TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica Superior de Jaén

# Detección de Ansiedad en Redes Sociales

Autor: Jesús Zafra Moreno

Grado: Ingeniería Informática

Director: María Dolores Molina González
Departamento del director: Ing. Telecomunicación (EPSL)

Fecha: 21/06/2024

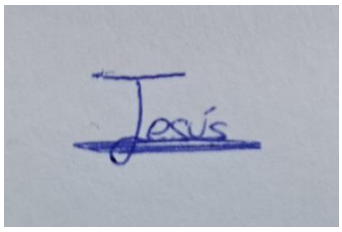
Universidad de Jaén

Escuela Politécnica Superior de Jaén
Departamento de Informática

Doña María Dolores Molina González, tutora del Proyecto Fin de Carrera titulado: **Detección de ansiedad en redes sociales**, que presenta Jesús Zafra Moreno, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, junio de 2024

Alumno:



Jesús Zafra Moreno

Tutora:

MOLINA GONZALEZ MARIA DOLORES - 26009191D

Firmado digitalmente por MOLINA GONZALEZ MARIA DOLORES - 26009191D
Fecha: 2024.06.21 09:51:10 +02'00'

María Dolores Molina González

# Índice

# Índice de Figuras

# Índice de Tablas

## Agradecimientos

Me gustaría aprovechar la elaboración de mi Trabajo Fin de Grado, el cual supone la culminación de este importante capítulo de mi vida académica, una etapa llena de aprendizaje y dificultades, para agradecer a las personas que me han acompañado y que han sido pilares fundamentales en este camino.

Principalmente, agradecer a mis padres el apoyo incondicional que han tenido siempre en mí, cuyo amor, sacrificio y dedicación han sido la fuerza que me ha impulsado a alcanzar mis metas, mostrando persistentemente su confianza y creyendo siempre en mi talento. También a mi hermana, por ser mi alegría constante y, en general, a toda mi familia, por celebrar mis alegrías y logros. Siempre me han inculcado el valor del esfuerzo, haciéndome llegar hasta donde estoy ahora mismo. Sin ellos nada de esto sería posible.

A Marta, mi amor y compañera, a quien le debo el equilibrio emocional que ha sido esencial en este viaje. En ella encuentro mi lugar en el que evadirme del mundo, alcanzando la tranquilidad y felicidad que me aporta. Me ha enseñado a ver la vida desde un punto más sereno, logrando que sea mejor persona.

Agradecer a mis amigos, "Los Borderline". Alguien dijo una vez que "los amigos son la familia que elegimos", no podría estar más de acuerdo con esta afirmación, sé que ante cualquier problema puedo acudir a ellos en busca de ayuda. Y como no, también agradecer a mis amigas, especialmente a mis dos Anas, que sin su apoyo moral constante, estos años tan intensos se hubieran hecho el doble de complicados.

Finalmente, agradecer a mi tutora María Dolores, por tener plena disponibilidad hacia mí cuando he necesitado ayuda y por irme guiando en el desarrollo de este proyecto, contribuyendo en otro escalón más de mi desarrollo personal y académico.

## Resumen

Aunque años anteriores ya se habían observado los primeros signos de cambio de paradigma en la comunicación y sociabilización, en esta última década la manera en las que las personas se relacionan ha experimentado notables transformaciones, principalmente a causa de la aparición de las redes sociales y su posterior crecimiento exponencial. A través de estas plataformas, millones de personas interactúan entre sí, compartiendo mediante textos sus emociones, experiencias y opiniones acerca de diferentes temas con el resto de usuarios.

Este foco de información que permite conocer prácticamente en tiempo real los sentimientos y las opiniones de los usuarios ha creado la necesidad de explotar este fenómeno, siendo aprovechado para diversos propósitos (estudios de mercado, investigación…). En este proyecto, se aborda la realización de un estudio exhaustivo de dichos sentimientos en busca de identificar la posible presencia de ansiedad en los usuarios de las redes sociales.

Para llevar a cabo lo anteriormente comentado, este trabajo se centrará en realizar una aplicación software en Python, con interfaz web para facilitar la interacción, consistente en un sistema de reconocimiento automático de la ansiedad a partir de una muestra de mensajes obtenida de la red social Telegram. Este trabajo se enmarca dentro de los conceptos del procesamiento del lenguaje natural y la analítica de datos. Se realizará un estudio de los mensajes enviados por los usuarios y, a raíz de unos criterios preestablecidos y el aprendizaje, clasificará a los usuarios según si padecen o no dicho trastorno mental.

**Palabras clave:** Procesamiento de Lenguaje Natural, Análisis de Ansiedad, Telegram, Aprendizaje automático

## Abstract

Although in previous years the first signs of a paradigm shift in communication and socialisation had already been observed, in the last decade the way in which people relate other another has undergone notable transformations, mainly due to the emergence of social networks and their subsequent exponential growth. Through these platforms, millions of people interact with each other, sharing their emotions, experiences and opinions on different topics with other users using texts.

This focus of information that allows us to know the feelings and opinions of users in practically real time has created the need to exploit this phenomenon, being used for different purposes (market studies, research...). In this project, an exhaustive study of these feelings is carried out in order to identify the possible presence of anxiety in the users of social networks.

To carry out the aforementioned, this work will focus on developing a software application in Python, with a web interface to facilitate interaction, consisting of an automatic anxiety recognition system based on a sample of messages obtained from the Telegram social network. This work is framed within the concepts of natural language processing and data analytics. A study of the messages sent by users will be carried out and, based on pre-established criteria and learning, users will be classified according to whether or not they suffer from this mental disorder.

**Keywords:** Natural Processing Language, Anxiety Analysis, Telegram, Machine Learning

# 1. Introducción

El propósito principal de este proyecto es realizar un análisis de los sentimientos de los usuarios que interactúan en las redes sociales para detectar signos de ansiedad. Aunque existen numerosas plataformas disponibles, como Twitter[1] o Facebook[2], que son utilizadas por las personas para compartir contenido, las restricciones para la extracción de grandes cantidades de información impuestas por estas plataformas han llevado a limitar el estudio a la red social Telegram[3], centrándose en analizar los mensajes enviados por los miembros de algunos grupos como fuente de datos.

El proyecto se enfocará en la extracción y preprocesamiento de mensajes de un grupo de Telegram dedicado a la ansiedad, junto con un corpus previamente etiquetado en el mismo tema. Utilizando este corpus para entrenar algoritmos de aprendizaje automático (*machine learning*, ML) en el campo del Procesamiento del Lenguaje Natural (PLN), se clasificarán los fragmentos de texto extraídos como positivos o negativos en términos de ansiedad. La efectividad de estos algoritmos se evaluará mediante diversas métricas, con el objetivo de mejorar la comprensión de los sentimientos de los usuarios en las redes sociales respecto a la ansiedad.

## 1.1. Motivación

El mundo actual gobernado por todo el universo social, en el cual existe la continua necesidad de compartir nuestro día a día, ha convertido a las personas en esclavos de los dispositivos electrónicos.

Este exceso de información que las personas muestran en las redes sociales, puede ser utilizada para gran cantidad de fines benévolos o malignos. En este caso, se quiere realizar una interpretación de dicha información en busca de detectar un problema que cada vez tiene más personas, como es la ansiedad

Según el Instituto Nacional de Salud Mental (National Institute of Mental Health, en inglés) [1], el **trastorno de ansiedad** se refiere a una condición psicológica en la cual los individuos experimentan un estado persistente de preocupación, miedo o aprehensión,

[1] https://twitter.com/
[2] https://www.facebook.com/
[3] https://telegram.org/

acompañado por síntomas físicos como tensión muscular, sudoración, palpitaciones y dificultad para respirar. Estos síntomas pueden ser desproporcionados en relación con la situación que los desencadena y pueden interferir con las actividades diarias, las relaciones interpersonales y el bienestar general.

En los últimos años, desde los jóvenes hasta las instituciones gubernamentales, han hecho hincapié en la importancia que tiene la salud mental. Por ello, esta herramienta de detección de la ansiedad en las redes sociales no solo ayudará a los expertos a identificar a qué personas dirigir sus esfuerzos, sino que también permitirá una detección temprana que evite que la situación alcance límites indeseables. Si bien esta herramienta no constituye un diagnóstico definitivo, dado que la evaluación precisa del trastorno de ansiedad debe ser realizada por un profesional de la salud mental, sí puede desempeñar un papel crucial como un mecanismo auxiliar para detectar usuarios que presenten signos de este trastorno, direccionando así la atención y el apoyo hacia quienes lo necesiten.

Con respecto a la decisión de realizar el Trabajo Fin de Grado (TFG) sobre este tema, pese a que no es una propuesta propia, al momento de seleccionar entre los distintos temas sugeridos por la Escuela Politécnica Superior de Jaén, me gratificaba poder destinar mis conocimientos adquiridos a lo largo de los distintos cursos del grado en Ingeniería Informática para ayudar a los demás, concretamente en el mundo de la salud mental. Tengo la posibilidad de poder combinar mi interés en el PLN con mi afición por las redes sociales y mi sentimiento altruista de ayudar a las personas con problemas. Soy fiel defensor de que hay que poner el foco en este aspecto y aumentar los esfuerzos para la detección precoz de los problemas mentales.

Me enfrento a un reto tedioso debido a la complejidad que supone determinar si una persona sufre un trastorno mental, como es la ansiedad, principalmente a causa de la subjetividad de ciertos factores, pero las aportaciones que realice al mundo de la investigación en este campo serán complacientes para mí.

## 1.2. Objetivos

Este TFG tiene como objetivo principal determinar la posibilidad de que un usuario tenga ansiedad analizando sus textos escritos en redes sociales. Para lograr alcanzar dicho objetivo y así contribuir a la mejora de los sistemas existentes de clasificación de trastornos

mentales, la propuesta es desarrollar e integrar recursos lingüísticos adaptados al español y al trastorno. Este sistema, a partir de la introducción de un conjunto de datos con un formato específico, debe determinar si el comentario expresa ansiedad.

Por tanto, los objetivos específicos de este proyecto son:

- Estudiar en qué consiste el Análisis de Emociones y su importancia.
- Desarrollar un lexicón de palabras asociadas al trastorno tratado.
- Desarrollar una herramienta que extraiga de un texto si existe o no la posibilidad de que el usuario sufra de ansiedad.
- Diseñar una interfaz web que permita al usuario introducir los conjuntos de datos a examinar y obtener un resultado.

## 1.3. Planificación temporal

En este apartado del trabajo se procede a la descripción de las actividades vinculadas a la gestión del proyecto. Una adecuada administración de los esfuerzos es una tarea decisiva en el desarrollo de cualquier proyecto y, por ende, en el logro de sus metas. Ayuda a poner en valor el trabajo invertido y facilita la organización en la elaboración del mismo, distribuyendo correctamente los plazos entre las distintas tareas a realizar. Se hará uso de los diagramas de Gantt para facilitar la visualización de la planificación a cumplir.

El **diagrama de Gantt** [2] es una herramienta gráfica diseñada para mostrar la planificación del tiempo asignado a diversas tareas o actividades durante un período específico. Está compuesto por un eje vertical en el que se identifican las actividades que componen el proyecto y un eje horizontal que muestra en un calendario la extensión temporal de cada una de las actividades. Su objetivo es mejorar la visibilidad de la programación realizada, consiguiendo a través de la posición de cada tarea a lo largo del tiempo poder identificar las posibles relaciones e interdependencias que puedan existir entre ellas.

Este proyecto corresponde al Trabajo Fin de Grado en el Grado de Ingeniería Informática de la Universidad de Jaén. Tiene una carga de 12 créditos que equivale a 300 horas de trabajo, comprendido en 9 horas presenciales de orientación o tutela individualizada y 291 horas de trabajo autónomo.

Aunque este trabajo me fue asignado de manera definitiva el 22 de noviembre de 2023, tras acordar ciertos términos y condiciones con la tutora, se determinó que la ejecución del proyecto se pospusiera hasta principios de 2024. Por consiguiente, se estima que el proyecto inicie aproximadamente el 2 de enero de 2024 y se prevé que concluya alrededor del 1 de julio de 2024.

Inicialmente, se llevaría a cabo una fase previa hasta mediados del mes de febrero enfocada a labores de investigación para conocer en mayor profundidad el tema y para iniciar la documentación de la parte teórica del proyecto. Tendría una duración de un poco más de 6 semanas, con una estimación de trabajo de aproximadamente 6-7 horas por semana, equivalente a 1-2 horas diarias, excluyendo los fines de semanas como días laborables. Tras concertar una nueva reunión con la tutora en la que se certifique con qué metodología enfocar los distintos algoritmos, se iniciará el desarrollo del proyecto en su plenitud. Se dispondrá de aproximadamente 19 semanas, sin contar los períodos vacacionales, donde se estiman en torno a unas 15 horas a la semana de trabajo. Esto equivaldría a 2-4 horas diarias si se excluyen los fines de semana, al igual que en la fase anterior.

Después de establecer con claridad los objetivos y las tareas a realizar en el proyecto, se elabora una planificación temporal que muestre el plan de acción. Esta programación incorpora dos diagramas de Gantt que representan la planificación inicial (Figura 1 y Figura 2), así como otro par que muestra la ejecución real del trabajo (Figura 3 y Figura 4).

Planificación de tareas

En cualquier iniciativa de investigación y desarrollo de software, es obligatorio llevar a cabo una fase preliminar de indagación antes de emprender cualquier tipo de implementación. Esta etapa tiene como objetivo recopilar información esencial para comprender adecuadamente la finalidad del proyecto, preparando así todos los recursos necesarios para iniciar de manera fructífera la ejecución del mismo.

En pos de adquirir el conocimiento necesario, se da inicio a la investigación explorando las diversas fuentes bibliográficas relacionadas con el análisis de sentimientos en las redes sociales, focalizando esta búsqueda en el ámbito de la ansiedad. Dado que este aspecto ha despertado interés reciente, la documentación disponible es limitada y mayormente en inglés, lo que implica un esfuerzo adicional en comparación con otras circunstancias.

Además, resulta conveniente revisar estudios anteriores vinculados a este proyecto, ya que contribuirán a definir la hoja de ruta y proporcionarán referencias fundamentales.

La información recapitulada será empleada para establecer los objetivos que se persiguen con la ejecución del proyecto, formulando también preguntas que serán abordadas a lo largo de todo el proceso de desarrollo. Este enfoque proporciona una visión teórica y metodológica desde la perspectiva informática, la cual será documentada de manera continua para plasmar de manera detallada la información relevante.

Después de comprender y asimilar los distintos conceptos presentados, así como los conocimientos adquiridos durante el grado, estos se convertirán en las pautas fundamentales para tomar decisiones sobre qué algoritmos implementar. El proyecto se enfocará en una serie de algoritmos basados en el PLN, por lo que sería conveniente realizar una investigación exhaustiva y una selección cuidadosa en este campo y en los algoritmos de ML adecuados para la detección de emociones en texto, específicamente la ansiedad, los cuales se utilizarán principalmente en el proyecto.

Habiendo completado estas labores previas de contextualización e investigación, se avanza hacia la ejecución de las actividades centradas en el desarrollo más específico del proyecto.

Antes de dar comienzo a la implementación, es necesario realizar la captura y subsiguiente preprocesamiento de los datos extraídos. Si bien el corpus de datos destinado al entrenamiento de los modelos generados en el proyecto se proporciona previamente, junto con su correspondiente etiquetado, los datos obtenidos en la captura se utilizarán específicamente para realizar pruebas y obtener resultados sobre los algoritmos.

Esta distinción permite llevar a cabo las verificaciones posteriores con el propósito de contrastar los resultados conseguidos. Cabe destacar que los datos consistirán en mensajes extraídos de algún grupo de Telegram y se organizarán en varios archivos con formato JSON.

Esos datos extraídos requieren someterse a un proceso de procesamiento antes de su utilización, principalmente porque involucran texto, pero también pueden incluir otros elementos como emojis que deben convertirse a texto. Además, es clave validar la integridad de los datos. A lo largo de la ejecución de esta etapa del proyecto, se documentará de manera minuciosa todo el proceso realizado, dejando constancia de cómo se ha ido realizando.

Una vez que se han establecido los objetivos, definido los métodos a implementar y recopilados los datos necesarios, se inicia la fase desarrollo y entrenamiento de modelos. Estos modelos se encargarán de analizar una serie de mensajes formateados, introducidos como datos, para determinar si un usuario está experimentando ansiedad o no. Se utilizará una serie de algoritmos de ML en conjunto con técnicas de PLN para analizar mensajes de redes sociales.

Tras la implementación de los algoritmos, se procede a realizar la integración de un léxico especializado diseñado para la detección de términos relacionados con la ansiedad. Este proceso fortalece la capacidad del sistema para identificar y clasificar de manera más precisa los elementos vinculados con el estado emocional específico que estamos investigando.

Luego de haber desarrollado los modelos, se ejecutan pruebas de evaluación utilizando los datos extraídos previamente, con el objetivo de verificar la eficacia de los algoritmos. Estos algoritmos y procedimientos estarán sujetos a ajustes y optimizaciones continuas basadas en los resultados obtenidos en las pruebas. Este enfoque iterativo garantiza una mejora constante y una adaptación precisa a los desafíos específicos identificados al inicio del proyecto.

Todo este proceso estará respaldado por su correspondiente documentación pormenorizada en la que se aborden los diferentes aspectos de métodos implementados, así como los diversos parámetros seleccionados para su ejecución adecuada. Asimismo, se incluirán los resultados obtenidos al poner a prueba los algoritmos con los datos extraídos.

Adicionalmente, con el propósito de mejorar la interacción del usuario con los algoritmos, se lleva a cabo la implementación de una interfaz web que permite integrar de manera más accesible los métodos desarrollados. Cada algoritmo tendrá acceso a un directorio que albergará los archivos necesarios para su entrenamiento. Mediante la interfaz, el usuario podrá seleccionar el algoritmo a ejecutar e introducir el conjunto de datos que desea evaluar.

Después de concluir el desarrollo integral del código, se procederá a finalizar la redacción de la memoria de prácticas. A lo largo de todo el proceso, se ha documentado detalladamente el avance de la implementación, aunque ciertos apartados requerirán completarse, como es el caso de la conclusión tras analizar todos los resultados obtenidos.

Asimismo, se realizará una comprobación exhaustiva de todo el contenido del documento, asegurándose de que el formato sea coherente y uniforme. Se efectuarán adaptaciones según sea necesario para garantizar que la edición sea semejante y el contenido esté cohesionado.

Como tarea final, tras la finalización del proyecto, y en adición a la revisión previa de la documentación, se hará una evaluación general del trabajo desde una perspectiva técnica. Esto implicará verificar el cumplimiento de todos los objetivos previamente establecidos, examinar detenidamente el diseño y desarrollo del trabajo, así como comprobar la coherencia entre los resultados obtenidos, entre otros aspectos.

También se elaborará la presentación requerida para exponer de manera detallada el trabajo ante el tribunal asignado, proporcionando un resumen conciso del proyecto. Formalmente, culmina este Trabajo Fin de Grado con la entrega y la posterior defensa.

Planificación inicial

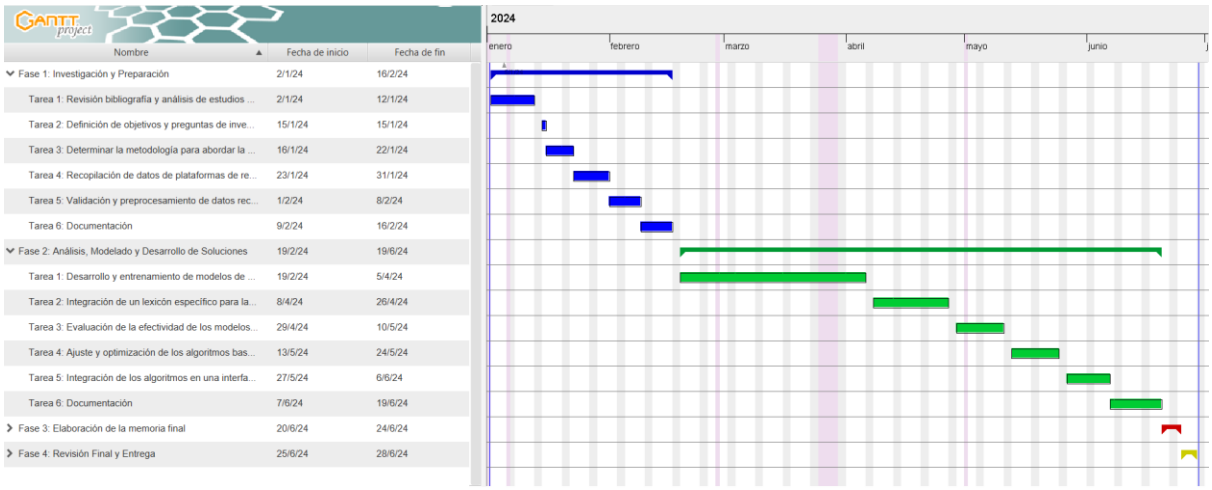
Las representaciones gráficas presentadas en la Figura 1 y Figura 2 ilustran el diagrama de Gantt que refleja la planificación inicial del proyecto. Se visualizan las diferentes actividades a realizar distribuidas en cuatro fases distintas: investigación y preparación; análisis, modelado y desarrollo de soluciones; elaboración de la memoria final y, por último, revisión final y entrega.

En este cronograma se presenta la disposición general de las tareas previstas necesarias para llevar a cabo el desarrollo completo del proyecto, organizadas en las distintas fases anteriormente comentadas. Se han establecido unos plazos (a visualizar en las ilustraciones) y un orden específico para cada una de las tareas, lo que contribuye a una planificación temporal efectiva:

1. Investigación y Preparación.
    a. Revisión bibliografía y análisis de estudios previos sobre la ansiedad en las redes sociales.
    b. Definición de objetivos y preguntas de investigación.
    c. Determinar la metodología para abordar la situación y resolver el problema.
    d. Recopilación de datos de plataformas de redes sociales.

e. Validación y preprocesamiento de datos recolectados.
f. Documentación.

2. Análisis, Modelado y Desarrollo de Soluciones.
    a. Desarrollo y entrenamiento de modelos de Aprendizaje Automático utilizando técnicas de Procesamiento de Lenguaje Natural (PLN) para analizar mensajes de redes sociales.
    b. Integración de un lexicón específico para la detección de términos relaciones con ansiedad.
    c. Evaluación de la efectividad de los modelos implementados.
    d. Ajuste y optimización de los algoritmos basados en los resultados obtenidos durante las pruebas.
    e. Integración de los algoritmos en una interfaz común para facilitar la interacción del usuario.
    f. Documentación.
3. Elaboración de la memoria final.
    a. Redacción de las conclusiones del trabajo.
    b. Revisión y edición del documento final.
4. Revisión Final y Entrega.
    a. Revisión final del trabajo desde la perspectiva técnica.
    b. Preparación de la presentación técnica (Defensa).




| Nombre | Fecha de inicio | Fecha de fin |
|---|---|---|
| Fase 1: Investigación y Preparación | 2/1/24 | 16/2/24 |
| Tarea 1: Revisión bibliografía y análisis de estudios ... | 2/1/24 | 12/1/24 |
| Tarea 2: Definición de objetivos y preguntas de inve... | 15/1/24 | 15/1/24 |
| Tarea 3: Determinar la metodología para abordar la ... | 16/1/24 | 22/1/24 |
| Tarea 4: Recopilación de datos de plataformas de re... | 23/1/24 | 31/1/24 |
| Tarea 5: Validación y preprocesamiento de datos rec... | 1/2/24 | 8/2/24 |
| Tarea 6: Documentación | 9/2/24 | 16/2/24 |
| Fase 2: Análisis, Modelado y Desarrollo de Soluciones | 19/2/24 | 19/6/24 |
| Tarea 1: Desarrollo y entrenamiento de modelos de ... | 19/2/24 | 5/4/24 |
| Tarea 2: Integración de un lexicón específico para la... | 8/4/24 | 26/4/24 |
| Tarea 3: Evaluación de la efectividad de los modelos... | 29/4/24 | 10/5/24 |
| Tarea 4: Ajuste y optimización de los algoritmos bas... | 13/5/24 | 24/5/24 |
| Tarea 5: Integración de los algoritmos en una interfa... | 27/5/24 | 6/6/24 |
| Tarea 6: Documentación | 7/6/24 | 19/6/24 |
| Fase 3: Elaboración de la memoria final | 20/6/24 | 24/6/24 |
| Fase 4: Revisión Final y Entrega | 25/6/24 | 28/6/24 |

**Figura 1:** Planificación inicial del proyecto Parte 1

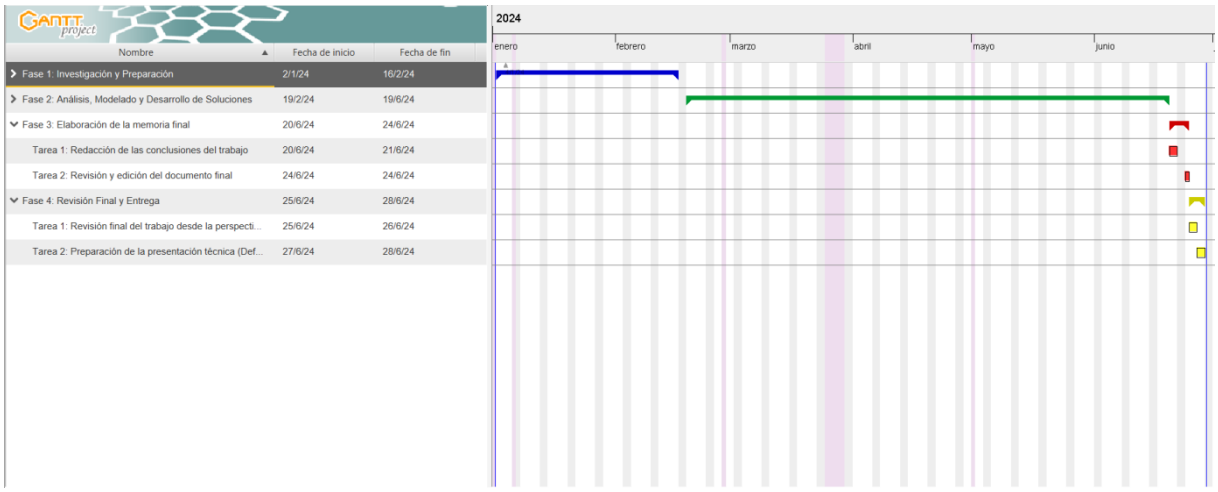



**Figura 2:** Planificación inicial del proyecto Parte 2

Pese a que existe una etapa específica en el plan de ejecución del proyecto destinada a la elaboración de la memoria del mismo, este será un proceso continuo que se llevará a cabo a lo largo de las fases precedentes, siendo en dicha etapa donde se realizará una revisión integral, completando y ajustando el contenido del documento. El propósito es contar con una memoria que evolucione a medida que avanza el proyecto, documentando de manera precisa la progresión alcanzada.

De manera análoga, a la hora de desarrollar el código de los algoritmos que componen este proyecto, cabe destacar que, aunque se ha programado una tarea concreta dirigida a optimización de dichos algoritmos, este esfuerzo no se limitará a una única tarea. Más bien, será una labor que se extenderá durante el período de implementación, permitiendo mejoras y adaptaciones constantes a medida que se avanza con su desarrollo.

Sin embargo, si bien se ha comentado que tanto la documentación como la optimización del código son un proceso continuo, se establecen claramente puntos de finalización que deben ser alcanzados antes de pasar a la siguiente fase. Estos puntos de finalización actúan como hitos o metas específicas que indican que ciertos aspectos o tareas críticas han sido completados satisfactoriamente antes de proceder al siguiente paso del proyecto.

Existe una probabilidad significativa de que esta estimación inicial realizada pueda experimentar ciertas modificaciones, dada la posible influencia de diversos factores que podrían requerir ajustes y adaptaciones en la planificación, por lo que se debe proporcionar un margen de flexibilidad. Es por ello que, a continuación, se muestra el esfuerzo real que

finalmente ha sido necesario, respaldado por las justificaciones pertinentes para cada cambio ejecutado.

Esfuerzo real

Sobre la hipótesis anteriormente planteada acerca de los incumplimientos de los plazos preestablecidos para completar las distintas tareas del proyecto, se ha podido certificar que, debido a diversos imprevistos que surgieron y que la temporalización estaba mal realizada, sobre todo por el desconocimiento de las dificultades que surgirían, los tiempos no se han cumplido.

Tomando como base la planificación inicial, se han efectuado anotaciones detalladas de los progresos diarios alcanzados en el desarrollo del proyecto, creando lo que se conoce como un cuaderno de campo. Haciendo uso de este registro, se ha elaborado un nuevo diagrama de Gantt, presentado en la Figura 3 y Figura 4, que refleja el esfuerzo real que ha supuesto este Trabajo Fin de Grado. Este diagrama actualiza las tareas y los plazos según las demandas del proyecto, adelantándose la fecha de entrega a mediados de junio, a la espera de la defensa ante el tribunal de la universidad.

En lo que respecta a la estructuración, las tareas a desempeñar no han sufrido ninguna modificación, exceptuando que se ha especificado las tareas de documentación realizadas a lo largo de cada fase. Aparte de este punto, el cronograma inicial se mantiene inalterado. No obstante, se han realizado ajustes significativos en los plazos conforme se han ido cumplimentando cada una de las labores programadas.

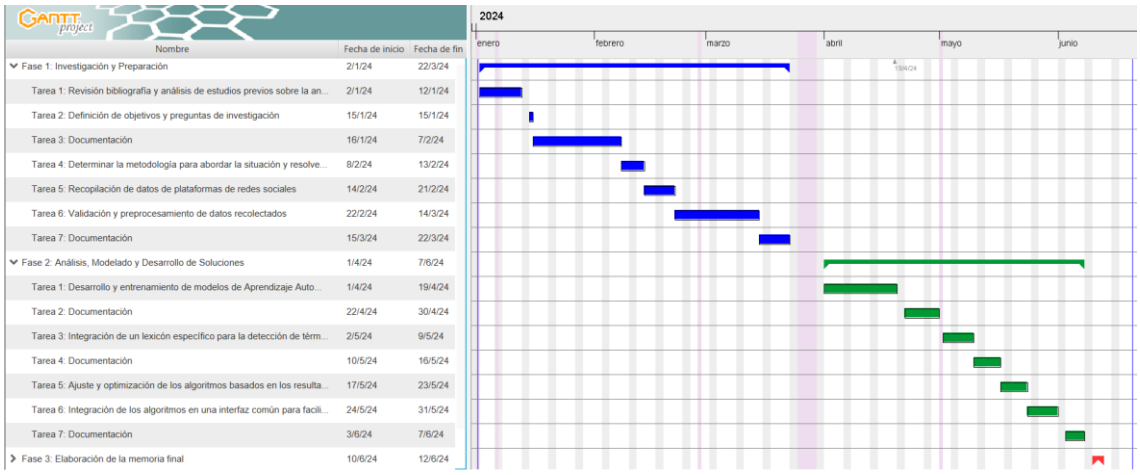



**Figura 3:** Esfuerzo real del proyecto Parte 1

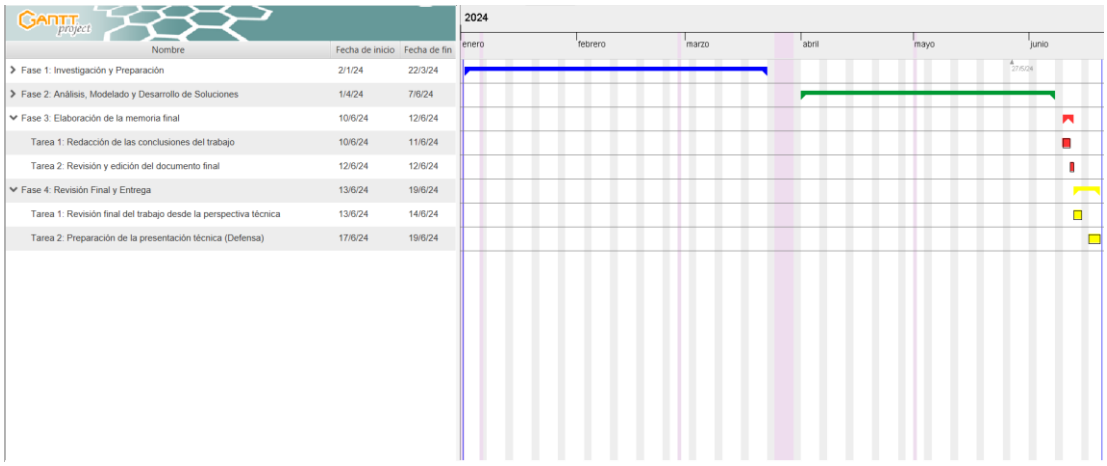



**Figura 4:** Esfuerzo real del proyecto Parte 2

## 1.4. Presupuesto

Una vez se ha elaborado la planificación del proyecto, se presenta de manera detallada el presupuesto desglosado del mismo. Este presupuesto incluye una especificación minuciosa de los diferentes gastos necesarios para la ejecución del proyecto. A continuación, se desglosan las principales categorías de gastos:

Mano de obra

Tal y como se ha mencionado en el apartado anterior, el proyecto está diseñado para ser desarrollado en un total de 300 horas. De estas, 9 horas están específicamente dedicadas a tareas de tutoría. Para una mejor comprensión, se ha realizado una ponderación detallada de los días y horas dedicadas a cada tarea. En la Tabla 1 se presenta este recuento de horas, desglosado por actividad.

| **Tareas** | **Dedicación (horas)** |
|---|---|
| Investigación y preparación | 36 |
| Diseño | 20 |
| Implementación | 160 |
| Pruebas | 20 |
| Documentación | 55 |
| Reuniones | 9 |
| **Tiempo de dedicación total (horas)** | 300 |

**Tabla 1:** Estimación de horas invertidas en el Proyecto

Con este cálculo de horas y una estimación del salario correspondiente al personal informático cualificado para realizar estas tareas, tomando como referencia los salarios de empresas similares en el sector, se ha elaborado la Tabla 2. Esta tabla muestra el gasto necesario en personal (costes sin I.V.A).

| Cargo | Nº de horas | Coste/hora | Total (€) |
|---|---|---|---|
| Analista/Diseñador | 56 | 22 € | 1232 |
| Programador | 160 | 19 € | 3040 |
| Documentalista | 55 | 15 € | 825 |
| **Total** | | | 5097 € |

**Tabla 2:** Presupuesto de Mano de Obra

Recursos

Para llevar a cabo el desarrollo completo de este proyecto, es necesario contar con una serie de recursos tanto de hardware como de software. A continuación, se detalla el equipo informático adquirido específicamente para este proyecto, junto con la Tabla 3 que refleja sus costes:

- Ordenador personal: Portátil Gaming HP VICTUS 16-r0020ns, con procesador i7, 16GB de RAM, 512GB SSD, tarjeta gráfica Nvidia GeForce RTX 4050 de 6GB, pantalla de 16,1", y sistema operativo Windows 11.
- Ratón: Ratón inalámbrico Logitech M185.
- Monitor auxiliar: Monitor LG Ultragear 27GQ50F-B de 27", LED FullHD, 165Hz, con tecnología FreeSync Premium.

| Descripción | Coste (€) |
|---|---|
| Portátil Gaming HP VICTUS | 1099 |
| Ratón inalámbrico Logitech M185 | 12 |
| Monitor LG Ultragear 27GQ50F-B 27" | 150 |
| **Total** | 1261 € |

**Tabla 3:** Presupuesto de Recursos Hardware

En lo que respecta al software, no ha sido necesario adquirir ningún tipo de licencia adicional, ya que el sistema operativo Windows 11 venía preinstalado en el portátil. Las herramientas de software necesarias para el proyecto se indican en la Tabla 4:

| Descripción | Coste (€) |
|---|---|
| Microsoft Office Profesional 2010 | 0 |
| Visual Studio Code | 0 |
| Google Chrome | 0 |
| GanttProject | 0 |
| **Total** | 0 € |

**Tabla 4:** Presupuesto de Herramientas Software

Resumen del Presupuesto

El coste de este proyecto se sustenta en tres pilares fundamentales: mano de obra, equipamiento hardware y herramientas software. Las dos primeras categorías representan los gastos más significativos, ya que se optó por no utilizar herramientas que requirieran el pago de licencias. Además, existen otros gastos extras derivados del desarrollo del proyecto, como la conexión a Internet de hasta 100MB con el operador *MÁSMÓVIL* y los gastos en copistería. En la Tabla 5 se refleja el presupuesto general de este proyecto.

| Descripción | Coste Total (€) |
|---|---|
| Equipo de Trabajo | 5097 |
| Equipamiento Hardware | 1261 |
| Herramientas Software | 0 |
| Costes extra | 180 |
| **Total del proyecto** | 6538 € |

**Tabla 5:** Presupuesto Total del Proyecto

# 2. Estado del arte

La contextualización desempeña un rol primordial en la comprensión adecuada de la solución, especialmente para aquellos segmentos de la sociedad que carecen de conocimientos sobre estos aspectos. Por lo tanto, resulta importante explicar la fundamentación teórica que respalde esta necesidad, con el fin de facilitar un entendimiento más amplio y accesible para un público menos familiarizado con la materia.

## 2.1. Estudio de las emociones

El estudio de las emociones es un campo de investigación en crecimiento debido a su relevancia actual. Por lo tanto, la siguiente información contribuirá a fortalecer el marco teórico necesario para comprender este proyecto desde un punto de vista más psicológico y social. Se abordará la temática de manera progresiva, comenzando desde un enfoque más general y luego adentrándose con mayor profundidad en el tema específico del proyecto.

Emociones

Las emociones desempeñan una función vital en la configuración de los fenómenos sociales, sirviendo como catalizadores que influyen en nuestras interacciones diarias. En las últimas décadas, se ha intensificado el interés en el análisis de estas complejas respuestas emocionales, marcando un hito en la investigación sociológica. Este proceso de estudio debe persistir y evolucionar, buscando la completa integración de las emociones en la perspectiva sociológica general. Comprender cómo las emociones afectan la toma de decisiones, las relaciones interpersonales y la estructura social es esencial para un análisis sociológico completo y preciso.

A la hora de abordar una definición precisa de las emociones, nos enfrentamos a una tarea tediosa, ya que los seres humanos sólo podemos experimentar la vida de manera emocional, lo cual implica un concepto intrínsecamente complejo. Diversos autores especializados en este tema han propuesto una amplia variedad de definiciones, destacándose la formulada por el sociólogo estadounidense Norman K. Denzin. En su libro ‘On Understanding Emotion’ publicado en 1984, Denzin describe las **emociones** [3] como “una experiencia corporal viva, veraz, situada y transitoria que impregna el flujo de conciencia de una persona, que es percibida desde el interior y recorriendo el cuerpo, y que, durante el

transcurso de su vivencia, sume a la persona y a sus acompañantes en una realidad nueva y transformada – la realidad de un mundo constituido por la experiencia emocional".

Es aconsejable realizar una distinción entre las diversas categorías de estados afectivos, diferenciando entre emociones primarias y secundarias. Las emociones primarias se consideran respuestas universales, esencialmente fisiológicas, evolutivamente relevantes y biológica y neurológicamente innatas. Por otro lado, las emociones secundarias son una combinación de las primarias, fuertemente condicionadas por factores sociales y culturales. La clasificación de las emociones primarias [3] puede variar según los autores. Por ejemplo, Kemper incluye el miedo, la ira, la depresión y la satisfacción como emociones primarias, mientras que Turner añade la satisfacción-felicidad, la aversión-miedo, la aserción-ira, la depresión-tristeza y el sobresalto-sorpresa. Además, se incluyen como emociones secundarias la culpa, la vergüenza, el amor y la decepción.

Es importante destacar la significativa contribución del eminente psicólogo Paul Ekman al estudio de las emociones, siendo uno de los más destacados del siglo XX y pionero en el análisis de las expresiones faciales para cada emoción. Ekman identificó 7 emociones básicas [4], las cuales son: la tristeza, la ira, la sorpresa, el miedo, el asco, el desprecio y la alegría. En relación con este proyecto, su estudio sobre la tristeza es especialmente relevante, siendo considerada como la emoción más duradera en el tiempo, manifestándose en la expresión facial y la voz, y a menudo asociada con el llanto. Además, el miedo, otra emoción clave, se desencadena como respuesta a la percepción de una amenaza de daño mental, lo que provoca una reacción de huida o escondite.

A pesar de que las emociones humanas pueden parecer inicialmente simples e intuitivas, en realidad, esconden una profunda complejidad y problemática. No pueden ser simplemente consideradas como respuestas mecánicas o fisiológicas a las variaciones en el entorno. Diversos factores influyen en la experiencia emocional del individuo, tales como:

- La intensidad con la que el sujeto valora consciente y/o inconscientemente los hechos acontecidos.
- A qué o a quién se atribuya la causa o responsabilidad de esos hechos.
- La identidad social activa en ese momento.
- La identificación del sujeto con otras personas, grupos o colectivos.

Todas estas circunstancias se basan en una serie de teorías propuestas por diversos autores que han destacado la variedad de influencias en las emociones humanas. Estas teorías ponen en relieve la complejidad de las experiencias emocionales, en las cuales la naturaleza inesperada del entorno del individuo cobra protagonismo. De ahí surge la dificultad para establecer reglas que permitan determinar ciertas emociones de manera objetiva. Entre las teorías más relevantes, cabe destacar:

- El autor Brody L [5], en su libro 'Gender, Emotion and the Family', introduce la Teoría de la Apreciación. Esta teoría postula que los seres humanos no son simplemente mecanismos biológicos sensibles, sino que antes de experimentar o expresar una emoción, valoramos cognitivamente los elementos del entorno. Esta perspectiva destaca la complejidad y la importancia de la evaluación cognitiva en el proceso emocional.
- Lawler EJ, Thye SR y Yoon J [6], en su obra 'Social Exchange and Micro Social Order', presentan las Teorías de Atribución, donde sostienen que la emoción experimentada no dependerá únicamente del hecho en sí mismo, sino también de la atribución causal que realice el sujeto. Esta perspectiva enfatiza la influencia de la interpretación subjetiva en la experiencia emocional, añadiendo una capa de complejidad a la comprensión de las interacciones sociales y el orden microsocial.
- Turner JH y Stets JE [7], en su obra 'The Sociology of Emotions', presentan la Teoría de las Expectativas. Según esta teoría, la valoración de un objeto, hecho o persona por parte del individuo dependerá significativamente de las expectativas previas que tenga. Este enfoque destaca cómo las expectativas pueden modular y modificar la experiencia emocional resultante, proporcionando una perspectiva valiosa sobre la dinámica de las emociones en el ámbito sociológico.

Toda esta información nos lleva a la reflexión de que las emociones constituyen la expresión corporal de la relevancia que un hecho del mundo natural o social tiene para el sujeto. Este fenómeno se sustenta en tres elementos esenciales: el individuo, el acontecimiento en cuestión y la valoración subjetiva que se infiere de las circunstancias que rodean a ambos elementos.

¿Cómo influyen las emociones a la forma de comunicar?

Debido a ser un término empleado en múltiples campos, el concepto de **comunicación** recibe diversas definiciones. El sociólogo y profesor francés Wolton [8], reconocido por sus estudios en comunicación, aborda este término desde tres perspectivas. La primera implica entender la comunicación como una experiencia antropológica esencial en la que se produce un intercambio con el otro. La segunda perspectiva la interpreta como un conjunto de técnicas que facilita la transmisión de información, siguiendo así el curso del desarrollo tecnológico del último siglo. Por último, se puede concebir la comunicación como una necesidad social funcional, es decir, como un intercambio entre sociedades, no solo entre individuos.

Todos estos significados atribuidos a dicho concepto convergen en la necesidad de interactuar. Interactuar no solo implica poner en contacto, sino que también supone un intercambio mutuo. Este término adquiere aún más relevancia cuando se hace referencia al mundo de las redes sociales. En términos generales, se establece una relación entre un emisor y un receptor mediante el uso de medios tecnológicos, permitiendo al usuario participar y comunicarse con los recursos que se le proporcionan. Varios autores destacan la importancia de la preparación de los usuarios receptores para tener la capacidad de recibir, interpretar y valorar los mensajes recibidos, siendo las emociones un factor fundamental en esta comprensión.

Frente a cualquier información, es posible experimentar una reacción emocional, pero para facilitar la comunicación es esencial procesar la información y obtener una respuesta consciente. Las respuestas emocionales están condicionadas por diversos factores, siendo el medio de transmisión de la información uno de los más influyentes. La velocidad de transmisión, la necesidad de una respuesta inmediata o cómo se interpela al espectador son elementos clave que pueden incidir en dichas respuestas.

Por eso, es fundamental saber gestionar las emociones en diversas situaciones, destacando la importancia de los procesos de regulación emocional, enmarcados dentro del concepto de inteligencia emocional. En 1990, Salovey, un psicólogo estadounidense, junto con colaboradores, definió la **Inteligencia Emocional** [9] como la fusión de la inteligencia intrapersonal e interpersonal. La describió como la "forma de inteligencia social que implica la capacidad de supervisar a uno mismo y a otros, sus sentimientos y emociones, para diferenciar entre ellos, utilizando esta información para conducir a la vez, el pensamiento y la

acción". Esta capacidad de autoconciencia y control emocional, crucial en un contexto social, se desarrolla durante la maduración personal, influida significativamente por la familia y los docentes. El usuario, en última instancia, debe poseer la habilidad de gestionar las distintas emociones según el entorno.

Las redes sociales y las emociones

Con el avance de las Tecnologías de la Información y la Comunicación, las redes sociales se han convertido en un medio de comunicación instantáneo y eficaz. Este fenómeno ha generado un aumento significativo en la participación de la población, especialmente entre los más jóvenes. A través de diversas plataformas, los usuarios comparten no solo sus intereses y gustos, sino también sus inquietudes y emociones, transformando así la forma en que nos relacionamos y socializamos.

El surgimiento de este fenómeno, al igual que cualquier novedad, ha desencadenado una serie de investigaciones en busca de conclusiones sobre su posible impacto en las personas. El mundo cibernético se presenta como un espacio profundamente influyente para aquellos usuarios que deciden sumergirse en él. Javier Serrano-Puche [10] destaca en su artículo que Internet no solo sirve como un canal para expresar emociones, sino que también desempeña una función importante en la construcción de la subjetividad individual. Este medio no solo genera emociones en los usuarios, sino que también influye en cómo esas emociones se manifiestan.

Los estudios [11] han revelado que las personas pueden contagiarse emociones a lo largo del tiempo a través de Internet. La felicidad, la soledad o la depresión pueden propagarse entre individuos socialmente conectados en la era digital. Las redes sociales, en particular, sirven como instrumentos cotidianos para la comunicación y conexión, manifestando una amplia gama de emociones, desde el miedo y la ira hasta la alegría, la sorpresa y la tristeza, junto con sus variaciones.

Además, estos estudios subrayan que compartir emociones en línea puede tener efectos beneficiosos, pero también plantean amenazas potenciales derivadas de contenidos cargados de emociones. Según las investigaciones del académico Hidalgo [12], “el intercambio social en línea puede influir tanto en el bienestar general como en la satisfacción

vital de los individuos, revelando la complejidad de la interacción emocional en el mundo digital".

La red social seleccionada es **Telegram** [13], una aplicación de mensajería gratuita enfocada en la velocidad y seguridad. Actualmente, cuenta con más de 700 millones de usuarios activos mensuales. Al igual que WhatsApp, permite el envío de mensajes de texto o voz, así como compartir fotos, videos y archivos de cualquier tipo. También posibilita la creación de grupos y canales de difusión. La característica distintiva de Telegram radica en ser una plataforma de mensajería basada en la nube con sincronización constante.

Esta aplicación es ideal para el contexto que nos ocupa, ya que dispone de distintos grupos, cada uno para los diferentes problemas emocionales o relacionados con la salud mental, en los que se puede expresar y buscar apoyo o ayuda. Con una capacidad para hasta 200,000 miembros, ofrece funciones como respuestas, menciones y hashtags que contribuyen a mantener el orden y la eficiencia en la comunicación. Además, cuenta con un histórico de mensajes, lo que permite a los nuevos miembros acceder a conversaciones anteriores que pueden ser de utilidad. Para unirse al grupo fácilmente, basta con buscar el nombre del grupo en el buscador de la aplicación y solicitar anexión.

Ansiedad

En el continuo estudio de las emociones humanas, este proyecto se enfoca en el análisis de las emociones vinculadas a trastornos mentales, centrándose específicamente en la *ansiedad*. Este fenómeno complejo suscita un interés significativo en diversos campos de investigación. En el trasfondo de las experiencias emocionales, la ansiedad emerge como un componente esencial que ejerce una influencia sustancial en el bienestar tanto psicológico como físico de los individuos.

Este concepto ha experimentado una evolución histórica a medida que la disciplina de la psicología se ha desarrollado. Diversas teorías han surgido a lo largo del tiempo, proporcionando enfoques variados sobre este fenómeno complejo. Estos encuadres se han desarrollado de manera simultánea, interactuando y enriqueciéndose mutuamente, lo que ha permitido la evolución y la delimitación progresiva del concepto, profundizando así en su comprensión. Entre algunos de estos enfoques se encuentran, por ejemplo, el enfoque psicofisiológico [14], que se centra en los procesos psicológicos subyacentes a la conducta

mediante el registro y análisis de respuestas fisiológicas, y el enfoque psicodinámico, que analiza la actividad interna del individuo a través del método introspectivo.

Los **trastornos de ansiedad** (TA) [15] representan patologías mentales comunes que, con frecuencia, generan sufrimiento y discapacidad, contribuyendo así a una carga sustancial en los ámbitos social y económico. Según datos proporcionados por la Organización Mundial de la Salud (OMS), estos trastornos son más prevalentes en mujeres, registrando un 7.7%, en comparación con el 3.6% observado en hombres. Este desequilibrio de género subraya la relevancia crítica de comprender y abordar los TA, no solo desde una perspectiva clínica, sino también considerando sus implicaciones sociodemográficas.

Las personas afectadas por un trastorno de ansiedad pueden experimentar un temor o preocupación desproporcionados frente a situaciones específicas, como crisis de angustia o eventos sociales, o, en el caso del trastorno de ansiedad generalizada, ante una amplia variedad de situaciones cotidianas. Estos síntomas suelen persistir durante un periodo prolongado, generalmente de varios meses, y con frecuencia llevan a las personas a evitar las situaciones que desencadenan su ansiedad. Además, se identifican otros síntomas [16] asociados a los trastornos de ansiedad, entre ellos:

- Dificultad para concentrarse o tomar decisiones.
- Irritabilidad, tensión o inquietud.
- Náuseas o malestar abdominal.
- Palpitaciones.
- Sudoración, tiritones o temblores.
- Trastornos del sueño.
- Sensación de peligro inminente, pánico o fatalidad.

Existen diversos tipos de trastornos de ansiedad, e incluso es posible que las personas experimenten varios de ellos de manera simultánea, lo que complica la comprensión y el abordaje de estos desafíos psicológicos. La interacción de estos trastornos puede generar una sintomatología única y compleja, desafiando la categorización clásica y destacando la necesidad de enfoques de tratamiento integradores y personalizados. Según la OMS [16], dentro de estos trastornos se encuentran:

1. Trastorno de ansiedad generalizada: caracterizado por una preocupación persistente y excesiva sobre actividades o eventos cotidianos.
2. Trastorno de angustia: se manifiesta a través de crisis de angustia acompañadas por el temor a que se repitan.
3. Trastorno de ansiedad social: se caracteriza por niveles elevados de miedo y preocupación frente a situaciones sociales donde la persona pueda sentirse humillada, avergonzada o rechazada.
4. Agorafobia: involucra un miedo excesivo, preocupación y evitación de situaciones que podrían desencadenar pánico, haciéndose sentir atrapado, indefenso o avergonzado.
5. Trastorno de ansiedad por separación: implica un miedo o preocupación desmedidos por estar separado de personas con las que se tiene un fuerte vínculo emocional.
6. Fobias específicas: estas son miedos intensos e irracionales hacia objetos o situaciones particulares, que llevan a conductas de evitación y generan considerable angustia.
7. Mutismo selectivo: se manifiesta como la incapacidad constante para hablar en ciertas situaciones sociales, a pesar de tener la capacidad de expresarse cómodamente en otros entornos; siendo algo que afecta principalmente a los niños.

Padecer este trastorno tiene una gran influencia en lo que respecta a la calidad de vida de las personas que lo sufren. La calidad de vida [17], según Fernández-López y Hernández-Mejía, abarca la percepción de las capacidades, limitaciones, síntomas y características psicosociales que permiten a un individuo realizar funciones de manera satisfactoria para él mismo. Es decir, este concepto, surgido en el campo de la salud, se utiliza para evaluar el bienestar y la satisfacción de los individuos, considerando diversos aspectos que influyen en su percepción de la vida. Los primeros estudios relacionados con este aspecto en el ámbito de la ansiedad datan de finales de los años 90, lo que significa que hay menos investigaciones disponibles en comparación con otros trastornos mentales.

Sin embargo, a pesar de tener un número menor de investigaciones, se ha llegado a la conclusión [17], respaldada por la coincidencia de todos los estudios, de que cuando se padece un trastorno de ansiedad, especialmente el trastorno de ansiedad generalizada (TAG), la calidad de vida del individuo se ve afectada en diversos grados, especialmente cuando está

acompañada de otro trastorno. Este deterioro se refleja en la presencia de limitaciones en el desempeño laboral, familiar y social, así como en la percepción distorsionada de bienestar.

## 2.2. Procesamiento del Lenguaje Natural

Para llevar a cabo este estudio sobre la ansiedad en las redes sociales, el procesamiento del lenguaje natural se erige como el eje técnico central en torno al cual se desarrollará todo el proyecto. Se abordarán los distintos aspectos relacionados con la extracción, análisis y comprensión de los mensajes textuales en la plataforma Telegram, dada la presencia de restricciones en otras plataformas. Además, se describirán las técnicas y algoritmos que se utilizarán para identificar patrones y tendencias relacionadas con la ansiedad en el discurso digital de los usuarios. Este enfoque integral permitirá obtener posteriormente una visión profunda y precisa del impacto de la ansiedad en el entorno de las redes sociales, así como proporcionar valiosos 'insights'[4] para la salud mental y el bienestar de los usuarios.

El **Procesamiento del Lenguaje Natural** [18] es una disciplina que se sitúa en la intersección de las ciencias de la computación, la inteligencia artificial y la lingüística. Se encarga de investigar cómo las computadoras pueden interactuar con el lenguaje humano y de analizar los aspectos computacionales de las lenguas naturales.

Actualmente, esta disciplina experimenta un notable crecimiento, impulsado por la ascendente necesidad de analizar y comprender el lenguaje de manera computacional en diversos campos y aplicaciones. Sin embargo, se enfrenta a ciertas dificultades al interpretar el texto a analizar, incluyendo la ambigüedad inherente de las lenguas naturales, donde, por ejemplo, una palabra puede tener varios significados. Además, pueden surgir desafíos para detectar la separación entre las palabras, especialmente en la lengua hablada donde no hay pausas entre las palabras, así como la recepción imperfecta de los datos.

La arquitectura del PLN [18], se organiza en cuatro niveles que abarcan desde el análisis de la oración hasta su comprensión completa. Estos niveles incluyen:

[4] Insights: percepciones o entendimientos significativos obtenidos a partir del análisis de datos o información

- Análisis morfológico: implica examinar las palabras para identificar sus raíces, rasgos flexivos, prefijos, sufijos y otros elementos, con el objetivo de comprender cómo se estructuran y forman las palabras.
- Análisis sintáctico: consiste en analizar la estructura de las oraciones para entender las reglas según las cuales se construyen y combinan las palabras.
- Análisis semántico: se centra en extraer el significado de la frase para resolver ambigüedades léxicas y estructurales.
- Análisis pragmático: implica examinar el texto más allá de los límites de la oración para, por ejemplo, determinar los antecedentes referenciales de los pronombres.

Con respecto a su utilización, el PLN se emplea en una amplia variedad de aplicaciones. Además del análisis y la comprensión del lenguaje, como es el caso que nos concierne, se aplica en áreas como la traducción automática, la respuesta a preguntas, la síntesis de voz, entre otras. Este campo tiene un alcance diverso y relevante en múltiples industrias y sectores.

Una vez que se ha establecido el contexto de esta disciplina, se procede a profundizar en las diversas fases o tareas del PLN [19], que reflejan los procedimientos más comunes utilizados en tareas relacionadas con esta área. A continuación, en la Figura 5, se presenta un esquema gráfico que resume de manera visual lo que podría ser una *pipeline*[5] genérica para el desarrollo de sistemas de PLN.

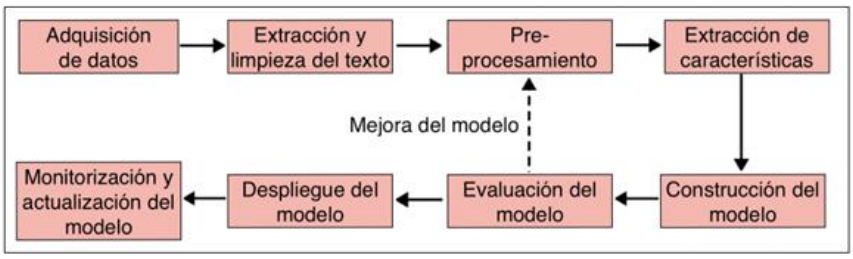



**Figura 5:** *Pipeline* genérica para tareas de PLN

[5] Pipeline: secuencia de procesos o etapas interconectadas que se utilizan para llevar a cabo una tarea o función de manera eficiente y sistemática.

Las etapas que integran el *pipeline* son:

1. Adquisición de los datos: los datos son elementos primordiales en los sistemas de aprendizaje automático. Para obtener conjuntos de datos útiles, se utilizan diversas técnicas como el uso de bases de datos públicas y web scraping[6]. El conjunto de datos de texto resultante se conoce como "corpus".
2. Extracción y limpieza del texto: implica extraer texto puro de los datos de entrada, eliminando información no textual como metadatos[7], y luego convertirlo al formato deseado. Aunque no se utilizan técnicas específicas de PLN, puede ser costosa en tiempo y esfuerzo, y tiene implicaciones en las siguientes etapas del proceso.
3. Pre-procesamiento: aunque ya se posee un texto puro, el software de PLN suele trabajar en contextos de frases y necesita separar el texto al menos en palabras. Además, a menudo se deben eliminar caracteres especiales y dígitos, o convertir todo en minúsculas. Estas acciones se realizan durante el preprocesamiento del texto, que incluye pasos como *tokenización*, eliminación de *stopwords*, stemming, lematización y conversión a minúsculas, entre otros.
4. Extracción de características: para utilizar algoritmos de aprendizaje automático para construir modelos, se necesita introducir los datos de texto al algoritmo. Esto se logra mediante la extracción de características, un conjunto de métodos que capturan las características del texto y las convierten en vectores numéricos comprensibles y manipulables por los algoritmos de ML (representación numérica del texto).
5. Construcción del modelo: implica diseñar un modelo estadístico o de aprendizaje automático, ajustar sus parámetros con los datos de entrenamiento, utilizar un procedimiento de optimización y, finalmente, usarlo para realizar predicciones en nuevos conjuntos de datos. Una ventaja de trabajar con características de valores numéricos es que permite elegir cualquier modelo de ML e incluso combinar varios de ellos.
6. Evaluación del modelo: la evaluación del modelo construido se puede realizar de varias formas, siendo la más común la medición del rendimiento del modelo en

[6] Web scraping: técnica utilizada para extraer automáticamente datos de páginas web.
[7] Metadato: información descriptiva que proporciona detalles sobre otros datos, como su origen, formato, autoría o contexto.

nuevos conjuntos de datos. Esto depende de dos factores principales: el uso de la métrica adecuada para la evaluación y la implementación correcta del proceso de evaluación. Las métricas pueden variar según la tarea de PLN y la fase de evaluación. Además, existen dos tipos de evaluación: intrínseca y extrínseca. La evaluación intrínseca se centra en objetivos intermedios del *pipeline*, mientras que la extrínseca evalúa el rendimiento del modelo en su objetivo final.

7. Despliegue de software: en la mayoría de las aplicaciones, el modelo de PLN forma parte de un sistema más amplio. Por lo tanto, una vez que el modelo ha sido probado y funciona correctamente, debe ser desplegado en un entorno de producción como parte integral del sistema.
8. Monitorización y actualización del modelo: al igual que en cualquier proyecto de desarrollo de software, el modelo se supervisa de forma continua después de su implementación. Además, se continúa recopilando datos adicionales para mantener actualizado el modelo con base en la información más reciente.

Algoritmos o modelos en PLN

Para realizar el análisis de sentimientos, específicamente en relación con la ansiedad, se emplean diversas técnicas [20], tales como:

- Técnicas basadas en aprendizaje automático supervisado: utilizan conjuntos de datos etiquetados para entrenar modelos, donde cada texto está asociado con una etiqueta de sentimiento conocida. Se extraen características del texto y se utilizan como entradas para el modelo. Algoritmos comunes incluyen árboles de decisión, Support Vector Machines (SVM), redes neuronales y Naive Bayes. La calidad de los modelos depende de la calidad de los datos etiquetados y las características seleccionadas.
- Técnicas basadas en aprendizaje automático no supervisado: buscan aprender por sí mismos, por lo que se emplean para descubrir patrones en datos de texto no etiquetados.
- Técnicas basadas en léxico: utilizan diccionarios de palabras previamente etiquetadas con su polaridad para asignar puntuaciones a cada palabra en el texto. Esto permite determinar el sentimiento del texto. Ejemplos de léxicos incluyen

SentiWordNet[8], Opinion Lexicon[9] y VADER[10]. Sin embargo, estas técnicas pueden tener limitaciones, como la falta de contexto o la incapacidad para detectar sarcasmo o ironía.

- Técnicas Híbridas: combinan métodos supervisados y no supervisados con técnicas basadas en léxico para aprovechar las fortalezas de cada enfoque. Mejoran la precisión y robustez del análisis de sentimientos al utilizar múltiples fuentes de información y enfoques complementarios.

Herramientas y Plataformas

Considerando la parte más técnica y dedicada a la programación en sí, la implementación de las diversas técnicas mencionadas anteriormente, que constituirán los modelos empleados para el análisis objetivo del proyecto, se llevará a cabo en el lenguaje de programación Python. Este lenguaje, ampliamente difundido en el ámbito de las tecnologías emergentes, ha experimentado un crecimiento notable en los últimos años. Para la escritura del código y su ejecución, se empleará el entorno de desarrollo integrado Visual Studio Code. Además, se utilizará el entorno de programación "Colab" de Google como apoyo para realizar pruebas auxiliares y colaborativas. Es importante evaluar detenidamente las características, capacidades y limitaciones de estas herramientas antes de su implementación en el proyecto.

El lenguaje de programación **Python** [21] se destaca como un lenguaje de alto nivel interpretado, reconocido por su énfasis en la legibilidad del código y su versatilidad en el desarrollo de una amplia gama de aplicaciones. Es conocido por su compatibilidad multiplataforma debido a su soporte parcial para la orientación a objetos, la programación imperativa y, en menor medida, la programación funcional. Mantenido por *Python Software Foundation*, este lenguaje dinámico y multiplataforma opera bajo una licencia de código abierto.

**Visual Studio Code** (*VS Code*) [22] es un editor de código fuente desarrollado por Microsoft que ofrece una experiencia de desarrollo ágil y personalizable. Su gran flexibilidad se debe a la amplia variedad de extensiones disponibles, lo que lo convierte en una herramienta versátil para desarrolladores de diversos lenguajes y tecnologías. Además de su

[8] https://github.com/aesuli/SentiWordNet
[9] https://www.kaggle.com/datasets/nltkdata/opinion-lexicon
[10] https://github.com/cjhutto/vaderSentiment

interfaz intuitiva, *VS Code* ofrece una serie de características destacadas, como el resaltado de sintaxis, la depuración integrada, el control de versiones y una comunidad activa de usuarios y desarrolladores que contribuyen con extensiones y soporte. Para trabajar con Python en *VS Code*, basta con instalar su extensión oficial, que ofrece funciones como completado automático, linting y gestión de entornos virtuales, simplificando así el desarrollo en este popular lenguaje de programación.

A modo auxiliar, el entorno de programación *Google Colaboratory*, también conocido como "**Colab**" [23], es un servicio de *Google Research* que permite a los usuarios escribir y ejecutar código de Python en el navegador de forma gratuita. Es especialmente útil para tareas de aprendizaje automático, análisis de datos y fines educativos. Colab proporciona un entorno de cuaderno alojado de *Jupyter* sin necesidad de configuración, con acceso a recursos informáticos como GPUs sin coste adicional. Sin embargo, se prioriza a los usuarios activos en el cuaderno y se imponen restricciones para evitar abusos y garantizar recursos equitativos. Estas limitaciones son necesarias para mantener el servicio gratuito y funcional para todos los usuarios.

Este cuaderno posibilita la combinación de celdas de texto para incluir explicaciones y celdas de código que se ejecutarán en una máquina virtual asignada a la cuenta de Google vinculada al cuaderno. Las máquinas virtuales se eliminan después de un período de inactividad y están sujetas a un ciclo de vida máximo establecido por el servicio de Colab. Los archivos generados se almacenan en Google Drive y pueden descargarse en formato de cuaderno de Jupyter de código abierto (.ipynb).

# 3. Desarrollo

Una vez se han adquirido los conocimientos necesarios para abordar la problemática central de este proyecto, que consiste en detectar posibles casos de ansiedad entre los usuarios de las redes sociales, el siguiente paso es diseñar una solución efectiva. Con esta base de conocimientos, se procede a llevar a cabo las fases preliminares previas a la implementación de los modelos, las cuales incluyen la preparación del corpus de datos y la determinación de los modelos que se van a construir para alcanzar los objetivos establecidos en este trabajo.

## 3.1. Datos de entrada y salida

El análisis de este proyecto se basa en mensajes de texto obtenidos de diversos grupos de Telegram destinados a usuarios con trastorno de ansiedad. Aunque en este caso el objeto de estudio sea mensajes de texto, esta metodología puede ser extrapolada a diferentes tipos de texto. El enfoque del análisis estará determinado por la presencia o ausencia de ansiedad, etiquetando los mensajes según el nivel de ansiedad asociado a cada uno.

Aunque existen diferentes opciones para realizar la etiquetación de los mensajes, en este proyecto se consideran principalmente dos planteamientos. Una opción es simplificar la asignación a dos valores: 1 si presenta el trastorno y 0 si no lo presenta, permitiendo una evaluación clara y directa del mismo. Alternativamente, se puede optar por un enfoque más complejo que determine un rango entre 0 y 1, según el porcentaje de positividad que los modelos detecten en el conjunto de texto analizado.

En cuanto a los datos, los mensajes de texto que se analizarán se clasifican según la forma en que se obtienen. En líneas generales, existen dos vertientes: utilizar datos previamente extraídos y organizados por otras personas, principalmente destinados al entrenamiento de modelos; o extraer los datos por cuenta propia, específicamente para el análisis.

La primera opción, que implica el uso de conjuntos de datos previamente preparados por terceros, marcará la pauta para el formato de datos utilizado en el proyecto, ya que es fundamental que este formato esté unificado. En caso de no optar por este tipo de conjunto de datos, se recurrirá a una técnica comúnmente empleada para la recolección de datos: el *web scraping* en grupos de Telegram dedicados a la ansiedad. Estos datos extraídos, en forma de

mensajes de texto, requerirán un procesamiento adecuado antes de su análisis posterior, adaptándolos al formato requerido.

### 3.1.1. Formato de entrada

Es necesario comprender el formato en el que se reciben los datos, es decir, los mensajes de texto que serán objeto de análisis. Estos mensajes deben someterse a un proceso de formateo para garantizar que coincidan con el formato de los datos de entrenamiento disponibles. En este trabajo se aborda la tarea de procesar mensajes provenientes de grupos de Telegram, los cuales llegan sin ningún tipo de etiquetado predefinido. Esta falta de etiquetado implica que los modelos de análisis que se implementen posteriormente deberán asumir la responsabilidad de categorizar o etiquetar estos mensajes según corresponda.

Ahora bien, sin adentrarse en detalles técnicos de programación, es importante comprender cómo se estructuran estos mensajes dentro del entorno de Telegram. El proceso de obtención de mensajes desde un grupo implica la creación de un objeto de tipo 'Messages', que comúnmente se representa como una lista de mensajes. Cada uno de estos mensajes, a su vez, se representa mediante otro objeto de tipo 'Message', el cual contiene información detallada sobre el contenido, la fecha, la hora y otros metadatos relevantes. Este enfoque estructurado facilita el análisis y la manipulación de los mensajes dentro del contexto del estudio.

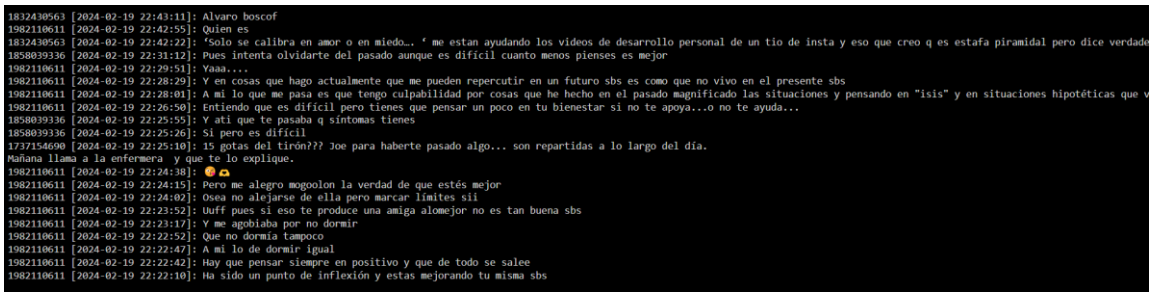



**Figura 6:** Formato de entrada mensajes Telegram

Por lo tanto, para poder realizar un análisis efectivo de estos mensajes de texto, es necesario comprender la estructura y el formato en los que se presentan inicialmente. Solo con esta comprensión podremos desarrollar y aplicar adecuadamente los modelos y algoritmos de análisis requeridos para extraer información significativa de los datos disponibles.

### 3.1.2. Formato de salida

Como se ha mencionado anteriormente, el punto de partida es un corpus de datos formateado compuesto por archivos JSON, cada uno de los cuales representa las interacciones de un sujeto en un grupo de Telegram específico. Estos archivos sirven como la base sobre la cual se llevará a cabo el análisis de los mensajes de texto.

Cada archivo JSON en este corpus de datos, tal y como se puede observar en la Figura 7, contiene tres campos fundamentales que son esenciales para el análisis posterior: el identificador único del mensaje ("id_message"), el contenido del mensaje ("message") y la fecha y hora en que el mensaje fue emitido ("date"). Estos campos proporcionan la información necesaria para interpretar el contenido y el contexto temporal de cada mensaje dentro del grupo de Telegram.


```
[
    {
        "id_message": 51299246790,
        "message": "Buenos días a tod @s ! ! No sé si esta será la finalidad del grupo . Pero cuando estás en plena crisis buscas recursos por todas partes cara desesperada . Estoy yendo a terapia y me diagnosticaron TAG ( trastorno de ansiedad generalizada ) desde hace mucho tiempo . El problema es que cuando me dan esas crisis de no poder parar ni un segundo mis pensamientos acabo agotada y ya no sé si con depresión . Quería saber si a alguno de vosotros también os pasa algo parecido . Por poner un ejemplo : estoy estudiando una carrera y cuando voy a ponerme a estudiar con ganas , me boicotea mi pensamiento del : total para que ? Y me empieza la angustia , los nervios , la desesperacion ... y así con todo . Os pasa algo parecido ? Habéis logrado vencerlo ? Gracias por leerme !",
        "date": "2021-05-06 10:58:48"
    },
    {
        "id_message": 7820167190,
        "message": "Gracias a los 2 ! Tranquiliza saber que no estamos solos en esto cara feliz con ojos sonrientes bíceps flexionado tono de piel claro medio",
        "date": "2021-05-06 13:01:27"
    },
    {
        "id_message": 79467366390,
        "message": "No me había dado cuenta . Voy a echarle un vistazo , gracias cara feliz con ojos sonrientes",
        "date": "2021-05-06 13:05:20"
    },
    {
        "id_message": 63297881224,
        "message": "Si , a veces persisten y persisten y te bloquean",
        "date": "2021-05-06 19:56:47"
    },
```


**Figura 7:** Formato de salida mensajes Telegram (Formato JSON)

Por consiguiente, al momento de disponer los datos extraídos del grupo de Telegram en el formato de entrada indicado, es necesario que se reconfiguren los datos para que se ajusten al formato JSON especificado, manteniendo los mismos campos mencionados, que capturan la información más relevante de cada mensaje. Esta reconversión garantiza la consistencia y la coherencia en la estructura de los datos, lo que facilita su posterior procesamiento y análisis por parte de los modelos y algoritmos diseñados para este fin, aprovechando al máximo la riqueza de información contenida en los mensajes de texto y realizando un análisis exhaustivo y significativo de las interacciones dentro del grupo.

## 3.2. Procesamiento de datos

El procesamiento de datos es vital en el análisis de la información, donde la abundancia de datos experimenta una gestión eficiente y efectiva. Este proceso implica la transformación de datos crudos en información significativa y útil, a través de una serie de operaciones que incluyen la recolección, almacenamiento, manipulación y análisis de datos. Con el avance de la tecnología, el procesamiento de datos ha evolucionado exponencialmente, permitiendo aprovechar grandes volúmenes de datos para obtener *insights* valiosos y tomar decisiones informadas. En este contexto, es importante comprender los principios, métodos y herramientas del procesamiento de datos para aprovechar al máximo su potencial y generar valor añadido en diferentes campos de aplicación.

### 3.2.1. Adquisición y extracción de datos

Al abordar el tema de los datos a utilizar, se puntualizó la existencia previa de un corpus de datos ya formateado e incluso etiquetado, diseñado específicamente para el entrenamiento de los modelos. Sin embargo, también se mencionó la necesidad de extraer datos adicionales de grupos de Telegram, los cuales no están formateados ni etiquetados previamente. Estos datos sin procesar serán sometidos a un proceso de transformación y análisis mediante los modelos que se implementarán más adelante. En consecuencia, para utilizar eficazmente estos datos, es imprescindible realizar su extracción de los grupos de Telegram.

Uno de estos grupos, denominado "Aprendiendo a vivir con la ansiedad" y con el identificador "@enluchaconstante" [24], será la principal fuente de datos para un posterior análisis. Este grupo está dirigido a personas diagnosticadas con trastorno de ansiedad y ofrece un espacio para compartir experiencias y recursos de ayuda. Está moderado por bots y administradores que mantienen el orden y expulsan a usuarios que incumplen las normas.

Además, se identificó otro grupo en línea llamado "Superando la ansiedad", con el identificador "@superando_la_ansiedad", también dedicado al mismo propósito. Este grupo fue descubierto a través de la investigación en foros de Internet sobre grupos de Telegram para la ansiedad. A diferencia del primero, este grupo cuenta con la participación ocasional de psicólogos y ofrece un entorno donde los usuarios pueden buscar apoyo profesional de forma privada. Sin embargo, extraer mensajes de este grupo presenta dificultades debido a la

eliminación periódica de mensajes (cada cierto número de días o semanas) y la restricción en el acceso a los identificadores de los usuarios, manteniendo en mayor medida el anonimato de los mismos.

Al abordar la política de privacidad de Telegram y los aspectos éticos de extraer mensajes de estos grupos, es esencial proteger la privacidad de los usuarios. Para mantener este anonimato, se han utilizado identificadores en lugar de nombres reales, preservando los datos personales de los participantes. Sin embargo, es importante considerar el consentimiento informado y los posibles riesgos éticos, especialmente en grupos donde los usuarios pueden ser vulnerables. Se debe abordar con cuidado y reflexión, asegurando el respeto hacia la privacidad y el bienestar emocional de los participantes.

En cuanto al proceso extracción de datos, se requerirá realizar *web scraping* de los grupos de Telegram inicialmente seleccionados. El **web scraping** [25] es un método automatizado utilizado para extraer grandes cantidades de datos de sitios web de manera rápida. Aunque su definición está asociada principalmente a los sitios web, este concepto puede aplicarse de manera análoga para extraer mensajes automáticamente de un grupo de Telegram. En este caso, en lugar de analizar HTML, se utilizarán técnicas específicas para interactuar con la API de Telegram.

Para este propósito, se implementará un enfoque similar al utilizado para sitios web, pero adaptado específicamente a la extracción de mensajes de los grupos de Telegram. Se creará un **crawler** [26], también conocido como rastreador, indexador o araña, que es un pequeño programa informático diseñado para analizar páginas web de forma automática. Su función principal es seguir los enlaces encontrados en cada sitio web y almacenar una copia de todo el contenido en sus bases de datos. Por lo tanto, se desarrollará un programa o script informático que, utilizando el identificador del grupo, extraiga y almacene en ficheros los mensajes de los usuarios deseados. Es un gran desafío debido a las restricciones de acceso a la API de Telegram y a sus estrictas políticas de privacidad. Aunque existen métodos para obtener estos datos, no todos son éticos ni están permitidos por Telegram. En este caso, se buscará evitar el uso de prácticas invasivas y respetar las políticas de privacidad establecidas por la plataforma.

Desde una perspectiva técnica, el *crawler* se ha desarrollado utilizando el lenguaje de programación Python en el entorno de desarrollo Visual Studio Code, con pruebas realizadas

durante el desarrollo en el entorno de Google Colab. La utilización de esta dualidad de entornos facilita el proceso de desarrollo. Esto se debe a que, mientras que VS Code ofrece versatilidad y la integración de herramientas de control de versiones para hacer un seguimiento de los cambios, la utilización de Google Colab como plataforma de desarrollo alternativa para realizar ciertas pruebas de código ofrece ventajas significativas. Por ejemplo, proporciona un entorno de ejecución en la nube con acceso a recursos computacionales sin necesidad de configuración local.

Este sistema de extracción de datos generado permite recopilar mensajes y otros datos relevantes de los grupos seleccionados, lo que posibilita el análisis y estudio de las interacciones dentro de dichos grupos. La aplicación de este enfoque tecnológico permite automatizar el proceso de extracción de datos de manera eficiente y escalable, lo que contribuye significativamente al desarrollo de investigaciones y análisis en diversas áreas de estudio, abriendo la puerta a un análisis más profundo y exhaustivo de los datos.

La API de Telegram ofrece la posibilidad de interactuar con los mensajes y datos de los grupos. A través de un script, se establecerá una conexión con la API de Telegram para extraer los datos necesarios. Es importante destacar que este proceso requiere permisos y autorizaciones adecuadas. Los pasos básicos para llevar a cabo este proceso son los siguientes:

1. Crear una aplicación en Telegram: para utilizar la API de Telegram, es necesario crear una aplicación en el sitio web oficial de Telegram. Para ello, se inicia sesión en https://my.telegram.org/auth con una cuenta propia de Telegram y se sigue las instrucciones para crear una nueva aplicación. Al completar este proceso, se recibirá un ID de aplicación y una clave de API (API hash) que serán necesarios para la autenticación.
2. Instalar una biblioteca cliente de Telegram**:** para simplificar la interacción con la API de Telegram, se recomienda utilizar una biblioteca ‘cliente’. Existen varias opciones disponibles para diferentes lenguajes de programación. Dado que se va a trabajar con Python, se utilizará la biblioteca *Telethon*, aunque siempre se puede buscar la biblioteca más adecuada para cada caso específico.
3. Autenticación con la API: para autenticarse con la API de Telegram, se utiliza el ID de la aplicación y la clave de API obtenida al crear la aplicación. Dependiendo de la biblioteca que se utilice, habrá métodos específicos para la autenticación. Por

lo general, esto implica proporcionar el ID de la aplicación, el número de teléfono personal y el código de verificación que se recibirá a través de un mensaje de Telegram.

4. Acceder al grupo y extraer datos: una vez realizada la autenticación, se puede usar métodos proporcionados por la biblioteca para acceder al grupo del que se desea extraer datos. Se puede recuperar mensajes, información del grupo, detalles de los usuarios, entre otros datos, según las necesidades específicas requeridas. La forma exacta de hacer esto variará según la biblioteca que se esté utilizando, así que será necesario consultar la documentación de la biblioteca para obtener más detalles.
5. Procesar y almacenar los datos: tras haber recuperado los datos del grupo, podrán ser procesados según sea necesario y ser almacenados en una base de datos u otro formato para su posterior análisis o uso.

Una vez contextualizados los pasos necesarios para generar el script de extracción de datos del grupo de Telegram, procedemos a entrar en más detalle en este proceso, abordando minuciosamente los distintos aspectos que involucra.

En el primer paso, que implica la creación de una aplicación en Telegram para obtener las credenciales de acceso a la API, se inicia sesión con una cuenta de Telegram propia en la URL proporcionada. Esto concede acceso al portal de desarrollo de Telegram, donde se pueden crear y administrar aplicaciones. Al crear la aplicación en este portal, se completan ciertos detalles como el nombre, la plataforma de ejecución (en este caso, "Escritorio" para una aplicación de escritorio en Windows, Linux o macOS), una descripción, entre otros.

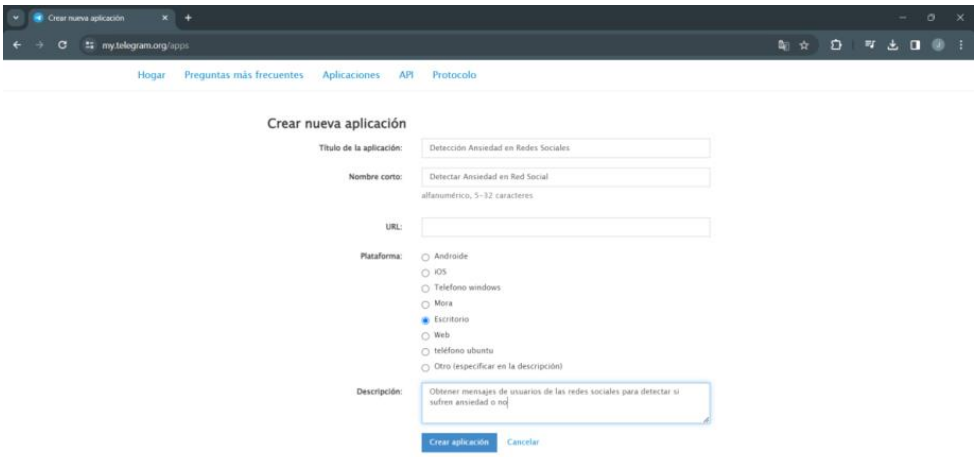



**Figura 8:** Creación Aplicación Telegram

Después de completar todos los detalles de la aplicación, se recibirán las credenciales necesarias para acceder a la API. Estas credenciales generalmente consisten en un ID de aplicación y una clave de API (API hash). Es fundamental almacenar estas credenciales de forma segura para autenticarse cuando sea necesario. De esta manera, se podrá interactuar con la API, recordando siempre revisar y cumplir con los términos de servicio de Telegram en todo momento.

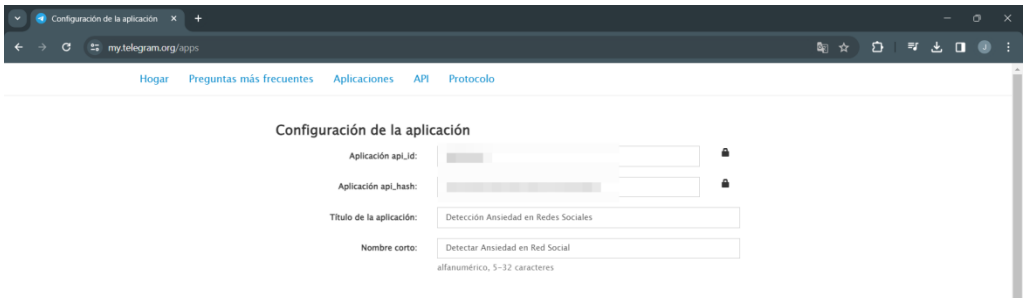



**Figura 9:** Credenciales acceso API Telegram

Tras completar el primer paso de creación de la aplicación web, avanzamos con los siguientes pasos orientados a la implementación del script de extracción de mensajes del grupo de Telegram. En el segundo paso, se evalúa la biblioteca cliente que se utilizará para facilitar la interacción con la API de Telegram, específicamente la biblioteca *Telethon* diseñada para Python.

Las bibliotecas cliente de Telegram son conjuntos de herramientas desarrolladas por la comunidad que simplifican la interacción con la API de Telegram desde diferentes lenguajes de programación. Estas bibliotecas encapsulan la complejidad de las solicitudes HTTP, la gestión de la autenticación y el procesamiento de respuestas, permitiendo concentrarse en la lógica específica de la aplicación en lugar de preocuparse por los detalles de bajo nivel. Para nuestro proyecto en Python, *Telethon* es un ejemplo destacado de estas bibliotecas, ofreciendo una amplia gama de funcionalidades y siendo ampliamente utilizada en la comunidad de desarrollo de Telegram.

Para utilizar una biblioteca cliente, primero se debe instalar en el entorno de desarrollo, usualmente a través de un gestor de paquetes como ‘pip’ en la terminal de VS Code con el comando “pip install telethon”. Luego, se importa en el proyecto para acceder a sus clases y métodos y así interactuar con la API de Telegram. En caso de dudas, la documentación [27] de esta biblioteca proporcionará más detalles y orientación específica.

Para el siguiente paso, que se centra en la autenticación en la API de Telegram, la biblioteca *Telethon* proporciona un método específico. Antes de proceder, es importante mencionar la presencia de dos variables globales que representan el ID de la aplicación ('api_id') y la clave secreta de la aplicación ('api_hash'). Se utiliza el método "TelegramClient()" con los parámetros que incluyen el nombre de la sesión para almacenar los datos de la sesión ('session_name') y las dos variables globales mencionadas anteriormente. Esta conexión con el cliente de Telegram se establece de manera asíncrona, lo que permite realizar múltiples tareas simultáneamente y asegura que la aplicación sea escalable. Dentro de este contexto creado, se pueden realizar diversas operaciones como enviar mensajes, obtener información de usuarios, entre otras. Al finalizar el mismo, la biblioteca *Telethon* se encarga automáticamente de cerrar la conexión correctamente.

Esto nos lleva al penúltimo paso del proceso, que implica acceder al grupo y extraer los datos necesarios. Es importante destacar que, para obtener información de un grupo en Telegram, es necesario ser miembro del mismo. Por lo tanto, para evitar complicaciones, se recomienda unirse a los grupos mencionados.

Con el objetivo de hacer el script lo más general posible y reutilizable para diferentes grupos o usuarios, se procede a la extracción de sus identificadores numéricos al comienzo del script. Para el grupo de Telegram, se obtiene a partir del identificador textual del grupo (sin el símbolo '@'), mientras que para el usuario se utiliza su nombre en el grupo. Es necesario revisar el listado de miembros del grupo en Telegram para determinar qué usuario se desea analizar.

Además, se incluyó la opción extra que permite listar todos los usuarios y seleccionar manualmente el identificador del nombre de usuario deseado a partir de la salida del entorno de desarrollo. Una vez obtenidos estos identificadores, se almacenan en variables globales para su uso posterior en el script.

En el ámbito de la programación, es relevante comentar que para extraer el identificador numérico del grupo de Telegram se utiliza el método "get_entity()" a partir del cliente creado previamente, pasando como parámetro la variable que contiene el identificador textual del grupo. Este método permite solicitar información específica sobre una entidad en Telegram, como un usuario o un grupo. Es importante señalar que esta operación se realiza de manera asíncrona, al igual que la creación del cliente.

De manera similar, para obtener el identificador del usuario del cual se desean extraer los mensajes, se obtiene primero el listado de participantes del grupo de Telegram utilizando el método "get_participants()" del cliente creado. Este método se invoca pasando como parámetro el identificador numérico del grupo obtenido anteriormente. A partir del listado de participantes, se verifica cuál coincide con el nombre del usuario deseado, y se almacena su identificador. En caso de realizar esta extracción a partir de la selección manual del identificador desde la salida del entorno de programación, el proceso es similar, pero en lugar de comparar nombres, se registra directamente el identificador proporcionado por el usuario que interactúa con el script.

Una vez se tienen todos los datos necesarios para extraer los mensajes de un usuario en un grupo de Telegram, se procede con la extracción. Antes de iniciarla, es obligatorio indicar el número total de mensajes a extraer del usuario con el identificador proporcionado. Se recomienda recopilar alrededor de 50 mensajes para obtener una muestra considerable y variada de sus intervenciones en el grupo. Para evitar una ejecución indefinida, se implementa una interrupción si la ejecución excede los 5 minutos, garantizando así un proceso controlado y eficiente. Esto previene situaciones en las que, debido a la gran cantidad de mensajes a revisar, no se encuentre el número total solicitado o no existan tantos mensajes del usuario en el grupo de Telegram. Los métodos utilizados se ejecutan de manera asíncrona para gestionar eficientemente estas situaciones.

Se implementan dos funciones destinadas a ser utilizadas posteriormente. La primera tiene como objetivo obtener los mensajes más recientes del grupo. Para ello, se utiliza el método "get_messages()" del cliente creado, pasando como parámetros el identificador numérico del grupo y el número de mensajes a extraer. La segunda función se encarga de formatear los mensajes en formato JSON, incluyendo los campos mencionados al inicio del apartado. Cabe destacar que, para realizar el formateo de la fecha y la hora al horario de España, ha sido necesario usar la función "timezone" de la librería importada ‘pytz’, realizando dicha importación mediante el gestor de paquetes correspondiente.

El núcleo de la implementación consistirá en un bucle en ejecución hasta que se obtenga el número deseado de mensajes de texto del usuario o se produzca una interrupción. En cada iteración, se verificará si el mensaje corresponde al usuario especificado. En caso afirmativo, se formateará el mensaje y se almacenará en una lista para su posterior escritura

en el archivo. Para evitar revisar constantemente los mismos mensajes, se comprobará el último mensaje del grupo revisado para determinar cuáles son los siguientes a extraer.

El último paso de este proceso implica almacenar los datos formateados obtenidos en archivos para su posterior exportación y uso en el análisis del proyecto. Estos archivos serán guardados en el directorio raíz del disco local del ordenador, lo que facilitará el acceso y almacenamiento independientemente de la ejecución. Específicamente, se almacenarán en un directorio denominado 'Ficheros sobre Ansiedad'. En caso de que este directorio no exista al ejecutar el script, se creará automáticamente para almacenar los ficheros de datos extraídos. Luego, el contenido del archivo, que consiste en el conjunto de mensajes formateados en formato JSON, se escribirá en el archivo utilizando el modo de escritura ('w').

### 3.2.2. Preprocesamiento del texto

Después de extraer los datos con los que se trabajará, es necesario realizar un proceso de preprocesamiento del texto para su posterior análisis. El preprocesamiento es una técnica fundamental en el procesamiento del lenguaje natural, ya que permite preparar adecuadamente los datos con el fin de simplificar y optimizar su manipulación por parte de los algoritmos de análisis.

Para realizar este proceso de preprocesamiento y el análisis subsiguiente, se ha empleado la librería NLTK y spaCy. Después de una comparación de rendimiento con otras librerías, se determinó que ambas eran opciones adecuadas que proporcionaban resultados óptimos. Su instalación es simple, solo requiere ejecutar "pip install NLTK" y "pip install spacy" en la terminal del entorno de desarrollo. Una vez instaladas, es posible desarrollar una función que realice el preprocesamiento de los datos utilizando las herramientas de ambas bibliotecas previamente importadas y descargadas. Esta función transformará el texto en su forma preprocesada, lista para ser evaluada en términos de polaridad por otra función. Es importante mencionar que, para la biblioteca spaCy, es necesario incorporar un modelo específico para el idioma español, capacitado para una variedad de tareas en dicho idioma. En este caso, se optó por la extensión "lg", que corresponde a una versión grande del modelo, lo que implica una mayor cantidad de datos y capacidades, mejorando así la precisión y comprensión del contexto.

En el preprocesamiento del texto, existen numerosos mecanismos que se pueden utilizar para adaptar el conjunto de datos a las necesidades específicas del experimento. Estos mecanismos pueden ser utilizados de manera encadenada, siguiendo una serie de pasos lógicos visualizados en la Figura 10.

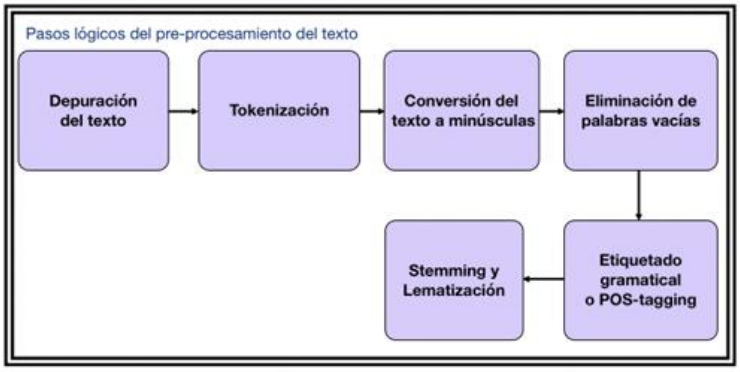



**Figura 10:** Pasos en el pre-procesamiento del texto

A continuación, se detallan algunas de las técnicas más comunes:

- Depuración del texto

  Los mensajes de las redes sociales, en este caso Telegram, a menudo contienen elementos particulares que requieren un tratamiento especial, como menciones, palabras mal escritas o emoticonos. Este proceso de limpieza del texto implica revisar todos los componentes para eliminar o transformar ciertos caracteres o elementos. Esto asegura que los datos estén adecuadamente preparados para que los algoritmos de extracción de características puedan trabajar con ellos de manera efectiva.

  El proceso de limpieza o depuración del texto se divide en dos fases. La primera fase implica la transformación de los emoticonos utilizados por los usuarios en texto y la eliminación de letras repetidas adicionales en una palabra (por ejemplo, modificar la palabra "tooooooonto" por "tonto"), lo que mejora la legibilidad del texto completo. La segunda fase consiste en eliminar aquellos elementos del texto que no aporten información relevante al análisis, como enlaces, menciones, entre otros.

- Conversión de Emoticonos a Texto Descriptivo

  Con respecto a los emoticonos, estos son pequeños dibujos que han evolucionado desde simples combinaciones de signos ortográficos, como por ejemplo :) o :(, hasta convertirse en dibujos gráficos a lo largo del siglo XXI. Estos iconos han adquirido una gran relevancia en las redes sociales debido a su simplicidad y capacidad para comunicar ideas de manera rápida y efectiva. Además, su universalidad los hace comprensibles para personas de diversas culturas y lenguas. Aunque podría considerarse prescindir de ellos, se ha optado por mantenerlos con el fin de aprovechar su capacidad para enriquecer la interpretación del texto.

  El proceso de transformación de emoticonos a texto se realiza utilizando la librería de Python llamada ‘emoji’, que proporciona herramientas específicas para trabajar con emoticonos. Por cada archivo de datos almacenado en el directorio del disco local, se aplica el método “demojize()” de la librería importada, el cual convierte los emoticonos en texto descriptivo. Sin embargo, es importante tener en cuenta que este texto resultante puede incluir diversos signos de puntuación y está en inglés. Por tanto, se procede a ajustar el formato y traducir estas descripciones al español. Para llevar a cabo la traducción, se importa otra librería al entorno de desarrollo, que permite la traducción de palabras utilizando Google Translate. Esto se realiza mediante el comando “pip install googletrans==4.0.0-rc1”.

- Corrección ortográfica

  Posteriormente, se lleva a cabo una tarea de transformación de las palabras para lograr trabajar con términos en su forma "canónica". Sin embargo, es importante señalar ciertos aspectos. Tras un período prolongado de implementación para desarrollar esta funcionalidad, se han enfrentado diversas dificultades que han llevado a tomar decisiones sobre cómo abordar este aspecto en la programación.

  Al trabajar con datos extraídos de mensajes de Telegram, nos encontramos con mucho ruido en el texto debido a que hay numerosas palabras mal escritas. Esto se debe a que algunas personas tienen deficiencias en cuanto a sus habilidades

ortográficas y a que la forma de expresarse en las redes sociales, influida por la constante necesidad de teclear velozmente, tiende a incluir abreviaturas o alargar ciertas palabras, escribiendo letras repetidas, sobre todo vocales. Este tipo de errores de escritura dificultan en gran medida el procesamiento del mensaje y la comprensión del contexto.

Existen algunos métodos para mitigar este problema, como por ejemplo, una REST API desarrollada por Microsoft que puede ser usada en Python para corregir potencialmente la ortografía de las palabras. Sin embargo, requiere una subscripción para su utilización, por lo que se descarta dicha opción. También existen varias bibliotecas y herramientas en Python que proporcionan capacidades de autocorrección. Una de las bibliotecas más populares es `autocorrect`, que es una implementación simple de autocorrección basada en estadísticas de frecuencia de palabras. Sin embargo, tiene limitaciones y puede no ser capaz de corregir todos los errores ortográficos de manera precisa.

Es por ello que, en vez de corregir todos los errores ortográficos en los datos, se centrará la corrección en las palabras con varias letras repetidas. Para lograrlo, se recorre de igual manera el texto de los archivos en busca de este tipo de palabras, eliminando las letras repetidas consecutivas y teniendo en cuenta los casos especiales que puedan existir. Sin embargo, cuando estas letras repetidas aparecen en medio de la palabra, surgen varias dificultades para detectar la forma real de la palabra. Por esta razón, se ha tomado la decisión de abordar este aspecto de la siguiente manera:

- Si las letras repetidas consecutivas están al principio o al final de la palabra, se reduce a una sola letra, excepto en el caso de "ll" o “www” (enlaces).
- Si las letras repetidas consecutivas están en medio de la palabra, su tratamiento es más complicado. Esto se debe a que existen palabras con letras repetidas consecutivas en su forma "canónica", lo que dificulta determinar cuándo se debe reducir el número de letras. Una opción para evitar reducir ese conjunto de letras a una sola sería mantener un listado de estas palabras y realizar una comprobación, pero esto sería muy ineficiente debido a la cantidad de

comprobaciones necesarias. Por ello, se decide que, si hay más de dos letras repetidas consecutivas, se reducirán a dos. Si solo hay dos, se mantienen como están. Además, debido a esta decisión, se omitirá la comprobación para los casos de doble "ll" o doble "rr", ya que en esos casos las letras repetidas no se eliminan.

Por consiguiente, para cada contenido del campo "message" del JSON, se corrigen las palabras con letras repetidas consecutivas y se sobrescribe el archivo leído del directorio del disco local.

- Eliminación de enlaces (URLs)

Es común que algunos mensajes del grupo de Telegram contengan un enlace a una página externa. Esto no sirve en el análisis, y por tanto, se elimina. Este proceso de eliminación de elementos especiales se puede logar eficientemente utilizando expresiones regulares en Python, las cuales ayudan a identificar los patrones que se desean eliminar o modificar. El módulo “re” de Python se especializa en el manejo de expresiones regulares. La función re.sub() es clave en este proceso, ya que reemplaza las coincidencias de un patrón con un texto específico. Recibe dos parámetros principalmente: el primero es el patrón que va a buscar y el segundo es por lo que la va a sustituir. En caso de dejar este segundo parámetro vacío, se procedería a eliminar aquellas cadenas que coincidan con el patrón indicado. El patrón es:

```
contenido_modificado = re.sub(r'(https?://|www\.)\S+?(?=\s|")', '', contenido)
```

A la hora de eliminar los enlaces, se tiene en cuenta que son enlaces validos aquellos que comienzan por “www.”, “https://” o “http://” y terminan con el primer espacio que se encuentre o la comilla final del campo JSON.

- Eliminación de saltos de línea

Los saltos de línea se codifican como ‘\n’ y pueden eliminarse para evitar problemas. Al extraer datos del chat de Telegram, este salto de línea se codifica textualmente con dicho formato, por lo que es necesario encontrar y eliminar esta coincidencia. El patrón es:

```
contenido_modificado = contenido.replace('\\n', '')
```

- Eliminación de correos electrónicos (emails)

  En el proceso de análisis, se decide eliminar los correos electrónicos del texto debido a su naturaleza como datos privados y a que no ofrecían información relevante para el análisis en cuestión. La protección de la privacidad es imprescindible y preservar la confidencialidad de la información personal es una prioridad ética. Al excluir estos datos sensibles, se pone el foco en los aspectos pertinentes del análisis. El patrón es:

```
contenido_modificado = re.sub(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-] +\.
[A-Z|a-z]{2,}\b', '', contenido)
```

- Eliminación de menciones a usuarios

  Las menciones se componen de un símbolo '@' seguido del nombre de un usuario, lo que activa una notificación para dicho usuario cuando se incluye en un mensaje del grupo. Además, se consideran posibles errores en las menciones, como aquellas que contienen un espacio entre el símbolo '@' y el nombre de usuario. Este proceso se lleva a cabo después de eliminar las direcciones de correo electrónico para evitar la eliminación parcial de mensajes que contienen emails, dejando la otra parte del texto sin procesar. El patrón es:

```
contenido_modificado = re.sub(r'@ ?\S+?(?=\s|'')', '', contenido)
```

- Eliminación de los signos de puntación

  Los signos de puntuación, en muchas ocasiones, no aportan información relevante al algoritmo, por lo que se procede a su eliminación. Para ello, se utiliza el módulo "string", en el cual se puede encontrar una lista de los signos de puntuación existentes a los que se les añadirá '¿' y '¡' debido a que la lista está construida para el idioma inglés. Es muy importante realizar esta fase después de la eliminación de enlaces, correos electrónicos y menciones, ya que

las tres contienen signos de puntuación que son necesarios para su identificación.

Para ello, se empleará el método ‘maketrans()’ para crear una tabla de traducción que se utilizará para eliminar signos de puntuación de las distintas cadenas del texto. Este método toma como argumentos dos cadenas de texto del mismo tamaño y una tercera cadena opcional. Estas cadenas deben tener la misma longitud y representan los caracteres que se traducirán, los caracteres con los que se traducirán y los caracteres que se eliminarán, respectivamente. En este caso, las tres cadenas están vacías, lo que significa que no se realizará ninguna traducción y simplemente se eliminarán los caracteres especificados en “string.punctuation”. Por tanto, el resultado es una tabla de traducción que eliminará todos los signos de puntuación de una cadena de texto cuando se aplique a través del método ‘translate()’.

- Tokenización

La **tokenización** es uno de los procesos más importantes en el PLN que implica dividir un texto en unidades más pequeñas llamadas "tokens". Estos tokens pueden ser palabras individuales, subpalabras, caracteres o incluso frases completas, dependiendo del nivel de granularidad requerido para una tarea específica.

Este proceso es clave para la tarea de PLN que aborda este trabajo, consistente en el análisis de sentimientos relacionados con la ansiedad. Al dividir el texto en tokens, se crea una representación estructurada que puede ser comprendida y manipulada más fácilmente por los ordenadores.

La implementación de este proceso de tokenización utiliza la librería NLTK, que ha sido importada previamente, y requiere la descarga de los recursos necesarios. Para cada uno de los archivos con datos en el directorio local, se extraerá el texto del campo “message” y se tokenizará utilizando el método “word_tokenize()”, el cual generará una lista de los tokens extraídos. También se puede tokenizar con Spacy llamando al método “nlp” del modelo con el texto que se desea tokenizar. Esto te devuelve un objeto “Doc” que contiene los tokens del texto, con sus respectivas propiedades como etiquetas POS, entidades nombradas, etc.

La forma con la que se realiza este proceso implica la selección de tokens de tamaño 1, lo que significa que se trata de palabras individuales. No obstante, en los textos existen expresiones que requieren ser consideradas como un conjunto para captar su significado contextual, como sucede con "libro de cocina" o "viaje de negocios", así como con algunos nombres propios de personas o lugares, como "País Vasco". Estas combinaciones tienen un significado específico que se pierde si se descomponen en palabras individuales, ya que su significado conjunto puede ser distinto al de las palabras por separado. Para abordar este desafío, se recurre a la técnica de los **N-gramas**, la cual consiste en tratar un número 'n' de palabras como una unidad indivisible (tokens).

- Conversión de mayúsculas en minúsculas

Esta estrategia se emplea para simplificar el texto, al convertir todas las letras del conjunto de datos a minúsculas. De esta manera, palabras que difieran únicamente en el uso de mayúsculas o minúsculas, como "Mesa", “MESA” y "mesa", pueden tratarse como equivalentes. Sin embargo, en ciertas situaciones, esta acción puede resultar en la pérdida de significado, como en el caso de cambiar "CIA" (Agencia Central de Inteligencia) a "cia", que se confunde con la palabra "compañía".

Para realizar esta conversión a minúscula, se hace uso de la función “.lower()”, la cual convierte a minúsculas todas las palabras del texto al que se aplique dicha función.

- Eliminación de palabras vacías *(stop words)*

Se trata de aquellas palabras que no aportan un valor semántico significativo, como las preposiciones, conjunciones, etc. A pesar de ello, estas palabras son frecuentes en el texto, lo que justifica su eliminación para reducir el número de palabras a analizar.

La librería NLTK proporciona una lista de palabras vacías para varios idiomas, incluyendo el español. Esta lista se importa y descarga como parte del paquete correspondiente. Una vez obtenida la lista para el idioma español, se compara cada

uno de los tokens en los campos 'message' de los archivos, eliminando aquellos que coincidan con algún elemento de la lista de palabras vacías.

- Etiquetado gramatical o POS-tagging

  Aunque para el propósito de este trabajo esta técnica tendrá poco impacto, en algunas ocasiones es necesario obtener el etiquetado gramatical de cada palabra, especialmente cuando se desea distinguir nombres propios de personas o lugares para evitar su análisis. Este etiquetado implica determinar la morfología de las palabras del texto, que están diferenciadas como tokens, utilizando en este caso la librería spaCy, la cual ofrece funcionalidades más fáciles de manejar para este objetivo.

- Stemming

  El **stemming** es un proceso utilizado en el procesamiento de lenguaje natural para reducir las palabras a su forma base o raíz, denominada "stem". Su objetivo principal es simplificar las palabras para que diferentes formas de una misma palabra se reduzcan a una única forma común. Por ejemplo, las palabras "corriendo", "corre", "correría" se reducirían al mismo stem "corr". Esto ayuda a mejorar la eficiencia en tareas como el análisis de sentimientos al agrupar palabras similares bajo una misma forma.

  Sin embargo, es importante tener en cuenta que el *stemming* puede producir resultados imperfectos, ya que puede generar “stems” que no son palabras válidas o puede no capturar adecuadamente las variaciones de sentido de algunas palabras. Además, el *stemming* no tiene en cuenta el contexto de las palabras, lo que puede llevar a reducciones inexactas en algunos casos. A pesar de estas limitaciones, el stemming sigue siendo una técnica útil en muchas aplicaciones de procesamiento de lenguaje natural.

  En la implementación, se emplea el algoritmo "SnowballStemmer", el cual es importado de la librería NLTK. Destaca por su eficiencia y precisión al reducir palabras a su forma raíz. Este algoritmo opera siguiendo un conjunto de reglas

predefinidas para el idioma español, diseñadas específicamente para eliminar sufijos comunes y derivar sistemáticamente la forma raíz de una palabra.

- Lematización

La **lematización** es otro proceso utilizado en el procesamiento de lenguaje natural para reducir las palabras a su forma base, conocida como "lema". A diferencia del *stemming*, que simplemente elimina los sufijos para obtener la raíz de una palabra, la lematización busca reducir las palabras a su forma canónica o de diccionario. Esto significa que, en lugar de solo truncar las palabras, la lematización tiene en cuenta la morfología y el significado de las palabras en el contexto de la oración.

Por ejemplo, mientras que el *stemming* puede reducir palabras como "corriendo" y "corre" a la misma forma "corr", la lematización las reduciría ambas a su lema común, que es "correr". Este enfoque más sofisticado permite una mayor precisión en la reducción de palabras, ya que considera la categoría gramatical y las reglas lingüísticas específicas de cada palabra. Sin embargo, la lematización tiende a ser más computacionalmente intensiva que el *stemming* debido a su complejidad, pero ofrece resultados más precisos, especialmente en contextos donde se requiere comprensión semántica más profunda.

La implementación de esta metodología se lleva a cabo utilizando el atributo "token.lemma_" de la librería spaCy. Este atributo devuelve el lema de un token, que es la forma base de una palabra. Al procesar un texto con spaCy, cada palabra se convierte en un token, y cada token tiene asociado un lema que representa la forma canónica de esa palabra.

Después de explicar cada una de las técnicas para realizar el preprocesamiento del texto, se incluye a continuación la Tabla 6, que ejemplifica la aplicación de estas técnicas:

| | |
|---|---|
| **Texto** | “Holaa chicoss! Esta última semana he tenido menos crisis de ansiedad\nOs dejo este enlace sobre el temaa 😁: https://www.youtube.com/watch?v=UxUybYT_WZo” |

| | |
|---|---|
| **Limpieza de texto** | "Hola chicos Esta última semana he tenido menos crisis de ansiedad Os dejo este enlace sobre el tema cara radiante con ojos sonrientes" |
| **Tokenización** | ["Hola", "chicos", "Esta", "última", "semana", "he", "tenido", "menos", "crisis", "de", "ansiedad", "Os", "dejo", "este", "enlace", "sobre", "el", "tema", "cara", "radiante", "con", "ojos", "sonrientes"] |
| **Conversión a minúsculas** | ["hola", "chicos", "esta", "última", "semana", "he", "tenido", "menos", "crisis", "de", "ansiedad", "os", "dejo", "este", "enlace", "sobre", "el", "tema", "cara", "radiante", "con", "ojos", "sonrientes"] |
| **Eliminación de palabras vacías** | ["hola", "chicos", "última", "semana", "menos", "crisis", "ansiedad", "dejo", "enlace", "tema", "cara", "radiante", "ojos", "sonrientes"] |
| **Estructura gramatical** | [["hola", "PROPN"], ["chicoss", "PROPN"], ["última", "ADJ"], ["semana", "NOUN"], ["menos", "ADV"], ["crisis", "NOUN"], ["ansiedad", "NOUN"], ["dejo", "VERB"], ["enlace", "NOUN"], ["tema", "NOUN"], ["cara", "VERB"], ["radiante", "ADJ"], ["ojos", "NOUN"], ["sonrientes", "ADJ"]] |
| **Stemming** | ["hol", "chic", "ultim", "seman", "men", "crisis", "ansied", "dej", "enlac", "tem", "car", "radiant", "ojos", "sonrient"] |
| **Lematización** | ["hola", "chicos", "último", "semana", "menos", "crisis", "ansiedad", "dejar", "enlace", "tema", "cara", "radiante", "ojo", "sonrisa"] |

**Tabla 6:** Ejemplo de Preprocesamiento del Texto

**Nota a destacar:** si bien en general se realizarán correcciones en los mensajes, algunos algoritmos y modelos pueden obtener conclusiones más precisas al utilizar datos sin procesar, aprovechando la variabilidad presente en el texto. Además, no es imprescindible aplicar todas las técnicas de preprocesamiento de texto a los datos de entrada. Por lo general, se realizará una limpieza inicial del texto, y luego, se ejecutará un preprocesamiento básico mediante tokenización, eliminación de palabras vacías, conversión de las palabras a minúscula y lematización.

### 3.2.3. Datos de entrenamiento y de prueba

La disponibilidad de un conjunto de datos previamente formateados y etiquetados proporciona una base sólida para el análisis de la ansiedad en los mensajes de los usuarios en los grupos de Telegram. No obstante, es importante señalar que en el ámbito de la ansiedad, encontrar conjuntos de datos en español tratados con anterioridad para el análisis no es una tarea sencilla. Esto se debe a que la investigación en este trastorno es relativamente reciente en comparación con otros trastornos mentales, y la mayoría de los conjuntos de datos disponibles están en inglés, que es el idioma principal en el que se suelen realizar los análisis en este campo. A menudo, estos conjuntos de datos incluyen sistemas de etiquetado asociados y son comúnmente utilizados como ejemplos en análisis de este tipo.

En este proyecto, se cuenta con un conjunto de datos con dichas características proporcionado por la tarea **MentalRiskES** del IberLEF 2023 [24], que establece el patrón de formato y etiquetado a seguir para mantener la coherencia en el análisis. Los datos de dicho corpus fueron recopilados del grupo de Telegram "Aprendiendo a vivir con la ansiedad" (‘@enluchaconstante’), también utilizado en este proyecto. Tras la compilación del conjunto de datos, se realizó un preprocesamiento similar al descrito en el apartado anterior, manteniendo los ficheros de los usuarios con entre 50 y 100 mensajes, y truncando a los más recientes cuando fue necesario.

En dicha tarea, se indica que para diseñar las pautas de anotación, se utilizaron definiciones y conceptos de la OMS[11] y la SEMI[12]. Además, se proporcionaron ejemplos de conversaciones, preguntas frecuentes y un esquema gráfico para facilitar la comprensión de las etiquetas. Las etiquetas para el trastorno de ansiedad, dado que el estudio también hace referencia a otros trastornos, son "Sufrir" (usuarios que experimentan situaciones relacionadas con la ansiedad) y "Control" (usuarios sin síntomas). Estas etiquetas se determinaron tras un análisis preliminar del corpus.

El proceso de anotación se llevó a cabo en la plataforma *Prolific*[13], estableciendo requisitos específicos para los anotadores: competencia en español, título universitario y edad entre 22 y 45 años, asegurando una representación equilibrada de género. Los anotadores

[11] Organización Mundial de la Salud (OMS)
[12] Sociedad Española de Medicina Interna (SEMI)
[13] https://www.prolific.com/

usaron la plataforma *Doccano*, y se desarrolló una herramienta para conectar *Prolific* y *Doccano*. Se realizó una prueba de un mes con 100 sujetos y diez anotadores para refinar las directrices y calcular el tiempo de etiquetado. Posteriormente, se llevaron a cabo varias tareas de anotación en *Prolific*, agrupando 100 usuarios por tarea.

De esta manera, se creó un corpus general con dos ficheros de texto con las etiquetas, uno denominado "gold_a" y otro "gold_b". El primero de ellos utiliza una codificación binaria, etiquetando a los usuarios con el valor "1" si se detecta ansiedad y "0" si no la presentan. Mientras que el segundo fichero adopta un enfoque más matizado, asignando a cada archivo JSON un valor numérico en un rango de 0 a 1, reflejando el grado de ansiedad experimentado por el usuario, donde 0 indica la ausencia total de ansiedad y 1 representa la presencia máxima de este trastorno.

Es importante destacar que ambos ficheros de etiquetas contienen la identificación de los usuarios, generándose una correspondencia con los archivos JSON que contienen los mensajes de los usuarios. Esta asociación de datos entre los mensajes y las etiquetas de ansiedad proporciona una oportunidad única para explorar las relaciones entre el contenido de los mensajes y el estado emocional de los individuos.

La división de los datos en conjuntos de entrenamiento y prueba se realizará de manera manual para generar un corpus de entrenamiento efectivo para la interfaz web, que puede recibir cualquier tipo de fichero externo. Este corpus se ha confeccionado mediante la recopilación de los ficheros de la tarea anterior que mejor resultado han mostrado al entrenar los modelos, tras evaluar sus predicciones. En cuanto a los ficheros de prueba, se ha realizado una selección de 11 ficheros, cada uno con un valor de etiquetación diferente en un rango entre 0 y 1, lo que permite una representación variada. De este modo, el modelo aprenderá patrones diversos de ansiedad y se podrá evaluar su efectividad para detectar la presencia o ausencia de ansiedad.

Además, también se utilizará un conjunto de datos sin etiquetar obtenido del mismo grupo de Telegram, pero de manera autónoma, para realizar pruebas finales una vez completadas todas las verificaciones previas con el corpus etiquetado. Esta etapa permitirá evaluar exhaustivamente el rendimiento de los modelos implementados y afinar cualquier ajuste necesario para lograr resultados óptimos.

La aplicación de esta estrategia fortalecerá la robustez y la capacidad de generalización del modelo ante datos no vistos durante el entrenamiento, contribuyendo así al desarrollo de un análisis preciso y generalizable de la ansiedad en los mensajes de usuarios. Además, se garantizará la integridad y la confidencialidad de los datos de los usuarios durante todo el proceso.

### 3.2.4. Vectorización de palabras

La detección de emociones en el texto, especialmente la ansiedad, no es directa, ya que las palabras que denotan emociones específicas no siempre están explícitas en el texto o, incluso si lo están, su significado puede ser ambiguo debido a la polisemia o la semántica contextual. Por ejemplo, el término "frío" puede referirse tanto a una baja temperatura ("hace frío afuera") como metafóricamente a una actitud emocional de distanciamiento o falta de afecto ("su respuesta fue fría").

Los métodos de aprendizaje automático requieren datos que puedan ser interpretados por los algoritmos, lo cual es difícil de lograr con datos de texto. Estos datos necesitan ser convertidos en representaciones numéricas comprensibles para los ordenadores.

Por consiguiente, tras las etapas iniciales de preprocesamiento del texto, que incluyen el formateo y la limpieza de los datos, en un clasificador de texto basado en aprendizaje automático que requiere el análisis de las palabras del texto, es necesario llevar a cabo un proceso de **vectorización**. Este proceso implica convertir las palabras en vectores numéricos y se conoce como **extracción de características**. Su objetivo es permitir que los algoritmos trabajen con los datos y reducir el conjunto de palabras a las más representativas para identificar el nivel de ansiedad en el texto.

Existen varias técnicas para representar numéricamente la información del texto de manera efectiva, entre ellas:

- Bolsa de Palabras o *Bag-of-Words (BoW)*

  La técnica de la **BoW** implica crear un inventario exhaustivo de todas las palabras presentes en un corpus lingüístico. Para cada texto en particular, se genera un vector numérico donde cada palabra del corpus se marca con un 1 si está presente en el texto y con un 0 si no lo está, incrementándose según el número de ocurrencias. En

esencia, se utiliza el concepto de corpus lingüístico, que abarca todo el conjunto de texto utilizado en el proyecto.

Sin embargo, esta técnica tiene limitaciones, ya que no considera el orden ni el contexto en el que aparece una palabra en el documento. Mantener este contexto es tiene gran importancia para el procesamiento de lenguaje natural, ya que puede influir significativamente en su significado y uso.

En resumen, esta técnica se utiliza para contar palabras, construyendo un vocabulario con las mismas. Un ejemplo ilustrativo sería:

- Disponer de un conjunto de textos:
    - texto1 = “El perro corre en la calle”
    - texto2 = “El gato juega en el jardín”
    - texto3 = “La casa está en la calle”
- Obtener un vocabulario formado por una lista de palabras distintas:
    - [‘el’, ‘perro’, ‘corre’, ‘en’, ‘la’, ‘calle’, ‘gato’, ‘juega’, ‘jardín’, ‘casa’, ‘está’]
- Transformar texto a formato vector (vectorización) y aplicar *BoW*:

| | ‘el’ | ‘perro’ | ‘corre’ | ‘en’ | ‘la’ | ‘calle’ | ‘gato’ | ‘juega’ | ‘jardín’ | ‘casa’ | ‘está’ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| texto1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| texto2 | 2 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| texto3 | 0 | 0 | 0 | 1 | 2 | 1 | 0 | 0 | 0 | 1 | 1 |

**Tabla 7:** Ejemplo de *Bag of Words*

- El texto vectorizado sería:
    - texto1 = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
    - texto2 = [2, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0]
    - texto3 = [0, 0, 0, 1, 2, 1, 0, 0, 0, 1, 1]

Para implementar esta técnica, se empleará el paquete 'sklearn', una potente biblioteca para el aprendizaje automático. Se creará una instancia de

'CountVectorizer' que se encuentra en 'sklearn.feature_extraction.text', con los parámetros de configuración iniciales. Utilizando el método 'fit_transform()' de esta clase, se procesa el conjunto de datos preprocesado para construir el array asociado. Este array tendrá como dimensiones el número de mensajes en el conjunto de datos y el número máximo de términos establecido previamente. Posteriormente, con el método 'transform()', se podrán obtener los arrays del texto pasado por parámetro a partir del entrenamiento previo. Esto significa que los nuevos mensajes se convertirán en vectores numéricos usando el mismo conjunto de términos que se estableció durante el entrenamiento inicial.

Al configurar la BoW, algunos parámetros se pueden establecer inicialmente, como por ejemplo, la exclusión de palabras que aparezcan en más del 70% de los mensajes del conjunto de datos (*max_df*=0.7). Los dos parámetros más importantes son el tamaño de la bolsa de palabras (*max_features*), que determina el número de palabras incluidas, y el número mínimo de veces que una palabra debe aparecer en el corpus para ser considerada en la construcción de la bolsa de palabras (*min_df*). Estos parámetros se ajustan durante la selección de hiperparámetros junto con el entrenamiento del modelo. Después de realizar diversas pruebas y evaluaciones, se ha determinado que, para el conjunto de datos utilizado, los parámetros más óptimos son los siguientes:

- ***stop_words*** *= None* → No especificar las palabras vacías *('stopwords')* en español, ya que se eliminan durante el preprocesamiento.
- ***min_df*** *= 3* → Considerar los términos que aparecen en al menos 3 documentos.
- ***max_df*** *= 1.0* → Descartar los términos que aparecen en más del 100% de los documentos, es decir, considerar todos los términos.
- ***max_features*** *= 2000* → Considerar los 2000 términos más frecuentes.
- ***ngram_range*** *= (1, 2)* → Estudiar unigramas y bigramas.

- *Term Frequency – Inverse Document Frequency (TF–IDF)*

Esta técnica puede ser empleada de forma independiente, aunque también puede ir ligada al uso de BoW. Tras generar los vectores para cada texto individual, se utiliza esta técnica que proporciona una métrica para evaluar la relevancia de cada término

en el documento en comparación con el conjunto total de documentos. Esta métrica se fundamenta en la relación entre el número de veces que un término aparece en un documento y el número de documentos en los que se encuentra.

La fórmula utilizada para calcular esta métrica es la siguiente:

$$tf - idf = \frac{count(palabras, texto) * NúmeroTotalTextos}{count(NúmeroTextos, palabra)}$$

**Figura 11:** Fórmula de TF-IDF

Por lo tanto, según esta fórmula, un valor elevado en esta medida indica una frecuencia significativa de un término dentro del documento, mientras que su frecuencia en el corpus general es baja. En otras palabras, el valor de TF-IDF aumenta proporcionalmente al número de veces que un término aparece en un documento y disminuye con su frecuencia en la colección de documentos, permitiendo distinguir palabras más comunes de otras, siendo siempre su valor mayor o igual a 0.

De esta manera, esta técnica también puede utilizarse para identificar las palabras vacías (*stopwords*). Estableciendo un umbral máximo para el valor de *IDF*, las palabras que superen dicho umbral se consideran demasiado comunes y, por consiguiente, no aportan un valor significativo en términos de sentimiento o emoción a los textos. En caso de optar por eliminar las *stopwords* durante el preprocesamiento del texto, este paso no será necesario durante la vectorización.

No obstante, a pesar de las mejoras introducidas en este modelo en comparación con el anterior, persisten problemas similares, como la falta de consideración de la semántica o el contexto, así como dificultades cuando se trata de conjuntos de datos grandes.

En cuanto a la implementación de esta técnica, se recurrirá nuevamente al mismo paquete. Sin embargo, en esta ocasión, se creará una instancia de 'TfidfVectorizer', también disponible en 'sklearn.feature_extraction.text', con los parámetros de configuración iniciales. Del mismo modo, se emplea el método 'fit_transform()' para generar el array asociado al conjunto de datos, que tendrá un tamaño igual al número de mensajes en el conjunto de datos y al número máximo de términos

previamente establecido. Además, mediante el método 'transform()', será posible obtener los arrays del texto pasado como parámetro, basándose en el entrenamiento previo.

Siguiendo una metodología similar, se pueden definir los parámetros de configuración para la vectorización TF-IDF. Después de varias pruebas durante el proceso de selección de hiperparámetros, se han identificado los siguientes parámetros óptimos para el conjunto de datos utilizado:

- ***stop_words*** *= None* → No especificar las palabras vacías *('stopwords')* en español, ya que se eliminan durante el preprocesamiento.
- ***min_df*** *= 2* → Considerar los términos que aparecen en al menos 2 documentos.
- ***max_df*** *= 1.0* → Descartar los términos que aparecen en más del 100% de los documentos, es decir, considerar todos los términos.
- ***max_features*** *= 2000* → Considerar los 2000 términos más frecuentes.
- ***ngram_range*** *= (1, 1)* → Estudiar exclusivamente unigramas.
- ***sublinear_tf*** *= False* → No aplicar transformación logarítmica a la frecuencia de términos.

- *Word Embeddings* o Representaciones distribuidas de palabras

A diferencia de enfoques anteriores que no consideraban el contexto de las palabras, esta metodología busca crear una representación de palabras donde aquellas con significados similares se asocien a vectores cercanos en un espacio vectorial de alta dimensionalidad. Durante el proceso de obtener esta representación numérica de las palabras, se realizan comparaciones entre ellas para identificar relaciones o similitudes conceptuales. Es decir, cada palabra se codifica en un vector que captura las relaciones y similitudes dentro del corpus lingüístico utilizado. Esto facilita la captura automática de variaciones en la puntuación, ortografía, etc., lo que contribuye a una comprensión más profunda del texto.

Este método presenta varias características interesantes:

- Semántica de palabras: captura la semántica de las palabras basándose en sus relaciones con otras palabras en el espacio.
- Relación entre palabras: pueden identificar relaciones lingüísticas entre palabras, como sinónimos o antónimos, o relaciones en el contexto.
    - Ejemplo: la diferencia entre los vectores “rey” y “reina” es parecida a la diferencia entre los vectores “hombre” y “mujer”.
- Operaciones algebraicas: los vectores pueden ser sujetos a operaciones algebraicas como sumas o restas para descubrir relaciones entre palabras.
    - Ejemplo: “Rey” - “Hombre” + “Mujer” produce un resultado aproximadamente igual a “Reina”.
- Generalización: tiene la capacidad de generalizar para palabras que no han sido observadas durante el entrenamiento.
    - Ejemplo: si el modelo ha aprendido la relación entre “perro” y “animal”, podrá inferir la relación “gato” y “animal”, incluso si esa relación no fue explícitamente enseñada durante el entrenamiento.

Existen numerosas técnicas o algoritmos [28] que se pueden aplicar en esta fase y su elección depende del objetivo que se persiga. Una estrategia para implementar este enfoque es representar las palabras en un sistema de coordenadas donde aquellas relacionadas se encuentren próximas entre sí, resaltando la técnica de **Word2vec**.

- Word2vec

  **Word2vec**, desarrollado por Google, emplea una red neuronal superficial para analizar un corpus de texto y tratar de predecir qué palabras serán vecinas, es decir, cuáles están asociadas o son similares. Este sistema cuenta con dos variantes arquitectónicas: continuous bag-of-words (CBOW) y continuous skip-gram. La selección de la arquitectura y la configuración de los parámetros dependen de la tarea específica a realizar. Los estudios han demostrado que CBOW produce resultados más rápidos, mientras que skip-gram se adapta mejor a las palabras menos comunes.

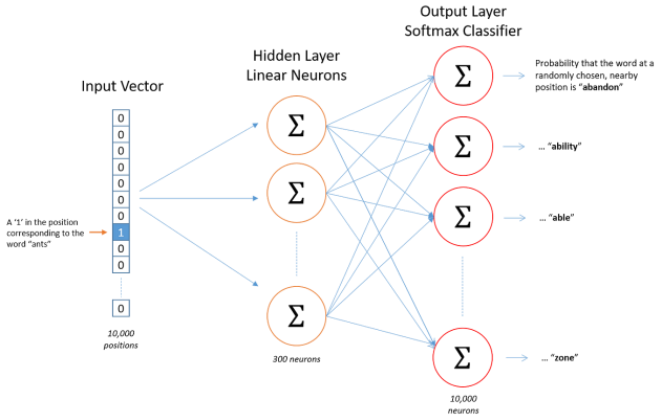



**Figura 12:** Red Neuronal *Word2vec*

Los vectores asociados a cada palabras [29] tienen una naturaleza binaria, donde cada entrada puede tener solo dos valores posibles: 0 o 1. Dichos valores pueden representar diversos conceptos según el contexto y el tipo de vector, como la presencia o ausencia de ciertas características, la activación de neuronas específicas en una red neuronal, o la pertenencia a una categoría determinada, entre otras interpretaciones. Tienen baja dimensión, por lo que se obtienen vectores de dimensiones 50 a 500. Dentro de los vectores, es posible distinguir:

- Un *vector denso*, aquel en el que la mayoría de las entradas son diferentes de cero.
- Un *vector disperso*, como la codificación one-hot, que tiene muchas entradas con valor cero.

Word2vec es un modelo que opera tomando un gran conjunto de datos como entrada y, a partir de los contextos en los que las palabras aparecen en este corpus, "aprende" a representarlas en un espacio vectorial común. Simplificando, asigna a cada palabra un vector inicial con valores aleatorios y luego ajusta estos vectores utilizando una red neuronal de dos capas y la información del contexto en el que cada palabra aparece.

Profundizando sobre las distintas variantes de Word2vec, se puede indicar que *CBOW* utiliza una ventana de palabras circundantes para predecir la palabra central, sin tener en cuenta el orden de las palabras en el contexto. Por otro lado, el modelo *Skip-Gram* toma la palabra central como entrada y trata de predecir las palabras circundantes en su ventana contextual. Esta variante pondera más

las palabras cercanas en el documento que las más alejadas. Aunque CBOW es más rápido, Skip Gram muestra un mejor rendimiento en la predicción de palabras poco frecuentes. Por lo tanto, es recomendable evaluar ambas arquitecturas para determinar cuál funciona mejor en nuestro experimento.

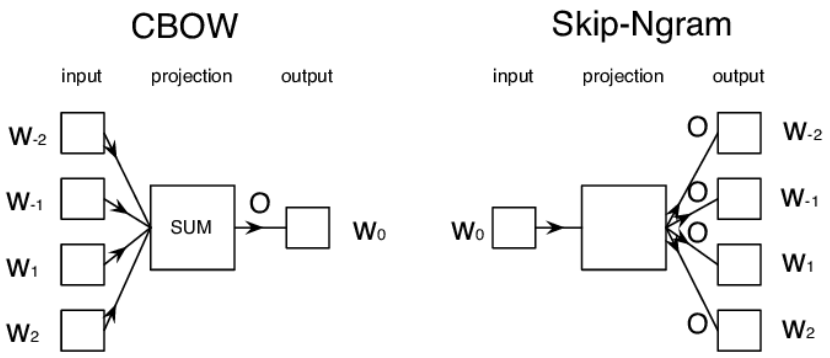



**Figura 13:** CBOW vs Skip-Gram

Para implementar Word Embedding con *Word2vec* se importará la librería 'gensim', la cual está preparada para el procesamiento de lenguaje natural. Se construirá un modelo 'Word2vec' procedente de 'gensim.models', el cual recibirá el conjunto de datos preprocesado, al igual que se hacía con anteriores modelos, junto a una configuración de parámetros inicial. Los parámetros a definir más importantes son el tamaño de los vectores de las palabras (*size*); la ventana máxima de contexto, la cual fija la cantidad máxima de palabras adyacentes a una palabra (*window*); el número mínimo de ocurrencias de una palabra en el corpus para ser considerada en la construcción de los vectores (*min_count*) y la cantidad de subprocesos utilizados para entrenar el modelo (*workers*). Para especificar la variante de esta metodología a utilizar, se asignará a la variable "sg" el valor 1 para utilizar la variante Skip Gram o el valor 0 para la variante CBOW.

Tras realizar diversas pruebas durante el proceso de selección de hiperparámetros, se han determinado los siguientes valores de parámetros óptimos para el conjunto de datos analizado:

- ***vector_size*** = *175* → Especificar la dimensión de los vectores de palabras que el modelo generará. Por ejemplo: 5, 100, 200…

- ***window*** *= 5* → Especificar la ventana máxima de contexto, es decir, la cantidad máxima de palabras adyacentes a una palabra. Por ejemplo: 3, 5, 7…
- ***min_count*** *= 2* → Establecer el umbral para determinar qué número de apariciones de una palabra son necesarias para contar con ella. Por ejemplo: 1, 2, 5…
- ***workers*** *= 4* → Especificar la cantidad de subprocesos/hilos usados para entrenar el modelo, marcando de esta manera la velocidad de entrenamiento.

## 3.3. Técnicas de clasificación supervisada

En el ámbito del aprendizaje supervisado, la clasificación se destaca por su enfoque en categorizar los datos en diversas clases específicas. Los algoritmos se dedican a asignar a qué categoría pertenecen los datos de entrada según sus características particulares.

El **aprendizaje supervisado** [30] implica trabajar con datos etiquetados, lo que significa que cada conjunto de datos está asociado con una etiqueta que indica su categoría. Este tipo de aprendizaje se esfuerza por identificar similitudes entre los datos que comparten la misma etiqueta, lo que permite distinguir entre los conjuntos con etiquetas diferentes. De esta manera, el objetivo es poder clasificar nuevas muestras que no tienen etiquetas asignadas, asignándoles una de las categorías disponibles.

Los conjuntos de datos de entrenamiento en aprendizaje supervisado tienen una estructura típica, que comprende:

- Dados unos datos $(x_1, y_1),\ldots, (x_n, y_n)$ donde $x = \{x_1, x_2,\ldots, x_D\}$
  - N es el número de ejemplos.
  - X representa las características siendo D las dimensiones.
  - Y representa las etiquetas.
- Un algoritmo de aprendizaje trata de obtener una función $G: X \rightarrow Y$
  - X es la entrada.
  - Y es la salida.

Algunos de los algoritmos más comunes en este tipo de aprendizaje, los cuales se implementarán en este proyecto, son:

- Support Vector Machines

**Support Vector Machines** [31], o Máquinas de Soporte Vectorial en español, es un algoritmo de aprendizaje supervisado desarrollado por Vladimir Vapnik. Este algoritmo se visualiza como puntos en un espacio, con el objetivo de crear un plano, conocido como hiperplano, para dividir estos puntos y generar particiones homogéneas entre clases. Se busca encontrar un hiperplano óptimo que maximice el margen entre él y los puntos más cercanos de cada clase. Se ilustra un ejemplo en la Figura 14, donde se muestran diferentes hiperplanos; observando que H1 y H2, aunque separan las clases, no lo hacen de manera óptima, mientras que H3 sí lo logra.

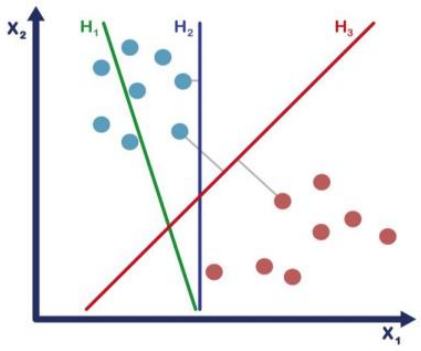



**Figura 14:** Diferentes hiperplanos en un SVM

El algoritmo SVM se refiere a los puntos más cercanos al hiperplano, donde cada clase debe tener al menos uno. Estos puntos de soporte permiten definir el hiperplano de máximo margen. Dado que el hiperplano puede no separar todos los puntos de una clase con precisión, el algoritmo introduce un parámetro *C* para compensar los errores durante el entrenamiento, permitiendo algunos errores de clasificación mientras los penaliza.

Aunque la forma más básica de crear un hiperplano es de manera lineal, esta puede no ser siempre la mejor solución para los datos. Debido a las limitaciones de las máquinas de aprendizaje lineal en problemas del mundo real, se recurre a soluciones como las funciones *kernel*, que proyectan los datos a espacios de alta dimensionalidad. Esto incrementa la capacidad computacional [32] de las máquinas de aprendizaje lineal. Dependiendo del tipo de *kernel* utilizado, se obtendrán diferentes hiperplanos.

Las SVM son destacadas por su facilidad de uso en comparación con las Redes Neuronales, y su capacidad para adaptarse bien a modelos relativamente complejos, así como su resistencia a los datos ruidosos. Sin embargo, tienen desventajas como la necesidad de probar diferentes *kernels* para encontrar el mejor hiperplano, lo que puede hacer que el entrenamiento del modelo sea lento.

- Naive Bayes

**Naive bayes** [33] es un algoritmo que se fundamenta en el Teorema de Bayes y los Métodos Bayesianos. Los clasificadores basados en estos principios utilizan los datos para calcular la probabilidad de que un dato de entrada pertenezca a una clase específica. Cuando se aplica a datos no etiquetados, el clasificador utiliza estas probabilidades para determinar la clase más probable.

Este tipo de clasificadores se emplea especialmente en problemas donde es necesario considerar múltiples atributos de información simultáneamente para estimar la probabilidad de un evento. Naive Bayes no es el único algoritmo que utiliza métodos Bayesianos para clasificar, pero es el más común en la clasificación de texto, convirtiéndose prácticamente en el estándar.

Existen varios tipos de algoritmos Naive Bayes, siendo dos de los más conocidos el Multinomial Naive Bayes y el Gaussian Naive Bayes, utilizados en la clasificación de textos, cada uno con su fórmula correspondiente.

Este algoritmo presenta diversas ventajas y desventajas que pueden influir en su uso. Entre sus fortalezas más destacadas se encuentran su rapidez y simplicidad, su capacidad para manejar datos ruidosos, la necesidad de pocos ejemplos para entrenar y la facilidad para obtener probabilidades estimadas para una predicción. Sin embargo, entre sus debilidades se incluye la suposición de que todas las clases son igualmente importantes, su subóptimo rendimiento con conjuntos de datos que contienen una gran cantidad de clases numéricas y la menor fiabilidad de las probabilidades estimadas en comparación con las clases predichas.

- Árboles de decisión

  Los **árboles de decisión** [34] son herramientas de clasificación que han sido ampliamente utilizadas en diversos campos científicos, como puede ser la biología computacional y la bioinformática. Estos clasificadores funcionan construyendo una serie de preguntas jerárquicas sobre las características de los datos, que guían la asignación de una etiqueta de clase a cada ítem.

  La construcción de un árbol de decisión comienza con el análisis de un conjunto de datos de entrenamiento con etiquetas de clase conocidas. A través de un proceso iterativo, se añaden nodos al árbol que plantean preguntas sobre los datos, dividiendo así el conjunto de datos en subconjuntos más homogéneos en términos de sus clases. Los nodos terminales, o hojas, indican la clase final asignada a cada ítem. Para evaluar la calidad de estas divisiones, se utilizan métricas como la entropía, que mide la pureza de los subconjuntos resultantes.

  Una de las ventajas de los árboles de decisión es su capacidad de ser interpretables, lo que los hace más accesibles que otros modelos como las redes neuronales o las máquinas de soporte vectorial. Sin embargo, presentan el riesgo de sobreajustarse a los datos de entrenamiento, por lo que se emplean técnicas como la poda del árbol para mejorar su generalización. Además, la robustez de los árboles de decisión se potencia mediante enfoques en conjunto, como los bosques aleatorios y el "*boosting"*, que combinan múltiples árboles para mejorar la precisión de las predicciones.

- Random Forest

  El algoritmo de **Random Forest** [35], desarrollado por Breiman y Cutler, es una técnica poderosa y popular en el ámbito del aprendizaje automático. Este método utiliza varios árboles de decisión para formar un modelo integral, aprovechando el método de *bagging* ("*bootstrap aggregating*") para generar diversos modelos a partir de los datos originales. El *bagging* implica la creación de múltiples conjuntos de datos de entrenamiento mediante el muestreo con reemplazo, los cuales se utilizan para entrenar modelos individuales. Posteriormente, las predicciones de

estos modelos se combinan mediante votación en el caso de clasificación. Se puede observar un ejemplo de este proceso en la Figura 15.

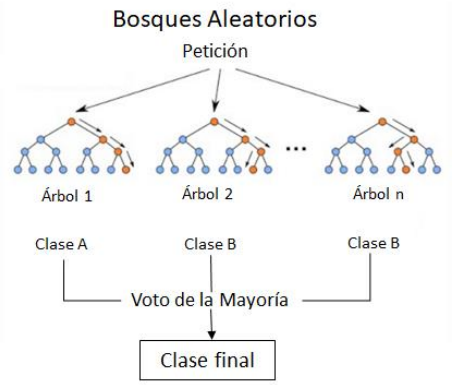



**Figura 15:** Ejemplo del funcionamiento de los bosques aleatorios

El algoritmo presenta varias ventajas y desventajas. Entre sus fortalezas destacan su capacidad para manejar datos ruidosos, su utilidad en conjuntos de datos con numerosas clases diferentes y su eficiencia para procesar grandes volúmenes de datos en poco tiempo. Además, proporciona una medida de la importancia de las características, lo que puede ser útil para interpretar el modelo y seleccionar características relevantes. Sin embargo, sus debilidades incluyen la falta de interpretabilidad y la dificultad para ajustarse a los datos de manera precisa. A pesar de estos inconvenientes, su robustez y capacidad para manejar conjuntos de datos grandes con alta dimensionalidad lo convierten en una de las herramientas más utilizadas por los científicos de datos y los practicantes de aprendizaje automático.

- K–vecinos *(KNN)*

**K-vecinos** [36] es una técnica popular en el ámbito del aprendizaje automático, reconocida por su simplicidad conceptual y su versatilidad en una variedad de aplicaciones. El principio fundamental de KNN es clasificar un objeto según la mayoría de las etiquetas de clase de sus vecinos más cercanos en un conjunto de datos de entrenamiento. Este enfoque intuitivo se basa en la idea de que los objetos con características similares tienden a pertenecer a la misma clase.

Su implementación implica una serie de pasos a seguir. Primero, se determina el número de vecinos (K) que se utilizarán para clasificar un nuevo objeto. Luego, se calcula la distancia entre el objeto de prueba y cada punto en el conjunto de datos de

entrenamiento, utilizando una métrica de distancia como la euclidiana o la distancia de Manhattan. Una vez que se han identificado los vecinos más cercanos, se utiliza un esquema de votación para asignar una etiqueta de clase al objeto de prueba, generalmente seleccionando la clase más común entre sus vecinos.

Sin embargo, la eficacia de KNN puede variar según varios factores. La elección del valor de *K* puede influir en la precisión del modelo, con valores bajos que pueden llevar a un sobreajuste y valores altos que pueden resultar en un subajuste. Además, la elección de la métrica de distancia puede afectar significativamente el rendimiento, especialmente en conjuntos de datos de alta dimensionalidad. Otras consideraciones incluyen la necesidad de normalizar los datos para evitar sesgos debido a las diferencias en la escala de las características. A pesar de sus limitaciones y sensibilidad a los parámetros, es una herramienta util gracias a su simplicidad y flexibilidad.

- Logistic Regression

El modelo de **Logistic Regression** [37] se utiliza para analizar la relación entre un conjunto de predictores y un resultado binario (con dos posibles valores). Este modelo se centra en las probabilidades de que ocurra un evento de interés específico. Se emplea el logaritmo natural de las probabilidades como una función de los predictores para establecer esta relación. En términos sencillos, el modelo describe cómo cambia la probabilidad de que ocurra el evento en función de los cambios en los predictores, proporcionando una medida llamada *odds ratio* que indica cuánto aumentan o disminuyen las probabilidades del evento con cada unidad adicional en el predictor.

El *odds ratio* es una herramienta común en el análisis de datos en estudios epidemiológicos y otras investigaciones científicas, aunque a veces se confunde con el riesgo relativo. Este ratio puede dar una impresión exagerada del efecto de un predictor si se interpreta incorrectamente. Sin embargo, en este modelo, se presenta habitualmente porque es la estimación más directa del modelo. Para entender mejor el impacto de los predictores, el *odds ratio* puede interpretarse como un porcentaje de cambio en las probabilidades del evento. Si se requieren ajustes adicionales, se pueden incorporar más predictores al modelo para refinar las estimaciones. Aunque

el proceso de encontrar las mejores estimaciones en Logistic Regression es más complejo que en regresión lineal y requiere herramientas computacionales, es ampliamente utilizado por su capacidad para manejar situaciones en las que el resultado es binario.

## 3.4. Técnicas de clasificación no supervisada

El **aprendizaje no supervisado** se refiere a trabajar con datos no etiquetados, es decir, donde no se dispone de información previa sobre el tipo de datos. Su objetivo es descubrir patrones o estructuras inherentes en los datos sin la guía de una salida deseada. Este enfoque busca identificar similitudes entre los datos para agruparlos en función de sus características compartidas. Los datos de entrada tienen la siguiente organización:

- Dados unos datos de entrada ($X_1 \ldots X_N$)
  - $X$ es un vector de D dimensiones o características.
  - N es el número de ejemplos.

Debido a su naturaleza, este tipo de aprendizaje se utiliza principalmente para la agrupación o compresión de datos. Algunos de los algoritmos más comunes son KNN (vecinos más cercanos), agrupación jerárquica, detección de anomalías, entre otros. Estos algoritmos son útiles si no se dispone de datos previamente etiquetados para el entrenamiento. Sin embargo, dado que sí se disponen de etiquetas y que los resultados obtenidos en esta categoría no son los más óptimos, se decide enfocar este tipo de aprendizaje únicamente en la formación de un lexicón de palabras relacionadas con el trastorno de la ansiedad.

### 3.4.1. Enfoque basado en lexicón

Otro enfoque para realizar el análisis de la ansiedad en los mensajes extraídos es a través de un lexicón. Según el Diccionario de Enseñanza y Aprendizaje de Lenguas [38], un **lexicón** se define como “un componente que contiene un listado de los elementos léxicos de una lengua, así como información sobre sus propiedades estructurales”. En el contexto de este proyecto, el lexicón consistirá en un listado de palabras de distintas clases morfológicas que ayuden a determinar si el usuario que envía el conjunto de mensajes está experimentando ansiedad.

Abordar esta perspectiva plantea un gran desafío debido a la escasez de investigaciones previas destinadas a analizar los trastornos de ansiedad. Existe poco material de apoyo al que recurrir para afrontar este trabajo, siendo mayormente en inglés lo que está disponible. Por lo tanto, se decide enfrentar este aspecto siguiendo las siguientes metodologías:

- Creación de un lexicón

  Tal y como se ha señalado anteriormente, existen pocos listados de palabras que contribuyan a la identificación de posibles casos de ansiedad en los mensajes de texto. Por esta razón, se ha decidido crear un listado propio utilizando el corpus de datos etiquetado.

  Para lograrlo, se ha revisado el corpus junto con su etiquetado y se han extraído aquellos archivos etiquetados con un alto grado de ansiedad. Se ha utilizado la clasificación que asigna a los archivos de mensajes un valor en un rango entre 0 y 1, según el nivel de ansiedad detectado. Con el objetivo de evitar que el listado creado a partir de estos archivos fuera demasiado amplio y, por lo tanto, aumentara el margen de error, se ha optado por seleccionar únicamente aquellos archivos etiquetados con un nivel igual o superior a 0.9, es decir, aquellos etiquetados con 0.9 y 1.0.

  El lexicón creado estará compuesto por los verbos, sustantivos y adjetivos más frecuentes en el conjunto de archivos seleccionados con un alto grado de ansiedad detectada. De esta manera, se busca identificar similitudes en la terminología utilizada por otros usuarios que deseamos analizar y otros que ya han sido analizados. Se ha establecido un umbral de ocurrencias de palabras del listado en el conjunto de los mensajes y, en caso de superarlo, se determina que ese usuario es positivo en ansiedad.

  Al utilizar este método, el etiquetado del nuevo conjunto de datos analizado será '1' en caso de que el usuario sea positivo en ansiedad y '0' en caso contrario, ya que al comparar términos no es posible determinar un nivel concreto dentro de un rango.

El gran inconveniente es que ni la metodología utilizada ni el etiquetado realizado proporcionan certezas sobre los resultados obtenidos, pero pueden ayudar en este proceso de análisis.

- Uso de un lexicón creado por terceros

Tras indagar en busca de listados de palabras relacionadas con la ansiedad, se encontró un blog [39] que había creado un pequeño lexicón de 70 palabras asociadas con la ansiedad. Al igual que con el listado propio creado, se siguió un proceso de etiquetado comparando las palabras de los mensajes de texto de cada archivo con las palabras del listado. Sin embargo, los resultados obtenidos en una primera fase de pruebas y testeo fueron tan desalentadores que se decidió no tomar este lexicón con referencia para obtener conclusiones. También se ha utilizado otro listado perteneciente al proyecto ISOL[14], proporcionado por la tutora del trabajo, el cual está compuesto por una recopilación exhaustiva de palabras negativas utilizadas por usuarios de redes sociales.

[14] https://sinai.ujaen.es/investigacion/recursos/isol

# 4. Implementación

Con los cimientos del proyecto establecidos, el siguiente paso implica la implementación concreta de los conceptos y estrategias previamente delineados para abordar la detección de posibles casos de ansiedad entre los usuarios de las redes sociales. Aprovechando el sólido conocimiento adquirido, esta sección se enfoca en la traducción práctica de la metodología en un entorno aplicado, lo que llevará desde la teoría hasta la práctica, garantizando la efectividad de las soluciones obtenidas.

## 4.1. Tecnologías utilizadas

Tal y como se ha puntualizado en apartados anteriores, el proyecto se ha desarrollado utilizando el lenguaje Python en el entorno de desarrollo Visual Studio Code. Además del preprocesamiento correspondiente del texto a analizar y su posterior proceso de vectorización, ha sido necesario implementar algoritmos de aprendizaje supervisado, respaldados por la biblioteca *sklearn*.

**SKlearn**[15], abreviatura de *scikit-learn*, es una biblioteca de código abierto en Python que ofrece una amplia gama de algoritmos de aprendizaje automático para tareas de clasificación, regresión, agrupación, entre otras. Su flexibilidad y facilidad de uso la convierten en una elección popular para proyectos de análisis de datos y PLN.

El objetivo principal del proyecto se centra en la evaluación de textos nuevos para detectar ansiedad mediante técnicas de aprendizaje supervisado. En el caso de utilizar una etiquetación de los archivos de entrenamiento con valores binarios (0 y 1), indicando la presencia o ausencia de ansiedad en el texto, se aplicará un enfoque de clasificación. Por otro lado, si se emplea una etiquetación que asigna un valor numérico entre 0 y 1 según el grado de ansiedad detectado en el texto, se optará por un enfoque de regresión.

En general, todas las funciones de los algoritmos reciben tanto el conjunto de entrenamiento, previamente preprocesado y vectorizado, con sus respectivas etiquetas para entrenar el modelo, como el conjunto de prueba, también preprocesado y vectorizado, que permitirá evaluar el rendimiento de cada uno. En la creación del modelo, se requiere un

[15] https://scikit-learn.org/stable/

constructor específico para cada algoritmo a ejecutar con estos datos. Los constructores necesarios son:

- **Algoritmos de aprendizaje supervisado – Enfoque Clasificación**
  - Support Vector Machines *(SVM)*
    - Importar el módulo 'svm' de la librería 'sklearn'.
    - Usar constructor 'SVC()'.
      - El parámetro 'kernel' define el tipo de núcleo que se utilizará en el algoritmo para transformar los datos. Los núcleos permiten que el algoritmo encuentre una frontera de decisión óptima en un espacio dimensional diferente.
        - Valores: 'lineal', 'poly', 'rbf', 'sigmoid'

```
from sklearn import svm                    # Support Vector Machines – SVM

# Creación del modelo
model_svm = svm.SVC(kernel = 'linear')
```

  - Naive Bayes
    - Importar el clasificador 'MultinomialNB' de la librería 'sklearn.naive_bayes'.
    - Usar constructor 'MultinomialNB()'.
      - Sin parámetros.

```
from sklearn.naive_bayes import MultinomialNB          # Naive Bayes

# Creación del modelo
model_nb = MultinomialNB()
```

  - Árboles de Decisión
    - Importar el módulo 'tree' de la librería 'sklearn'.
    - Usar constructor 'DecisionTreeClassifier()'.
      - Sin parámetros.

```
from sklearn import tree                          # Árboles de Decisión

# Creación del modelo
model_decision_tree = tree.DecisionTreeClassifier()
```

- Random Forest
  - Importar el clasificador 'RandomForestClassifier' de la librería 'sklearn.ensemble'.
  - Usar constructor 'RandomForestClassifier()'.
    - El parámetro 'n_estimators' especifica el número de árboles en el bosque. Un mayor número de árboles generalmente mejora la precisión del modelo, pero también incrementa el tiempo de ejecución.
    - El parámetro 'random_state' controla la aleatoriedad del modelo. Al fijar un valor específico, se asegura que los resultados sean reproducibles cada vez que se entrena el modelo con los mismos datos. Cualquier número entero puede ser utilizado, lo que permitirá que, con el uso de diferentes valores, se pueda producir diferentes resultados.

```
from sklearn.ensemble  import RandomForestClassifier       #Random Forest

# Creación del modelo
model_rf = RandomForestClassifier(n_estimators = 200, random_state = 42)
```

- K–Vecinos (KNN)
  - Importar el clasificador 'KNeighborsClassifier' de la librería 'sklearn.neighbors'.
  - Usar constructor 'KNeighborsClassifier()'.
    - El parámetro 'n_neighbors' especifica el número de vecinos que se utilizarán para clasificar un punto de datos desconocido.

```
from sklearn.neighbors import KNeighborsClassifier                  # K–vecinos

# Creación del modelo
model_knn = KNeighborsClassifier (n_neighbors = 8)
```

- Logistic Regression
  - Importar el módulo 'LogisticRegression' de la librería 'sklearn.linear_model'.
  - Usar constructor 'LogisticRegression()'.

- El parámetro 'max_iter' especifica el número máximo de iteraciones que el algoritmo de optimización puede realizar. Esto es útil para asegurar que el modelo converja.

```
from sklearn.linear_model import LogisticRegression        # Logistic Regression

# Creación del modelo
model_lr = LogisticRegression (max_iter = 100)
```

- **Algoritmos de aprendizaje supervisado – Enfoque Regresión**
  - Support Vector Machines *(SVM)*
    - Importar la librería 'svm'.
    - Usar constructor 'SVR()'.
      - Parámetro 'kernel'.

```
from sklearn import svm                    # Support Vector Machines – SVM

# Creación del modelo
model_svm = svm.SVR(kernel = 'linear')
```

  - Árboles de Decisión
    - Importar el módulo 'tree' de la librería 'sklearn'.
    - Usar constructor 'DecisionTreeRegressor()'.
      - Sin parámetros.

```
from sklearn import tree                    # Árboles de Decisión

# Creación del modelo
model_decision_tree = tree.DecisionTreeRegressor()
```

  - Random Forest
    - Importar el regresor 'RandomForestRegressor' de la librería 'sklearn.ensemble'.
    - Usar constructor 'RandomForestRegressor()'.
      - Parámetro 'n_estimators'.
      - Parámetro 'random_state'.

```
from sklearn.ensemble  import RandomForestRegressor      #Random Forest

# Creación del modelo
model_rf = RandomForestRegressor(n_estimators = 200, random_state = 42)
```

- K–Vecinos *(KNN)*
  - Importar el regresor 'KNeighborsRegressor' de la librería 'sklearn.neighbors'.
  - Usar constructor 'KNeighborsRegressor()'.
    - Parámetro 'n_neighbors'.

```
from sklearn.neighbors import KNeighborsRegressor                # K–vecinos

# Creación del modelo
model_knn = KNeighborsRegressor(n_neighbors = 8)
```

- Logistic Regression
  - Importar el módulo 'LogisticRegression' de la librería 'sklearn.linear_model'.
  - Usar constructor 'LogisticRegression()'.
    - Parámetro 'max_iter'.

```
from sklearn.linear_model import LogisticRegression         # Logistic Regression

# Creación del modelo
model_lr = LogisticRegression (max_iter = 100)
```

Una vez construido el modelo, se ejecuta la función "fit()" con el conjunto de entrenamiento y sus etiquetas correspondientes. Posteriormente, se ejecuta la función "predict()" con el conjunto de prueba, lo que devolverá las etiquetas asociadas a esos datos.

Además, existe la opción de realizar un **ensemble** tanto para tareas de clasificación como de regresión. En este caso, se entrenan los múltiples modelos de aprendizaje supervisado, mencionados anteriormente, sobre los vectores de características de los datos preprocesados. Cada modelo se entrena de manera independiente utilizando el mismo conjunto de datos vectorizados.

Para hacer predicciones, se utiliza una estrategia de votación por mayoría, donde cada modelo hace una predicción y la predicción final se determina por la mayoría de votos entre los modelos. Este enfoque mejora la precisión al combinar las fortalezas de diferentes modelos, aumenta la robustez del sistema y reduce la probabilidad de sobreajuste (*overfitting*), ya que los errores de un modelo pueden ser corregidos por otros.

Sin embargo, estas no son las únicas metodologías para determinar si un usuario sufre un trastorno de ansiedad. En lugar de utilizar algoritmos de aprendizaje supervisado, también se puede optar por un **enfoque no supervisado** mediante la formación de diferentes lexicones que recopilen palabras para comparar con el texto del usuario, buscando un número específico de coincidencias.

En el caso de la creación de un lexicón propio, como se indicó anteriormente, se realizará una recopilación de la terminología utilizada por los usuarios etiquetados con un grado de ansiedad de 0.9 y 1.0, enfocándose en los verbos, sustantivos, adjetivos y adverbios que más se repitan. Se utilizará la función “Counter()”, que permitirá contabilizar el número de apariciones de los diferentes términos en la totalidad de los textos analizados y obtener las frecuencias deseadas. El lexicón estará formado por 15 verbos más frecuentes, así como por el mismo número de sustantivos y adjetivos, obviando la utilización de los adverbios ya que se comprobó que no aportaban significativamente al resultado. Tras esta recopilación, la estrategia a seguir para etiquetar cada nuevo fichero será contabilizar el número de términos que aparecen en el listado creado y, cuando se supere un umbral marcado, se determinará como positivo o negativo.

En el caso de utilizar lexicones proporcionados, simplemente hay que repetir esta última parte de comparación, también buscando comprobar un número específico de ocurrencias.

A continuación, se adjunta un enlace a un repositorio en *GitHub* que contiene el código de la interfaz, distintos archivos ejecutables para realizar las diferentes funcionalidades del proyecto de manera individual, un directorio con ficheros para realizar pruebas y los conjuntos de entrenamiento a utilizar:

- https://github.com/jzm00007/tfg.git

## 4.2. Funcionamiento

Con el propósito de reflejar el funcionamiento de este proyecto, se utilizará un diagrama de flujo cuyo objetivo es detallar y visualizar el proceso de detección de ansiedad en mensajes de usuarios extraídos de Telegram. Este diagrama permitirá identificar claramente las etapas del proyecto, desde la extracción y preprocesamiento de datos hasta la aplicación de modelos de aprendizaje automático y la generación de resultados. La representación visual facilitará no solo la planificación y ejecución del proyecto, sino también la comunicación de los pasos y decisiones involucradas a otros interesados en el trabajo. Además, el diagrama servirá como una herramienta de referencia que asegura que cada etapa del proyecto se ejecute de manera organizada y eficiente.

Un **diagrama de flujo** [40] es una representación gráfica de un proceso o algoritmo. Utiliza símbolos estandarizados como rectángulos, diamantes, óvalos y flechas para ilustrar las operaciones, decisiones y secuencias de pasos involucrados. Estos diagramas son herramientas útiles para planificar, documentar y comunicar procesos complejos de manera clara y concisa, facilitando la comprensión y el análisis de las actividades involucradas. A través de la visualización estructurada que proporciona, es posible identificar posibles cuellos de botella, redundancias o áreas que requieren mayor atención y ajuste, optimizando así el desarrollo y la implementación del proyecto.

Antes de visualizar el proceso completo del trabajo a través de dicho diagrama, se procede a describir los pasos principales que se han incluido junto con sus acciones necesarias para el correcto funcionamiento:

1. Inicio del sistema.
2. Extracción de Datos de Telegram.
    - Conectar a la API de Telegram.
    - Obtener mensajes de usuarios.
    - Almacenar datos en formato JSON *(id_message, message, date)*
3. Preprocesamiento de Textos.
    - Limpieza del texto *(eliminar caracteres especiales, URLs, emojis)*
    - Conversión a minúsculas.
    - Tokenización.

- Eliminación de palabras vacías *(stopwords)*
- Lematización.

4. Vectorización/Extracción de Características.

   - Opción 1: Bag-of-Words.
   - Opción 2: TF–IDF.
   - Opción 3: Word2vec.

5. Entrenamiento del Modelo.

   - Opción 1: Algoritmos de Clasificación
     - SVM.
     - Naive Bayes.
     - Árboles de Decisión.
     - Random Forest.
     - K–vecinos.
     - Logistic Regression.
     - Ensemble.
   - Opción 2: Algoritmos de Regresión.
     - SVM.
     - Árboles de Decisión.
     - Random Forest.
     - K–vecinos.
     - Logistic Regression.
     - Ensemble.
   - Opción 3: Aplicación de Lexicones.

6. Evaluación del Modelo.

   - Métricas de rendimiento *(precisión, recall, F1-score)*

7. Predicción.

   - Aplicar el modelo entrenado a nuevos mensajes.
   - Generar resultados con probabilidad de ansiedad *(no certezas)*

8. Generación de Resultados.
    - Mostrar las predicciones obtenidas sobre los documentos introducidos.
9. Fin del sistema.

Tras hacer un repaso por cada uno de los puntos a recorrer, a continuación se muestra de manera visual el diagrama de flujo que refleja el funcionamiento completo del sistema:

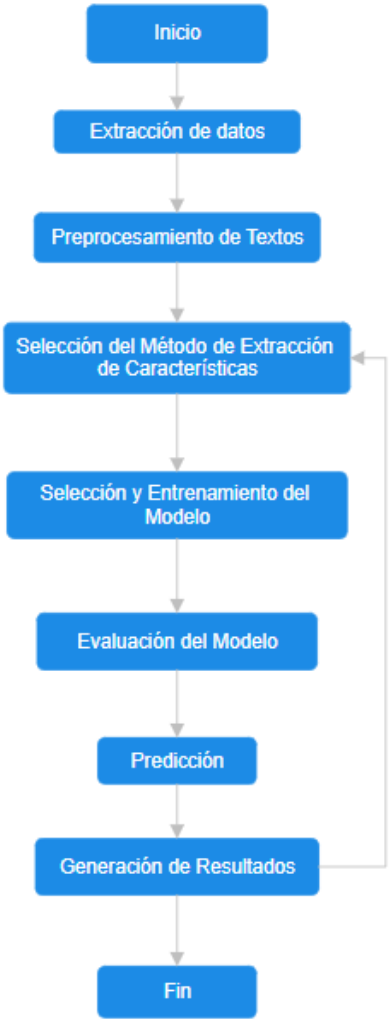



**Figura 16:** Diagrama de Flujo del sistema

# 5. Pruebas y Resultados

Con la implementación de la solución al proyecto completada, el enfoque se traslada ahora hacia la fase de pruebas y evaluación de resultados. En este punto crítico, se llevarán a cabo una serie de pruebas exhaustivas para validar la efectividad y la precisión de la metodología desarrollada para detectar posibles casos de ansiedad entre los usuarios de las redes sociales. Se evaluará el rendimiento de los modelos y, además, se realizará un análisis profundo de los resultados obtenidos. La presentación sistemática de estos hallazgos, tanto cuantitativos como cualitativos, permitirá validar la viabilidad de la solución, así como ofrecer percepciones significativas sobre su impacto y utilidad en la práctica.

## 5.1. Análisis de errores

Para verificar el funcionamiento del sistema construido, se realizará una experimentación en la que se observarán las diferentes métricas obtenidas para cada uno de los algoritmos según el proceso de extracción de características utilizado. Esto permitirá extraer conclusiones sobre cuál modelo funciona mejor.

Para este fin, se utilizará el conjunto de datos etiquetado proporcionado por la tarea *MentalRiskES*, que contiene 150 archivos con mensajes. Se seleccionarán 11 archivos, cada uno con un valor de etiquetado diferente en el rango de 0 a 1, para ser usados como conjunto de prueba y evaluar el rendimiento del modelo. El resto de los archivos se utilizarán para entrenar el modelo. El mismo conjunto de prueba se empleará en el modo de etiquetado binario para asegurar equidad entre ambos tipos de etiquetado. Al obtener las métricas de validación, se realizará un balance promedio proveniente de varias ejecuciones para obtener un resultado representativo del rendimiento del modelo.

Por un lado, en el caso del enfoque clasificatorio, las **métricas de validación** utilizadas son las siguientes:

- Exactitud (Accuracy): mide el porcentaje de casos en los que el modelo ha acertado. Sin embargo, cuando las clases no están balanceadas, se recomienda no utilizar esta métrica, sugiriendo en su lugar las siguientes métricas.
- Precisión: mide la capacidad del modelo para evitar falsos positivos y falsos negativos. Se calcula para cada clase y luego se obtiene la media ponderada.

- Recall: evalúa la capacidad del modelo para clasificar correctamente todas las muestras positivas y negativas. Se calcula para cada clase y luego se obtiene la media ponderada.
- Medida F1 Ponderada: es la media ponderada de la *precisión* y el *recall*, donde ambas métricas tienen igual importancia. Esta métrica es la más deseable para comparar modelos.

Para entender a fondo las métricas mencionadas anteriormente, es necesario comprender los siguientes conceptos, que se utilizan para describir los resultados de una clasificación binaria:

- Verdaderos Positivos (VP): son los casos en los que el modelo predice correctamente la clase positiva.
- Verdaderos Negativos (VN): son los casos en los que el modelo predice correctamente la clase negativa.
- Falsos Positivos (FP): son los casos en los que el modelo predice incorrectamente la clase positiva.
- Falsos Negativos (FN): son los casos en los que el modelo predice incorrectamente la clase negativa.

Estas categorías se pueden resumir en una tabla de contingencia o matriz de confusión, que se utiliza para evaluar el rendimiento del modelo:

| | ***Predicción Positiva*** | ***Predicción Negativa*** |
|---|---|---|
| ***Clase Positiva*** | Verdadero Positivo (VP) | Falso Negativo (FN) |
| ***Clase Negativa*** | Falso Positivo (FP) | Verdadero Negativo (VN) |

**Tabla 8:** Tabla de contingencia o Matriz de confusión

A partir de estos valores, se pueden calcular las métricas anteriores, siendo estos valores los elementos de sus fórmulas.

Por otro lado, en el enfoque de regresión, las **métricas de validación** utilizadas son:

- Coeficiente de Determinación ($R^2$): es una medida estadística que indica qué tan bien se ajustan los valores predichos de un modelo a los valores observados. Representa la proporción de la varianza en la variable dependiente que es predecible a partir de las variables independientes. En caso de que el valor sea

menor que 0, esto indica que el modelo es peor que un modelo promedio (la media de los valores observados).

- Error Cuadrático Medio (MSE): es la media de los cuadrados de los errores o desviaciones, es decir, la diferencia entre los valores predichos por el modelo y los valores observados. Penaliza más fuertemente los errores grandes debido a la naturaleza cuadrática de la función. Un valor de MSE más bajo indica un mejor ajuste del modelo a los datos.
- Error Absoluto Medio (MAE): es la media de los errores absolutos entre los valores predichos y los valores observados. A diferencia del MSE, no penaliza tanto los errores grandes, ya que no eleva los errores al cuadrado. Un valor de MAE más bajo indica un mejor ajuste del modelo a los datos.

Una vez comentadas las métricas y los conjuntos de datos de entrenamiento y prueba que se emplearán en la ejecución del proyecto para su evaluación, los resultados obtenidos para cada algoritmo de aprendizaje supervisado, según el proceso de vectorización, en el **enfoque de clasificación** son los siguientes:

- Support Vector Machines

La Tabla 9 muestra los resultados obtenidos de la ejecución del modelo *SVM* con cada una de las variantes de metodologías planteadas para la extracción de características, junto a la Tabla 10 que presenta la tabla de contingencia.

| | **Accuracy** | **Precission** | **Recall** | **Medida F1** |
|---|---|---|---|---|
| **BoW** | 54.55% | 53.90% | 54.55% | 53.77% |
| **TF-IDF** | 72.73% | 81.82% | 72.73% | 69.61% |
| **Word2vec – Skip-Gram** | 54.55% | 75.21% | 54.55% | 38.50% |
| **Word2vec – CBOW** | 54.55% | 75.21% | 54.55% | 38.50% |

**Tabla 9:** Validación del clasificador *SVM* con los distintos métodos de vectorización

| | **VP** | **FP** | **VN** | **FN** |
|---|---|---|---|---|
| **BoW** | 4 | 3 | 2 | 2 |
| **TF-IDF** | 6 | 3 | 2 | 0 |
| **Word2vec – Skip-Gram** | 6 | 5 | 0 | 0 |
| **Word2vec – CBOW** | 6 | 5 | 0 | 0 |

**Tabla 10:** Matriz de confusión del clasificador *SVM* con los distintos métodos de vectorización

El método de vectorización TF-IDF con SVM es la mejor opción, ya que destaca con la Medida F1 más alta (69.61%) y una precisión notable (81.82%). Esto indica que, aunque tiene algunas falsas alarmas (FP=3), el modelo es eficaz en identificar correctamente las muestras positivas (FN=0). Además, TF-IDF ofrece un balance adecuado entre precisión y recall, lo que se traduce en un rendimiento global más confiable en la clasificación.

Por otro lado, aunque Word2vec (tanto Skip-Gram como CBOW) muestra una alta precisión (75.21%), la Medida F1 es significativamente más baja (38.50%), indicando un desbalance notable entre precisión y recall. Ambos métodos tienen una alta tasa de falsos positivos (FP=5), y no logran identificar correctamente las muestras negativas (VN=0), lo que reduce la exactitud global.

BoW muestra el rendimiento más bajo en todas las métricas. La precisión y el recall están equilibrados pero son bajos, lo que refleja una capacidad limitada para clasificar correctamente las muestras.

- Naive Bayes

La Tabla 11 muestra los resultados obtenidos de la ejecución del modelo de *Naive Bayes* con cada una de las variantes de metodologías planteadas para la extracción de características. Sin embargo, las variantes de Word2vec no están incluidas debido a que Naive Bayes supone independencia entre características, mientras que Word2vec genera vectores correlacionados que representan relaciones semánticas, lo que contradice esta suposición. Además, se adjunta la Tabla 12 que presenta la tabla de contingencia.

| | **Accuracy** | **Precission** | **Recall** | **Medida F1** |
|---|---|---|---|---|
| **BoW** | 81.82% | 86.36% | 81.82% | 80.84% |
| **TF-IDF** | 54.55% | 75.21% | 54.55% | 38.50% |

**Tabla 11:** Validación del clasificador *Naive Bayes* con los distintos métodos de vectorización

| | **VP** | **FP** | **VN** | **FN** |
|---|---|---|---|---|
| **BoW** | 6 | 2 | 3 | 0 |
| **TF-IDF** | 6 | 5 | 0 | 0 |

**Tabla 12:** Matriz de confusión del clasificador *Naive Bayes* con los distintos métodos de vectorización

BoW con Naive Bayes es la mejor opción con la Medida F1 más alta (80.84%) y una precisión notable (86.36%). Esto indica que el modelo es eficaz en identificar correctamente las muestras positivas (FN=0) y la mayoría de las negativas (VN=3), y tiene una baja tasa de falsas alarmas (FP=2). Esta efectividad se debe a la simplicidad y la naturaleza de independencia de las características que BoW puede proporcionar.

Por otro lado, TF-IDF muestra una Medida F1 significativamente más baja (38.50%), a pesar de tener una precisión relativamente alta (75.21%). Esto indica un desbalance considerable entre precisión y recall. Además, presenta una alta tasa de falsos positivos (FP=5) y no logra identificar correctamente ninguna de las muestras negativas (VN=0), lo que reduce su exactitud global.

- Árboles de decisión

La Tabla 13 muestra los resultados obtenidos de la ejecución del modelo de *Árboles de Decisión* con cada una de las variantes de metodologías planteadas para la extracción de características, junto a la Tabla 14 que presenta la tabla de contingencia.

| | **Accuracy** | **Precission** | **Recall** | **Medida F1** |
|---|---|---|---|---|
| **BoW** | 72.73% | 73.05% | 72.73% | 72.26% |
| **TF-IDF** | 90.91% | 92.21% | 90.91% | 90.75% |
| **Word2vec – Skip-Gram** | 81.82% | 86.36% | 81.82% | 80.84% |
| **Word2vec – CBOW** | 63.64% | 63.64% | 63.64% | 63.64% |

**Tabla 13:** Validación del clasificador *Árboles de Decisión* con los distintos métodos de vectorización

| | **VP** | **FP** | **VN** | **FN** |
|---|---|---|---|---|
| **BoW** | 5 | 2 | 3 | 1 |
| **TF-IDF** | 6 | 1 | 4 | 0 |
| **Word2vec – Skip-Gram** | 6 | 2 | 3 | 0 |
| **Word2vec – CBOW** | 4 | 2 | 3 | 2 |

**Tabla 14:** Matriz de confusión del clasificador *Árboles de Decisión* con los distintos métodos de vectorización

El método de vectorización TF-IDF con Árboles de Decisión destaca con la Medida F1 más alta (90.75%) y una precisión notable (92.21%). Esto indica que el modelo

es muy eficaz en identificar correctamente las muestras positivas y tiene una baja tasa de falsas alarmas (FP=1).

Por otro lado, Word2vec en su variante de Skip-Gram muestra una Medida F1 bastante alta (80.84%) y una precisión también elevada (86.36%), aunque con una tasa de falsos positivos ligeramente superior a TF-IDF (FP=2). Con una exactitud del 81.82%, esta metodología de vectorización también es una opción fuerte, aunque no tan eficaz como TF-IDF.

Respecto al resto de métodos, aunque BoW muestra un rendimiento aceptable con una Medida F1 de 72.26%, su precisión y recall, así como su exactitud, son inferiores a los anteriormente comentados. Word2vec en su variante de CBOW muestra el rendimiento más bajo en todas las métricas, indicando una capacidad limitada para clasificar correctamente las muestras.

- Random Forest

La Tabla 15 muestra los resultados obtenidos de la ejecución del modelo de *Random Forest* con cada una de las variantes de metodologías planteadas para la extracción de características, junto a la Tabla 16 que presenta la tabla de contingencia.

| | **Accuracy** | **Precission** | **Recall** | **Medida F1** |
|---|---|---|---|---|
| **BoW** | 63.64% | 78.18% | 63.64% | 56.06% |
| **TF-IDF** | 63.64% | 64.40% | 63.64% | 61.69% |
| **Word2vec – Skip-Gram** | 81.82% | 81.82% | 81.82% | 81.82% |
| **Word2vec – CBOW** | 81.82% | 86.36% | 81.82% | 80.84% |

**Tabla 15:** Validación del clasificador *Random Forest* con los distintos métodos de vectorización

| | **VP** | **FP** | **VN** | **FN** |
|---|---|---|---|---|
| **BoW** | 6 | 4 | 1 | 0 |
| **TF-IDF** | 5 | 3 | 2 | 1 |
| **Word2vec – Skip-Gram** | 5 | 1 | 4 | 1 |
| **Word2vec – CBOW** | 6 | 2 | 3 | 0 |

**Tabla 16:** Matriz de confusión del clasificador *Random Forest* con los distintos métodos de vectorización

En este caso, el método de vectorización Word2vec tanto en su variante de Skip-Gram como en su variante de CBOW son las mejores opciones con una Medida F1 alta (81.82% y 80.84% respectivamente). Skip-Gram tiene un balance perfecto entre precisión y recall (81.82%), mientras que CBOW tiene una ligera ventaja en precisión (86.36%) pero una sutil desventaja en F1-Score.

Ambos métodos Word2vec muestran un equilibrio sólido entre precisión y recall, reflejado en una alta exactitud (81.82%). Word2vec en su variante de Skip-Gram tiene menos falsos positivos (FP=1) que CBOW (FP=2), lo que puede ser ventajoso dependiendo del contexto de la aplicación.

Por otro lado, aunque BoW muestra una precisión relativamente alta (78.18%), su Medida F1 (56.06%) y exactitud (63.64%) son considerablemente más bajas que las de Word2vec. La alta tasa de falsos positivos (FP=4) y un único verdadero negativo (VN=1) reflejan un desbalance significativo.

En lo que respecta a TF-IDF, tiene un rendimiento ligeramente mejor que BoW en términos de Medida F1 (61.69%) pero sigue siendo inferior a Word2vec. La precisión y el recall son equilibrados, pero la exactitud y la capacidad general de clasificación son menores comparadas con los métodos Word2vec.

- K–vecinos *(KNN)*

La Tabla 17 muestra los resultados obtenidos de la ejecución del modelo de *K–vecinos* con cada una de las variantes de metodologías planteadas para la extracción de características, junto a la Tabla 18 que presenta la tabla de contingencia.

| | **Accuracy** | **Precission** | **Recall** | **Medida F1** |
|---|---|---|---|---|
| **BoW** | 63.64% | 79.80% | 63.64% | 59.74% |
| **TF-IDF** | 63.64% | 64.40% | 63.64% | 61.69% |
| **Word2vec – Skip-Gram** | 72.73% | 73.05% | 72.73% | 72.26% |
| **Word2vec – CBOW** | 81.82% | 81.82% | 81.82% | 81.82% |

**Tabla 17:** Validación del clasificador *K-vecinos* con los distintos métodos de vectorización

| | VP | FP | VN | FN |
|---|---|---|---|---|
| **BoW** | 2 | 0 | 5 | 4 |
| **TF-IDF** | 5 | 3 | 2 | 1 |
| **Word2vec – Skip-Gram** | 5 | 2 | 3 | 1 |
| **Word2vec – CBOW** | 5 | 1 | 4 | 1 |

**Tabla 18:** Matriz de confusión del clasificador *K-vecinos* con los distintos métodos de vectorización

El método de vectorización Word2vec en su variante de CBOW con K-vecinos es la mejor opción con una Medida F1 más alta (81.82%), una precisión equilibrada (81.82%) y una alta exactitud (81.82%). Esto indica que el modelo es muy eficaz en identificar correctamente tanto las muestras positivas como las negativas. La matriz de confusión muestra una baja tasa de falsos positivos (FP=1) y una alta tasa de verdaderos negativos (VN=4), lo que es favorable para una clasificación confiable.

Word2vec en su variante de Skip-Gram también muestra un buen rendimiento con una Medida F1 de 72.26% y una precisión de 73.05%. Tiene una exactitud de 72.73%, lo que es bastante sólido, aunque no alcanza el nivel de la anterior, teniendo una tasa de falsos positivos ligeramente superior (FP=2) comparado con CBOW.

En los otros casos, BoW tiene una alta precisión (79.80%), pero su Medida F1 (59.74%) y exactitud (63.64%) son considerablemente más bajas. La matriz de confusión muestra una alta tasa de falsos negativos (FN=4), lo que reduce su efectividad global. TF-IDF muestra un rendimiento ligeramente mejor que BoW en términos de Medida F1 (61.69%) y tiene una precisión y recall equilibrados. Sin embargo, su exactitud y capacidad general de clasificación son inferiores comparadas con los métodos Word2vec.

- Logistic Regression

La Tabla 19 muestra los resultados obtenidos de la ejecución del modelo de *Logistic Regression* con cada una de las variantes de metodologías planteadas para la extracción de características, junto a la Tabla 20 que presenta la tabla de contingencia.

| | Accuracy | Precission | Recall | Medida F1 |
|---|---|---|---|---|
| **BoW** | 54.55% | 53.90% | 54.55% | 53.77% |
| **TF-IDF** | 54.55% | 75.21% | 54.55% | 38.50% |
| **Word2vec – Skip-Gram** | 54.55% | 75.21% | 54.55% | 38.50% |
| **Word2vec – CBOW** | 54.55% | 75.21% | 54.55% | 38.50% |

**Tabla 19:** Validación del clasificador *Logistic Regression* con los distintos métodos de vectorización

| | VP | FP | VN | FN |
|---|---|---|---|---|
| **BoW** | 4 | 3 | 2 | 2 |
| **TF-IDF** | 6 | 5 | 0 | 0 |
| **Word2vec – Skip-Gram** | 6 | 5 | 0 | 0 |
| **Word2vec – CBOW** | 6 | 5 | 0 | 0 |

**Tabla 20:** Matriz de confusión del clasificador *Logistic Regression* con los distintos métodos de vectorización

El método de vectorización BoW con Logistic Regression es la mejor opción con una Medida F1 más alta (53.77%), una precisión equilibrada (53.90%) y una exactitud (54.55%). Esto indica que el modelo es moderadamente eficaz en identificar correctamente tanto las muestras positivas como las negativas. La matriz de confusión muestra una tasa equilibrada de falsos positivos y verdaderos negativos, lo que es favorable para una clasificación confiable.

El resto de los casos presentan las mismas métricas, consistentes en una alta precisión (75.21%), pero con una Medida F1 (38.50%) y una exactitud (54.55%) inferiores. La matriz de confusión indica una alta tasa de falsos positivos y ningún verdadero negativo, lo que afecta a su efectividad global al no detectar casos negativos.

- Ensemble

La Tabla 21 muestra los resultados obtenidos de la ejecución del *Ensemble* con todos los modelos anteriores, concluyendo con el voto de la mayoría, con cada una de las variantes de metodologías planteadas para la extracción de características. Además, se adjunta la Tabla 22 que presenta la tabla de contingencia. En caso de empate entre las predicciones de los distintos algoritmos, se decide no clasificar la

instancia, lo cual puede afectar la contabilización, ya que puede haber casos en los que se evalúe el rendimiento con menos de 11 archivos.

| | **Accuracy** | **Precission** | **Recall** | **Medida F1** |
|---|---|---|---|---|
| **BoW** | 77.78% | 84.13% | 77.78% | 75.93% |
| **TF-IDF** | 70.00% | 80.00% | 70.00% | 64.00% |
| **Word2vec – Skip-Gram** | 63.64% | 64.39% | 63.64% | 61.69% |
| **Word2vec – CBOW** | 72.73% | 73.05% | 72.73% | 72.26% |

**Tabla 21:** Validación del *Ensemble* con los distintos métodos de vectorización

| | **VP** | **FP** | **VN** | **FN** |
|---|---|---|---|---|
| **BoW** | 5 | 2 | 2 | 0 |
| **TF-IDF** | 6 | 3 | 1 | 0 |
| **Word2vec – Skip-Gram** | 5 | 3 | 2 | 1 |
| **Word2vec – CBOW** | 5 | 2 | 3 | 1 |

**Tabla 22:** Matriz de confusión del *Ensemble* con los distintos métodos de vectorización

A la hora de realizar un Ensemble, la mejor opción es el método de vectorización BoW, que muestra un rendimiento fuerte con una Medida F1 del 75.93% y una precisión elevada del 84.13%. Su exactitud es del 77.78%, lo que indica un buen balance entre las predicciones correctas de las clases positivas y negativas. La matriz de confusión muestra un número de falsos negativos igual a cero (FN=0), lo que sugiere que el modelo es muy eficaz en identificar todas las muestras positivas.

Word2Vec en su variante CBOW podría ser otra buena opción, mostrando métricas similares, pero ligeramente inferiores, con una Medida F1 del 72.26% y una precisión del 73.05%. La matriz de confusión muestra una tasa aceptable de falsos positivos (FP=2) y verdaderos negativos (VN=3), aunque es menos eficaz que BoW.

Respecto a los otros métodos de vectorización, TF-IDF tiene un rendimiento más bajo en términos de Medida F1 (64.00%) y exactitud (70.00%). Aunque la precisión es alta (80.00%), su efectividad global se ve reducida por una mayor tasa de falsos positivos (FP=3). Word2Vec en su variante Skip-Gram tiene los peores resultados, mostrando un rendimiento inferior comparado con otros métodos.

Para comparar entre todos los algoritmos con sus respectivos resultados utilizando cada metodología de vectorización, lo más recomendable es utilizar la "Medida F1 Ponderada". Esta métrica considera tanto la precisión como el recall, lo que la hace robusta en situaciones de desbalance de clases. La ponderación garantiza que cada clase contribuya proporcionalmente a la medida, lo que la hace útil para comparar modelos en contextos donde las clases pueden tener diferentes tamaños o importancias relativas. Por lo tanto, con estos resultados, se puede concluir que el algoritmo que mejor rendimiento ha obtenido es el de Árboles de Decisión con el proceso de vectorización TF–IDF debido a su capacidad para manejar datos de alta dimensión y extraer patrones relevantes de manera efectiva, lo que resulta en una clasificación precisa y consistente, como se evidencia por la alta puntuación en la medida F1 ponderada de 90.75%.

También cabe destacar que, en lo que respecta a los métodos de vectorización, Word2vec en su variante de CBOW ha demostrado ser la mejor opción en múltiples modelos, incluyendo Random Forest, K-Vecinos y el modelo Ensemble. Ofrece un excelente balance entre precisión y recall, resultando en altos valores de Medida F1. Esto se debe a que captura bien las relaciones semánticas y contextuales de las palabras, mejorando el rendimiento del modelo en la clasificación.

En general, los resultados muestran una tendencia a preferir métodos de vectorización basados en TF-IDF y Word2vec (CBOW y Skip-Gram) combinados con clasificadores como Árboles de Decisión y Random Forest. Es recomendable, además, explorar otros métodos de vectorización y ajustar los hiperparámetros para mejorar potencialmente aún más el rendimiento.

De manera similar, se presentan los resultados obtenidos para cada algoritmo de aprendizaje supervisado, según el proceso de vectorización, en el **enfoque de regresión**:

- Support Vector Machines

  La Tabla 23 muestra los resultados obtenidos de la ejecución del modelo de *SVM* con cada una de las variantes de metodologías planteadas para la extracción de características.

| | $R^2$ | MSE | MAE |
|---|---|---|---|
| **BoW** | –0.2749 | 0.1275 | 0.3215 |
| **TF-IDF** | 0.3184 | 0.0682 | 0.2193 |
| **Word2vec – Skip-Gram** | –0.8538 | 0.1854 | 0.3507 |
| **Word2vec – CBOW** | –0.6918 | 0.1692 | 0.3393 |

**Tabla 23:** Validación del regresor *SVM* con los distintos métodos de vectorización

En este enfoque, el modelo de SVM con TF-IDF presenta el mejor coeficiente de determinación (0.3184), indicando que es el método de vectorización que mejor explica la variabilidad de la variable dependiente. Además, tiene el menor error cuadrático medio (0.0682), lo que sugiere que el modelo predice con mayor precisión los valores observados, y el menor error absoluto medio (0.2193), lo que refuerza su capacidad para hacer predicciones más precisas en comparación con los otros métodos.

El resto de métodos de vectorización muestran resultados desalentadores. Word2vec en su variante de Skip-Gram visualiza el peor desempeño con un $R^2$ de –0.8538, indicando que el modelo con este método no logra capturar la variabilidad de la variable dependiente. Además, tiene el mayor MSE (0.1854) y MAE (0.3507), reflejando una pobre precisión en las predicciones. En el caso de la variante CBOW, aunque ligeramente mejor que Skip-Gram, también muestra un desempeño negativo en $R^2$ (–0.6918) y altos valores de MSE (0.1692) y MAE (0.3393), indicando un ajuste inadecuado del modelo. Finalmente, BoW tiene un $R^2$ de –0.2749, lo que indica un ajuste pobre, aunque mejor que los métodos Word2vec, y los valores de MSE (0.1275) y MAE (0.3215) son también más bajos, pero significativamente más altos que los de TF-IDF, lo que refleja un desempeño moderado.

- Árboles de decisión

La Tabla 24 muestra los resultados obtenidos de la ejecución del modelo de *Árboles de Decisión* con cada una de las variantes de metodologías planteadas para la extracción de características.

| | $R^2$ | MSE | MAE |
|---|---|---|---|
| **BoW** | –0.3727 | 0.1372 | 0.2818 |
| **TF-IDF** | 0.2273 | 0.0773 | 0.2455 |
| **Word2vec – Skip-Gram** | –1.4182 | 0.2418 | 0.3818 |
| **Word2vec – CBOW** | –2.7000 | 0.3700 | 0.5182 |

**Tabla 24:** Validación del regresor *Árboles de Decisión* con los distintos métodos de vectorización

Al igual que en el caso anterior, para Árboles de Decisión, el método de vectorización TF-IDF presenta el mejor coeficiente de determinación (0.2273). Además, tiene el menor error cuadrático medio (0.0773) y el menor error absoluto medio (0.2455).

Para Word2vec en su variante de Skip-Gram se comprueba un desempeño muy pobre con un $R^2$ de –1.4182, indicando que el modelo con este método no logra capturar la variabilidad de la variable dependiente. Asimismo, tiene un alto MSE (0.2418) y MAE (0.3818). En el caso de su variante CBOW, muestra el peor desempeño con un $R^2$ de –2.7000, altos valores de MSE (0.3700) y MAE (0.5182), indicando un ajuste inadecuado.

Si se analiza el método BoW, este visualiza también un pobre ajuste con un $R^2$ de –0.3727, aunque mejor que los métodos Word2vec. Los valores de MSE (0.1372) y MAE (0.2818) son más bajos que los de Word2vec pero significativamente más altos que los de TF-IDF, por lo que tiene un desempeño moderado.

- Random Forest

La Tabla 25 muestra los resultados obtenidos de la ejecución del modelo de *Random Forest* con cada una de las variantes de metodologías planteadas para la extracción de características.

| | $R^2$ | MSE | MAE |
|---|---|---|---|
| **BoW** | 0.2208 | 0.0779 | 0.2154 |
| **TF-IDF** | 0.5150 | 0.0485 | 0.1648 |
| **Word2vec – Skip-Gram** | 0.0580 | 0.0942 | 0.2163 |
| **Word2vec – CBOW** | 0.0543 | 0.0946 | 0.2292 |

**Tabla 25:** Validación del regresor *Random Forest* con los distintos métodos de vectorización

Nuevamente, el método de vectorización TF-IDF para Random Forest presenta el mejor coeficiente de determinación (0.5150), tiene el menor error cuadrático medio (0.0485) y también muestra el menor error absoluto medio (0.1648).

Word2vec, tanto en su variante de Skip-Gram como de CBOW, muestran un desempeño moderadamente bajo con un $R^2$ de 0.0580 y 0.0543 respectivamente, indicando que el modelo con estos métodos de vectorización captura solo una pequeña fracción de la variabilidad de la variable dependiente. Además, tiene un MSE y MAE relativamente alto, reflejando una menor precisión en las predicciones debido a un ajuste inadecuado del modelo.

En el caso de BoW, tiene un $R^2$ de 0.2208, lo que indica un ajuste moderado, mejor que los métodos Word2vec pero no a TF-IDF, al igual que los valores de MSE (0.0779) y MAE (0.2154).

- K–vecinos *(KNN)*

La Tabla 26 muestra los resultados obtenidos de la ejecución del modelo de *K–vecinos* con cada una de las variantes de metodologías planteadas para la extracción de características.

| | **$R^2$** | **MSE** | **MAE** |
|---|---|---|---|
| **BoW** | –0.0470 | 0.1047 | 0.2739 |
| **TF-IDF** | –0.8581 | 0.1858 | 0.3602 |
| **Word2vec – Skip-Gram** | –0.4616 | 0.1462 | 0.2636 |
| **Word2vec – CBOW** | –0.1112 | 0.1111 | 0.2398 |

**Tabla 26:** Validación del regresor *K-vecinos* con los distintos métodos de vectorización

A diferencia de los modelos anteriores, para K–vecinos, el método de vectorización Word2vec en su variante de CBOW presenta el mejor coeficiente de determinación (–0.1112) entre las opciones disponibles, aunque sigue siendo negativo, lo que indica que el modelo no es particularmente bueno para capturar la variabilidad de la variable dependiente. Tiene un MSE (0.1111) y MAE (0.2398) relativamente bajos en comparación con los otros métodos, lo que sugiere un rendimiento ligeramente mejor en términos de precisión de predicción.

Aunque TF-IDF era el mejor en los casos anteriores, para K–vecinos presenta el peor coeficiente de determinación (–0.8581), lo que indica un ajuste muy pobre y una incapacidad significativa para capturar la variabilidad de la variable dependiente. También tiene el mayor MSE (0.1858) y MAE (0.3602),.

Para el resto de los casos, BoW tiene un $R^2$ de –0.0470, lo que indica un ajuste ligeramente mejor pero aún insuficiente, y los valores de MSE (0.1047) y MAE (0.2739) son moderados en comparación con los otros métodos, reflejando un desempeño razonable. Word2vec en su variante de Skip-Gram tiene un $R^2$ de –0.4616, reflejando un desempeño deficiente, y los valores de MSE (0.1462) y MAE (0.2636) son elevados.

- Logistic Regression

La Tabla 27 muestra los resultados obtenidos de la ejecución del modelo de *Logistic Regression* con cada una de las variantes de metodologías planteadas para la extracción de características.

| | **$R^2$** | **MSE** | **MAE** |
|---|---|---|---|
| **BoW** | –1.1545 | 0.2155 | 0.3727 |
| **TF-IDF** | –2.5000 | 0.3500 | 0.5000 |
| **Word2vec – Skip-Gram** | –2.5000 | 0.3500 | 0.5000 |
| **Word2vec – CBOW** | –2.5000 | 0.3500 | 0.5000 |

**Tabla 27:** Validación del regresor *Logistic Regression* con los distintos métodos de vectorización

Al igual que en el enfoque clasificatorio de este modelo, el método de vectorización BoW es la mejor opción, obteniendo mejores métricas que el resto de métodos. El enfoque BoW proporciona un $R^2$ de –1.1545, un MSE de 0.2155 y un MAE de 0.3727, indicando un rendimiento notablemente mejor, aunque igualmente ineficiente. En contraste, los métodos TF-IDF y Word2vec en sus variantes Skip-Gram y CBOW presentan las mismas métricas, con un valor de $R^2$ de –2.5000, un MSE de 0.3500 y un MAE de 0.5000, reflejando un rendimiento nefasto.

- Ensemble

La Tabla 28 muestra los resultados obtenidos de la ejecución del *Ensemble* con todos los modelos anteriores, resultantes de la media del conjunto de ellos, con cada una de las variantes de metodologías planteadas para la extracción de características.

| | $R^2$ | MSE | MAE |
|---|---|---|---|
| **BoW** | 0.0585 | 0.0941 | 0.2136 |
| **TF-IDF** | 0.1512 | 0.0849 | 0.2374 |
| **Word2vec – Skip-Gram** | –0.3671 | 0.1367 | 0.2736 |
| **Word2vec – CBOW** | –0.3704 | 0.1370 | 0.2855 |

**Tabla 28:** Validación del *Ensemble* con los distintos métodos de vectorización

El análisis de los resultados muestra que el método de vectorización TF-IDF proporciona el mejor rendimiento en el enfoque de Ensemble, con un $R^2$ de 0.1512, un MSE de 0.0849 y un MAE de 0.2374, indicando una capacidad moderada para predecir el valor objetivo.

El método BoW también muestra un rendimiento aceptable, con un $R^2$ de 0.0585, un MSE de 0.0941 y un MAE de 0.2136, lo que sugiere que es una opción viable, aunque ligeramente inferior a TF-IDF.

En contraste, los métodos basados en Word2vec (Skip-Gram y CBOW) tienen resultados considerablemente peores, con valores de $R^2$ negativos (–0.3671 y –0.3704, respectivamente) y métricas de error (MSE y MAE) más altas. Esto refleja una capacidad predictiva mucho menor en comparación con BoW y TF-IDF, evidenciando que estos métodos no son adecuados para este tipo de análisis en el contexto de Ensemble.

En este caso, para determinar qué algoritmo es mejor, generalmente se prefiere una métrica que evalúe la capacidad predictiva del modelo y que refleje tanto la precisión como la magnitud de los errores de predicción. Por tanto, el Error Absoluto Medio (MAE) sería una buena opción, eligiendo el algoritmo cuyo dicho valor sea más bajo en tu conjunto de datos de prueba. Esto indicaría que el modelo tiene una mejor capacidad para hacer predicciones precisas y que los errores de predicción son, en promedio, más pequeños.

Al observar los resultados de MAE en las diferentes tablas, se puede concluir que el mejor modelo es el Random Forest con el método de vectorización TF-IDF, ya que tiene el MAE más bajo de 0.1648. Esto indica que, en promedio, tiene errores de predicción más pequeños en comparación con los otros modelos y métodos de vectorización evaluados. Además, en el resto de métricas también es el más óptimo, ya que tiene el mayor Coeficiente de Determinación ($R^2$) con un valor de 0.5150, indicando una mejor capacidad para explicar la variabilidad de la variable dependiente, y el menor Error Cuadrático Medio (MSE) con valor de 0.0485, mostrando el menor error en la predicción de los valores observados, lo que refleja una alta precisión.

Por tanto, esta proporciona el mejor equilibrio entre explicabilidad y precisión, resultando en predicciones más fiables y precisas. Además, es aconsejable evitar el uso de Word2vec para modelos de regresión en contextos similares, dado su rendimiento significativamente inferior en comparación con los otros métodos de vectorización.

Finalmente, se presentan los resultados obtenidos para cada uno de los lexicones utilizados, puntualizando que los resultados son relativos, ya que no se utiliza una metodología exacta:

| | **Accuracy** | **Precission** | **Recall** | **Medida F1** |
|---|---|---|---|---|
| **Lexicón propio** | 36.36% | 55.30% | 36.36% | 36.36% |
| **Lexicón proporcionado – Listado palabras ansiedad** | 27.27% | 40.26% | 27.27% | 29.70% |
| **Lexicón proporcionado – Listado palabras negativas** | 27.27% | 40.26% | 27.27% | 29.70% |

**Tabla 29:** Validación de los diferentes lexicones utilizados

| | **VP** | **FP** | **VN** | **FN** |
|---|---|---|---|---|
| **Lexicón propio** | 2 | 1 | 2 | 6 |
| **Lexicón proporcionado – Listado palabras ansiedad** | 2 | 2 | 1 | 6 |
| **Lexicón proporcionado – Listado palabras negativas** | 2 | 2 | 1 | 6 |

**Tabla 30:** Matriz de confusión de los lexicones con los distintos métodos de vectorización

Debido a que esta implementación se basa en comparar términos para contabilizar la frecuencia de aparición, no es una estrategia que ofrezca certezas, pudiendo variar los resultados de la metodología dependiendo de la cantidad de ficheros seleccionados para evaluar o de cuáles ficheros sean.

El Lexicón Propio es el que presenta el mejor rendimiento entre los tres lexicones evaluados, destacándose con la Medida F1 más alta (36.36%) y la mayor precisión (55.30%). Esto indica que el modelo es más eficaz en identificar correctamente tanto las muestras positivas como las negativas comparado con los otros lexicones. Aunque el accuracy del Lexicón Propio es bajo, su alta precisión sugiere que los positivos clasificados son más confiables. La matriz de confusión muestra una tasa de falsos positivos baja (FP=1) y una tasa de verdaderos negativos aceptable (VN=2), lo cual es favorable para una clasificación más confiable.

Sin embargo, los Lexicones Proporcionados, tanto el de palabras relacionadas con la ansiedad como el de palabras negativas, muestran un desempeño inferior, con precisión y medida F1 más bajas en comparación con el lexicón propio, siendo ambas de 40.26% y 29.70% respectivamente. Presentan una tasa de falsos positivos (FP=2) más alta y un verdadero negativo más bajo (VN=1), lo que indica una menor efectividad en la clasificación correcta de las muestras.

Para futuros proyectos que utilicen análisis de sentimientos o categorización basada en lexicones, es recomendable optimizar y refinar el lexicón propio, ya que ha mostrado un mejor desempeño comparativo, o considerar el uso combinado o adaptativo de lexicones para mejorar la precisión y recall.

# 6. Conclusiones y Trabajos Futuros

Una vez finalizado el desarrollo e implementación de este proyecto, junto con su correspondiente análisis de resultados, el último paso es realizar un repaso exhaustivo para evaluar si se han alcanzado los objetivos establecidos inicialmente. Esto implica identificar y abordar cualquier problema que haya surgido durante el proceso, así como extraer conclusiones relevantes que ayuden a mejorar en futuros proyectos.

## 6.1. Conclusiones

En este trabajo, se ha explorado el ámbito del procesamiento de lenguaje natural, centrándose específicamente en el análisis de sentimientos en redes sociales, con un enfoque particular en la ansiedad. Se llevó a cabo una fase inicial de investigación para recopilar información que pudiera contribuir al desarrollo de un sistema propio para detectar el problema de salud mental de la ansiedad, con la calidad necesaria para un proyecto relevante como un Trabajo Fin de Grado. A lo largo del TFG, se ha procurado explicar con detalle las diferentes etapas de manera clara y didáctica, con el propósito de facilitar la comprensión de los procedimientos a lectores nuevos en la materia.

En la tarea de programación, se han desarrollado una serie de algoritmos de aprendizaje supervisado, tanto en el enfoque clasificatorio como en el de regresión, según el formato de etiquetado empleado, que determinarán si el texto recibido, correspondiente a los mensajes del usuario a analizar, indica la presencia de trastorno de ansiedad o no. Esta predicción se basará en un entrenamiento previo, donde todo el texto será preprocesado y vectorizado para que sea tratado de manera efectiva por los modelos generados. En última instancia, también se puede utilizar una serie de lexicones incorporados que buscarán correspondencias de palabras para determinar si el usuario sufre o no de ansiedad.

Tras realizar un balance general, se puede determinar que los objetivos iniciales planteados para este proyecto se han alcanzado. Se ha llevado a cabo un estudio sobre el análisis de las emociones, centrándose en la ansiedad, y se ha desarrollado una herramienta que permite determinar a partir del texto introducido si un usuario sufre dicho trastorno. Todo esto se ha apoyado en una interfaz de usuario que facilitará la interacción del usuario, permitiendo la carga de los archivos con los mensajes de los usuarios a analizar y la selección del algoritmo y metodología de vectorización deseada.

Actualmente, la búsqueda de información no representa el principal desafío, dada la abundancia de recursos en Internet. Cualquier individuo puede redactar sobre cualquier tema. El verdadero reto radica en discernir qué datos son relevantes para el progreso de nuestro proyecto y en evaluar con criterio y en función de nuestros conocimientos previos la calidad de la información hallada. En este sentido, el acceso a una amplia gama de artículos científicos, facilitado por ser miembros de la Universidad de Jaén, ha sido fundamental. Además, la disponibilidad de la tutora del proyecto ha agilizado enormemente el proceso, brindando la oportunidad de plantear cualquier duda que surja.

Sin embargo, durante el proceso, se ha hecho frente a ciertos desafíos que surgieron en diferentes etapas de la investigación. Estos problemas, aunque inicialmente pueden parecer obstáculos, se han logrado solventar con éxito. Por ejemplo, como es común en todo proceso de programación, pueden surgir ciertos impedimentos con los métodos utilizados, lo que requiere revisar las versiones de las librerías importadas para resolver ciertos solapamientos o incompatibilidades. Además, para obtener los mejores resultados posibles, se necesitó llevar a cabo un largo periodo de ajuste de parámetros exhaustivo para determinar cuáles proporcionaban una mejor predicción.

Finalmente, debido a las limitaciones de tiempo para profundizar en el proyecto y su naturaleza introductoria, se plantean posibles expansiones del trabajo a continuación. Estas expansiones potenciales incluyen la mejora de los resultados obtenidos y el aumento de la confianza en la capacidad de catalogar nuestro sistema construido en un primer intento como competitivo.

## 6.2. Trabajos futuros

Pese a que el trabajo realizado contribuye significativamente a la investigación actual sobre el análisis de los trastornos de ansiedad en los usuarios a través de las redes sociales, aún queda espacio para continuar profundizando. Se pueden añadir mejoras que refuercen la calidad de las distintas partes de la herramienta desarrollada.

En este proyecto se han implementado exclusivamente algoritmos de aprendizaje supervisado, aunque también se puede considerar la inclusión de lexicones como soporte adicional. Sin embargo, se podría explorar aún más, por ejemplo, integrando transformers. Los **Transformers** son modelos revolucionarios que han transformado la forma en que se

aborda el análisis de texto. Surgieron en un artículo de Google a finales de 2017, presentando una arquitectura innovadora que reemplazó las capas recurrentes por capas de atención. Estas capas codifican cada palabra en función del contexto, lo que permite que los modelos basados en Transformers comprendan mejor el significado y la relación entre las palabras. Desde entonces, se han convertido en el estado del arte en tareas de PLN, como generación de texto, resumen y clasificación. Entre sus ventajas, destacan su eficiente paralelización durante el entrenamiento y la inferencia.

Además, aunque este proyecto se ha centrado en el estudio de los trastornos de ansiedad, existe una amplia gama de investigaciones dedicadas a otros estados emocionales. En lugar de enfocarse exclusivamente en un solo trastorno, se podría adoptar un enfoque más holístico y explorar diversos problemas que afectan a la sociedad contemporánea. Al ampliar el análisis hacia otros tipos de trastornos y emociones, se pueden obtener una comprensión más completa de las complejidades del bienestar mental. Además, resulta especialmente relevante considerar la interrelación entre diferentes problemas de salud mental; por ejemplo, en el caso de este estudio, la relación entre la depresión y la ansiedad es notable. Por lo tanto, sería beneficioso para futuras investigaciones no solo abordar un espectro más amplio de emociones y trastornos, sino también explorar las conexiones y superposiciones entre ellos, lo que podría proporcionar una visión más integral de la salud mental y mejores enfoques de intervención.

En relación a la selección de distintas redes sociales para la extracción de datos, en este proyecto se ha optado por utilizar Telegram. Aunque esta elección ha resultado en la obtención de resultados satisfactorios, es importante reconocer que existen otras plataformas, como Twitter, donde los usuarios tienden a expresar con mayor frecuencia su estado emocional. Al utilizar Telegram, nos hemos centrado en la búsqueda de grupos donde los usuarios discuten y comparten sus experiencias emocionales, lo que puede introducir un sesgo en la investigación al limitarse a comunidades específicas relacionadas con problemas mentales. Por otro lado, en Twitter, los usuarios suelen expresarse de manera más libre y diversa, lo que permite encontrar una mezcla de todo tipo de emociones. Sin embargo, cabe destacar que actualmente Twitter presenta limitaciones significativas en cuanto a la extracción de datos. También existen otras redes sociales como Reddit[16], Facebook, entre otras, que están disponibles para los investigadores que deseen dirigir sus estudios hacia estas

[16] https://www.reddit.com/?rdt=42303

plataformas y así ampliar la variedad de datos disponibles y enriquecer el análisis de las emociones en línea.

En el ámbito del PLN, es notable la predominancia de corpus de datos en inglés, lo cual puede limitar la generalización de los resultados obtenidos y sesgar el desarrollo de modelos y herramientas hacia este idioma. Para abordar esta limitación y promover una investigación más inclusiva y global, resulta necesario expandir la disponibilidad de corpus en otros idiomas. Este enfoque no solo ampliaría las oportunidades de investigación para lingüistas y científicos del lenguaje en todo el mundo, sino que también enriquecería la comprensión de las diferencias lingüísticas y culturales. Asimismo, la creación de corpus multilingües permitiría el desarrollo de modelos de PLN más robustos y precisos, con aplicaciones potenciales en traducción automática, análisis de sentimientos y comprensión intercultural. Por lo tanto, un área prometedora para futuras investigaciones sería la recopilación y curación de corpus de datos en una variedad de idiomas, lo que contribuiría significativamente al avance de este campo en un contexto global.

Toda esta investigación proporciona una serie de aplicaciones prácticas y beneficios potenciales, como puede ser la identificación temprana de individuos en riesgo de trastornos de ansiedad, la personalización de intervenciones de salud mental o la mejora de la comprensión científica de la ansiedad y su impacto en la sociedad. Estas perspectivas abren la puerta a un campo prometedor en el que la tecnología y la investigación pueden unirse para mejorar la calidad de vida y el bienestar emocional de las personas.

# 7. Bibliografía

[1] Instituto Nacional de Salud Mental. (Abril 2023). Trastornos de ansiedad. Recuperado de https://www.nimh.nih.gov/health/topics/anxiety-disorders/index.shtml

[2] Diagrama de Gantt. (19 de agosto de 2023). En *Wikipedia*. https://es.wikipedia.org/w/index.php?title=Diagrama_de_Gantt&oldid=153182377

[3] Bericat Alastuey, E. (2012). Emociones. Sociopedia.isa, 1-13.

[4] EDPYN. (15 de septiembre de 2016). Paul Ekman: el rostro de las emociones. Recuperado de https://edpyn.com/blog-coaching/paul-ekman-rostro-emociones/

[5] Brody L (1999) Gender, Emotion, and the Family. Cambridge, MA: Harvard University Preses.

[6] Lawler EJ, Thye SR and Yoon J (2008) Social exchange and micro social order. American Sociological Review 73: 519–542.

[7] Stets JE and Turner JH (eds) (2006) Handbook of the Sociology of Emotions. Boston, MA: Springer.

[8] Rodrigo, E. M., García, R. S., & Martín, L. S. (2011). El complejo mundo de la interactividad: emociones y redes sociales. Revista Mediterránea de Comunicación: Mediterranean Journal of Communication, 2(1), 189-208.

[9] Jiménez, R. M. R., López, M. D. M. C., Gracia, P., Quintana, P. J. V., & López, M. J. T. (2013). Inteligencia emocional y comunicación: la conciencia corporal como recurso. REDU: Revista de docencia Universitaria, 11(1), 213.

[10] Serrano, J. (2016). Internet y emociones: nuevas tendencias en un campo de investigación emergente. Revista científica iberoamericana de comunicación y educación, 26. 19-26.

[11] García, E., & Heredia, N. (2017). Emociones y redes sociales en adolescentes Emotions and social networks in teenagers. Revista de estudios e investigación en psicología y educación.

[12] Kušen, E., Cascavilla, G., Figl, K., Conti, M., & Strembeck, M. (2017, August). Identifying emotions in social media: comparison of word-emotion lexicons. In 2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW) (pp. 132-137). IEEE.

[13] Telegram. (s.f.). Preguntas frecuentes. Recuperado el 1 de febrero de 2024 de https://telegram.org/faq/es

[14] Díaz Kuaik, I., & De la Iglesia, G. (2019). Ansiedad: revisión y delimitación conceptual.

[15] Delgado, E. C., De la Cera, D. X., Lara, M. F., & Arias, R. M. (2021). Generalidades sobre el trastorno de ansiedad. Revista Cúpula, 35(1), 23-36.

[16] Organización Mundial de la Salud. (2023, 27 de septiembre). Trastornos de ansiedad. Recuperado el 31 de enero de 2024 de https://www.who.int/es/news-room/fact-sheets/detail/anxiety-disorders

[17] Vetere, G. (2008). Nivel de funcionamiento y calidad de vida en pacientes con trastorno de ansiedad generalizada. Anuario de investigaciones, 15, 00-00.

[18] Procesamiento de lenguajes naturales. (2024, 24 de enero). *Wikipedia*. https://es.wikipedia.org/w/index.php?title=Procesamiento_de_lenguajes_naturales&oldid=157612573

[19] Alejandra, N. C. (2022, 23 junio). Técnicas de machine learning para procesamiento del lenguaje natural. Universidad de Oviedo. http://hdl.handle.net/10651/63977

[20] Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. Ain Shams engineering journal, 5(4), 1093-1113.

[21] Python. (2024, 27 de febrero). *Wikipedia*, La enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Python&oldid=158473800

[22] Visual Studio Code. (2024, 14 de marzo). *Wikipedia*, La enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=158794261

[23] Google. (s.f.). Colaboratory: Preguntas frecuentes. Recuperado el 29 de febrero de 2024, de https://research.google.com/colaboratory/intl/es/faq.html

[24] Alba M. Mármol Romero, Adrián Moreno Muñoz, Flor Miriam Plaza-del-Arco, M. Dolores Molina González, María Teresa Martín Valdivia, L. Alfonso Ureña-López, and Arturo Montejo Ráez. 2024. MentalRiskES: A New Corpus for Early Detection of Mental Disorders in Spanish. In Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pages 11204–11214, Torino, Italia. ELRA and ICCL.

[25] Web scraping. (2024, 1 de marzo). *Wikipedia*, La enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Web_scraping&oldid=158530420

[26] Linube. (s.f.). Crawlers: ¿Qué son y cómo funcionan? Recuperado de https://linube.com/blog/crawlers-que-son/

[27] Telethon. (s.f.). Documentación de Telethon. Recuperado de https://docs.telethon.dev/en/stable/

[28] Wang B, Wang A, Chen F, Wang Y, Kuo C. (2019). *Evaluating word embedding models: Methods and experimental results*. APSIPA Transactions on Signal and Information Processing, 8, E19. doi:10.1017/ATSIP.2019.12.

[29] Elastic. (s. f.). ¿Qué es el "word embedding"? Recuperado de https://www.elastic.co/es/what-is/word-embedding

[30] Técnica de Machine Learning para crear modelos predictivos a partir de datos de entrada y respuesta conocidos (2024, 7 abril). Recuperado de https://es.mathworks.com/discovery/supervised-learning.html

[31] Extensiones y aplicaciones (Máquinas de vectores soporte, SVM). Recuperado de http://verso.mat.uam.es/~joser.berrendero/cursos/Matematicas-IO/io-tema6- svm-16.pdf

[32] Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. The annals of statistics, 1171-1220. https://projecteuclid.org/download/pdfview_1/euclid.aos/1211819561

[33] H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS http://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf

[34] Kingsford, C., & Salzberg, S. L. (2008). What are decision trees? Nature biotechnology, 26(9), 1011-1013.

[35] Algoritmos de Clustering paralelos en Sistemas de Recuperación de Información Distribuidos. D. Daniel Jiménez González. (Mayo 2011) http://www.dsic.upv.es/docs/bib-dig/tesis/etd11182010-091738/Tesis.pdf

[36] Peterson, L. E. (2009). K-nearest neighbor. Scholarpedia, 4(2), 1883.

[37] LaValley, M. P. (2008). Logistic regression. *Circulation*, *117*(18), 2395-2399.

[38] DicenLen. (s.f.). Diccionario de Enseñanza y Aprendizaje de Lenguas. Recuperado de https://www.dicenlen.eu/es/diccionario/entradas/lexicon

[39] Moreno, J. (2019, 22 de abril). 70 palabras relacionadas con ansiedad. Recuperado de https://jackmoreno.com/2019/04/22/70-palabras-relacionadas-con-ansiedad/

[40] Diagrama de flujo. (2024, 8 de mayo). En *Wikipedia, La enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Diagrama_de_flujo&oldid=159976444

# Anexo 1 – Manual de instalación

El proyecto consiste en un directorio general que contiene dos subdirectorios: uno denominado ‘interfaz’, con todo el contenido necesario para generar y poder interactuar con la interfaz de usuario, y otro denominado ‘proyecto’, con varios archivos correspondientes a las distintas partes del proceso de este proyecto. Todo este sistema de directorios se abrirá en Visual Studio Code, creando un entorno de programación específico en el que se trabajará con Python. Las librerías adicionales necesarias se instalarán en la terminal del entorno utilizando el gestor de paquetes “pip”.

Por un lado, se procede a comentar el contenido del directorio ‘proyecto’, el cual está formado por los siguientes archivos que se ejecutan de manera independiente, obteniendo o almacenando los ficheros necesarios en el directorio “Ficheros sobre Ansiedad” situado en el directorio raíz del ordenador. La función de cada archivo y las librerías que son necesarias instalar son:

- *extraccion_datos.py*
    - **Función:** extraer los mensajes de usuarios de un grupo de Telegram, siendo necesario especificar el identificador del grupo y del usuario, con las correspondientes verificaciones de acceso previas.
    - **Almacenamiento:** los ficheros JSON extraídos se almacenan en el directorio “Archivos extraídos Telegram”, contenido en el directorio base anterior.
    - **Librerías a instalar**:
        - Pytz: utilizada para el manejo de franjas horarias. Esta librería facilita el trabajo con zonas horarias y la conversión entre diferentes zonas. Es necesaria cuando se trata con datos de usuarios de diferentes partes del mundo y se necesita una representación coherente del tiempo. La versión recomendada es *2023.3.post1*.
        - Telethon: es un cliente de Telegram. Esta librería permite interactuar con la API de Telegram para realizar tareas como la extracción de mensajes, la gestión de grupos, y más. Es necesaria para autenticar, acceder y extraer los datos de Telegram. La versión recomendada es *1.34.0*.

- *preprocesamiento_del_texto_limpieza.py*
    - **Función:** realizar el preprocesamiento del texto de los ficheros JSON previamente extraídos, centrándose únicamente en la limpieza del texto (conversión de emojis, normalización de palabras, eliminación de correos electrónicos, menciones, etc.)
    - **Almacenamiento:** obtiene y sobrescribe los ficheros JSON contenidos en el directorio "Preprocesamiento del texto" situado en el directorio anterior.
    - **Librerías a instalar**:
        - Emoji: utilizada para la conversión de los emojis gráficos en texto descriptivo. Esto es útil para mantener el significado de los emojis en un formato que los algoritmos de procesamiento de texto puedan entender y analizar. La versión recomendada es *2.11.1*.
        - Googletrans: utilizada para la traducción de texto a partir de Google Translate. Esto puede ser útil para normalizar el texto en un solo idioma si los mensajes están en múltiples idiomas. La versión recomendada es *4.0.0-rc1*.
- *preprocesamiento_del_texto.py*
    - **Función:** realizar el preprocesamiento del texto de los ficheros JSON previamente extraídos, llevando a cabo tareas de tokenización, conversión de palabras a minúsculas, eliminación de palabras vacías, entre otras.
    - **Almacenamiento:** obtiene y sobrescribe los ficheros JSON contenidos en el directorio "Preprocesamiento del texto" situado en el directorio anterior.
    - **Librerías a instalar:**
        - NLTK: proporciona herramientas de Procesamiento del Lenguaje Natural. Esta librería es útil para la tokenización, eliminación de palabras vacías (stop words), y otras tareas de preprocesamiento de texto. La versión recomendada es *3.8.1*.
        - SpaCy: proporciona herramientas avanzadas para el procesamiento de texto, incluyendo tokenización eficiente, etiquetado de partes del discurso, etc. La versión recomendada es *3.7.4*. Para su uso completo, es necesario instalar el modelo de lenguaje específico:
            - Ejecutar en la terminal: "python -m spacy download es_core_news_lg" para descargar el modelo de español.

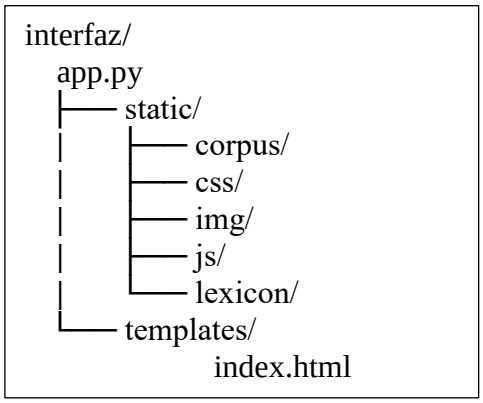
- *proceso_vectorizacion.py*
    - **Función:** realizar la extracción de características y el proceso de vectorización sobre el texto de los ficheros JSON preprocesados, utilizando diferentes metodologías.
    - **Almacenamiento:** obtiene y sobrescribe los ficheros JSON contenidos en el directorio "Vectorización" situado en el directorio anterior.
    - **Librerías a instalar:**
        - Scikit–learn: proporciona herramientas de Aprendizaje Automático. Esta librería es fundamental para realizar tareas de extracción y vectorización de características, así como para la implementación de modelos de aprendizaje automático. La versión recomendada es *1.4.2*.
        - Gensim: proporciona herramientas para el procesamiento de texto, incluyendo la implementación de modelos de temas (topics), la vectorización de texto, entre otros. La versión recomendada es *4.3.0*.
- *algoritmos_aprendizaje_supervisado.py*
    - **Función:** ejecución de los algoritmos de aprendizaje supervisado, tanto desde el enfoque clasificatorio como regresivo, sobre el texto de los ficheros a analizar.
    - **Almacenamiento:** obtiene los ficheros JSON contenidos en el directorio "Archivos entrenamiento" para entrenar los modelos, y los ficheros JSON contenidos en el directorio "Archivos test" para ser analizados con los modelos entrenados, situados en el directorio base anterior.
    - **Librerías a instalar:** no es necesario instalar nuevas librerías, ya que se utilizan las librerías mencionadas anteriormente.
- *lexicon.py*
    - **Función:** generación de un lexicón propio y predicción de los ficheros a analizar con dicho lexicón y otros lexicones proporcionado.
    - **Almacenamiento:** obtiene los ficheros JSON contenidos en el directorio "Archivos entrenamiento lexicon" para generar el lexicón propio, los ficheros JSON contenidos en el directorio "Archivos test" para ser analizados con los modelos entrenados, y los listados de los lexicones proporcionados, situados en el directorio base anterior.

- **Librerías a instalar:** no es necesario instalar nuevas librerías, ya que se utilizan las librerías mencionadas anteriormente.

Esta estructura garantiza que todos los componentes del proyecto estén organizados y accesibles, lo que facilita tanto el desarrollo como la ejecución de las distintas partes del proceso. Además, permite una gestión y mantenimiento del código más eficientes.

Por otro lado, se procede a examinar el contenido del directorio 'interfaz', el cual alberga los archivos que constituyen la interfaz de usuario. Esta interfaz está compuesta tanto por la parte Back-end en Flask como por la parte Front-end con HTML, CSS y JavaScript. Para su funcionamiento, es esencial instalar Flask, importando la versión *3.0.3*.

La estructura del directorio de la interfaz será la siguiente:




La parte del Back-end estará en el archivo 'app.py', el cual se encarga de la creación de la interfaz y de llevar a cabo todos los procesos relacionados con el proyecto, como el preprocesamiento, la extracción de características, el entrenamiento de los modelos y la predicción. Por otro lado, la parte del Front-end estará en el archivo 'index.html', el cual se encargará de la representación visual de la interfaz, creando los formularios correspondientes y mostrando los mensajes con los resultados.

Además, en el directorio 'static' se almacenarán tanto el corpus de entrenamiento de los modelos como el corpus utilizado para la creación del lexicón propio, así como las imágenes, el código CSS y JavaScript.

Para desplegar la interfaz, se ejecutará el archivo 'app.py', cargando todo el código y realizando el despliegue en la dirección http://127.0.0.1:5000, que se puede abrir en cualquier navegador.

# Anexo 2 – Manual de usuario

El despliegue de la interfaz de usuario ocurre en la dirección http://127.0.0.1:5000 y puede abrirse en cualquier navegador web. Debido a que se requiere realizar el preprocesamiento y la vectorización de todo el texto de los archivos necesarios para el entrenamiento, así como la creación del lexicón propio, la carga inicial de la interfaz puede ser lenta, tardando varios segundos. Sin embargo, una vez que la aplicación está completamente desplegada, su rendimiento es óptimo, aunque podría ralentizarse ligeramente si se introducen demasiados archivos para analizar.

La interfaz de usuario se visualiza de la siguiente manera:


**Figura 17:** Interfaz de Usuario

El procedimiento a seguir para interacturar con la interfaz es el siguiente:

- Seleccionar el algoritmo y el método de vectorización a utilizar.


**Figura 18:** Interfaz de Usuario – Selección de Algoritmo y Proceso de Vectorización

- Seleccionar los ficheros de mensajes de usuarios a analizar.

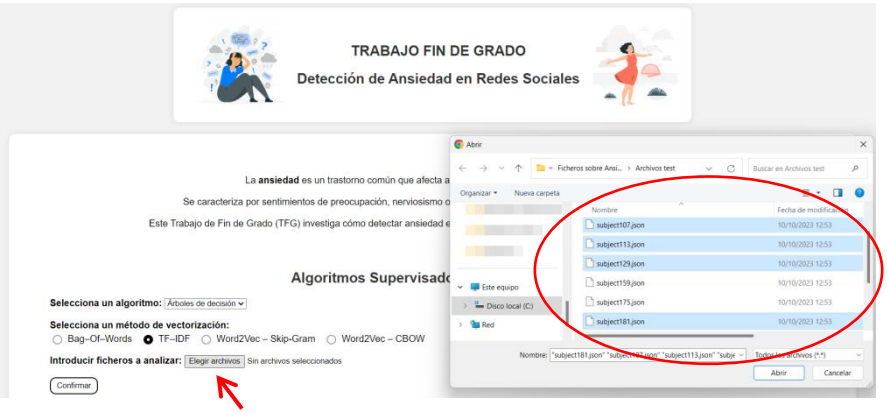



**Figura 19:** Interfaz de Usuario – Selección de ficheros a analizar

- Confirmar la operación a realizar.


**Figura 20:** Interfaz de Usuario – Confirmar operación a realizar

- Visualización de los resultados obtenidos.

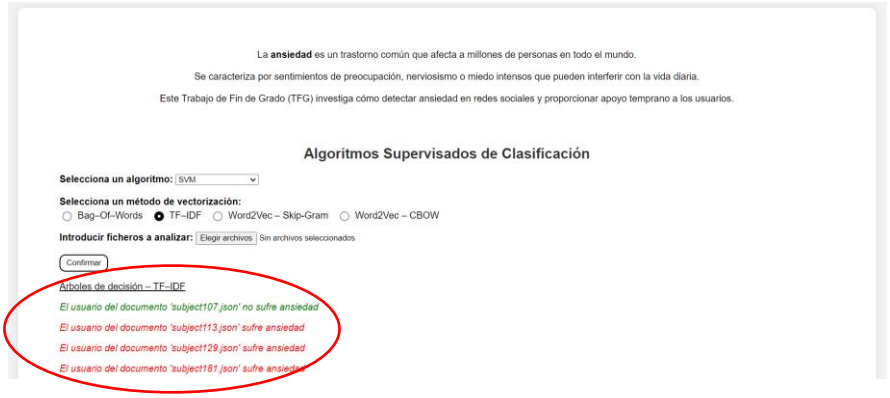



**Figura 21:** Interfaz de Usuario – Visualización de Resultados

Es importante destacar que si se selecciona el algoritmo Naive Bayes con alguna de las dos variantes de vectorización Word2vec, se producirá una incompatibilidad. En este escenario, la interfaz mostrará un mensaje indicando dicho error. Esta notificación garantiza una experiencia de usuario más informativa y ayuda a evitar confusiones o malentendidos durante el uso de la aplicación.


**Figura 22**: Interfaz de Usuario – Selección de Algoritmo y Proceso de Vectorización (Opción *Naive Bayes*)


**Figura 23:** Interfaz de Usuario – Visualización de Resultados (Opción *Naive Bayes*)

En el caso del Ensemble, se ejecuta el conjunto de los algoritmos anteriores bajo el mismo método de vectorización. La predicción final se determina mediante el voto de la mayoría, donde cada algoritmo aporta su predicción individual y la predicción final es el resultado de la mayoría. Se muestra no solo lo predicho por cada algoritmo, sino también la predicción final resultante. En caso de empate entre las predicciones de los distintos algoritmos, la predicción final no se evalúa y se establece como *None*. A continuación, se presenta la visualización de la interfaz:


**Figura 24:** Interfaz de Usuario – Selección de Algoritmo y Proceso de Vectorización (Opción *Ensemble*)


**Figura 25:** Interfaz de Usuario – Visualización de Resultados (Opción *Ensemble*)

Además, es importante destacar que los usuarios también tienen la opción de seleccionar algoritmos supervisados desde un enfoque regresivo, junto con su método de vectorización correspondiente. Asimismo, existe la posibilidad de seleccionar un lexicón de palabras que realice la clasificación en función de un número de ocurrencias específico. Esta flexibilidad en la selección de algoritmos y métodos de vectorización amplía significativamente las opciones disponibles para el análisis de datos, permitiendo a los usuarios adaptar el proceso según sus necesidades específicas con mayor precisión.


**Figura 26:** Interfaz de Usuario – Algoritmos Supervisados de Regresión

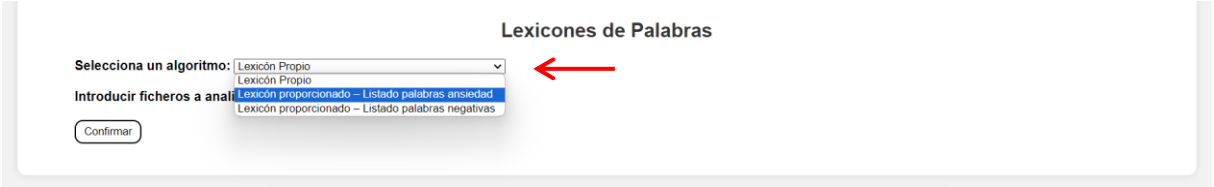



**Figura 27:** Interfaz de Usuario – Lexicones de Palabras