede observar que está bastante más equilibrado que el dataset de Financial PhraseBank. Esto va a permitir que, aunque haya pocos datos, podamos tener una mejor visión de si los modelos son capaces o no de predecir correctamente todas las clases.

A continuación, se van a visualizar las nubes de palabras divididas en los tres sentimientos que contiene el dataset.

Figura 13: Nube de palabras de los textos positivos en el dataset específico

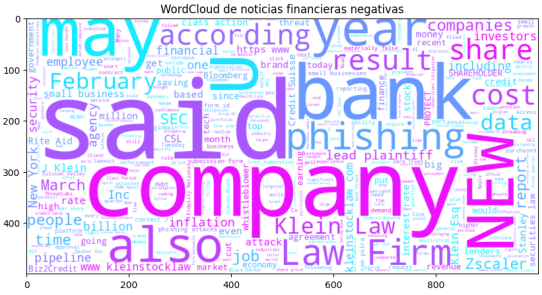

Figura 14: Nube de palabras de los textos negativos en el dataset específico

Figura 15: Nube de palabras de los textos neutros en el dataset específico

Se puede ver que las palabras más destacadas que aportan sentimientos dentro de cada nube de palabras son:

- Entre las palabras positivas se encuentran “year growth”, “expected”, “major”, “growth exhibit”...
- Entre las palabras negativas se encuentran “phishing”, “small business”, “inflation”...
- Entre las palabras neutras se encuentran “market”, “bank”, “share”, “company”, “stock”...

3.5. Procesamiento de lenguaje natural: Análisis de sentimientos.

3.5.1. Preparación de los datos

Una vez analizados los dos datasets, es necesario realizar una serie de técnicas para preparar los datos para que puedan ser introducidos en los modelos. En primer lugar, se ha de realizar una limpieza, aplicando diferentes técnicas de preprocesamiento de lenguaje natural. Una vez se tienen los datos limpios, se requiere generar *embeddings* para poder tener datos que sean entendibles por los modelos. Estos *embeddings* se realizarán de tres formas distintas, empleando los algoritmos de TF-IDF, Word2Vec y GLOVE, para comparar cuál de las combinaciones *embedding*-modelo funciona mejor a la hora de clasificar los sentimientos de las noticias.

3.5.1.1. Técnicas de preprocesado de los datos

El preprocesado de los datos nos asegura que se van a poder destacar las palabras y características más importantes del texto. Antes de realizar ninguna operación, las palabras deben ser divididas en partes del discurso y se deben realizar operaciones de preprocesamiento en cada una de las partes, teniendo en cuenta la estructura semántica del idioma. A continuación, se detallan las técnicas de preprocesado para NLP más comunes a día de hoy.

- **Tokenización:** La tokenización es un proceso en el procesamiento del lenguaje natural (NLP) que consiste en dividir un texto en unidades más pequeñas llamadas tokens. Los tokens pueden ser palabras individuales, caracteres, subcadenas o incluso n-gramas, que son secuencias continuas de n elementos, como letras o palabras.

  La tokenización tiene varios objetivos y aplicaciones en NLP. Al dividir un texto en tokens, se simplifica su análisis y procesamiento posterior. Los tokens pueden ser utilizados para construir un vocabulario, realizar conteos de palabras, realizar análisis de frecuencia y aplicar modelos de aprendizaje automático, entre otros. Además, la tokenización es útil para eliminar signos de puntuación y caracteres especiales, normalizar el texto y corregir errores ortográficos.

  Además de dividir el texto en palabras, la tokenización también puede considerar n-gramas. Un n-grama es una secuencia de n elementos consecutivos en

el texto, donde los elementos pueden ser caracteres o palabras. Los n-gramas son útiles para capturar información contextual y relaciones entre palabras en un texto.

- **StopWords:** En NLP es importante contar con una gran cantidad de datos, pero es más importante aún que los datos que tengamos nos aporten información. Las Stopwords son palabras que no contienen información muy relevante para el entrenamiento de los modelos de NLP, por lo tanto, se las denomina también *noisy words* y se debe considerar eliminarlas en el preprocesamiento de los datos. Para ello, hay que tener en cuenta el idioma de las noticias, ya que existen diferentes librerías de stopwords dependiendo del idioma.

  Sin embargo, de cara a analizar sentimientos, puede ser relevante incluir en el análisis y entrenamiento de los modelos algunas stopwords, así como los caracteres numéricos. En este caso, en el que se necesita que todas las oraciones mantengan su estructura y su significado, se ha realizado la prueba de eliminar las stopwords sobre un subconjunto de tres oraciones, cada una de ellas con su polaridad de sentimientos. En la imagen siguiente se puede ver como, al eliminar estas palabras, el sentimiento de la oración cambia por completo. Por lo tanto, se ha tomado la decisión de no realizar este paso en el preprocesado de los datos.

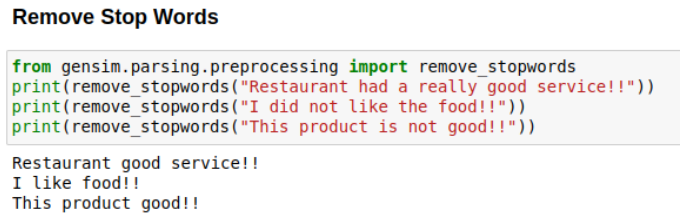

Figura 16: Resultado de eliminación de stopwords

- **Stemming:** Tiene como objetivo encontrar la raíz de una palabra. Las palabras pueden tener diferentes variantes morfológicas, como plurales, tiempos verbales o formas derivadas, pero estas variantes a menudo tienen interpretaciones semánticas similares. La técnica de stemming busca reducir el número total de palabras diferentes en un texto al reducir todas las variantes morfológicas a una forma base común, llamada raíz. Por ejemplo, las palabras "corriendo", "corre" y "corrió" pueden ser reducidas a la raíz "corr-", ya que todas comparten la misma raíz y tienen un significado similar relacionado con la acción de correr.

  Esta técnica es útil en el procesamiento de texto porque al reducir el número de palabras diferentes, se puede simplificar el análisis y el procesamiento posterior del texto. Esto, a su vez, puede mejorar el rendimiento y la eficiencia de los modelos de NLP, ya que se requiere menos tiempo y recursos para procesar un vocabulario más pequeño.

- **Lemmatization:** Tiene como objetivo encontrar la forma base o raíz léxica de una palabra. A diferencia del Stemming, que busca reducir las palabras a su forma más básica, la lematización considera el contexto y la categoría gramatical de las palabras para determinar su raíz.

El proceso de lematización implica utilizar un lematizador, que es un componente específico del NLP, capaz de identificar la forma base de las palabras. Por ejemplo, para el verbo "corriendo", el lematizador puede identificar su raíz léxica como "correr". De manera similar, para el sustantivo "ratones", el lematizador puede determinar que su raíz es "ratón".

Para llevar a cabo la lematización de manera precisa, es necesario tener en cuenta la tipología gramatical de las palabras. Esto implica etiquetar cada palabra con su categoría gramatical correspondiente, como sustantivo, verbo, adjetivo, adverbio, etc., utilizando un algoritmo de etiquetado POS (Part-of-Speech).

- **Part-of-speach (POS) tagging:** es un proceso en el procesamiento del lenguaje natural que se encarga de categorizar las palabras en una frase según su contexto dentro del discurso. El objetivo es asignar a cada palabra una etiqueta gramatical específica, como sustantivo, verbo, adjetivo, adverbio, pronombre, preposición, etc.

  El etiquetado POS se realiza mediante el uso de clasificadores, que son algoritmos responsables de determinar las etiquetas que representan la semántica de las palabras en el contexto de la frase en la que se encuentran. Estos clasificadores analizan diversos factores, como la posición de la palabra en la oración, su relación con otras palabras cercanas y las estructuras gramaticales típicas del idioma.

- **Named Entity Recognition:** es una metodología fundamental en NLP que tiene como objetivo identificar y clasificar términos específicos en un texto, como nombres de personas, organizaciones, lugares y otras entidades. El NER se ha convertido en una de las principales técnicas en NLP debido a su capacidad para predecir y etiquetar de manera precisa diferentes tipos de entidades en una frase. Estas entidades pueden incluir nombres de equipos deportivos, marcas comerciales, categorías de productos, nombres de personas, instituciones, países...

  Aunque para esta fase del proyecto no es necesaria esta técnica, de cara a próximos pasos, en los que se completará la construcción de la herramienta final, para ser capaces de realizar una búsqueda por un cliente concreto es necesario ser capaces de detectar de qué entidades habla la noticia. Para ello, se puede emplear esta técnica, que nos permite conocer qué entidades (compañías) aparecen en la noticia, permitiéndonos extraerlas y agrupar cada noticia por compañía.

  En Python existen diferentes librerías que permiten realizar este tipo de reconocimiento de entidades. Entre las más utilizadas destacan NLTK y Spacy. Ambas devuelven un detalle de todas las entidades existentes en una oración. A continuación, se muestra un ejemplo. Se ha tomado como entrada la siguiente oración:

  > The White House meeting came after Biden returned from a trip to Asia early to hammer out a deal ahead of the US Treasury's June 1 cut off date for Congress to authorize more borrowing.

A partir de esta oración, mediante la librería Spacy se puede generar una visualización de todas las entidades que contiene la oración:

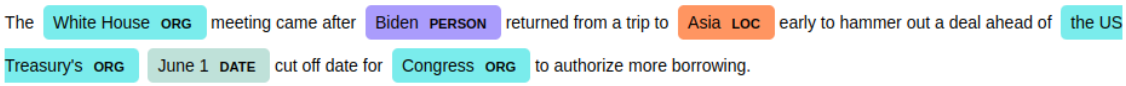

Figura 17: Técnica NER con librería Spacy

Como se puede ver, con esta técnica se podrán obtener todas las entidades que sean interesantes de cara a agrupar las diferentes noticias.

Tras el análisis de todas las técnicas de preprocesado que se ha realizado, se ha decidido aplicar a los datasets de entrenamiento y validación las técnicas de tokenizado y stemming, concretamente el método de Porter Stemmer, ya que es el más común dentro de los algoritmos de *stemming* disponibles en Python.[38].

La eliminación de Stopwords se descartó al ver que el significado de las oraciones podía cambiar significativamente. Puesto que es necesario conocer la polaridad de cada una de las palabras de las oraciones, se ha decidido mantener todas ellas.

En un estudio comparativo entre las técnicas de Stemming y lematización, se ha observado que se han obtenido mejores resultados en la clasificación con la técnica de Stemming. Esto se debe en parte a la dependencia de un algoritmo de etiquetado llamado Part-of-Speech (POS), utilizado en la lematización para identificar correctamente la categoría gramatical de cada palabra. Sin embargo, la precisión del POS puede variar y, en algunos casos, puede no identificar correctamente las categorías gramaticales de todas las palabras.

La lematización depende tanto del algoritmo de etiquetado POS como del proceso de encontrar la raíz léxica de las palabras. Si el algoritmo POS no logra identificar correctamente la categoría gramatical de una palabra, esto puede llevar a que la lematización produzca resultados erróneos. Por otra parte, el Stemming no está tan afectado por este problema, ya que se enfoca en reducir las palabras a su forma base sin tener en cuenta el contexto gramatical completo.

Es por esto que se ha decidido aplicar la técnica de Stemming tras la tokenización de los datos como técnica de preprocesamiento de NLP.

3.5.2. Generación de embeddings

Para generar los *embeddings*[8] que servirán como datos de entrada para el entrenamiento de los diferentes modelos de machine learning que se han seleccionado para este TFM, se han empleado tres algoritmos, TF-IDF[29][10], Word2Vec [26][11]y GLOVE[37][20], descritos en la sección de 2.3.1.2 estado del arte de este documento. El procedimiento seguido para la generación de los *embeddings* ha sido similar en los tres casos, variando únicamente el algoritmo con el que se generan los *embeddings*. Estos han de generarse para los dos datasets empleados en el estudio, tanto Financial Phrase Bank como el específico.

1. En primer lugar se aplican las técnicas de preprocesado que se ha decidido emplear para este dataset. Se tokenizan las oraciones en palabras individuales y, una vez tokenizadas, se aplica el algoritmo Porter Stemmer a los tokens. Una vez hecho esto, ya se puede pasar a generar los tokens.

2. Al tener los datos separados por palabras, se puede generar un diccionario de *embeddings*, en el que cada vector se corresponda con una palabra del diccionario. Para ello, se aplican los diferentes algoritmos de generación de *embeddings* sobre cada una de las palabras obtenidas a partir de la tokenización, y se almacenan los modelos de generación de *embeddings* tras haberlos entrenado en el conjunto de palabras generado, junto con el diccionario generado. El siguiente paso será entrenar los modelos. En este punto se dan dos escenarios:

    2.1. A la hora de entrenar los modelos de clasificación clásicos como por ejemplo Random Forest o Naive Bayes, como resultado de la aplicación del paso anterior, se dispone de un diccionario en el que cada vector representa una palabra. Lo que se hace a continuación es generar un vector por oración. Esto se consigue calculando la media de todos los vectores correspondientes a las palabras que componen la oración. Esta es una técnica muy habitual empleada en NLP. Si bien es cierto que hay parte de la información que se pierde al hacer la media de los *embeddings* de cada palabra, se mantiene la información sintáctica y el contexto de cada una de las palabras de la oración en el *embedding* final.

    Una vez generados los vectores para cada una de las oraciones, se puede comenzar a entrenar los modelos, estableciendo como entrada el vector resultante de la media de los vectores asociados a las palabras que conforman la oración a analizar.

    2.2. Por otro lado, en el caso de las redes neuronales LSTM (Long Short-Term Memory) y RNN (Recurrent Neural Network), es importante destacar que se han empleado los *embeddings* de las palabras previamente generados, y no los de las oraciones obtenidos a partir de la media como en el resto de modelos. Esto se debe a que los *embeddings* de cada palabra permiten capturar el contexto semántico de las palabras, lo cual resulta fundamental para el correcto funcionamiento de estos modelos. Sin el uso de los *embeddings* de palabras, la capacidad de comprensión y predicción de las redes neuronales LSTM y RNN se vería significativamente limitada, ya que no podrían capturar de manera efectiva las relaciones y significados de las palabras dentro de las oraciones. Por lo tanto, el entrenamiento con *embeddings* de palabras garantiza una mayor capacidad de contextualización y comprensión en estos modelos de procesamiento de lenguaje natural.

3.5.3. Modelos de Machine Learning

3.5.3.1. Modelos desde cero

En esta sección, se realiza un análisis de los modelos de Machine Learning comúnmente empleados en la detección de sentimientos [7]. Estos modelos son los que han sido entrenados sobre el dataset Financial Phrasebank y sobre los que se ha

llevado a cabo una comparación para tratar de encontrar el modelo que mejor se comporta clasificando sentimientos financieros.

Se abordarán los beneficios asociados con cada uno de estos modelos de clasificación utilizados en el proceso de selección y entrenamiento . Dicho análisis permitirá comprender en mayor profundidad las ventajas que cada modelo ofrece en el contexto de la detección de sentimientos.

Para el entrenamiento de todos los modelos seleccionados se ha empleado la técnica de *Grid Search*, de cara a encontrar los mejores hiperparámetros que hacen que el modelo tenga una mejor *performance* a la hora de clasificar sentimientos. A continuación, se van a exponer los diferentes modelos empleados en la comparación, indicando de cada uno de ellos los parámetros más útiles y significativos, es decir, los que conviene optimizar mediante Grid Search.

3.5.3.1.1. Modelo Naive Bayes

El modelo Naive Bayes[2] es especialmente útil en el análisis de sentimientos debido a su simplicidad y eficiencia computacional. Puede manejar grandes volúmenes de datos y es rápido en términos de entrenamiento y predicción. Además, el modelo puede manejar características categóricas y funciona bien incluso cuando hay una gran cantidad de características en comparación con el tamaño del conjunto de datos.

El modelo elegido para abordar la clasificación de sentimientos en un problema multiclase ha sido el MultinomialNB, perteneciente a la biblioteca de Python, Scikit Learn. Este modelo se caracteriza por su simplicidad y no requiere una configuración compleja de parámetros. En el proceso de búsqueda de hiperparámetros mediante Grid Search, se han considerado únicamente los parámetros alpha y fit_prior. Estos parámetros permiten ajustar el suavizado de Laplace y la inclusión o exclusión de las probabilidades a priori, respectivamente.

3.5.3.1.2. Modelo Decission Tree

Los árboles de decisión [6] son conocidos por su capacidad para capturar relaciones no lineales y complejas en los datos. En el contexto de la clasificación de sentimientos, un árbol de decisión puede analizar características lingüísticas y patrones en los textos para determinar la clase de sentimiento correspondiente. Los nodos del árbol representan condiciones sobre las características de entrada, mientras que las ramas representan las posibles decisiones basadas en esas condiciones. La estructura jerárquica del árbol permite una interpretación clara y explícita de cómo se toman las decisiones de clasificación.

Además, los árboles de decisión pueden manejar características categóricas y numéricas, y son menos susceptibles al sobreajuste en comparación con otros modelos más complejos. Esto los hace especialmente adecuados para problemas de clasificación de sentimientos en NLP, donde es importante comprender las razones detrás de las predicciones.

Entre los parámetros principales que se van a incluir en la búsqueda Grid Search se encuentran:

- criterion: Determina la medida de calidad utilizada para seleccionar la mejor división en cada nodo del árbol de decisión (por ejemplo, "gini" o "entropy").
- max_depth: Establece la profundidad máxima permitida para el árbol de decisión, controlando la complejidad y evitando el sobreajuste.
- min_samples_split: Especifica el número mínimo de muestras requeridas para realizar una división en un nodo interno del árbol.
- min_samples_leaf: Define el número mínimo de muestras requeridas en una hoja (nodo final) del árbol de decisión.
- max_features: Limita el número máximo de características consideradas al buscar la mejor división en cada nodo.

El resto de hiperparámetros configurables del Árbol de Decisión que no aparecen en el listado anterior se han mantenido con su valor por defecto, puesto que no se han considerado tan relevantes como los indicados.

3.5.3.1.3. Modelo Gradient Boosting

Gradient Boosting [4] es una técnica que combina múltiples modelos débiles, como árboles de decisión, en un modelo más fuerte y robusto. Esta metodología se enfoca en mejorar el rendimiento del modelo al ir iterando y corrigiendo los errores cometidos por los modelos previos.

En el contexto de la clasificación de sentimientos en NLP, Gradient Boosting tiene la capacidad de aprender características complejas y sutiles en los textos, lo que permite una representación más precisa de los sentimientos.

Además, este enfoque puede manejar características tanto categóricas como numéricas, lo que lo hace adecuado para el análisis de texto. Gradient Boosting también es conocido por su capacidad para manejar conjuntos de datos desequilibrados, como es el caso de los dos datasets de los que se dispone en este proyecto. Al combinar la potencia de múltiples modelos y abordar los desafíos específicos del problema, Gradient Boosting puede mejorar la precisión y la capacidad de generalización del modelo de clasificación de sentimientos.

Los parámetros que se han optimizado mediante la búsqueda de Grid Search han sido los siguientes:

- n_estimators: Determina el número de estimadores (árboles de decisión débiles) utilizados en el algoritmo de Gradient Boosting.
- learning_rate: Controla la contribución de cada estimador al modelo general y afecta la velocidad de convergencia del algoritmo.
- max_depth: Define la profundidad máxima permitida para cada estimador en el conjunto, lo que limita la complejidad y evita el sobreajuste.
- max_features: Establece el número máximo de características consideradas al buscar la mejor división en cada estimador.

- min_samples_split: Especifica el número mínimo de muestras requeridas para realizar una división en un nodo interno del estimador.
- min_samples_leaf: Define el número mínimo de muestras requeridas en una hoja (nodo final) del estimador.

3.5.3.1.4. Modelo Random Forest

Random Forest [15] es un modelo que combina múltiples árboles de decisión independientes, entrenados en diferentes conjuntos de datos de muestra y características, para obtener una predicción conjunta. Esta metodología de ensamble proporciona varias ventajas para la clasificación de sentimientos en NLP. En primer lugar, Random Forest puede capturar relaciones no lineales y complejas presentes en los textos, permitiendo una representación más precisa de los sentimientos. Además, al utilizar múltiples árboles, el modelo puede evitar el sobreajuste y mejorar la capacidad de generalización. Random Forest también puede manejar características tanto categóricas como numéricas, lo que es crucial en el análisis de sentimientos que involucra datos de texto, como es el caso de las noticias financieras. Esta característica es muy importante de cara a analizar cantidades y volúmenes de datos numéricos que puedan aparecer en una noticia financiera.

Los parámetros que se han optimizado en la búsqueda Grid Search han sido los siguientes:

- n_estimators: El número de árboles de decisión en el bosque aleatorio.
- max_depth: La profundidad máxima de cada árbol de decisión en el bosque.
- min_samples_split: El número mínimo de muestras requeridas para realizar una partición en un nodo interno.
- min_samples_leaf: El número mínimo de muestras requeridas para considerar un nodo como una hoja.
- max_features: El número máximo de características a considerar al buscar la mejor partición en cada nodo.

3.5.3.1.5. Modelo SVM

SVM[3] es un algoritmo de clasificación lineal que tiene la capacidad de separar eficientemente clases en un espacio de características de alta dimensión. En el contexto de la clasificación de sentimientos en NLP, SVM puede aprovechar características lingüísticas y semánticas complejas para discernir las emociones asociadas con los textos. Además, SVM es eficaz para manejar datos desequilibrados y resolver problemas multiclase, lo cual es muy útil en el caso de los datasets empleados, puesto que en ambos las clases están desbalanceadas.

SVM puede aprender fronteras de decisión no lineales mediante el uso de funciones de kernel, lo que lo hace altamente flexible y capaz de capturar relaciones no lineales en el lenguaje natural. También es conocido por su capacidad para generalizar bien en conjuntos de datos pequeños y lidiar con la maldición de la dimensionalidad.

En el caso de este modelo, únicamente se ha tratado de optimizar mediante Grid Search el parámetro C, que se encarga de controlar el equilibrio entre la maximización del margen y la clasificación incorrecta de puntos de datos.

3.5.3.1.6. Modelo RNN

Las RNN[9] son capaces de modelar relaciones secuenciales y capturar dependencias a largo plazo en el texto, lo cual es esencial para comprender y clasificar los sentimientos en un contexto lingüístico. Al procesar secuencias de palabras, las RNN pueden tener en cuenta el contexto previo para asignar un sentimiento específico a una palabra o frase determinada. Esto les permite capturar el flujo de información a través de la secuencia y considerar la evolución de los sentimientos a medida que se desarrolla el texto. Además, las RNN tienen la capacidad de aprender representaciones semánticas contextualizadas, lo que les permite capturar matices emocionales y comprender el tono general del texto. En el análisis de sentimientos en NLP, donde las palabras y las estructuras lingüísticas pueden afectar significativamente la polaridad de un texto, las RNN pueden aprovechar su capacidad para modelar la dependencia temporal y tomar decisiones de clasificación más precisas. Es por esto que este modelo se entrena a partir de *embeddings* de palabras, y no de oraciones completas, como hacen el resto de modelos anteriores.

La red neuronal RNN que se va a entrenar en este proyecto tiene la siguiente estructura:

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, None, 32)          320000

 simple_rnn_1 (SimpleRNN)     (None, 32)                2080

 dense_1 (Dense)              (None, 3)                 99

=================================================================
Total params: 322,179
Trainable params: 322,179
Non-trainable params: 0
```

Figura 18: Estructura de la red RNN

Este modelo se ha configurado con una función softmax en la ultima capa, de cara a normalizar las salidas así como mejorar la representación de la incertidumbre del modelo. A su vez, se ha establecido como función de pérdida la función *categorical_ crossentropy*, debido al problema multiclase.

Gradient vanishing problem: A medida que la información se desplaza de las neuronas de entrada a las de salida a lo largo de la red neuronal, ésta actualiza sus pesos en función del error y realiza retro propagación. Los pesos asignados aleatoriamente próximos a cero caen por debajo de cero a lo largo del tiempo con actualizaciones repetidas de los pesos para reducir el error. Este problema se denomina

problema de desaparición del gradiente. Y es por esto que surgieron las denominadas LSTM (*Long Short Term Memory*).

3.5.3.1.7. Modelo LSTM

Las LSTMs[5] son una variante de las redes neuronales recurrentes (RNN) que se destacan por su capacidad para capturar dependencias a largo plazo en secuencias de texto. Esto es especialmente relevante en la clasificación de sentimientos, ya que las palabras y las estructuras lingüísticas pueden estar interconectadas en un texto y afectar el tono emocional general.

Las LSTMs pueden aprender patrones temporales complejos y descubrir relaciones sutiles en el texto a través de sus unidades de memoria y compuertas. Esto les permite modelar la evolución de los sentimientos a medida que se desarrolla el texto y tomar decisiones de clasificación más precisas.

Además, las LSTMs son capaces de manejar secuencias de longitud variable, lo que es beneficioso para el análisis de sentimientos en NLP, donde las oraciones o textos pueden tener longitudes diferentes. Al procesar secuencias de palabras, las LSTMs pueden aprender representaciones semánticas contextualizadas y capturar matices emocionales sutiles en el texto.

La red neuronal LSTM que se va a entrenar en este proyecto tiene la siguiente estructura:

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding (Embedding)        (None, None, 64)          640000

 lstm (LSTM)                  (None, 64)                33024

 dense (Dense)                (None, 64)                4160

 dense_1 (Dense)              (None, 3)                 195

=================================================================
Total params: 677,379
Trainable params: 677,379
Non-trainable params: 0
```

Figura 19: Estructura de la red LSTM

La configuración de los parámetros de esta red es similar a la empleada en el modelo anterior (RNN), se ha establecido una función *softmax* en la ultima capa así como la función de pérdida *categorical_ crossentropy*.

Función de pérdida personalizada: Las funciones de pérdida en las redes neuronales desempeñan un papel fundamental para ajustar el modelo a la lógica empresarial. En un entorno de producción, es común contar con tres modelos distintos: uno optimista, otro neutro y otro pesimista. Estos modelos se utilizan en

función de diferentes escenarios financieros.

Cuando la empresa está experimentando una buena situación financiera, con una disponibilidad mayor de fondos y un bajo índice de incumplimiento, se emplea un modelo optimista. Este modelo se enfoca en detectar predominantemente noticias positivas, con el objetivo de aprovechar al máximo las oportunidades favorables. Por otro lado, en momentos en los que la situación financiera no es tan favorable, como en períodos de mayor incumplimiento o durante una crisis, se utiliza un modelo más conservador. Este modelo se centra en detectar de manera más efectiva las noticias negativas, ya que el principal objetivo es evitar el incumplimiento. Además, existe un modelo genérico que asigna un peso equilibrado a todas las predicciones, buscando ofrecer una perspectiva más balanceada y abordar diferentes escenarios.

Estos tres modelos permiten adaptar la detección de noticias y la toma de decisiones a las condiciones financieras cambiantes de la empresa. Al ajustar las funciones de pérdida en cada modelo, se logra una mayor alineación con los objetivos y la estrategia de negocio. Esto proporciona flexibilidad y capacidad de respuesta para enfrentar diferentes situaciones y maximizar los resultados en función de las circunstancias específicas.

Para mitigar este problema, se ha aplicado una función de pérdida que pondera las clases en función de su presencia en el conjunto de datos. Esto implica asignar un mayor peso a las clases minoritarias durante el proceso de entrenamiento, lo que permite al modelo aprender de manera más efectiva a detectar patrones y características específicas de esas clases subrepresentadas. Al hacerlo, se busca mejorar la capacidad de generalización del modelo y lograr una mejor precisión en la predicción de todas las clases, incluso aquellas con menor frecuencia en los datos de entrenamiento.

3.5.3.2. Modelos Preentrenados

3.5.3.2.1. Modelo FinancialBert

Tal y como se ha descrito en el estado del arte, este modelo es un modelo Bert al que se le ha hecho fine tunning sobre varios datasets, entre ellos el dataset Financial Phrasebank.[14]

Este modelo tiene varias aplicaciones, como predecir los precios de mercado, etc. Sin embargo, para este proyecto, se va a emplear el modelo "ahmedrachid/ FinancialBERT-Sentiment-Analysis", disponible en hugging face.

Para utilizar este modelo, simplemente se debe importar a través de la biblioteca transformers. Además, para asegurar su correcto funcionamiento, se debe importar el tokenizador específico correspondiente a este modelo, cuyo nombre coincide con el modelo mismo. Para utilizar el modelo, basta con crear un pipeline que incluya tanto el tokenizador como el modelo. Una vez configurado el pipeline, solo se deben proporcionar las noticias financieras al modelo, sin necesidad de generar embeddings, ya que esto es realizado automáticamente por el tokenizador. El modelo devolverá la etiqueta asignada a la noticia introducida (positiva, negativa o neutral), junto con un

score que indica el nivel de confianza con el que el modelo ha realizado su predicción.

Este modelo será empleado para validar los sentimientos de las noticias del dataset específico, que han sido etiquetadas por el equipo de riesgos. Este dataset contiene artículos de noticias completos, por lo que las oraciones y el contenido que se introduce al modelo es mucho más largo que el del dataset Financial PhraseBank. Por ello, en ocasiones es necesario particionar el artículo en varios bloques.

Una vez el modelo predice el sentimiento para cada uno de los bloques de la noticia, se realiza una ponderación de los sentimientos de cada bloque para asignarle un sentimiento final a la noticia completa. Esta ponderación se realiza de forma similar a la que se aplica cuando una noticia del dataset específico ha sido etiquetada por más de una persona. Se asignan los valores 1, 0 y -1 a los sentimientos "Positivo", "Negativo" y "Neutro" respectivamente, y se calcula una media. Si esta media es mayor que 0, la noticia se considera positiva, si es menor negativa, y si está muy cercana a 0, será neutra.

3.5.3.2.2. Modelo GPT3.5 Turbo

Dentro la API de OpenAI[24] existen multitud de modelos y métodos que se pueden emplear según las tareas que se quieran realizar. En concreto, para la realización de este proyecto se ha empleado el *endpoint* de "Completion". Este *endpoint* permite generar texto a partir de un *prompt* que se le envía como parámetro a la petición.

Además de seleccionar el *endpoint*, es necesario especificar el modelo[25] con el que se quiere generar el texto. Existen multitud de modelos, en el caso de este proyecto se han probado dos de ellos "gpt-3.5-turbo" y "text-davinci-003". El modelo que finalmente se comparará con FinancialBert será gpt-3.5-turbo, puesto que los resultados que aporta son equivalentes a los del modelo Davinci y, sin embargo, las llamadas a la API con este modelo son mucho más económicas.

Uno de los puntos clave a tener en cuenta con la utilización de este modelo es la forma en la que se genera el *prompt*. El *prompt* es el mensaje que se le envía al modelo para que haga lo que se le pida. En este caso, se quiere que clasifique noticias en función de sus sentimientos financieros y les asigne una de las tres clases establecidas. tras múltiples pruebas, el mensaje resultante enviado contiene lo siguiente:

> Decide whether a new's sentiment is positive, neutral, or negative and the percentage of accuracy of your response.
> New: "*text of the new*".
> Sentiment - Score:

Con este *prompt* el modelo suele devolver la respuesta estructurada según lo que se le indica. Sin embargo, en ocasiones devuelve más texto del que debería, justificando su decisión, puesto que al fin y al cabo es un modelo de generación de texto. Por lo tanto, hay veces en las que es necesario limpiar la respuesta y adecuarla a la estructura requerida, de cara a comparar los resultados con el modelo de FinancialBert, que devuelve sus predicciones acorde a la estructura indicada[39].

Capítulo 4

Desarrollo

4.1. Introducción

En este capítulo se explican todos los procesos y detalles del desarrollo más técnico del proyecto. Se comienza detallando las diferentes tecnologías que se han empleado para el desarrollo de este proyecto. A continuación, se muestra el dataset resultante tras aplicar las diferentes técnicas de preprocesamiento de NLP. Más adelante, se incluye una tabla con el resumen de los tiempos de ejecución del entrenamiento de los diferentes modelos y los hiperparámetros seleccionados para cada uno de ellos tras la optimización. A continuación, se explica el proceso de despliegue de la aplicación web de puntuación de noticias en GCP. Y, por último, se explica la estructura del código generado.

4.2. Tecnologías utilizadas

En esta sección se van a describir las principales tecnologías que han sido empleadas para la realización de este proyecto y cómo se relacionan entre sí. En concreto, las tecnologías empleadas se pueden dividir en dos grandes subgrupos: codificación e infraestructura.

- En la sección de codificación, la tecnología empleada ha sido **Python**, pues es el lenguaje de programación más utilizado en problemas de Machine Learning. Entre las principales librerías empleadas destacan las siguientes:
    - **Scikit Learn**. Esta librería contiene todos los métodos necesarios para la aplicación de los modelos de Machine Learning aplicados desde cero. Además, también contiene métodos de generación de vectores. Para este proyecto, se han utilizado TFIdfVectorizer del módulo de Feature Extraction para la generación de los *embeddings* con TF IDF, así como los métodos de modelos como Gradient Boosting o Random Forest del módulo de *ensemble*, o Decission Tree. Por último, se han empleado métodos de métricas para extraer resultados.
    - **Gensim**. Esta librería contiene métodos muy utiles para la aplicación de técnicas de NLP. En concreto, se han utilizado los métodos de preprocesamiento de tokenización y Porter Stemming, y los de generación de *embeddings* de Word2Vec.

- **Keras** y **Tensorflow**. Estas librerías han sido empleadas para crear y entrenar las redes neuronales RNN y LSTM desde cero empleadas en la comparación de los modelos.
- **Seaborn** y **Matplotlib**. Estas librerías han sido utilizadas para graficar los resultados de los modelos entrenados, en concreto, las curvas ROC y las matrices de confusión.
- **Streamlit**. Esta librería se ha empleado para la implementación de la aplicación web de puntuación de noticias.
- Librerías personalizadas de Ebury. Encargadas de la comunicación con la infraestructura de GCP. Mediante estas librerías es posible almacenar y leer datos de las diferentes tablas y Buckets de GCP, así como acceder al Secret Manager para la autenticación dentro de la infraestructura.

- En la sección de infraestructura, la tecnología empleada ha sido Google Cloud Platform. Para la gestión de bases de datos y almacenamiento de *ratings*, se ha utilizado Big Query. Los ficheros JSON en crudo generados después del *scrapping* se almacenan en Buckets, que son accedidos posteriormente por la aplicación web. Además, se ha empleado el Secret Manager de GCP para la autenticación mediante API de la aplicación y para acceder a los datos desde un *script* de Python. Por último, el despliegue de la aplicación se ha realizado utilizando Cloud Build de GCP. Estas herramientas y servicios de GCP han sido fundamentales para el éxito y eficiencia del proyecto.

4.3. Aplicación de técnicas de preprocesamiento de NLP

Se ha llevado a cabo la tokenización por palabra, evitando la utilización de bigramas u otras técnicas similares con el fin de simplificar el proceso y facilitar la generación de los *embeddings*. Tras la tokenización, es posible obtener una visualización de las palabras que más aparecen en el dataset.

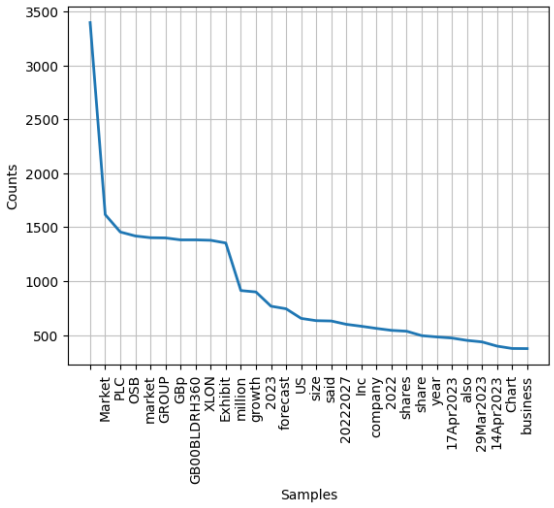

Figura 20: Top 30 palabras más comunes del dataset específico.

Tras la tokenización, se procede a aplicar el algoritmo de Porter Stemming a todas las palabras existentes en el dataset. A continuación, se muestra un ejemplo de algunas de las palabras originales y el resultado de aplicarles dicho *stemmer*.

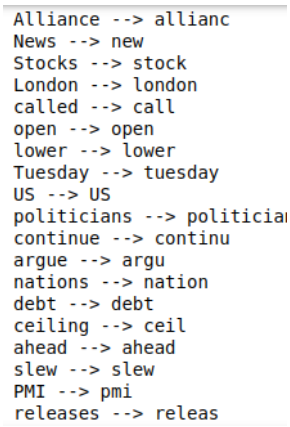

Figura 21: Resultado de la aplicación de Stemming a los tokens obtenidos.

Como se puede observar, algunas de las palabras (tokens) mantienen su formato original, como los nombres propios. Sin embargo, la mayoría de palabras derivadas, son reducidas a su raíz.

4.4. Aplicación de modelos de NLP

En esta sección se va a analizar cómo se han entrenado los diferentes modelos de Machine Learning desde cero. Se analizarán los hiperparámetros a optimizar, así como los diferentes tiempos de entrenamiento.

Tras la selección de los mejores parámetros para cada uno de los modelos con *grid search*, y con los datasets ya preparados con los vectores de cada oración generados a partir de los *embeddings* de cada una de las palabras, se procede al entrenamiento de cada uno de los modelos. En la sección de 5Pruebas y Resultados de este documento, se va a mostrar para cada uno de los modelos, el Classification Report resultante tras la validación de los modelos. A continuación, se va a indicar el tiempo de entrenamiento de cada uno de los modelos, así como los mejores hiperparámetros seleccionados para cada modelo, según los tipos de *embeddings*.

El procedimiento ha comenzado con la separación del dataset Financial Phrasebank en dos subconjuntos de datos, 80 % para entrenamiento y 20 % para validación. Una vez divididos los datos, los modelos son entrenados con el subconjunto de entrenamiento. Sobre este subconjunto es sobre el que se calculan los mejores hiperparámetros para cada modelo. Una vez entrenados, los modelos se almacenan en formato *pickle* para poder validarlos sobre los diferentes datasets sin tener que realizar el entrenamiento de nuevo. Tras el entrenamiento, se realiza una validación de cada uno de los modelos sobre el 20 % restante reservado para esta labor. Una vez validado el modelo, se genera el Classification Report y se calculan métricas como el Accuracy, Recall, F1 Score y Precision de cada uno de los modelos. Además, se grafican tanto la matriz de confusión resultante como las curvas ROC. Al ser un

problema multiclase, se ha visto la necesidad de adaptar la visualización de las curvas ROC, para poder ver cómo se comporta el modelo con cada una de las clases.

Por último, una vez entrenados y validados los modelos sobre el dataset Financial Phrase Bank, se realiza una última validación de los mismos modelos que se habían almacenado tras el entrenamiento, pero esta vez sobre el dataset específico, para comparar los resultados con los de FinancialBert.

4.4.1. Modelos desde cero. Entrenamiento

En esta sección se van a mostrar los tiempos de entrenamiento de cada uno de los modelos entrenados con los diferentes datasets generados a partir de la aplicación de diferentes algoritmos de generación de *embeddings*. Además, se indican los mejores parámetros para cada uno de los modelos, dependiendo de los diferentes escenarios de entrenamiento que se han encontrado mediante la técnica de Grid Search.

Modelo ML Embeding	**Tiempo de entrenamiento (ms)**	**Hiperparámetros seleccionados**
Naive bayes - TF-IDF	0.949	alpha:0.1 fit_prior: False
Naive bayes - Word2Vec	0.215	alpha:10.0 fit_prior: False
Naive bayes - GLOVE	0.112	alpha:1.0 fit_prior: False
Decission Tree - TF-IDF	29.164	criterion: entropy max_depth: 5 max_features: sqrt min_samples_leaf: 4 min_samples_split: 5
Decission Tree - Word2Vec	12.888	criterion: entropy max_depth: 5 max_features: sqrt min_samples_leaf: 4 min_samples_split: 5
Decission Tree - GLOVE	4.528	criterion: entropy max_depth: 5 max_features: sqrt min_samples_leaf: 4 min_samples_split: 5
Gradient Boosting - TF-IDF	264.049	n_estimators: 1000 learning_rate: 1.0 max_depth: 1 max_features: auto min_samples_split: 2 min_samples_leaf:

Cuadro 4.1: Entrenamiento de los modelos

Modelo ML Embeding	Tiempo de entrenamiento (ms)	Hiperparámetros seleccionados
Gradient Boosting - Word2Vec	51.6	n_estimators: 500 learning_rate: 1.0 max_depth: 1 max_features: auto min_samples_split: 2 min_samples_leaf:
Gradient Boosting - GLOVE	5.241	n_estimators: 100 learning_rate: 1.0 max_depth: 1 max_features: auto min_samples_split: 2 min_samples_leaf:
Random Forest - TF-IDF	1016.048	max_depth: None max_features: auto min_samples_leaf: 1 min_samples_split: 2 n_estimators: 200
Random Forest - Word2Vec	1030.258	max_depth: None max_features: auto min_samples_leaf: 4 min_samples_split: 10 n_estimators: 50
Random Forest - GLOVE	451.746	max_depth: 5 max_features: auto min_samples_leaf: 2 min_samples_split: 5 n_estimators: 100
SVM - TF-IDF	842.865	C: 10
SVM - Word2Vec	67.015	C: 1000
SVM - GLOVE	112.062	C: 10
RNN - TF-IDF	126.0527	No se han seleccionado parámetros para este modelo
RNN - Word2Vec	2.124	
RNN - GLOVE	1.178	
LSTM - TF-IDF	307.62	No se han seleccionado parámetros para este modelo
LSTM - Word2Vec	6.87	
LSTM - GLOVE	4.888	

Cuadro 4.2: Entrenamiento de los modelos

4.5. Despliegue de la aplicación web

De cara a desplegar la aplicación web para que los integrantes del equipo de riesgos tengan acceso a ella y puedan puntuar las noticias, es necesario realizar una serie de configuraciones. El despliegue de la aplicación se realizará en la infraestructura de GCP de la empresa, para que únicamente los empleados de la compañía puedan tener acceso a ella.

El proceso de despliegue comienza con la preparación de la aplicación, asegurándose de que todos los requisitos, incluidas las dependencias de Python, estén correctamente especificados en un archivo de requirements.txt. Luego, se procede a crear una imagen Docker que contenga todos los componentes necesarios para desplegar la aplicación en GCP. Es importante tener en cuenta que la imagen Docker debe ser construida de tal manera que incluya todas las dependencias necesarias para la aplicación.

Una vez creada la imagen Docker, se debe subir a Container Registry, el registro de contenedores de GCP. Esto permitirá que la aplicación esté disponible para ser desplegada en Cloud Run y se verifica que la aplicación esté correctamente configurada.

Para gestionar la autenticación de la aplicación, se configuran los secretos necesarios en Secret Manager, asegurándose de que los secretos estén almacenados de forma segura. Se configura un trigger en Cloud Run para que esté conectado al repositorio de GitHub de la aplicación, lo que permite que se redespliegue automáticamente cuando se realicen cambios en el repositorio.

Una vez que se haya verificado la conexión con el repositorio de GitHub, se procede a desplegar la aplicación en Cloud Run. Es importante asegurarse de que la aplicación esté correctamente configurada antes de desplegarla en Cloud Run. Una vez configurado el trigger de GitHub y desplegado el servicio en Cloud Run, cualquier cambio realizado en el repositorio de GitHub desencadenará automáticamente un redespliegue de la aplicación. Esto garantiza que los cambios se apliquen de forma rápida y eficiente en el entorno de producción.

4.6. Explicación del código generado

En esta sección se proporcionará una breve descripción del código generado para el proyecto. Para empezar, se ha desarrollado el código utilizando PyCharm, un entorno de desarrollo integrado que ofrece una plataforma eficiente para la creación de proyectos en Python. Además, las tres fases del proyecto han sido implementadas dentro del mismo proyecto de PyCharm. La estructura de directorios del proyecto se organiza de la siguiente manera:

- En primer lugar, en el directorio **config**, se almacena el fichero de configuración que contiene toda la información acerca de la ubicación de los directorios de los ficheros, la configuración de los *logs*, y las conexiones con los diferentes servicios de GCP como el *Secret Manager* o la URL de los *buckets*.

- En segundo lugar, en el directorio **data** se almacena una copia de los ficheros de noticias que se almacenan principalmente en los *buckets*. Este almacenamiento local tiene como principal motivo permitir el análisis de los datos de forma rápida, sin necesidad de realizar la conexión con GCP. En este directorio se almacenan además los datos procesados como los diccionarios de *embeddings* generados.

- En el directorio **images** se almacenan las gráficas generadas, así como la visualización de los resultados de aplicar las diferentes técnicas de preprocesamiento a los datos.
- En el directorio de **logs** se almacena un registro de los logs de todas las ejecuciones del código. Para ser capaces de detectar fallos y seguir el flujo de ejecución del código de forma más sencilla, sin necesidad de navegar por la terminal.
- En el directorio **models** se almacenan los modelos entrenados, tanto los modelos de ML empleados para clasificar los sentimientos de las oraciones como los modelos de vectorización y generación de embeddings.
- En el directorio **notebooks** se almacenan los diferentes notebooks empleados para realizar pruebas rápidas de código, así como el análisis exploratorio inicial de los datos.
- En el directorio **scripts** se almacena un *script* que permite instalar y replicar el entorno virtual empleado en la ejecución del proyecto.
- En el directorio **src** es donde se almacena el código del proyecto, dividido en diferentes módulos según la funcionalidad de cada *script*:
    - En primer lugar, en el directorio **data_connection** contienen los scripts para obtener las noticias a través de la API de Marketaux y almacenar los ficheros resultantes en Buckets. En estos scripts se realiza la conexión con la API, la obtención de las noticias completas, así como la limpieza de los datos y el almacenamiento del JSON resultante en GCP.
    - A continuación, en el directorio **flow_control** existe un único script que contiene métodos de cara a la autenticación con GCP y obtención de los datos del usuario conectado para su posterior utilización.
    - El siguiente directorio, **machine_learning** contiene todos los scripts relacionados con la aplicación de modelos de ML sobre los diferentes datasets. En concreto hay 4 scripts.
        - El primero de ellos, **bert**, contiene el código de validación del modelo BERT sobre el dataset específico. Este script almacena los resultados de las predicciones para su posterior análisis y comparación.
        - El segundo, **embeddings**, se encarga de generar los embeddings con los tres algoritmos empleados (TF-IDF, Word2Vec y Glove), y almacenarlos en ficheros que más adelante se emplearán para el entrenamiento de los modelos.
        - El siguiente script, **modesl_from_scratch**, contiene el código de todos los modelos entrenados para la comparación. Contiene también la aplicación de técnicas de preprocesamiento de los datos así como métodos para graficar las matrices de confusión y curvas roc de todos los modelos.
        - Por último, el script **openai_gpt** es similar al de BERT, contiene el código necesario para obtener las predicciones del modelo GPT3.5 turbo sobre el dataset específico.

- Por último, el directorio **utils** que contiene las queries realizadas para obtener la información que sea necesaria de la base de datos de Big Query. Así como otros métodos útiles para configuración del proyecto.
- El fichero **main.py** contiene el código de ejecución de la obtención de noticias. Este script hace uso de los métodos definidos en el data_connector
- El fichero **webapp.py** contiene la estructura del diseño de la interfaz de la aplicación web de puntuación de noticias, así como la lógica de mostrar las noticias y el almacenado de las puntuaciones en la base de datos destinada para ello.

- El directorio **venv** contiene la configuración del entorno virtual necesario para la ejecución del proyecto. Incluye librerías como streamlit, tensorflow o scikit-learn entre otras.
- El fichero **cloudbuild_webapp.yaml** contiene información sobre el despliegue de la aplicación en GCP. Indica el nombre del Docker en el que se debe desplegar así como la configuración del Docker una vez desplegado, como el tiempo de ejecución de la aplicación, etc.
- Por último,el fichero **dockerfile_webapp** contiene información acerca de la configuración del Docker, como los ficheros a incluir, el puerto al que conectarse desde el navegador para poder acceder y el comando necesario para ejecutar la aplicación.

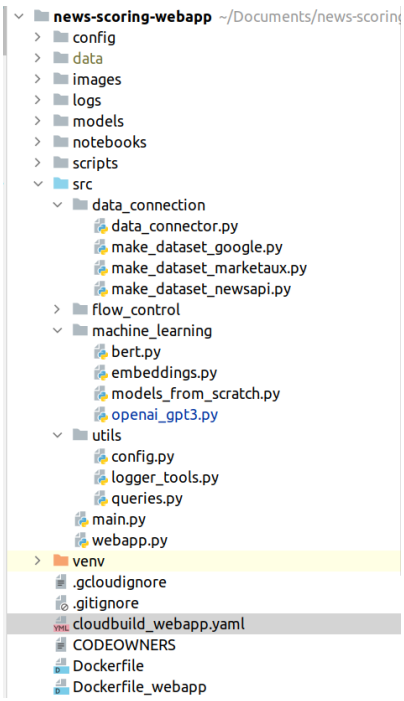

Figura 22: Estructura de código del proyecto

Capítulo 5

Pruebas y resultados

5.1. Introducción

En esta sección se detallan las pruebas que se han realizado relacionadas con el entrenamiento de los diferentes modelos que se han comparado durante el proyecto. Además, se muestran las comparaciones de los diferentes resultados obtenidos al aplicar diferentes algoritmos de *embeddings* sobre el dataset inicial. Por último, se muestra también una comparación de la aplicación de los modelos FinancialBert y GPT 3.5 Turbo sobre el dataset específico. De esta forma se podrá tener una idea de la precisión de las clasificaciones de ambos modelos sobre este dataset, y se podrá observar cómo de subjetivo puede llegar a ser la percepción de sentimientos en noticias.

5.2. Resumen de resultados

A continuación se muestra una tabla con el resumen de los resultados obtenidos para cada uno de los modelos. En la tabla se muestra el accuracy global que aporta cada uno de los modelos según los diferentes *embeddings*, tanto en entrenamiento con el dataset Financial PhraseBank como en validación con el dataset específico.

Como se puede observar, ninguno de los modelos ha dado buenos resultados con el dataset específico, por la forma en la que se ha introducido los datos a los modelos. En cambio, con el dataset de Financial PhraseBank se puede ver que los modelos proporcionan resultados bastante buenos con los embeddings de TF-IDF.

Modelo	Accuracy con dataset Financial PhraseBank	Accuracy con dataset específico
Naive bayes - TF-IDF	0.83	0.35
Naive bayes - Word2Vec	0.64	**0.40**
Naive bayes - GLOVE	0.53	0.25
Decission Tree - TF-IDF	0.75	0.39
Decission Tree - Word2Vec	0.70	0.36
Decission Tree - GLOVE	0.68	0.34
Gradient Boosting - TF-IDF	**0.89**	0.35
Gradient Boosting - Word2Vec	0.71	0.36
Gradient Boosting - GLOVE	0.67	**0.41**
Random Forest - TF-IDF	**0.86**	0.38
Random Forest - Word2Vec	0.71	0.36
Random Forest - GLOVE	0.70	0.36
SVM - TF-IDF	**0.90**	0.36
SVM - Word2Vec	0.78	0.36
SVM - GLOVE	0.67	0.36
RNN - TF-IDF	0.89	0.34
RNN - Word2Vec	0.78	0.31
RNN - GLOVE	0.65	0.30
LSTM - TF-IDF	0.88	0.36
LSTM - Word2Vec	0.79	0.33
LSTM - GLOVE	0.67	0.29
Financial Bert Preentrenado	-	**0.40**
GPT 3.5 Turbo Preentrenado	-	0.38

Cuadro 5.1: Comparación de resultados de todos los modelos con el dataset Financial Phrasebank y el dataset específico

5.3. Pruebas realizadas

5.3.1. Dataset Financial Phrasebank con modelos desde cero

En primer lugar, se llevan a cabo las pruebas correspondientes a los modelos creados desde cero. Una vez se han determinado los hiperparámetros más adecuados para cada modelo, se procede a entrenarlos utilizando el 80 % del conjunto de datos del Financial PhraseBank. Posteriormente, se realiza la validación de dichos modelos utilizando el 20 % restante del conjunto de datos.

Después del entrenamiento, se guarda cada modelo en un archivo de extensión ".pickle" con el fin de validarlos utilizando los diferentes conjuntos de datos y poder utilizarlos sin necesidad de volver a entrenarlos previamente. Para llevar a cabo la validación de los modelos, se carga cada uno de ellos en un script de Python y se realiza una predicción sobre el conjunto de validación reservado para esta labor, para obtener las clases que el modelo predeciría. Al tratarse de un conjunto de datos etiquetado, se puede comparar la predicción del modelo con las etiquetas reales, lo

que nos permite evaluar el desempeño del modelo de forma objetiva.

Con el propósito de evaluar el rendimiento de cada uno de los modelos, se han generado visualizaciones de las matrices de confusión y las curvas ROC para cada una de las predicciones realizadas. Asimismo, con el fin de obtener métricas numéricas para facilitar la comparación entre los modelos, se ha generado un Classification Report para cada uno de ellos. Este informe presenta las principales métricas de los modelos, como *accuracy*, *precission*, *recall*, *f1-score* y *support*, desglosados por cada una de las clases presentes en el conjunto de datos. Además, proporciona las métricas generales del modelo. En el Anexo A de este documento se incluyen todas las gráficas e informes correspondientes al entrenamiento de cada uno de los modelos utilizando los tres tipos de *embeddings* generados.

Los resultados obtenidos muestran que el tipo de *embedding* que ha demostrado un mejor desempeño es el de TF-IDF. Esto se debe principalmente a los volúmenes de datos con los que se ha trabajado, puesto que Word2Vec y Glove requieren de un corpus extenso para que puedan capturar todas las características relevantes presentes en un texto, mientras que TF-IDF al ser probabilístico no requiere de contexto sino que se basa en frecuencias que haya presentes.

Adicionalmente, se observa que el modelo que presenta los mejores resultados es el SVM con TF-IDF, alcanzando una precisión del 90 % y un AUC cercano a 1 en las tres curvas ROC.

El modelo SVM con TF-IDF ha demostrado un rendimiento destacado con este tipo de *embeddings* debido a su eficiencia en espacios de alta dimensionalidad. Al generar los *embeddings* de TF-IDF a partir del vocabulario completo, cada vector tiene un tamaño igual al tamaño del vocabulario, que en este caso consta de 1415 dimensiones. Además, este modelo exhibe una buena capacidad de generalización, lo que le permite clasificar correctamente noticias nuevas con las que no ha sido previamente entrenado. Dado que el dataset presenta un claro desbalanceo entre las clases, la capacidad de generalización resulta crucial para que el modelo sea capaz de capturar patrones en los datos, incluso en las clases minoritarias, y así realizar predicciones más precisas.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Negative	0.79	0.88	0.83	91
Neutral	0.92	0.97	0.95	418
Positive	0.89	0.73	0.80	171
Accuracy			0.90	680
Macro Avg	0.87	0.86	0.86	680
Weighted Avg	0.90	0.90	0.89	680

Cuadro 5.2: Classification Report del modelo SVM con TF-IDF

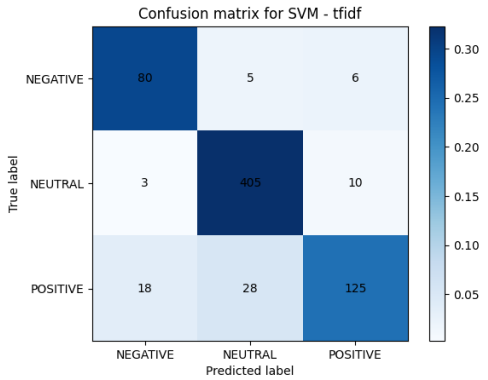

Figura 23: Matriz de confusión del modelo SVM con TF-IDF

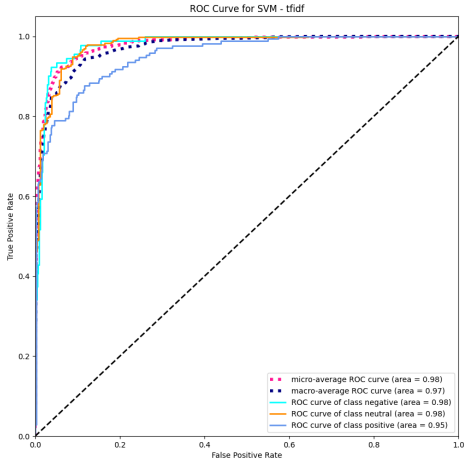

Figura 24: Curva ROC del modelo SVM con TF-IDF

Además del modelo SVM con TF-IDF, otros modelos también han demostrado un rendimiento notable en este estudio. Entre ellos se encuentran Gradient Boosting, Random Forest y las redes neuronales, tanto RNN como LSTM. Todos estos modelos fueron entrenados utilizando el conjunto de datos de embeddings generados mediante TF-IDF. Sus resultados se pueden observar en el Anexo A del documento.

Sin embargo, modelos con resultados peores han sido, en general, todos los modelos entrenados con los *embeddings* de GLOVE.

Una prueba adicional que se puede llevar a cabo consiste en evaluar los resultados de los modelos en función de los diferentes tipos de *embeddings* utilizados durante el entrenamiento. En este sentido, se pueden distinguir tres escenarios distintos.

- En primer lugar, al emplear los *embeddings* de TF-IDF, se ha demostrado que en términos generales todos los modelos han exhibido un desempeño y una precisión bastante equilibrados, logrando resultados mayormente satisfactorios. No obstante, es importante destacar que el modelo SVM continúa siendo el que ha arrojado las métricas más destacadas, seguido de cerca por la red neuronal LSTM. Por otro lado, el modelo de Árbol de Decisión simple ha mostrado los resultados menos favorables en comparación.

- En segundo lugar, al emplear los *embeddings* de Word2Vec, se han obtenido resultados menos favorables en comparación con los obtenidos mediante TF-IDF. En general, la mayoría de los modelos presentan dificultades para distinguir entre noticias positivas y negativas, lo cual puede estar relacionado con el proceso de generación de los *embeddings*. Aunque se evitó eliminar las Stopwords con el fin de preservar el significado y la polaridad de las oraciones, es posible que el modelo Word2Vec, entrenado en un corpus relativamente pequeño y con una escasez de noticias negativas en comparación con las neutrales y positivas, no cuente con suficiente información para detectar características negativas en el texto. En consecuencia, los modelos tienden a clasificar las

noticias como positivas por defecto, ya que fueron entrenados en un conjunto de datos que predominaba por noticias neutrales, y el modelo es capaz de identificar claramente este tipo de noticias.

- Por último, al aplicar los *embeddings* de GLOVE, se han obtenido resultados notablemente inferiores en comparación con los otros dos algoritmos mencionados. Similar a la observación realizada con Word2Vec, se ha identificado una tendencia en los modelos a clasificar las noticias como positivas o neutrales, pero con una menor capacidad para identificar noticias negativas.

5.3.2. Dataset específico con todos los modelos

Estas pruebas están relacionadas con la validación de todos los modelos utilizados en el estudio, tanto aquellos entrenados desde cero como los modelos preentrenados, utilizando el dataset específico. Sin embargo, es importante destacar que los resultados obtenidos al aplicar estos modelos sobre el dataset específico no han sido satisfactorios para ninguno de los modelos utilizados. En el Anexo B de este documento se incluyen todas las gráficas e informes correspondientes a la validación de cada uno de los modelos utilizando los tres tipos de *embeddings* generados con este dataset.

La principal diferencia entre estas pruebas y las anteriores, donde se validaron los modelos utilizando el dataset Financial Phrasebank, radica en la longitud de los textos utilizados para la validación. Mientras que el dataset Financial Phrasebank consiste en oraciones financieras de tamaño moderado, en este caso se ha realizado la validación utilizando artículos completos.

- En el caso de los modelos preentrenados, se ha enfrentado el desafío de la limitación de tamaño para los datos de entrada. En algunas ocasiones, ha sido necesario dividir el texto de entrada en fragmentos más pequeños y posteriormente ponderar los sentimientos predichos para cada fragmento. Esta aproximación ha permitido abordar la limitación de tamaño y obtener una visión más completa de los sentimientos en el texto completo.

- En el caso de los modelos entrenados desde cero, se han realizado pruebas utilizando tanto el artículo completo como el resumen de la noticia, disponible en la misma tabla en la que se almacena el texto, como datos de entrada. Desafortunadamente, en ambos casos se han obtenido resultados insatisfactorios para todos los tipos de *embeddings*. Como se mencionó anteriormente, es posible que la forma en que se generaron los *embeddings* haya tenido un impacto negativo en la capacidad de los modelos para generalizar de manera efectiva en este contexto particular.

El proceso de validación de los modelos sobre artículos completos ha tenido un impacto significativo en los resultados obtenidos. Al analizar los artículos que los modelos han clasificado incorrectamente, se ha observado una mayor confusión entre los artículos con sentimientos negativos y positivos. Esta confusión se debe a la presencia de oraciones con diferentes polaridades dentro de un mismo artículo. Dado que los modelos han sido entrenados utilizando oraciones con una sola polaridad, esta situación de ambigüedad en los artículos ha llevado a que los modelos se

equivoquen al clasificarlos. En esencia, es como lanzar una moneda al aire y, como resultado, ninguno de los modelos ha logrado ofrecer un rendimiento satisfactorio.

Con el objetivo de mejorar los resultados de los modelos preentrenados, se ha explorado la posibilidad de calcular una métrica de precisión ponderada teniendo en cuenta el *score* proporcionado en sus predicciones. Sin embargo, esta estrategia no ha logrado mejorar los resultados, ya que el *score* devuelto por los modelos tiende a ser casi siempre cercano al 100 %. Este *score* refleja la confianza que tienen los modelos en las respuestas que están proporcionando. A pesar de considerar este factor de confianza en el cálculo de las métricas, no se ha observado una mejora sustancial en los resultados. Esto sugiere que la alta confianza expresada por los modelos puede no ser un indicador confiable de la precisión real de sus predicciones. Es necesario explorar otras estrategias y métricas para evaluar y mejorar el rendimiento de los modelos preentrenados.

A continuación se muestran las matrices de confusión resultantes de validar los modelos preentrenados de Financial Bert y GPT 3.5 Turbo sobre el dataset específico, así como sus Classification Reports correspondientes.

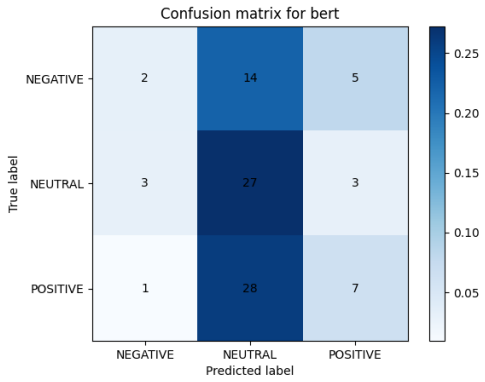

Figura 25: Matriz de confusión del modelo FinancialBert

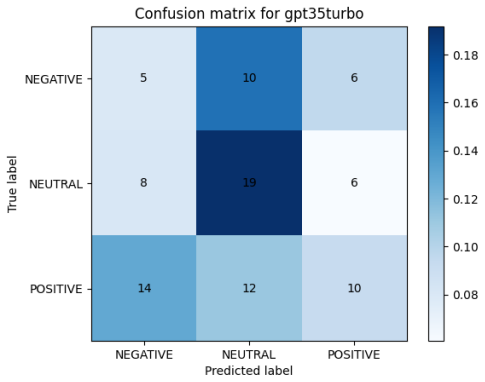

Figura 26: Curva ROC del modelo GPT 3.5 Turbo

Class	Precision	Recall	F1-Score	Support
Negative	0.33	0.10	0.15	21
Neutral	0.39	0.82	0.53	33
Positive	0.47	0.19	0.27	36
Accuracy			0.40	90
Macro Avg	0.40	0.37	0.32	90
Weighted Avg	0.41	0.40	0.34	90

Cuadro 5.3: Classification Report del modelo FinancialBert sobre el dataset específico

Class	Precision	Recall	F1-Score	Support
Negative	0.19	0.24	0.21	21
Neutral	0.46	0.58	0.51	33
Positive	0.45	0.28	0.34	36
Accuracy			0.38	90
Macro Avg	0.37	0.36	0.36	90
Weighted Avg	0.39	0.38	0.37	90

Cuadro 5.4: Classification Report del modelo GPT 3.5 Turbo sobre el dataset específico

Como se puede ver, ninguno de los modelos preentrenados aporta buenos resultados.

A continuación, se incluye un ejemplo de las oraciones que falla GPT turbo, las 14 noticias positivas que puntúa como negativas son un ejemplo claro del problema descrito anteriormente:

> Jefferies analyst Casey Haire highlighted NYCB's strong balance sheet positioning, thanks to the significant improvement in its liquidity profile resulting from the pending acquisition of Signature Bank assets. This FDIC-brokered acquisition is expected to provide NYCB with increased scale and diversification, which is seen as a positive development.
> While Jefferies holds an optimistic view overall, some investors remain cautious due to the current valuation discount compared to its peers. Although NYCB is expected to close the valuation gap in the future, this discount remains a concern.
> Additionally, while an attractive dividend yield of approximately 8 % is mentioned, some analysts also emphasize the need to monitor the long-term sustainability of this dividend payment level.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Introducción

En esta última sección se detallan las conclusiones obtenidas tras el desarrollo del proyecto y se indican una serie de próximos pasos que se podrían seguir de cara a mejorar y continuar con el desarrollo del proyecto.

6.2. Conclusiones

- La **elección del algoritmo de generación de *embeddings*** tiene un impacto significativo en los resultados obtenidos al entrenar los modelos de clasificación. Es importante considerar este aspecto al seleccionar el mejor algoritmo, ya que puede influir en los resultados deseados o en las características en las que se espera que el modelo se enfoque.

- Las **redes neuronales** son herramientas útiles para capturar y almacenar el contexto de las oraciones, y su capacidad de aprendizaje recurrente las hace adecuadas para tareas que involucran secuencias de texto. Sin embargo, al entrenar redes neuronales con conjuntos de datos pequeños y desbalanceados, no siempre se obtienen resultados satisfactorios.

  En el caso de **conjuntos de datos pequeños**, las redes neuronales encuentran **dificultades** para generalizar adecuadamente y capturar la diversidad de patrones presentes en los datos. Esto se debe a que la cantidad limitada de ejemplos de entrenamiento puede no ser suficiente para aprender las complejidades y variaciones presentes en el lenguaje.

- Los **modelos preentrenados** pueden proporcionar resultados prometedores, sin embargo, al validar estos modelos con un conjunto de datos específico, se evidencia que la **interpretación de los sentimientos** puede ser altamente **subjetiva** y depender de la persona que esté leyendo la noticia. Por lo tanto, al realizar la validación y comparación de los sentimientos generados por FinancialBert/GPT 3.5 Turbo con el equipo de riesgos, se observa que se presenten discrepancias significativas.

- Podría haber sido beneficioso calcular la media de las puntuaciones de las noticias de manera diferente en el conjunto de datos específico, o incluso dividirlo

en varios conjuntos de datos similares al caso de Financial PhraseBank. Sin embargo, debido a la **limitada cantidad de noticias puntuadas** disponibles, no fue viable llevar a cabo dicha separación.

- Se ha observado que la adaptación de los modelos a noticias más largas y complejas presenta dificultades debido a varias razones. En primer lugar, una noticia puede contener **múltiples sentimientos** dependiendo de la oración en la que se encuentre, lo que conlleva que al etiquetar la noticia en su totalidad, algunas oraciones puedan estar incorrectamente etiquetadas. En segundo lugar, siempre existe la posibilidad de **errores humanos** al momento de realizar el etiquetado. Además, dado que el tipo de problema es diferente, sería más justo evaluar el rendimiento de los modelos al **clasificar cada oración individualmente** y luego ponderar los resultados a nivel de noticia. Comparar esa ponderación final con el etiquetado realizada por el equipo de riesgos permitiría obtener una medida más precisa y equitativa del desempeño de los modelos.

6.3. Próximos pasos

- ***Embeddings* con el algoritmo de OpenAI**: Explorar la posibilidad de utilizar el algoritmo de *embeddings* de OpenAI para enriquecer la representación de los textos utilizados en el proyecto. Este enfoque puede ayudar a capturar mejor las características semánticas y contextuales de las noticias, mejorando potencialmente la precisión de los modelos de clasificación.

- **Probar otras formas de clasificación de noticias**: Ya que los modelos preentrenados han sido entrenados con oraciones más cortas y con un único sentimiento, se podría tratar de replicar este procedimiento pero con los modelos desde cero, para tratar de obtener mejores resultados.

- **Comparar con datos económicos de la empresa**: Incorporar datos económicos relevantes de la empresa en el análisis de sentimientos puede proporcionar una perspectiva adicional sobre la relación entre los eventos económicos y las opiniones expresadas en las noticias. Esto permitirá una evaluación más completa y contextualizada de los sentimientos asociados a cada empresa.

- **Agrupar las noticias por empresa mediante el algoritmo NER**: Utilizar el algoritmo de reconocimiento de entidades (NER) para identificar y agrupar las noticias según la empresa a la que se refieren. Esto facilitará el análisis individualizado de cada empresa y proporcionará información más precisa sobre las opiniones y eventos relacionados con cada una.

- **Eliminar barrera del idioma**: Investigar y desarrollar técnicas para abordar el análisis de sentimientos en noticias escritas en diferentes idiomas. Esto permitirá ampliar la aplicabilidad del proyecto a nivel global y aprovechar fuentes de información adicionales.

- **Modelos con *ensemble***: Implementar técnicas de *ensemble*, como Stacking o Bagging, utilizando los mejores modelos obtenidos individualmente. Esta

estrategia permitirá combinar las fortalezas de diferentes modelos y mejorar aún más la capacidad de predicción y generalización del sistema.

- **Usar la aplicación de etiquetado para recibir *feedback* de los modelos**: Utilizar la aplicación final como una herramienta de retroalimentación para los modelos de clasificación. Recopilar y analizar los comentarios y sugerencias de los usuarios para mejorar continuamente los modelos y adaptarlos a las necesidades y preferencias de los usuarios. Como se ha comentado, pueden existir diferentes escenarios económicos y es posible que se necesiten adaptar los modelos a cada uno de ellos para que clasifiquen las noticias de forma más optimista o pesimista según el caso. Con este **feedback** se podría adaptar el modelo final a las necesidades específicas del momento. Además, de cara a futuro, en el que se tendrá un dataset sin etiquetar, este **feedback** permitirá saber si los modelos están bien calibrados o es necesario ajustarlos.

- **Realizar un speech to text para analizar sentimientos en tiempo real**: Implementar un sistema de reconocimiento de voz (speech-to-text) para analizar en tiempo real los sentimientos expresados en las noticias financieras que los *dealers* ven cada día. Esto permitirá capturar información en tiempo real y reaccionar rápidamente a las noticias relevantes en los mercados financieros.

- **Expandir las fuentes de datos**: Utilizar otras fuentes de datos como Twitter o Bloomberg para conseguir solventar dos de los problemas encontrados. En primer lugar, para ampliar el corpus. En segundo lugar, tener la posibilidad de conseguir noticias de clientes con menor cobertura mediática.

- **Desarrollo de la aplicación final**: Desarrollar la aplicación final con una serie de funcionalidades clave. Una de ellas será la capacidad de buscar noticias por empresa, lo que permitirá a los usuarios acceder a todas las noticias disponibles sobre una empresa específica en el último año.

  Para cada noticia, se mostrará el sentimiento asignado por el modelo de análisis de sentimientos, lo que proporcionará información valiosa sobre la percepción financiera asociada a esa empresa. Además, se incluirá la importancia de la noticia, que se determinará en función de la fuente que la haya publicado. Esto ayudará a los usuarios a evaluar la relevancia y credibilidad de las noticias.

  La aplicación también incluirá enlaces directos a los artículos y mostrará la fecha de publicación de cada noticia. Además, se permitirá ordenar las noticias en función de su fecha de publicación, que por defecto serán mostradas en orden descendente desde las más recientes hasta las más antiguas.

  Una característica adicional de la aplicación será la capacidad de realizar una ponderación de los sentimientos de las noticias del último mes para asignar una puntuación de sentimiento global a la empresa. Esto permitirá a los usuarios obtener una visión general del sentimiento predominante hacia la empresa durante ese período específico.

  Adicionalmente, se implementará una gráfica temporal que mostrará la variación de los sentimientos de las noticias de la empresa a lo largo del último mes. Esta representación visual permitirá a los usuarios visualizar y comprender de manera intuitiva cómo ha evolucionado el sentimiento financiero de las noticias relacionadas con la empresa a lo largo del tiempo.

A continuación se incluye un boceto del diseño de la aplicación final que se pretende implementar.

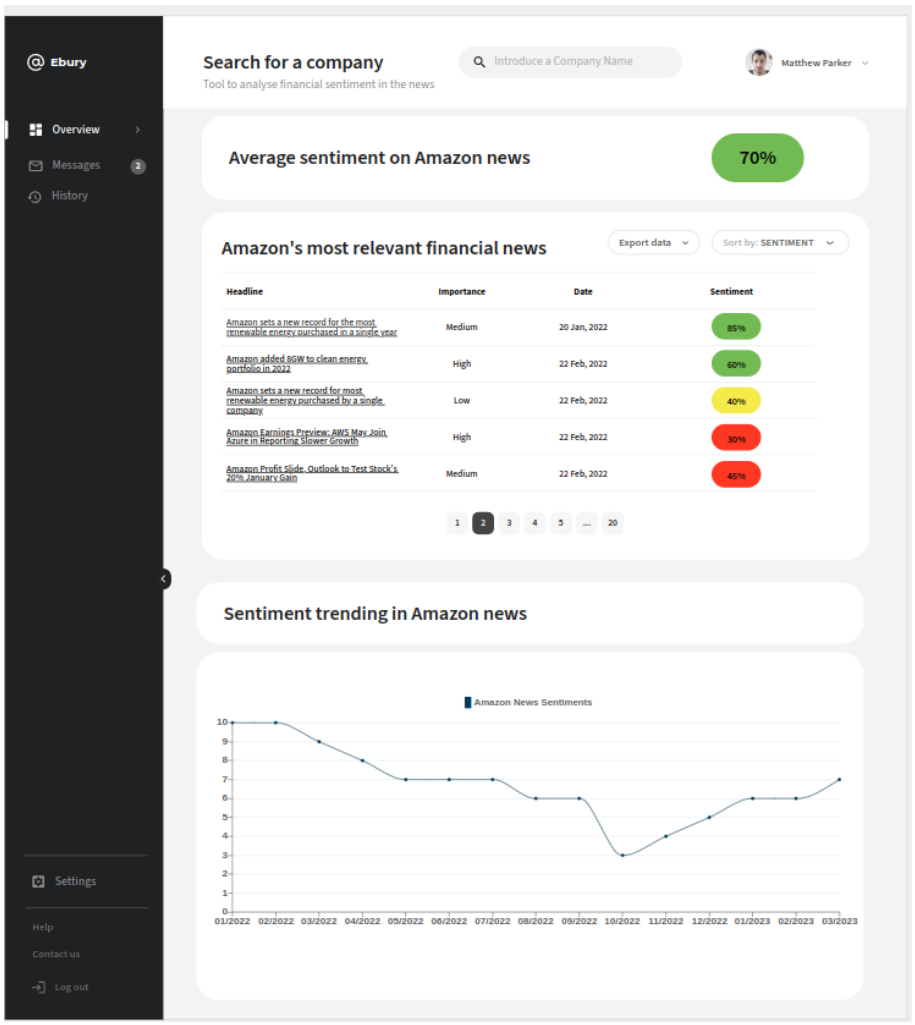

Figura 27: Boceto de visualización de la aplicación final

Apéndice A

Resultados del entrenamiento de los modelos

En este anexo se muestran los *classification reports*, las matrices de confusión y las curvas ROC resultantes de todos los modelos probados.

La organización de esta sección se divide según los diferentes modelos que se han probado. Dentro de cada uno de los modelos se muestran los resultados obtenidos con cada uno de los *embeddings*.

- El Classification Report contiene las principales métricas de clasificación de cada uno de los modelos. Estos informes han sido obtenidos de la validación de cada uno de los modelos, realizada con el 20 % de datos reservados para esta labor.

- Las matrices de confusión muestran el rendimiento de cada uno de los modelos de clasificación empleados en el análisis. Permite cuantificar los aciertos y errores del modelo para cada una de las clases.

- Las curvas ROC representan la capacidad de discriminación de un modelo en cada clase individual frente al resto de las clases.

A.1. Rendimiento del modelo Naive Bayes

A.1.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.67	0.64	0.65	91
Neutral	0.92	0.91	0.91	418
Positive	0.72	0.74	0.73	171
Accuracy			0.83	680
Macro Avg	0.77	0.76	0.77	680
Weighted Avg	0.83	0.83	0.83	680

Cuadro A.1: Classification Report del modelo Naive Bayes con TF IDF

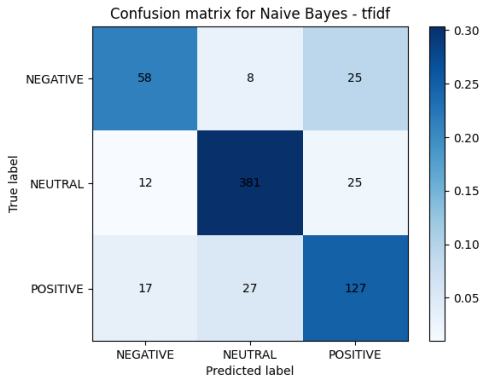

Figura 28: Matriz de confusión del modelo Naive Bayes con TF-IDF

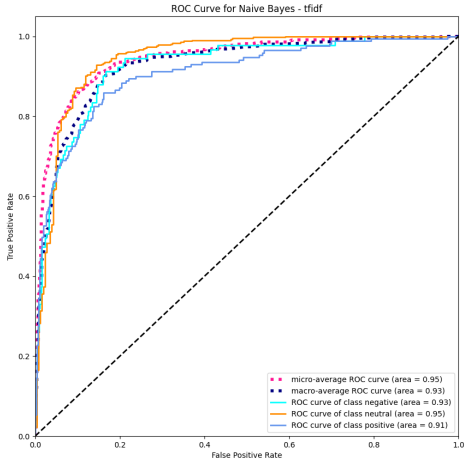

Figura 29: Curva ROC del modelo Naive Bayes con TF-IDF

A.1.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.40	0.40	0.40	91
Neutral	0.79	0.80	0.80	418
Positive	0.37	0.36	0.36	171
Accuracy			0.64	680
Macro Avg	0.52	0.52	0.52	680
Weighted Avg	0.63	0.64	0.63	680

Cuadro A.2: Classification Report del modelo Naive Bayes con Word2Vec

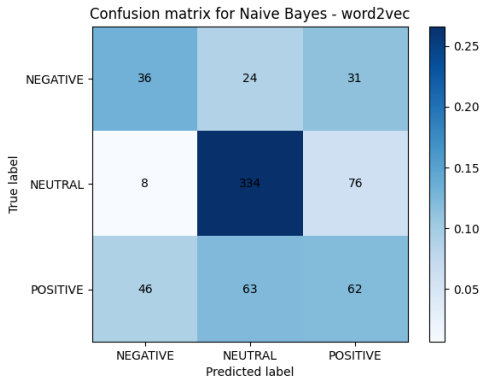

Figura 30: Matriz de confusión del modelo Naive Bayes con Word2Vec

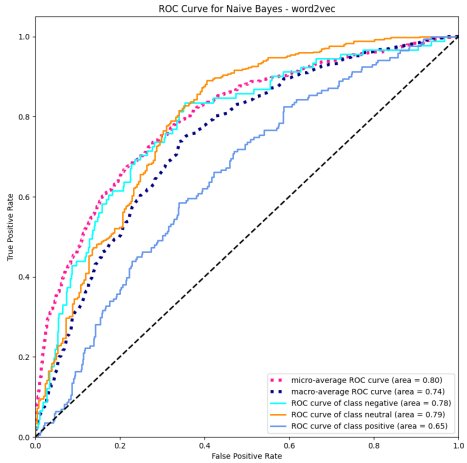

Figura 31: Curva ROC del modelo Naive Bayes con Word2Vec

A.1.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	0.26	0.56	0.36	91
Neutral	0.76	0.68	0.72	418
Positive	0.22	0.14	0.17	171
Accuracy			0.53	680
Macro Avg	0.41	0.46	0.42	680
Weighted Avg	0.56	0.53	0.53	680

Cuadro A.3: Classification Report del modelo Naive Bayes con GLOVE

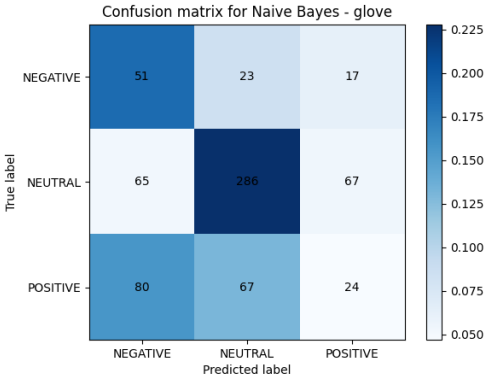

Figura 32: Matriz de confusión del modelo Naive Bayes con GLOVE

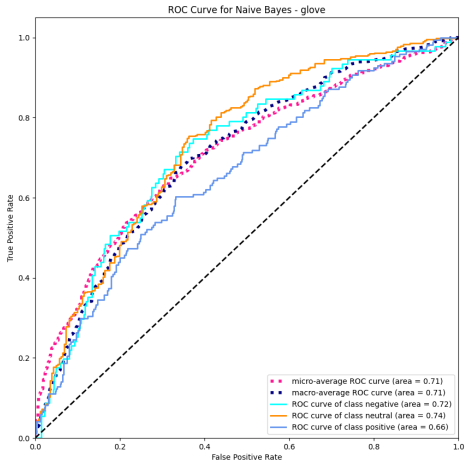

Figura 33: Curva ROC del modelo Naive Bayes con GLOVE

A.2. Rendimiento del modelo Decission Tree

A.2.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.51	0.58	0.55	91
Neutral	0.86	0.88	0.87	418
Positive	0.61	0.54	0.58	171
Accuracy			0.75	680
Macro Avg	0.66	0.67	0.66	680
Weighted Avg	0.75	0.75	0.75	680

Cuadro A.4: Classification Report del modelo Decission Tree con TF-IDF

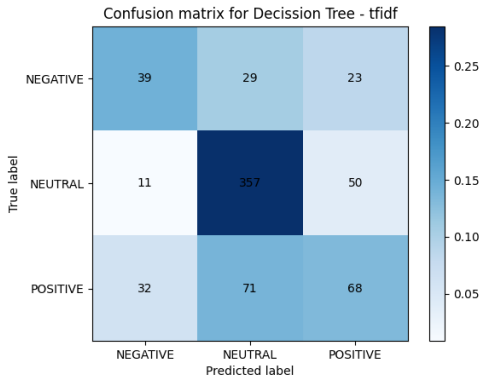

Figura 34: Matriz de confusión del modelo Decission Tree con TF-IDF

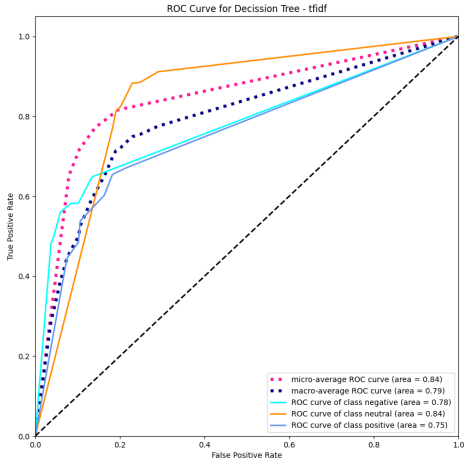

Figura 35: Curva ROC del modelo Decission Tree con TF-IDF

A.2.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.32	0.13	0.19	91
Neutral	0.82	0.89	0.85	418
Positive	0.49	0.55	0.52	171
Accuracy			0.70	680
Macro Avg	0.54	0.52	0.52	680
Weighted Avg	0.67	0.70	0.68	680

Cuadro A.5: Classification Report del modelo Decission Tree con Word2Vec

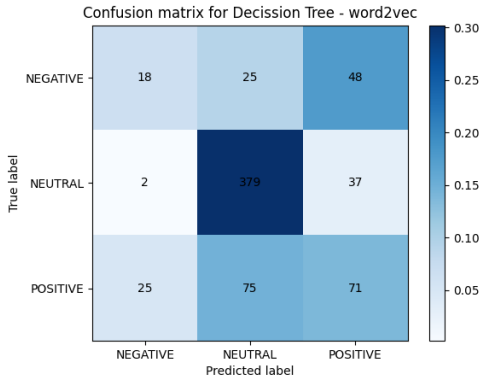

Figura 36: Matriz de confusión del modelo Decission Tree con Word2Vec

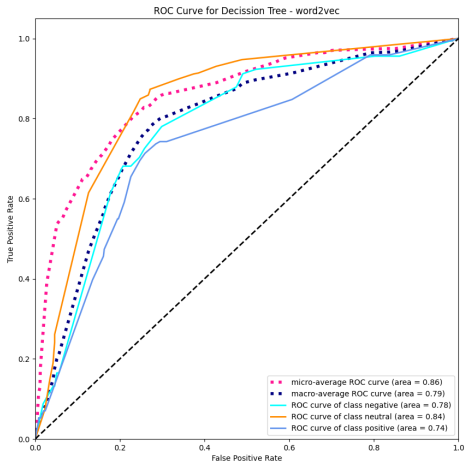

Figura 37: Curva ROC del modelo Decission Tree con Word2Vec

A.2.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	0.44	0.09	0.15	91
Neutral	0.70	0.97	0.82	418
Positive	0.55	0.27	0.36	171
Accuracy			0.68	680
Macro Avg	0.57	0.44	0.44	680
Weighted Avg	0.63	0.68	0.61	680

Cuadro A.6: Classification Report del modelo Decission Tree con GLOVE

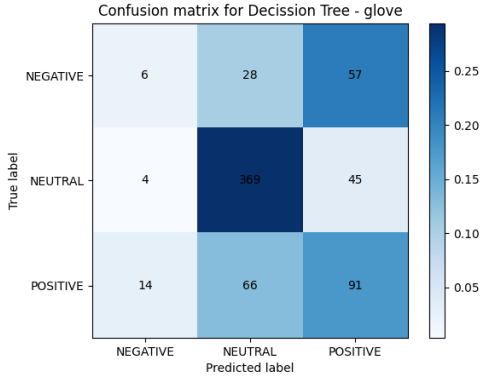

Figura 38: Matriz de confusión del modelo Decission Tree con GLOVE

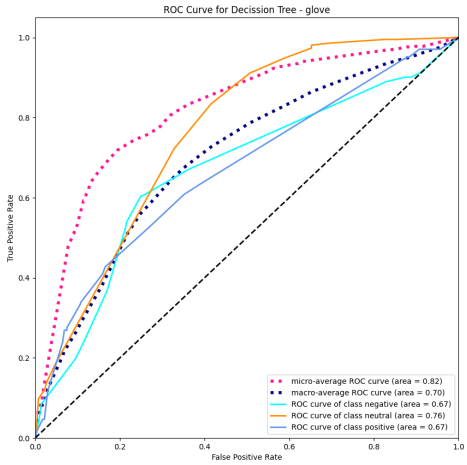

Figura 39: Curva ROC del modelo Decission Tree con GLOVE

A.3. Rendimiento del modelo Gradient Boosting

A.3.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.85	0.84	0.84	91
Neutral	0.90	0.97	0.93	418
Positive	0.90	0.74	0.81	171
Accuracy			0.89	680
Macro Avg	0.88	0.85	0.86	680
Weighted Avg	0.89	0.89	0.89	680

Cuadro A.7: Classification Report del modelo Gradient Boosting con TF IDF

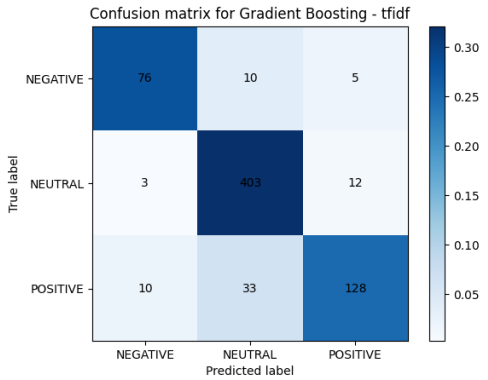

Figura 40: Matriz de confusión del modelo Gradient Boosting con TF-IDF

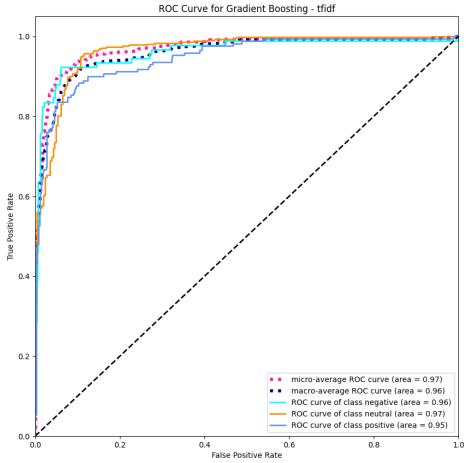

Figura 41: Curva ROC del modelo Gradient Boosting con TF-IDF

A.3.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.38	0.24	0.30	91
Neutral	0.82	0.89	0.85	418
Positive	0.53	0.52	0.52	171
Accuracy			0.71	680
Macro Avg	0.57	0.55	0.56	680
Weighted Avg	0.69	0.71	0.69	680

Cuadro A.8: Classification Report del modelo Gradient Boosting con Word2Vec

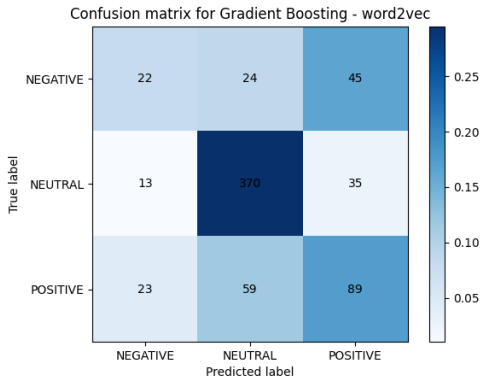

Figura 42: Matriz de confusión del modelo Gradient Boosting con Word2Vec

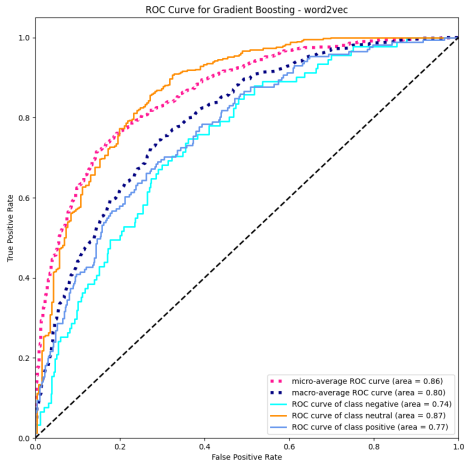

Figura 43: Curva ROC del modelo Gradient Boosting con Word2Vec

A.3.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	0.40	0.21	0.27	91
Neutral	0.75	0.88	0.81	418
Positive	0.49	0.39	0.44	171
Accuracy			0.67	680
Macro Avg	0.54	0.49	0.51	680
Weighted Avg	0.63	0.67	0.64	680

Cuadro A.9: Classification Report del modelo Gradient Boosting con GLOVE

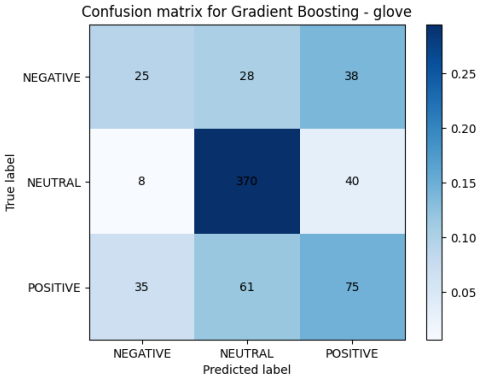

Figura 44: Matriz de confusión del modelo Gradient Boosting con GLOVE

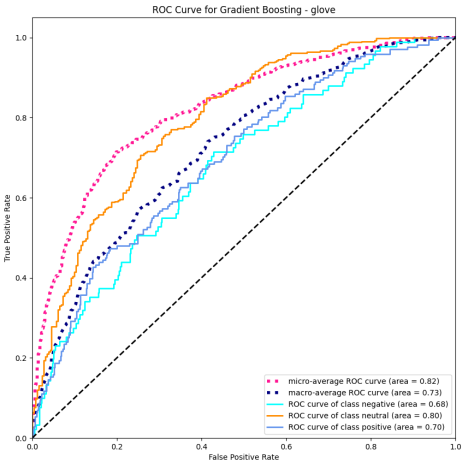

Figura 45: Curva ROC del modelo Gradient Boosting con GLOVE

A.4. Rendimiento del modelo Random Forest

A.4.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.91	0.56	0.69	91
Neutral	0.86	1.00	0.93	418
Positive	0.82	0.68	0.75	171
Accuracy			0.86	680
Macro Avg	0.86	0.75	0.79	680
Weighted Avg	0.86	0.86	0.85	680

Cuadro A.10: Classification Report del modelo Random Forest con TF IDF

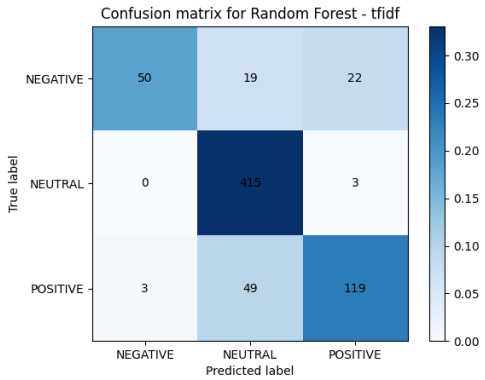

Figura 46: Matriz de confusión del modelo Random Forest con TF-IDF

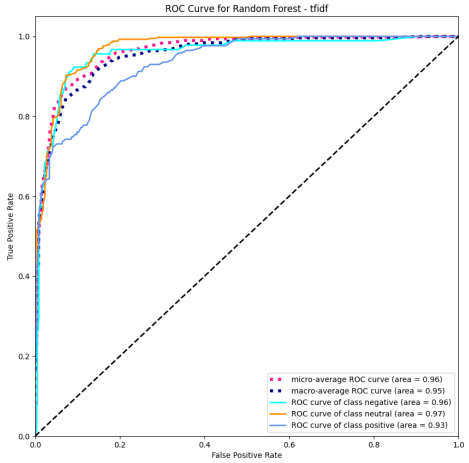

Figura 47: Curva ROC del modelo Random Forest con TF-IDF

A.4.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.37	0.21	0.27	91
Neutral	0.81	0.91	0.86	418
Positive	0.54	0.49	0.51	171
Accuracy			0.71	680
Macro Avg	0.57	0.54	0.55	680
Weighted Avg	0.68	0.71	0.69	680

Cuadro A.11: Classification Report del modelo Random Forest con Word2Vec

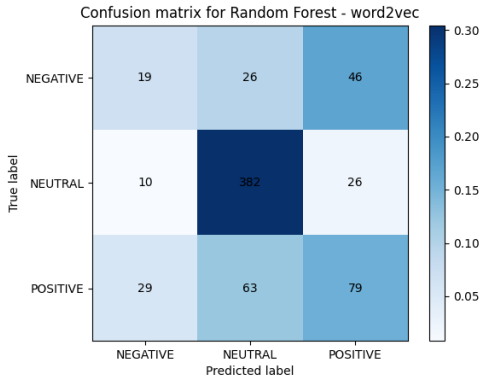

Figura 48: Matriz de confusión del modelo Random Forest con Word2Vec

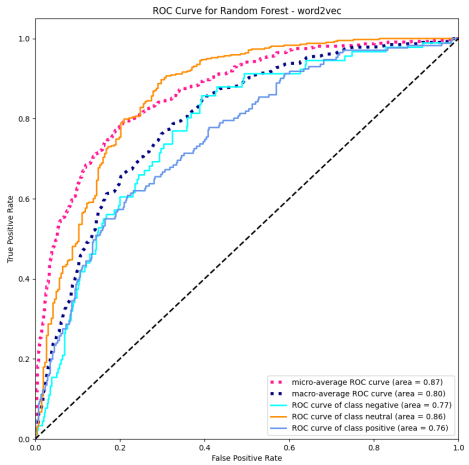

Figura 49: Curva ROC del modelo Random Forest con Word2Vec

A.4.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	0.44	0.12	0.19	91
Neutral	0.75	0.95	0.84	418
Positive	0.52	0.38	0.44	171
Accuracy			0.70	680
Macro Avg	0.57	0.49	0.49	680
Weighted Avg	0.65	0.70	0.65	680

Cuadro A.12: Classification Report del modelo Random Forest con GLOVE

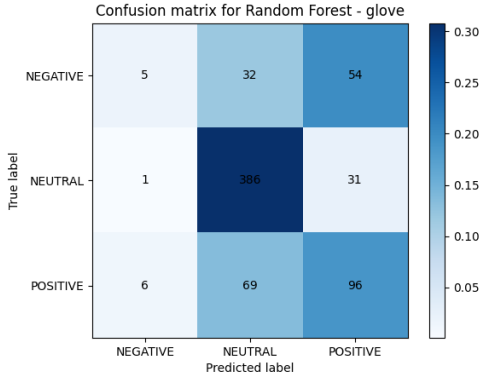

Figura 50: Matriz de confusión del modelo Random Forest con GLOVE

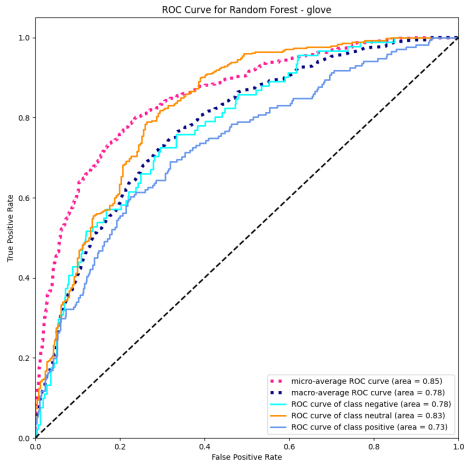

Figura 51: Curva ROC del modelo Random Forest con GLOVE

A.5. Rendimiento del modelo SVM

A.5.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.79	0.88	0.83	91
Neutral	0.92	0.97	0.95	418
Positive	0.89	0.73	0.80	171
Accuracy			0.90	680
Macro Avg	0.87	0.86	0.86	680
Weighted Avg	0.90	0.90	0.89	680

Cuadro A.13: Classification Report del modelo SVM con TF-IDF

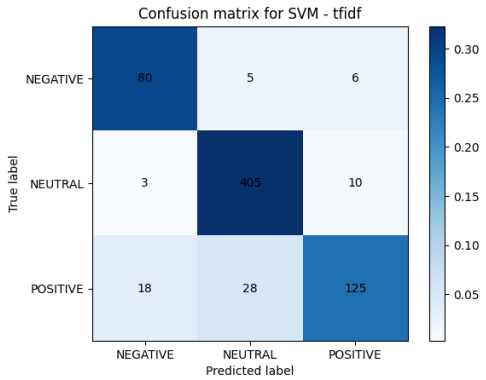

Figura 52: Matriz de confusión del modelo SVM con TF-IDF

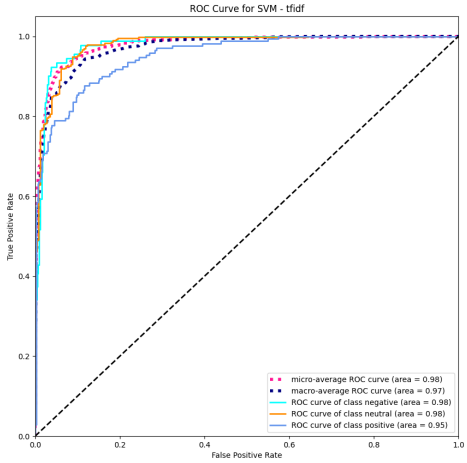

Figura 53: Curva ROC del modelo SVM con TF-IDF

A.5.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.78	0.27	0.41	91
Neutral	0.83	0.95	0.89	418
Positive	0.63	0.61	0.62	171
Accuracy			0.78	680
Macro Avg	0.75	0.61	0.64	680
Weighted Avg	0.77	0.78	0.76	680

Cuadro A.14: Classification Report del modelo SVM con Word2Vec

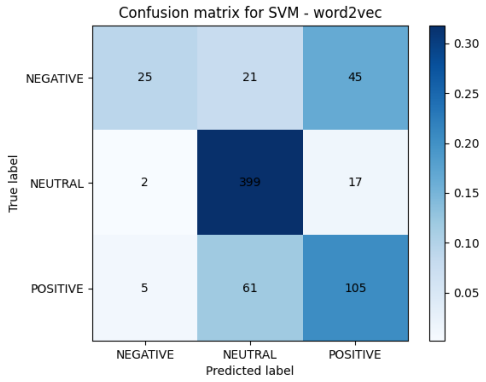

Figura 54: Matriz de confusión del modelo SVM con Word2Vec

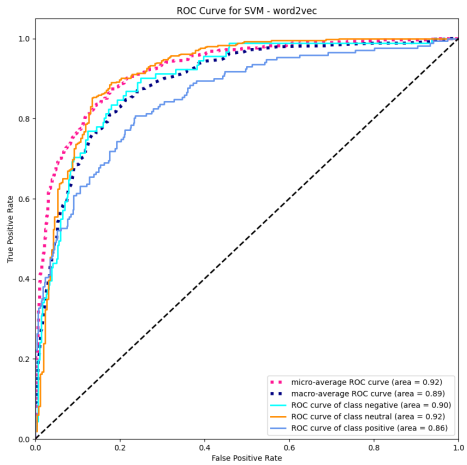

Figura 55: Curva ROC del modelo SVM con Word2Vec

A.5.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	91
Neutral	0.71	0.96	0.82	418
Positive	0.50	0.32	0.39	171
Accuracy			0.67	680
Macro Avg	0.40	0.43	0.40	680
Weighted Avg	0.56	0.67	0.60	680

Cuadro A.15: Classification Report del modelo SVM con GLOVE

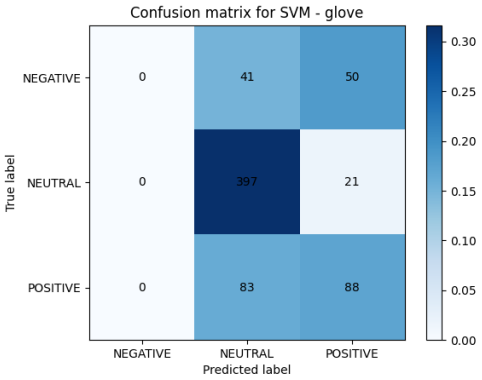

Figura 56: Matriz de confusión del modelo SVM con GLOVE

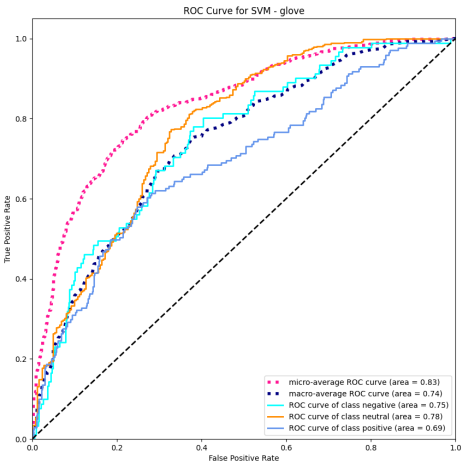

Figura 57: Curva ROC del modelo SVM con GLOVE

A.6. Matrices de confusión y curvas de pérdida del modelo RNN

A.6.1. Empleando embeddings del algoritmo TF-IDF

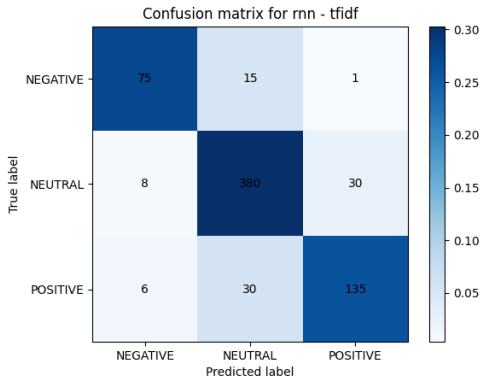

Figura 58: Matriz de confusión del modelo RNN con TF-IDF

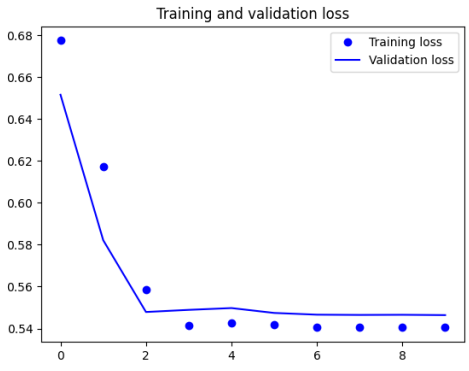

Figura 59: Curva de *loss* del modelo RNN con TF-IDF

A.6.2. Empleando embeddings del algoritmo Word2Vec

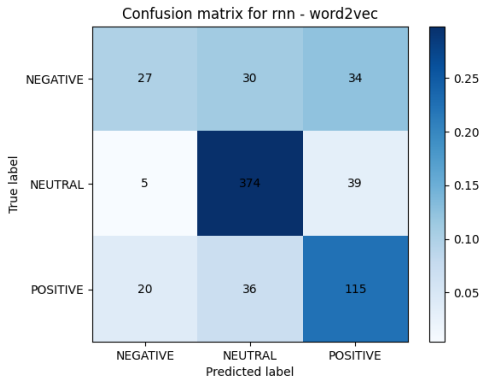

Figura 60: Matriz de confusión del modelo RNN con Word2Vec

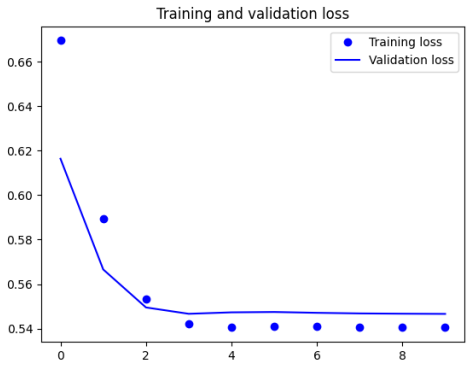

Figura 61: Curva de *loss* del modelo RNN con Word2Vec

A.6.3. Empleando embeddings del algoritmo GLOVE

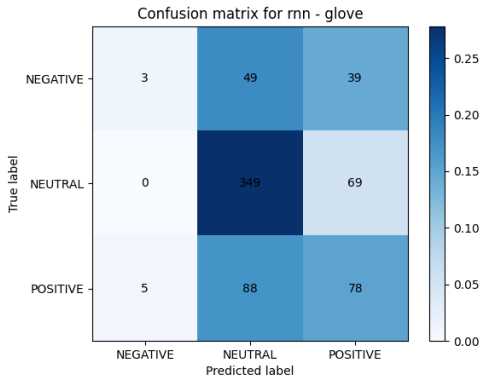

Figura 62: Matriz de confusión del modelo RNN con GLOVE

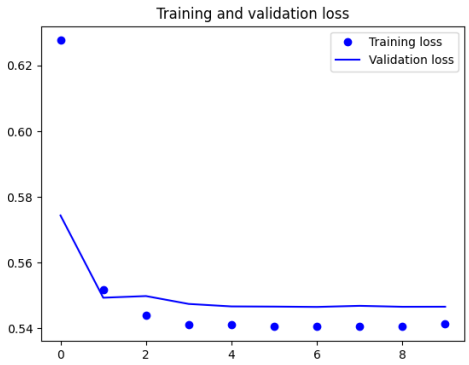

Figura 63: Curva de *loss* del modelo RNN con GLOVE

A.7. Matrices de confusión y curvas de pérdida del modelo LSTM

A.7.1. Empleando embeddings del algoritmo TF-IDF

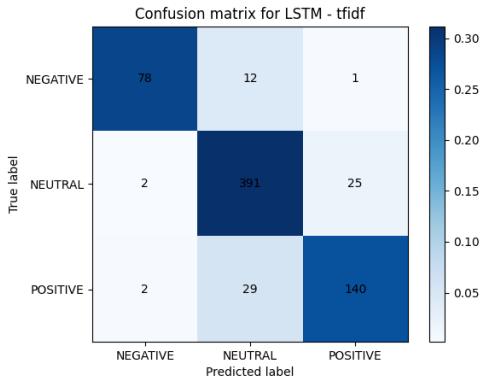

Figura 64: Matriz de confusión del modelo LSTM con TF-IDF

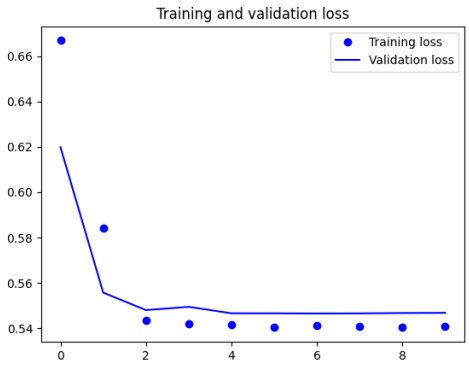

Figura 65: Curva de *loss* del modelo LSTM con TF-IDF

A.7.2. Empleando embeddings del algoritmo Word2Vec

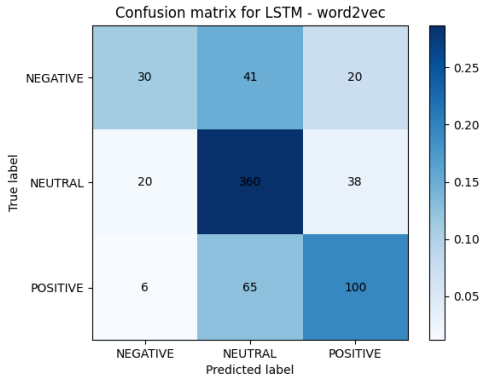

Figura 66: Matriz de confusión del modelo LSTM con Word2Vec

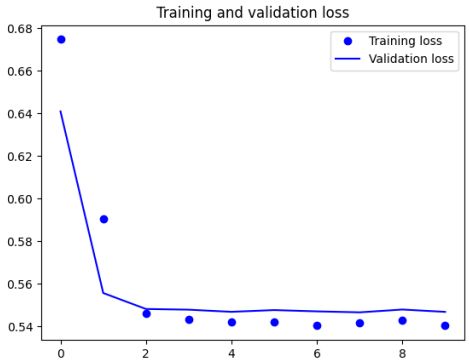

Figura 67: Curva de *loss* del modelo LSTM con Word2Vec

A.7.3. Empleando embeddings del algoritmo GLOVE

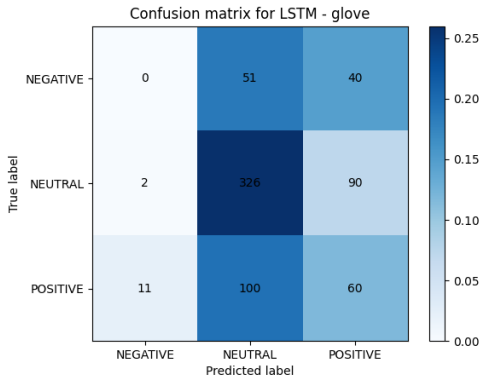

Figura 68: Matriz de confusión del modelo LSTM con GLOVE

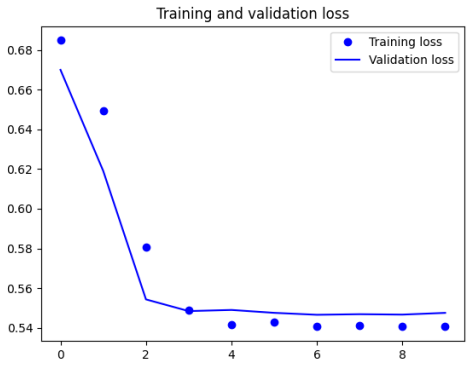

Figura 69: Curva de *loss* del modelo LSTM con GLOVE

Apéndice B

Resultados de la validación de los modelos

En este anexo se incluyen los classification reports y las métricas de cada uno de los modelos entrenados en el proyecto.

Al igual que en el anexo anterior La organización de esta sección se divide según los diferentes modelos que se han probado. Dentro de cada uno de los modelos se muestran los resultados obtenidos con cada uno de los embeddings.

B.1. Rendimiento del modelo Naive Bayes

B.1.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.36	0.94	0.52	34
Positive	0.00	0.00	0.00	37
Accuracy			0.35	92
Macro Avg	0.12	0.31	0.17	92
Weighted Avg	0.13	0.35	0.19	92

Cuadro B.1: Classification Report del modelo Naive Bayes con TF-IDF

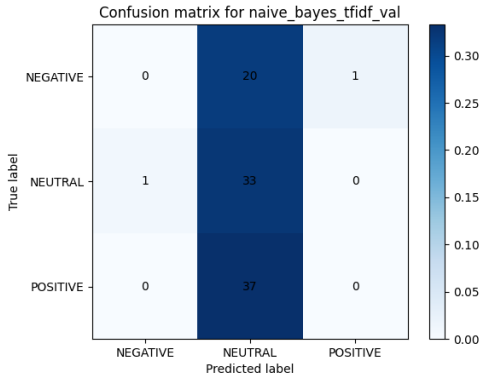

Figura 70: Matriz de confusión del modelo Naive Bayes con TF-IDF

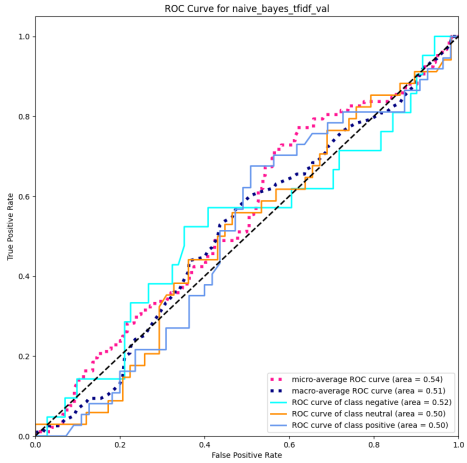

Figura 71: Curva ROC del modelo Naive Bayes con TF-IDF

B.1.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.00	0.00	0.00	31
Positive	0.40	1.00	0.57	35
Accuracy			0.40	87
Macro Avg	0.13	0.33	0.19	87
Weighted Avg	0.16	0.40	0.23	87

Cuadro B.2: Classification Report del modelo Naive Bayes con Word2Vec

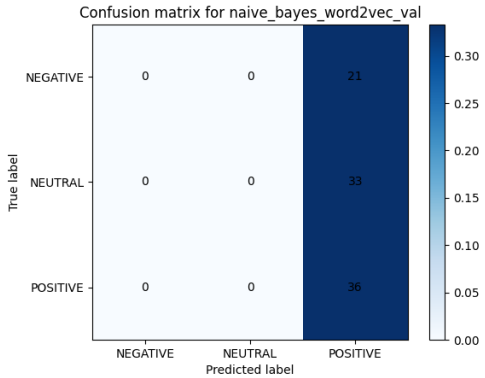

Figura 72: Matriz de confusión del modelo Naive Bayes con Word2Vec

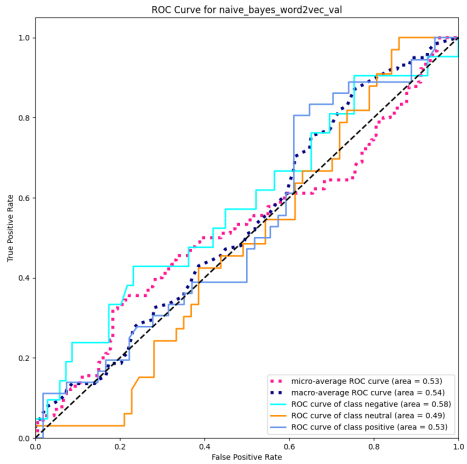

Figura 73: Curva ROC del modelo Naive Bayes con Word2Vec

B.1.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	0.24	1.00	0.39	21
Neutral	1.00	0.03	0.06	31
Positive	0.00	0.00	0.00	35
Accuracy			0.25	87
Macro Avg	0.41	0.34	0.15	87
Weighted Avg	0.42	0.25	0.12	87

Cuadro B.3: Classification Report del modelo Naive Bayes con GLOVE

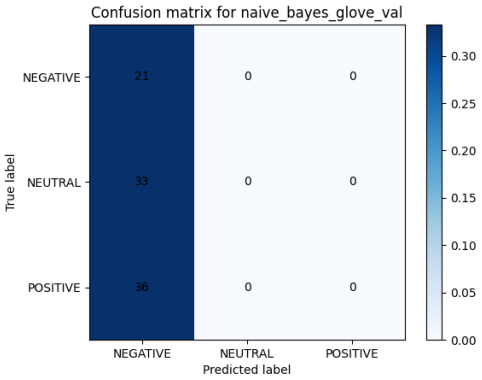

Figura 74: Matriz de confusión del modelo Naive Bayes con GLOVE

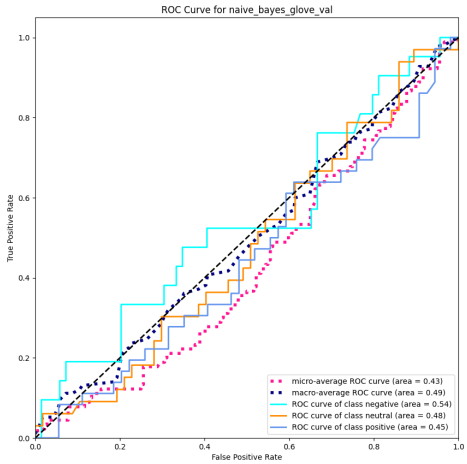

Figura 75: Curva ROC del modelo Naive Bayes con GLOVE

B.2. Rendimiento del modelo Decission Tree

B.2.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.33	0.05	0.08	21
Neutral	0.38	0.94	0.54	34
Positive	0.60	0.08	0.14	37
Accuracy			0.39	92
Macro Avg	0.44	0.36	0.26	92
Weighted Avg	0.46	0.39	0.28	92

Cuadro B.4: Classification Report del modelo Decission Tree con TF-IDF

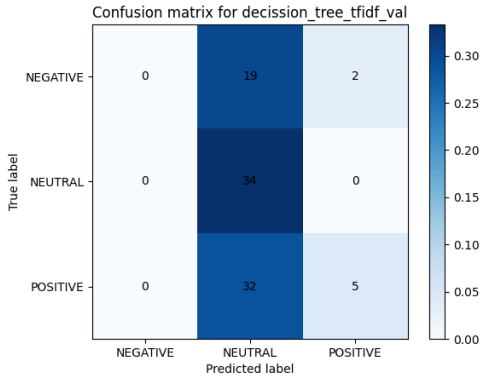

Figura 76: Matriz de confusión del modelo Decission Tree con TF-IDF

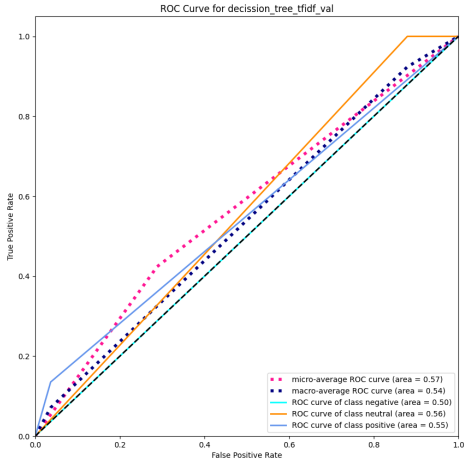

Figura 77: Curva ROC del modelo Decission Tree con TF-IDF

B.2.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.36	1.00	0.53	31
Positive	0.00	0.00	0.00	35
Accuracy			0.36	87
Macro Avg	0.12	0.33	0.18	87
Weighted Avg	0.13	0.36	0.19	87

Cuadro B.5: Classification Report del modelo Decission Tree con Word2Vec

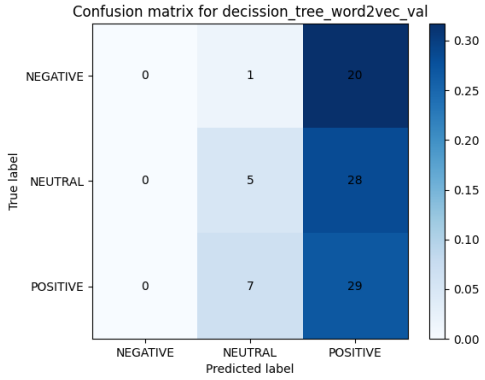

Figura 78: Matriz de confusión del modelo Decission Tree con Word2Vec

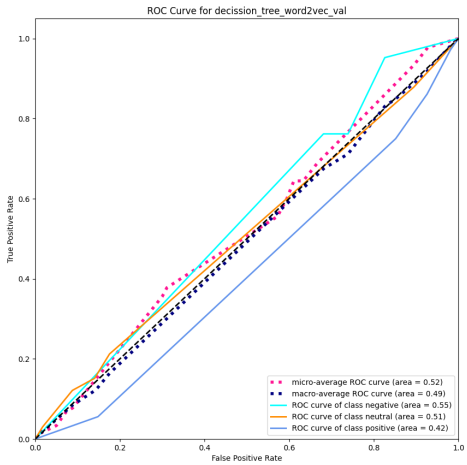

Figura 79: Curva ROC del modelo Decission Tree con Word2Vec

B.2.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.35	0.97	0.52	31
Positive	0.00	0.00	0.00	35
Accuracy			0.34	87
Macro Avg	0.12	0.32	0.17	87
Weighted Avg	0.13	0.34	0.18	87

Cuadro B.6: Classification Report del modelo SVM con TF-IDF

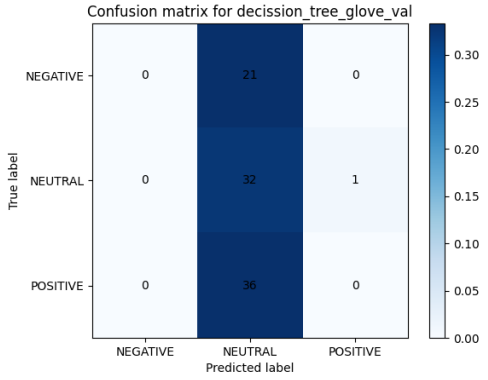

Figura 80: Matriz de confusión del modelo Decission Tree con GLOVE

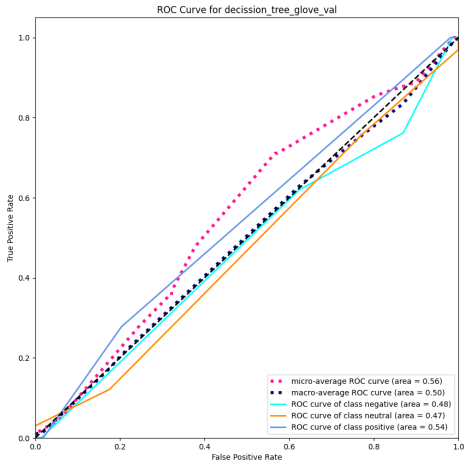

Figura 81: Curva ROC del modelo Decission Tree con GLOVE

B.3. Rendimiento del modelo Gradient Boosting

B.3.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.40	0.10	0.15	21
Neutral	0.35	0.65	0.45	34
Positive	0.33	0.22	0.26	37
Accuracy			0.35	92
Macro Avg	0.36	0.32	0.29	92
Weighted Avg	0.35	0.35	0.31	92

Cuadro B.7: Classification Report del modelo Gradient Boosting con TF-IDF

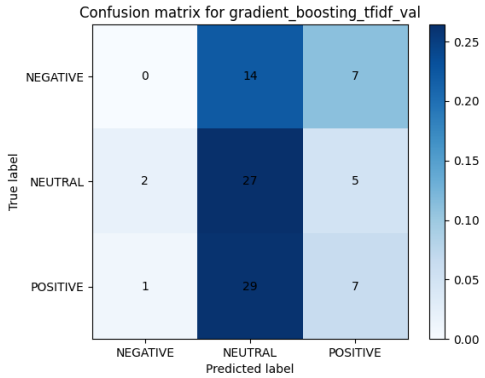

Figura 82: Matriz de confusión del modelo Gradient Boosting con TF-IDF

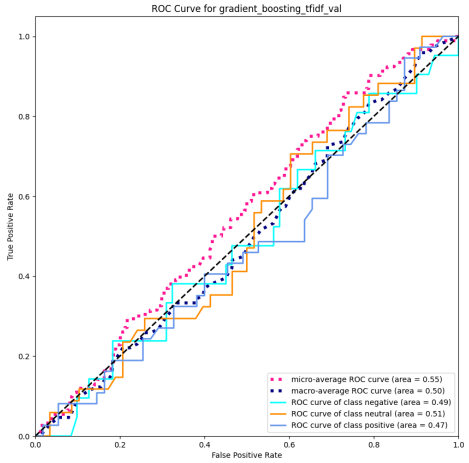

Figura 83: Curva ROC del modelo Gradient Boosting con TF-IDF

B.3.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.36	1.00	0.53	31
Positive	0.00	0.00	0.00	35
Accuracy			0.36	87
Macro Avg	0.12	0.33	0.18	87
Weighted Avg	0.13	0.36	0.19	87

Cuadro B.8: Classification Report del modelo Gradient boosting con Word2Vec

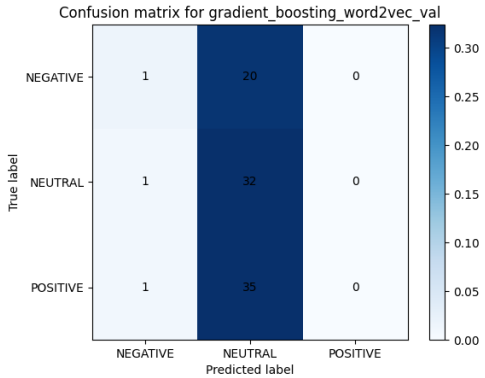

Figura 84: Matriz de confusión del modelo Gradient Boosting con Word2Vec

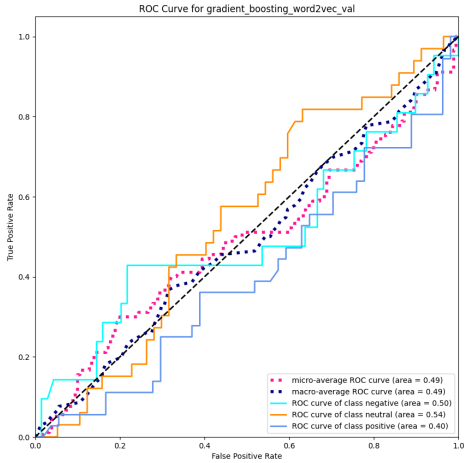

Figura 85: Curva ROC del modelo Gradient Boosting con Word2Vec

B.3.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	1.00	0.05	0.09	21
Neutral	0.38	0.97	0.55	31
Positive	0.62	0.14	0.23	35
Accuracy			0.41	87
Macro Avg	0.67	0.39	0.29	87
Weighted Avg	0.63	0.41	0.31	87

Cuadro B.9: Classification Report del modelo Gradient Boosting con GLOVE

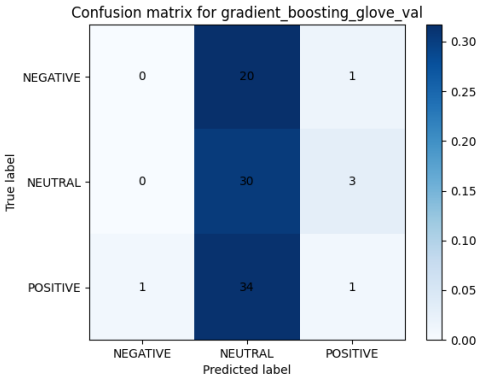

Figura 86: Matriz de confusión del modelo Gradient Boosting con GLOVE

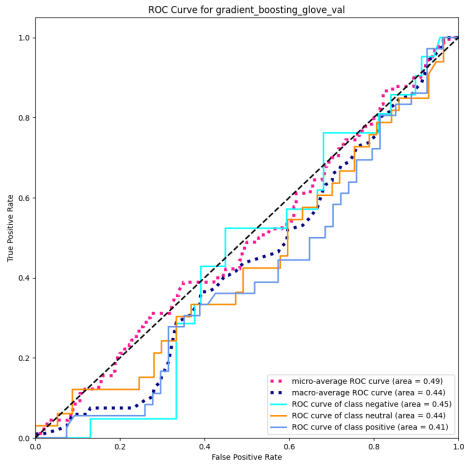

Figura 87: Curva ROC del modelo Gradient Boosting con GLOVE

B.4. Rendimiento del modelo Random Forest

B.4.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.38	1.00	0.55	34
Positive	0.33	0.03	0.05	37
Accuracy			0.38	92
Macro Avg	0.24	0.34	0.20	92
Weighted Avg	0.28	0.38	0.22	92

Cuadro B.10: Classification Report del modelo Random Forest con TF-IDF

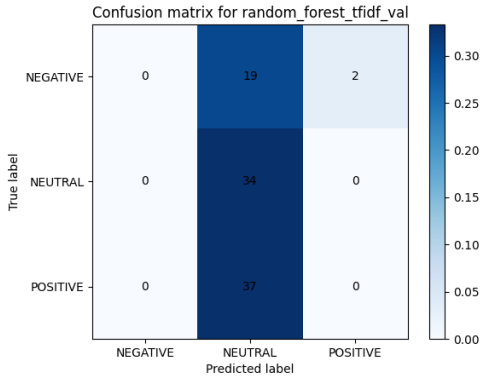

Figura 88: Matriz de confusión del modelo Random Forest con TF-IDF

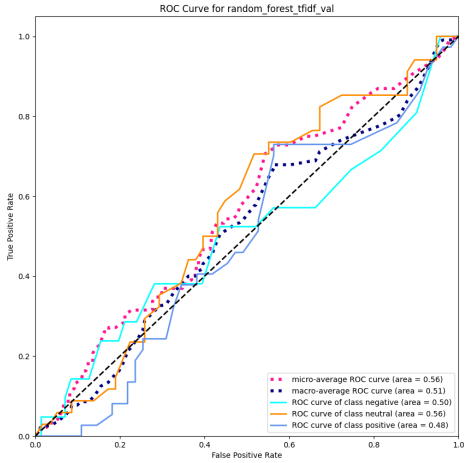

Figura 89: Curva ROC del modelo Random Forest con TF-IDF

B.4.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.36	1.00	0.53	31
Positive	0.00	0.00	0.00	35
Accuracy			0.36	87
Macro Avg	0.12	0.33	0.18	87
Weighted Avg	0.13	0.36	0.19	87

Cuadro B.11: Classification Report del modelo Random Forest con Word2Vec

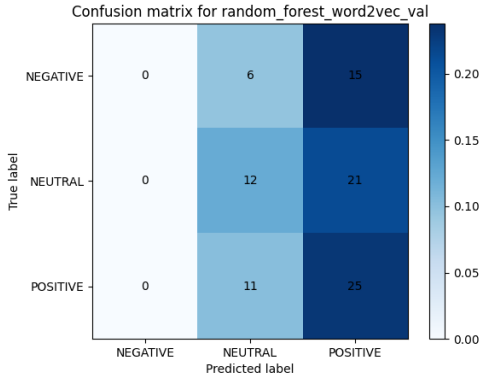

Figura 90: Matriz de confusión del modelo Random Forest con Word2Vec

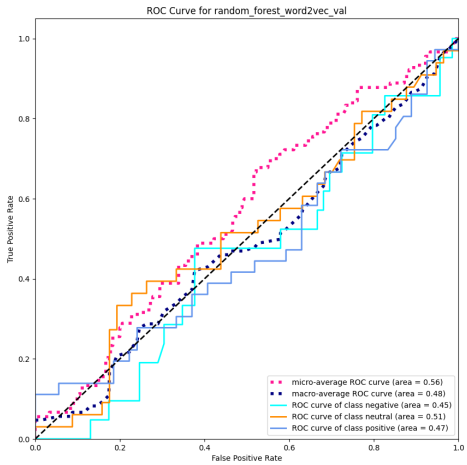

Figura 91: Curva ROC del modelo Random Forest con Word2Vec

B.4.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.36	1.00	0.53	31
Positive	0.00	0.00	0.00	35
Accuracy			0.36	87
Macro Avg	0.12	0.33	0.18	87
Weighted Avg	0.13	0.36	0.19	87

Cuadro B.12: Classification Report del modelo Random Forest con GLOVE

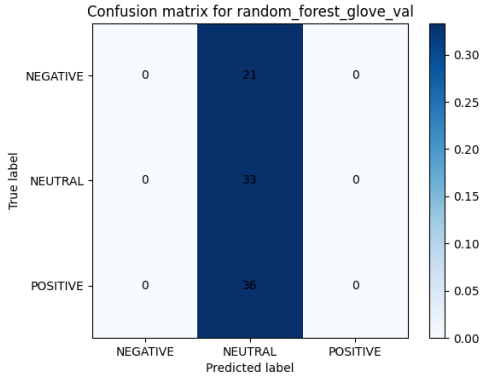

Figura 92: Matriz de confusión del modelo Random Forest con GLOVE

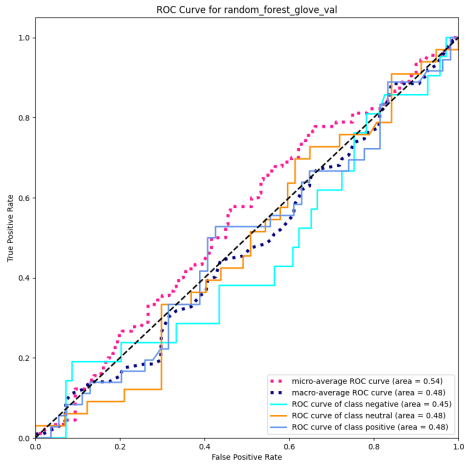

Figura 93: Curva ROC del modelo Random Forest con GLOVE

B.5. Rendimiento del modelo SVM

B.5.1. Empleando embeddings del algoritmo TF-IDF

Class	Precision	Recall	F1-Score	Support
Negative	0.25	0.05	0.08	21
Neutral	0.36	0.74	0.48	34
Positive	0.39	0.19	0.25	37
Accuracy			0.36	92
Macro Avg	0.33	0.32	0.27	92
Weighted Avg	0.35	0.36	0.30	92

Cuadro B.13: Classification Report del modelo SVM con TF-IDF

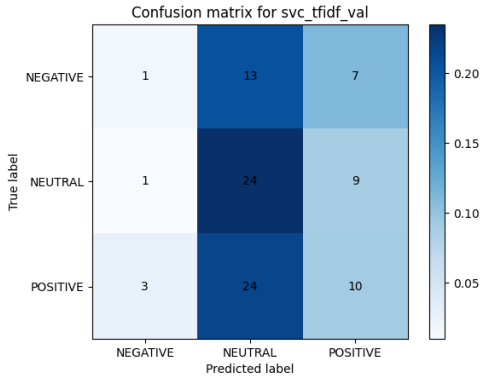

Figura 94: Matriz de confusión del modelo SVM con TF-IDF

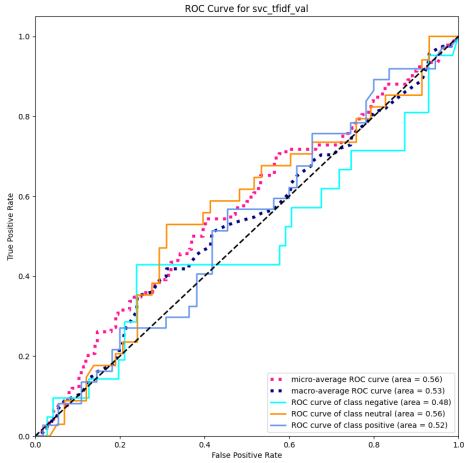

Figura 95: Curva ROC del modelo SVM con TF-IDF

B.5.2. Empleando embeddings del algoritmo Word2Vec

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.36	1.00	0.53	31
Positive	0.00	0.00	0.00	35
Accuracy			0.36	87
Macro Avg	0.12	0.33	0.18	87
Weighted Avg	0.13	0.36	0.19	87

Cuadro B.14: Classification Report del modelo SVM con Word2Vec

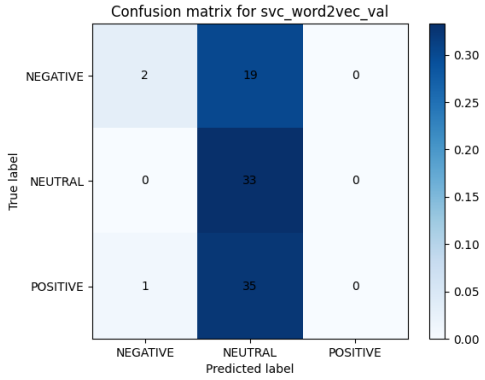

Figura 96: Matriz de confusión del modelo SVM con Word2Vec

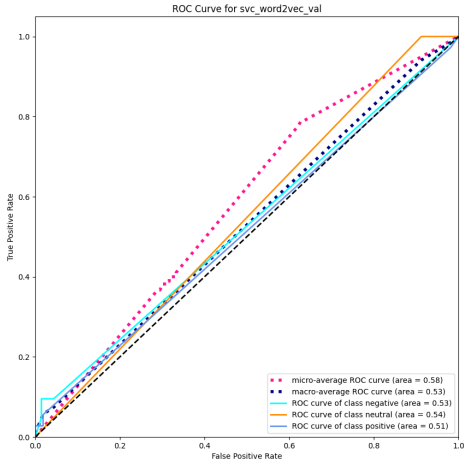

Figura 97: Curva ROC del modelo SVM con Word2Vec

B.5.3. Empleando embeddings del algoritmo GLOVE

Class	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	21
Neutral	0.36	1.00	0.53	31
Positive	0.00	0.00	0.00	35
Accuracy			0.36	87
Macro Avg	0.12	0.33	0.18	87
Weighted Avg	0.13	0.36	0.19	87

Cuadro B.15: Classification Report del modelo SVM con GLOVE

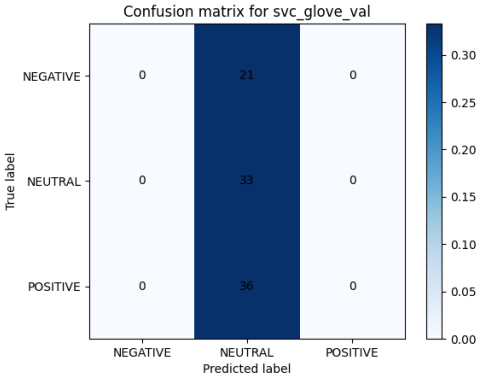

Figura 98: Matriz de confusión del modelo SVM con GLOVE

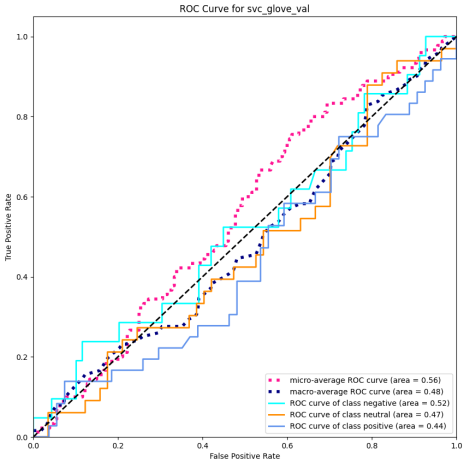

Figura 99: Curva ROC del modelo SVM con GLOVE

B.6. Matrices de confusión y curvas de pérdida del modelo RNN

B.6.1. Empleando embeddings del algoritmo TF-IDF

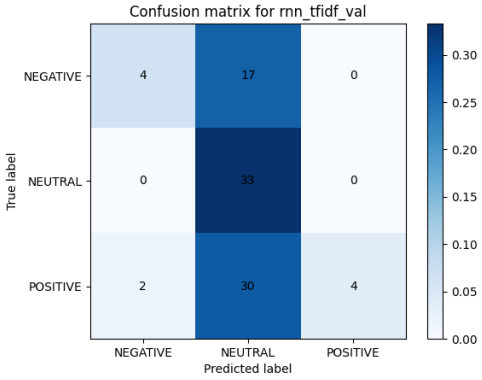

Figura 100: Matriz de confusión del modelo SVM con TF-IDF

B.6.2. Empleando embeddings del algoritmo Word2Vec

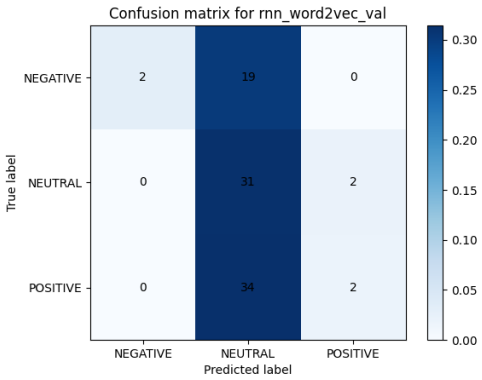

Figura 101: Matriz de confusión del modelo RNN con Word2Vec

B.6.3. Empleando embeddings del algoritmo GLOVE

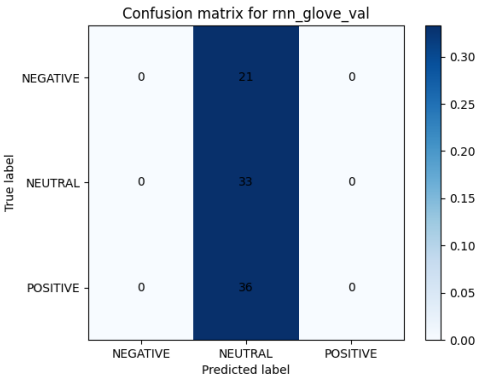

Figura 102: Matriz de confusión del modelo SVM con GLOVE

B.7. Matrices de confusión y curvas de pérdida del modelo LSTM

B.7.1. Empleando embeddings del algoritmo TF-IDF

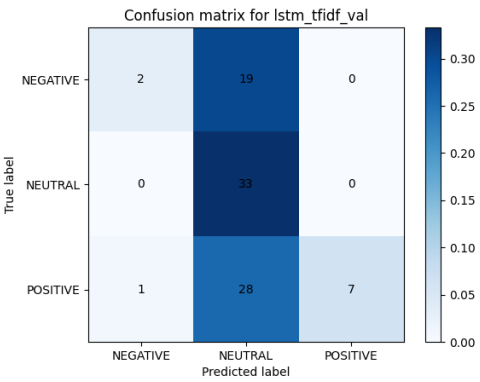

Figura 103: Matriz de confusión del modelo LSTM con TF-IDF

B.7.2. Empleando embeddings del algoritmo Word2Vec

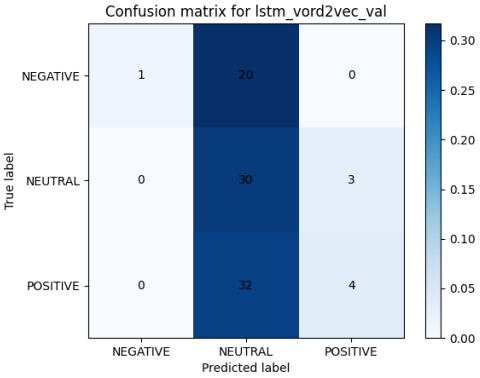

Figura 104: Matriz de confusión del modelo LSTM con Word2Vec

B.7.3. Empleando embeddings del algoritmo GLOVE

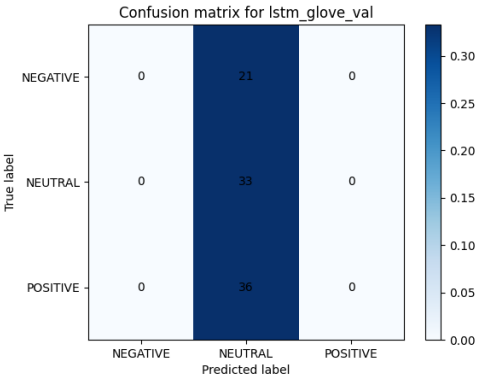

Figura 105: Matriz de confusión del modelo LSTM con GLOVE

Bibliografía

[1] Dogu Tan Araci. Finbert: Financial sentiment analysis with pre-trained language models. *Faculty of Science, University of Amsterdam*, 2020.

[2] Dr. Soumen Atta. Text classification with naive bayes in python: A complete guide. https://blog.devgenius.io/text-classification-with-naive-bayes-in-python-a-complete-guide-c01a7090869f. Consultado el 22.04.2023.

[3] Autor(es). Financial sentiment analysis using machine learning approach. https://wisdomml.in/financial-sentiment-analysis-using-machine-learning-approach/. Consultado el 05.05.2023.

[4] Autor(es). Natural language processing using gradient boosting machine and h2o library. https://www.section.io/engineering-education/natural-language-processing-using-gradient-boosting-machine-and-h2o-library/. Consultado el 15.05.2023.

[5] Autor(es). Python word2vec for text classification with lstm. https://python.plainenglish.io/python-word2vec-for-text-classification-with-lstm-d9e63e84f6ee. Consultado el 05.05.2023.

[6] Dipika Baad. Sentiment classification using word embeddings (word2vec). https://medium.com/swlh/sentiment-classification-using-word-embeddings-word2vec-aedf28fbb8ca. Consultado el 22.04.2023.

[7] Data Robot Blog. Using machine learning for sentiment analysis: A deep dive. https://www.datarobot.com/blog/using-machine-learning-for-sentiment-analysis-a-deep-dive/#:~:text=Sentiment%20analysis%20models&text=Logistic%20regression%20is%20a%20good,Random%20Forests%2C%20and%20Naive%20Bayes. Consultado el 05.02.2023.

[8] Jason Brownlee. What Are Word Embeddings? https://machinelearningmastery.com/what-are-word-embeddings/, Accedido el 22.03.2023.

[9] PARTH CHITTAWAR. Financial news sentiment analysis. https://www.kaggle.com/code/parthchittawar/financial-news-sentiment-analysis. Consultado el 01.05.2023.

[10] Wikipedia contributors. Tf-idf. https://en.wikipedia.org/wiki/Tf%E2%80%93idf, Consultado el 15.03.2023.

[11] Wikipedia contributors. Word2vec. https://en.wikipedia.org/wiki/Word2vec, Consultado el 30.01.2023.

[12] Argimiro Arratia. Gustavo Avalos. Alejandra Cabaña. Ariel Duarte-López. and Martí Renedo-Mirambell. Sentiment analysis of financial news: Mechanics and statistics. *Universitat Politecnica de Cataluña*, 2020.

[13] GeeksforGeeks. Python word embedding using word2vec. https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/. Consultado el 03.02.2023.

[14] Ahmed Hazourli. Financialbert - a pretrained language model for financial text mining. *Research Gate*, 02 2022.

[15] Chijioke Idoko. Randomforest classifier vs. multinomial naive bayes for a multi-output natural language. https://medium.com/analytics-vidhya/randomforest-classifier-vs-multinomial-naive-bayes-for-a-multi-output-natural-l Consultado el 16.05.2023.

[16] Jay Jalammar. How gpt-3 works: Visualizations and animations. Disponible en: http://jalammar.github.io/how-gpt3-works-visualizations-animations/. Consultado el 24.05.2023.

[17] Jay Jalammar. The illustrated bert, elmo, and co. (how nlp cracked transfer learning). http://jalammar.github.io/illustrated-bert/. Consultado el el 23.05.2023.

[18] Jay Jalammar. Illustrated transformer. https://jalammar.github.io/illustrated-transformer/. Consultado el 28.05.2023.

[19] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65, 2014.

[20] Jeffrey Pennington. Richard Socher. Christopher D. Manning. Glove: Global vectors for word representation. *4 Association for Computational Linguistics*, 2013.

[21] Marketaux. Marketaux api. https://www.marketaux.com/. Consultado el 01.02.2023.

[22] Cory Marklin). Natural language processing feature engineering using tf-idf. https://towardsdatascience.com/natural-language-processing-feature-engineering-using-tf-idf-e8b9d00e7e76. Consultado el 15.03.2023.

[23] OpenAI. Openai platform documentation - completion api introduction. https://platform.openai.com/docs/guides/completion/introduction. Consultado el 24.05.2023.

[24] OpenAI. Openai examples. https://platform.openai.com/examples, 2023. Consultado el el 13.03.2023.

[25] OpenAI. Openai models documentation. https://platform.openai.com/docs/models/overview, 2023. Consultado el el 13.03.2023.

[26] Sushant Patrikar. Sentiment Classification using Word Embeddings (Word2Vec). https://medium.com/swlh/sentiment-classification-using-word-embeddings-word2vec-aedf28fbb8ca, Accedido el 20.01.2023.

[27] Daniel Peña, Pilar Poncela, and Esther Ruiz. *Nuevos Métodos de Predicción Económica con Datos Masivos*. Funcas, 2020.

[28] Damien Puy. The power of text: How news sentiment influences financial markets. https://www.imf.org/en/Blogs/Articles/2019/12/16/blog-the-power-of-text. Consultado el el 28.05.2023.

[29] Shahzad Qaiser and Ramsha Ali. Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181, 07 2018.

[30] Ahmed Rachid. FinancialBERT-Sentiment-Analysis. https://huggingface.co/ahmedrachid/FinancialBERT-Sentiment-Analysis. Consultado el 20.01.2023.

[31] Şeyma Sarigil and Murat Koklu. *NATURAL LANGUAGE PROCESSING TECHNIQUES*, pages 205–217. Yaşar Hız, 06 2022.

[32] Streamlit. Streamlit. a faster way to build adn share data apps. python library. https://streamlit.io/. Consultado el 10.02.2023.

[33] CFI Team. Analysis of financial statements. https://corporatefinanceinstitute.com/resources/accounting/analysis-of-financial-statements/. Consultado el 28.05.2023.

[34] Alicia Tuovila. Bert explained: State of the art language model for nlp. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270. Consultado el el 23.05.2023.

[35] Alicia Tuovila. Financial analysis: Definition, importance, types, and examples. https://www.investopedia.com/terms/f/financial-analysis.asp#:~:text=Financial%20analysis%20is%20the%20process,to%20warrant%20a%20monetary%20investment. Consultado el 25.05.2023.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[37] Analytics Vidhya. Word Vectorization using GloVe. https://medium.com/analytics-vidhya/word-vectorization-using-glove-76919685ee0b, Consultado el 13.02.2023.

[38] Wikipedia, la enciclopedia libre. Algoritmo de porter. `https://es.wikipedia.org/wiki/Algoritmo_de_Porter`. Consultado el 20.03.2023.

[39] Andy Xie, Camille Bruckmann, and Tracy Qian. Sensitivity analysis on transferred neural architectures of bert and gpt-2 for financial sentiment analysis. *Journal of Financial Natural Language Processing*, 2022.