UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Análisis de sentimientos en Instagram usando ChatGPT

Autor:
Dña. Lucía Martín García
Tutor:
Dña. Noemí Merayo Álvarez

TÍTULO: Análisis de sentimientos en Instagram usando ChatGPT
AUTOR: **Dña. Lucía Martín García**
TUTOR: **Dña. Noemí Merayo Álvarez**
DEPARTAMENTO: **Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL
PRESIDENTE: Noemí Merayo Álvarez
SECRETARIO: Lara del Val Puente
VOCAL: Alfonso Bahillo Martínez
SUPLENTE: Juan Carlos Aguado Manzano
SUPLENTE: Ramón de la Rosa Steinz

FECHA:
CALIFICACIÓN:

Resumen de TFG

En este trabajo se ha realizado un estudio sobre el análisis de la respuesta emocional en comentarios de publicaciones en redes sociales, y más específicamente en Instagram. Dado el creciente uso de las redes sociales y la mayor exposición de los influencers a temas de salud mental, es fundamental comprender las emociones que sienten los usuarios frente a este tipo de publicaciones. Por esta razón, se aplicarán técnicas de Inteligencia Artificial y Procesamiento de Lenguaje Natural con el fin de optimizar modelos de clasificación de polaridad (positiva, negativa, indeterminada) y de emociones (amor/admiración, comprensión/empatía/identificación, gratitud, tristeza/pena, enfado/desprecio/burla, indeterminada) en los comentarios.

El objetivo principal de este trabajo es mejorar los resultados obtenidos en dos estudios previos sobre el análisis de sentimientos en redes sociales, uno centrado en temas de salud mental en Instagram y otro centrado en el ámbito de los videojuegos en la plataforma Twitch. Para lograr este objetivo, se ha ampliado el corpus inicial para redes sociales, incluyendo un mayor número de comentarios correspondientes a aquellas emociones que tenían una menor representación. Posteriormente, se ha utilizado el modelo gpt-3.5-turbo de OpenAI para optimizar los resultados de clasificación y se ha evaluado su rendimiento utilizando distintas métricas de precisión, comparando los resultados con estudios anteriores.

Palabras clave

Instagram, análisis de sentimientos, influencers, Inteligencia Artificial, Emociones, Polaridad, Twitch, gpt-3.5-turbo.

Abstract

In this work, a study has been conducted on the analysis of emotional responses in comments on social media posts, specifically on Instagram. Given the growing use of social media and the increased exposure of influencers to mental health topics, it is crucial to understand the emotions that users experience in response to such posts. For this reason, Artificial Intelligence and Natural Language Processing techniques will be applied to optimize models for classifying polarity (positive, negative, indeterminate) and emotions (love/admiration, understanding/empathy/identification, gratitude, sadness/pity, anger/contempt/mockery, neutral) in the comments.

The main objective of this work is to improve the results obtained in two previous studies on sentiment analysis in social media, one focused on mental health topics on Instagram and the other focused on the gaming domain on the Twitch platform. To achieve this goal, the initial social media corpus has been expanded by including a larger number of comments corresponding to those emotions that had less representation. Subsequently, OpenAI's gpt-3.5-turbo model was used to optimize classification results, and its performance was evaluated using various accuracy metrics, comparing the results with previous studies.

Keywords

Instagram, sentiment analysis, influencers, Artificial Intelligence, Emotions, Polarity, Twitch, gpt-3.5-turbo.

Agradecimientos

En primer lugar, quería agradecer a mis padres y a mi hermana por todo el apoyo que me han dado. Gracias por apoyarme siempre y haber estado ahí cuando os he necesitado.

También me gustaría agradecer a mi tutora Noemí, gracias por haber confiado en mí para la realización de este proyecto y por haberme ayudado para que este saliese adelante.

Por último, gracias a mis amigas de la resi, estos cuatro años no habrían sido lo mismo sin vosotras.

Índice

Índice de figuras

Índice de tablas

Índice de ecuaciones

1

Introducción

1.1 Motivación

El presente Trabajo de Fin de Grado (TFG) se centra en el análisis de la respuesta emocional en el contexto de las redes sociales; un tema de gran relevancia en la actualidad debido al uso masivo de estas plataformas y la creciente influencia de los influencers [1]. Este trabajo se motiva en el potencial de herramientas como GPT-3.5, desarrolladas por OpenAI, que han mejorado significativamente el Procesamiento de Lenguaje Natural (PLN) y permiten abordar de manera más eficaz el análisis de los comentarios generados en dichas plataformas. En nuestro caso, nos hemos enfocado en Instagram, plataforma en constante evolución que permite la interacción a través de mensajes directos y comentarios en publicaciones. Contando que tiene millones de usuarios activos y la gran cantidad de comentarios que puede recibir una publicación de una persona conocida, el análisis manual de estos datos se convierte en una tarea extremadamente laboriosa.

Por ello, herramientas como GPT-3.5, en las que se basa ChatGPT, emergen como soluciones innovadoras. Estas, permiten optimizar un modelo capaz de evaluar y categorizar comentarios en función del sentimiento que expresan. Este estudio se centra específicamente en el análisis de sentimientos y polaridad de comentarios en publicaciones de influencers que abordan problemas de salud mental. Al crear un corpus a partir de la recopilación de estos comentarios y entrenar el modelo gpt-3.5-turbo (basado en transformadores), podemos identificar si la reacción del público varía dependiendo de si el influencer es hombre o mujer, o si existen diferencias en las reacciones entre influencers del mismo género.

Otro factor que motiva este trabajo es su potencial contribución a la comunidad psicológica, ya que podría facilitar a los profesionales la evaluación del impacto de las redes sociales en la salud mental. Este análisis podría ayudar a identificar de manera más efectiva los principales factores que influyen en las reacciones emocionales de los usuarios.

Además, también se explorará la respuesta emocional en chats en vivo en la plataforma de videojuegos Twitch mediante los mismos algoritmos, con el objetivo de identificar las particularidades del lenguaje en el contexto de los videojuegos y las emociones generadas entre los usuarios de dicha red. Este enfoque nos permitirá comprender mejor cómo se expresa y se percibe la emoción en entornos digitales de alta interacción y cómo varía según el contexto y el tipo de contenido.

1.2 Objetivos

1.2.1 Objetivo principal

El principal objetivo de este trabajo es utilizar la herramienta de ChatGPT, más en concreto uno de los modelos de GPT-3.5, para poder optimizarlo en el análisis de la respuesta emocional en los comentarios de Instagram y en los chats de Twitch. En nuestro trabajo nos centraremos en la clasificación de la polaridad (positiva, negativa e indeterminada) siendo esta la misma para ambos casos y de emociones, la cual varía para Instagram (amor/admiración, gratitud, comprensión/identificación/empatía, tristeza/pena, enfado/desprecio/burla e indeterminado) y para Twitch (aprobación/empatía/confianza, desaprobación, decepción/tristeza, interés/anticipación/hype, enfado/ira e indeteminado).

Además, este trabajo está basado en los resultados obtenidos de dos estudios previos [2,3] uno de los cuales abordó un análisis en Instagram (contexto de la salud mental) mientras que el otro abordó el análisis en Twitch (contexto de los videojuegos). Sin embargo, el objetivo es la creación de un modelo que mejore los resultados obtenidos anteriormente, y para ello se ha realizado la ampliación de ambos corpus con la inclusión de comentarios con mayor diversidad lingüística y temáticas.

El objetivo final es la creación de cuatro modelos de GPT-3.5 (polaridad Instagram, emociones Instagram, polaridad Twitch y emociones Twitch) que sean

capaces con una alta precisión de clasificar la emoción o polaridad del comentario para cada uno de los casos.

1.2.2 Objetivos específicos

Para poder cumplir el objetivo principal de este TFG, se han establecido los siguientes objetivos específicos:

1. Ampliación del corpus de datos de Instagram, en el que se incluyen publicaciones de influencers masculinos para poder observar si la reacción es distinta en función del género. Para equilibrar la cantidad de comentarios de emociones y polaridad, se ha prestado atención a aquellas que mostraban un menor número en el corpus inicial para así poder balancearlo.

2. Evaluación del rendimiento de los modelos optimizados en la clasificación de emociones y polaridad mediante el uso de métricas de exhaustividad, puntuación F1 y precisión.

3. Comparación de los resultados de los modelos anteriores con el modelo actual para observar si ha habido una mejora.

1.3 Metodología

La metodología que se ha seguido para poder desarrollar los objetivos anteriormente descritos se ha estructurado en las fases siguientes:

1.3.1 Fase de documentación

En esta fase, se ha llevado a cabo un estudio detallado de los conceptos clave necesarios para el análisis de sentimientos en comentarios de publicaciones sobre salud mental en redes sociales, en concreto en Instagram, mediante la optimización de un modelo de OpenAI [4]. Para ello ha sido necesario explorar en profundidad las características de estos modelos como los pasos a seguir para llevar a cabo dicha optimización. También se han investigado herramientas y librerías que facilitarán la

implementación y ajuste del modelo gpt-3.5-turbo, asegurando así una base sólida para el desarrollo del trabajo.

1.3.2 Fase de análisis

Esta fase, fue centrada en la recopilación de los datos en Instagram, en la que fue necesaria la evaluación de diferentes estrategias para la selección de los datos para que nuestro modelo resultante tuviera el mayor rendimiento posible. Para ello, se analizaron diversas publicaciones y comentarios con el fin equilibrar el corpus inicial. Asimismo, se analizaron los resultados de trabajos anteriores para obtener una idea más clara de los resultados esperados y para identificar qué emociones podrían presentar mayores desafíos de interpretación. Este análisis previo fue fundamental para ajustar la selección de datos y asegurar que el modelo resultante fuera capaz de manejar con precisión tanto las emociones más comunes como aquellas más difíciles de interpretar en el contexto de la salud mental. Para la comparación y análisis de los resultados se han utilizado el TFG de Javier Estévez Asensio [2] y el TFG de Miguel Carralero Lanchares [3]. En el caso del contexto de videojuegos y Twitch se tomó el corpus ampliado generado en el TFG de Miguel Carralero Lanchares [3].

1.3.3 Fase de prueba

Se trata de la fase en la que se ha optimizado el modelo mediante los datos de entrenamiento y los hiperparámetros. Además, en esta fase se obtienen los resultados de las métricas de rendimiento para su comparación con los resultados de estudios previos para identificar si ha obtenido una mejora.

1.3.4 Fase de escritura

En esta se ha redactado el presente documento, en el que se han detallado, los objetivos, metodología, las pruebas, los resultados y las conclusiones a las que se han llegado en este trabajo. Además, se ha explicado de manera detallada los pasos que se han seguido, así como las decisiones que se han ido tomando a lo largo del proceso.

1.4 Estructura de la memoria

La memoria está estructurada en los siguientes capítulos:

Capítulo 1: Introducción, donde se aborda el problema a desarrollar con los objetivos propuestos y la metodología para poder cumplir dichos objetivos.

Capítulo 2: Estado del arte en el que se explica la relación que hay entre el modelo GPT-3.5, ChatGPT y el procesamiento de lenguaje natural. Además, se explica la relevancia del uso de los modelos que cuentan con la arquitectura *Transformer*. Por último, se proporciona una contextualización sobre el análisis de sentimiento en redes sociales y el impacto que pueden tener en nuestra salud mental.

Capítulo 3: Descripción de las principales herramientas utilizadas para el desarrollo del estudio y su análisis.

Capítulo 4: Se describe el corpus inicial, así como la estrategia seguida para la obtención de nuevas publicaciones para la ampliación del corpus. Se incluyen tanto los criterios de selección de publicaciones e influencers, así como el proceso de etiquetado de los comentarios.

Capítulo 5: Capítulo en el que se realiza el análisis de sentimientos y polaridad para Instagram utilizando el modelo gpt-3.5-turbo. Se detallan los datos de los que partimos, el proceso de optimización del modelo, el coste de dicho proceso y los resultados obtenidos. Por último, se evalúan los resultados obtenidos de las métricas de rendimientos y se comparan con los obtenidos en estudios previos.

Capítulo 6: Se detalla el análisis de emociones y polaridad en Twitch utilizando el modelo gpt-3.5-turbo. Para ello, se presentan los datos de partida con los que contamos, el coste del proceso de optimización, el proceso de optimización y la obtención de los resultados. Por último, se examinan los resultados obtenidos por las métricas de rendimiento y se comparan con los de estudios previos.

Capítulo 7: Se presentan las conclusiones finales, así como las dificultades encontradas y se proponen futuras líneas de investigación para la mejora del trabajo realizado.

2

Estado del arte

2.1 Introducción

Este capítulo abordará una explicación acerca de ChatGPT y de algunos de los modelos con los que cuenta OpenAI, siendo uno de ellos en el que nos hemos basado para la realización de este TFG. Además, se realizará un estado del arte sobre la relación que hay entre la salud mental y las redes sociales.

2.2 Marco teórico sobre chatgpt y los modelos de OpenAI

La Inteligencia Artificial (IA) ha experimentado un gran progreso, en gran parte gracias a los avances en el Aprendizaje Automático y en el Aprendizaje Profundo. Debido a estos avances las máquinas son capaces de realizar tareas que antes solo podían ser ejecutadas por personas, como la comprensión y generación de lenguaje natural, lo cual ha facilitado la creación de programas como ChatGPT [5].

Además, la introducción de la arquitectura Transformer ha marcado un hito en el procesamiento de lenguaje natural (PLN), ya que ha proporcionado una estructura que mejora la capacidad de los modelos de entender y generar texto. El Transformer utiliza un mecanismo de atención que evalúa la relevancia de distintas partes de un texto de entrada, lo cual facilita la mejor comprensión del contexto y una mayor precisión en las tareas del PLN. Dicho avance es la base de modelos como GPT-3.5 [6].

De esta manera en los últimos años, OpeanAI ha sido capaz de desarrollar múltiples modelos de lenguaje avanzado, entre los que podemos destacar a GPT-3.5 que representa una evolución significativa en la generación de texto coherente y natural.

GPT-3.5 ha sido entrenado con un gran volumen de datos, lo que le ha permitido aprender ciertos patrones complejos de lenguaje [7]. La elección del modelo basado en GPT-3.5, se debe a que, en el momento de la realización de este trabajo, era el modelo más avanzado disponible, superando en capacidad a las versiones anteriores.

Por otro lado, ChatGPT es una herramienta que está diseñada para mantener conversaciones con usuarios. Aprovechando las capacidades de GPT-3.5, ChatGPT puede mantener diálogos naturales, responder preguntas complejas y generar contenido de texto con fluidez y precisión. Esta herramienta se ha convertido en una referencia en la generación de texto automatizado [7,8].

El impacto de GPT-3.5 y ChatGPT en el procesamiento de lenguaje natural ha sido muy significativo. Debido, a que estos modelos han demostrado que es posible alcanzar un alto grado de fluidez y coherencia en la generación de texto [6].

Con GPT-3.5 y ChatGPT, OpenAI ha establecido nuevos estándares en la generación de lenguaje natural. A medida que estos modelos continúan evolucionando, se espera que se integren cada vez más en aplicaciones cotidianas, proporcionando a los usuarios una experiencia interactiva más intuitiva y eficaz. Además, el desarrollo continuo de estos modelos plantea interesantes posibilidades para el futuro del procesamiento del lenguaje, ampliando los límites de lo que las máquinas pueden lograr en términos de generación y compresión del lenguaje, como se presenta en este trabajo en el cual se analiza la respuesta emocional de comentarios de publicaciones en Instagram [7].

2.3 La salud mental en redes sociales

Las redes sociales han aumentado su crecimiento de manera considerable en los últimos años al punto de que más de la mitad de la población global utiliza alguna de ellas, alcanzando un porcentaje de uso del 93,2% y un tiempo medio de usuario de 2 horas y 23 minutos [9]. Actualmente hay más de 5 mil millones de usuarios de redes sociales por todo el mundo distribuidos entre distintas plataformas entre las que podemos destacar Facebook, Youtube, WhatsApp, Instagram y TikTok [10].

Con el auge de estas aparecen las figuras de los influencers que son aquellas personas que han ganado popularidad en las redes debido a su capacidad para atraer la

atención de la gente gracias a la variedad de contenido que abarcan entre los que se incluyen moda, belleza, tecnología y salud.

En los últimos años, estos han empezado a desafiar estigmas mostrando autenticidad y alejándose de imágenes idealizadas que pueden distorsionar la realidad y afectar la autoestima, especialmente entre jóvenes y adolescentes. La necesidad del público joven de mantenerse conectados y seguir las tendencias de los influencers son algunas de las razones por las que su salud mental puede verse afectada [11].

La relación entre la salud mental y las redes sociales es muy compleja. Por un lado, estas plataformas pueden ayudar a intensificar el estrés y la ansiedad debido a la presión que muchas personas sientan por recibir reconocimiento social. Asimismo, la búsqueda de aprobación del resto de personas a través de “me gusta” y comentarios positivos favorecen la creación de una dependencia emocional [11].

La comparación con las publicaciones que se suben de vidas aparentemente perfectas puede intensificar sentimientos de insuficiencia y baja autoestima. Esto provoca y contribuye a un ciclo de autoevaluación negativa, lo que puede acarrear diversos problemas de salud mental [11].

Otro punto para tener en cuenta es el sentimiento de soledad que pueden llegar a producir estas plataformas. Esto se debe a que en un mundo conectado por las redes combinado con un uso excesivo de estas lleva a que las relaciones cara a cara disminuyan [11].

Por otro lado, las redes sociales también pueden tener un efecto positivo en la salud mental debido al acceso que se tienen a grupos y asociaciones de apoyo. Además de permitir conectar a personas que tengan intereses parecidos, lo que puede ayudar a encontrar un apoyo emocional a aquellas que se puedan sentir aisladas en su entorno [11].

En conclusión, el uso de las redes sociales de manera excesiva y descontrolada puede provocar graves repercusiones en la salud mental de las personas. Por lo que es importante promover un uso responsable y consciente, educando especialmente a las jóvenes quienes son los principales consumidores y víctimas de los efectos negativos que

tienen, pero también a los influencers, que son generadores de este tipo de contenidos y que pueden tener una fuerte repercusión e impacto a nivel social entre los jóvenes.

3

Herramientas utilizadas

3.1 Introducción

Para la realización de este trabajo ha sido necesario el empleo de distintas herramientas y plataformas software de las que hablaremos a continuación.

3.2 Python

Python [12] es un lenguaje de alto nivel de programación interpretado y multiparadigma. Fue creado en 1991 por Guido van Rossumy cuenta con aplicaciones en distintos ámbitos, entre los que destacan el desarrollo Web, científico, educacional y desarrollo software.

Además, presenta una serie de características, que lo hacen una elección idónea para el desarrollo de este proyecto, siendo las siguientes:

- Bibliotecas de Ciencia de Datos [13]: Python cuenta con una gran cantidad de bibliotecas que nos facilitan el análisis y manipulación de los datos, así como la visualización y el mejor entendimiento del comportamiento del modelo y los resultados obtenidos.
- Sencillez y legibilidad de la sintaxis: está es muy intuitiva lo que permite el desarrollo rápido de proyectos. Lo que es muy importante a la hora de desarrollar el nuestro.
- Interacción con APIs: gracias a sus bibliotecas especiales y robustas, se nos permite interactuar con APIs externas como la de OpenAI que utilizamos en este proyecto.

En resumen, debido a las características anteriores, se ha podido observar que Python es un lenguaje de programación óptimo para el desarrollo y la optimización de

modelos utilizando OpenAI. No solo por su gran ecosistema de bibliotecas, sino también por su facilidad de uso y capacidad de integración.

3.3 Librerías empleadas

Para la realización de la optimización y obtención de los resultados de los distintos modelos, ha sido necesario el empleo de distintas librerías que nos ha permitido trabajar con los diferentes datos. Estas librerías son las que a continuación se detallan:

- Os: Cuenta con módulos diseñados para proveer de manera versátil distintas funcionalidades, en función del sistema operativo que se use [14].
- Openai: Permite acceder a la API de OpenAI desde aplicaciones escritas en Python. Para ello, cuenta con una selección de clases predefinidas para los recursos de la API [15].
- Wandb: Permite gestionar y optimizar el proceso de desarrollo de modelos. Además, permite registrar y visualizar métricas de entrenamiento u otros datos del proceso de optimización [16].
- Pandas: Utilizada para la manipulación y análisis de datos. Maneja estructuras de datos, como DataFrame, equivalentes a bases de datos [17].
- Json: Nos permite trabajar con datos en formato JSON. Esta librería proporciona entre otras funciones convertir datos entre estructuras de Python y JSON [18].
- Sklearn.metrics: Es una de las más utilizadas para la evaluación y medición del rendimiento de modelos de aprendizaje automático. Incluye funciones como el cálculo de métricas de clasificación y análisis de decisiones [19].
- Ratelimit: Nos ayuda a controlar la velocidad a la que se realizan solicitudes de una función. Esto resulta de gran utilidad cuando trabajamos con APIs que tiene alguna tasa de uso [20].
- Matplotlib.pyplot: Nos ayuda en la construcción de gráficos y visualización de datos de una manera más sencilla. Se trata de una de las librerías más populares debido a su gran conjunto de funciones [21].
- Numpy: Es imprescindible para la computación científica, en la que encontramos funciones matemáticas para operar con arrays y matrices [22].

- Ramdom: Nos proporciona herramientas para generar números pseudoaletorios y realizar distintas operaciones aleatorias [23].
- Collections: Permite la implementación de tipos de datos especializados de contenedores, lo cual puede conllevar a mejores rendimientos [24].
- Tiktoken: Esta biblioteca nos ayuda a manejar los tokens en modelos de procesamiento de lenguaje natural. Entre otras cosas nos ayuda en la tokenización y detokenización de textos [25].

En conclusión, estas librerías son herramientas fundamentales que facilitan el desarrollo del trabajo propuesto. Gracias a ellas y sus funciones seremos capaces de convertir, cargar, analizar, evaluar y visualizar los datos obtenidos.

3.4 Jupyter Notebook

Jupyter Notebook [26] es una plataforma web pionera para desarrollar y compartir documentos computacionales. Esta plataforma proporciona una experiencia de usuario simple y eficiente, que está enfocada en la creación y gestión de documentos.

Su principal ventaja en este proyecto es su capacidad para ofrecer un entorno interactivo. Esto nos permite ajustar parámetros y ver los resultados, mediante la integración de código, visualizaciones y anotaciones en un mismo documento. La integración de esta manera no solo nos facilita una integración rápida y la mejora continua del modelo, sino que también ofrece una mayor comprensión y documentación del proceso.

3.5 Wandb

Wandb [27] es una plataforma que proporciona herramientas avanzadas para la gestión y monitoreo de experimentos relacionados con la optimización de los modelos de aprendizaje automático diseñados. Estos seguimientos nos permiten registrar tanto los hiperparámetros como las métricas relacionado con los modelos que se van a optimizar. Esta plataforma es una de las que en más confían las grandes empresas líderes en el mundo de la inteligencia artificial, como es el caso de OpenAI.

3.6 Instagram

Instagram [28] es una red social y una plataforma que se ha consolidado como un espacio de interacción social a través de la compartición de vídeos, fotos y *stories*. Es principalmente popular en áreas como la moda, el estilo de vida y la gastronomía, aunque también abarca temas relacionados con la educación y el comercio. Esta plataforma se distingue por la comunicación entre creadores de contenido y seguidores mediante comentarios en post, mensajes directos o respondiendo a *stories*. Dichos comentarios suelen ser una fuente importante de información para poder medir el interés y compromiso de los usuarios, analizar la respuesta emocional que producen los contenidos y comprender cuáles son los patrones de interacción que se siguen en la misma. Todo ello, puede ofrecer información significativa a investigadores cuyo objetivo sea comprender el comportamiento de los usuarios en los entornos digitales.

3.7 Métricas de rendimiento

Para la evaluación de la eficacia de los modelos optimizados y poder compararlos con estudios anteriores, debemos utilizar una serie de métricas muy utilizadas.

En el caso de considerar un modelo que clasifique la polaridad de un comentario, si realizamos una predicción, se podrían obtener cuatros resultados posibles en función de la salida que se espera y el resultado que se obtiene:

- *True Positive* (TP): Es el verdadero positivo, este clasifica comentarios positivos correctamente como positivos.
- *False Positive* (FP): Es el falso positivo, este clasifica comentarios negativos como si fueran positivos.
- *False Negative* (FN): Es el falso negativo, este clasifica un comentario que es negativo como si fuera positivo.
- *True Negative* (TN): Es el verdadero negativo, este clasifica un comentario negativo correctamente.

Como se observa en la Figura 1, los datos anteriores se pueden representar en una matriz de confusión, donde las filas muestran las etiquetas verdaderas mientras que las columnas las predicciones.

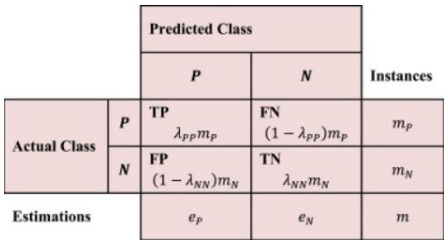


Figura 1: Matriz de confusión. *The impact of class imbalance in classification performance metrics based on the binary confusion matrix*, cap. 2, figura 1 – ScienceDirect [29]

Por otro lado, para poder evaluar el rendimiento de los modelos, se han empleado distintas métricas en función de los resultados obtenidos de la matriz de confusión. Estas son las que a continuación se detallan:

- Precisión (Precision): Mide la proporción de verdaderos positivos entre todos los casos que fueron clasificados como positivos (Ecuación 1).

$$\text{Precisión} = \frac{TP}{TP + FP}$$

Ecuación 1: Función de precisión

- Exhaustividad (Recall): Mide la proporción de verdaderos positivos entre aquellos casos que son positivos (Ecuación 2).

$$\text{Exhaustividad} = \frac{TP}{TP + FN}$$

Ecuación 2: Función de exhaustividad

- Puntuación F1 (F1-score): Mide la media armónica de la precisión y la exhaustividad (Ecuación 3).

$$\text{F1} = 2 \cdot \frac{\text{Precisión} \cdot \text{Exhaustividad}}{\text{Precisión} + \text{Exhaustividad}}$$

Ecuación 3: Función de puntuación F1

- Exactitud (Accuracy): Es la métrica que mide la proporción de todos los casos que han sido clasificados correctamente sobre el total (Ecuación 4).

$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN}$$

Ecuación 4: Función de exactitud

4

Ampliación del Corpus de Salud Mental en Instagram

4.1 Introducción

En este capítulo se detallará tanto el corpus inicial como el proceso de ampliación de dicho corpus. Para ello, se describirá la metodología empleada para la selección y obtención de nuevos comentarios procedentes de publicaciones en Instagram; planteándose como objetivo la obtención de comentarios de las emociones desbalanceadas, en nuestro caso se trataban de gratitud, tristeza/pena y enfado/desprecio/burla. De esta manera se deseaba que la base de datos resultante fuera más completa y equilibrada. Asimismo, nos centramos en incorporar a influencers masculinos no solo con el propósito de equilibrar el corpus en cuestión de género, sino también para poder observar si la reacción era distinta en función del género del influencer (masculino o femenino).

4.2 Descripción del corpus de datos inicial de salud mental en redes sociales

El corpus inicial se trata de una base de datos de salud mental compuesta por comentarios obtenidos en redes sociales, principalmente Instagram, y etiquetados mediante la emoción y polaridad de dicho comentario. Este corpus contaba con 2288 instancias integradas en un fichero .xlsx en el que podemos destacar tres columnas: texto, polaridad y emociones.

En primer lugar, en la columna de texto, encontramos los comentarios escritos por distintos usuarios de Instagram en las publicaciones seleccionadas. Estos se encuentran

en el formato original, a excepción de aquellos en lo que se mencionaba a una persona externa y por ello los hemos anonimizado.

En segundo lugar, en la columna de polaridad, encontramos la polaridad general al que se refiere el comentario, pudiendo ser positivo, negativo o indeterminado. Así, la polaridad positiva se refiere a aquellos comentarios que denotan un sentimiento favorable, como por ejemplo de admiración, "*Qué bonito que compartas con tanta generosidad y apertura esta experiencia!"*. Por otro lado, al contrario que en este caso, aquellos con polaridad negativa provienen de comentarios desfavorables, con tonos de burla o enfado, como podemos observar en el siguiente ejemplo, *"¿Para qué lo haces? ¡que ridículo! ¡No entiendo! todos pasamos por cosas, pero no andamos publicando, me parece que te encanta la atención"*.

Por último, la polaridad indeterminada es aquella en la que ya sea, por que no tengamos el contexto suficiente, o porque puede verse de ambas maneras, no tenemos claro cuál es su polaridad, esto se puede ver en comentarios como *"¿y qué hacemos cuando nos pasa esto?"*.

En tercer lugar, en la columna de emociones podemos diferenciar entre Amor/Admiración, Compresión/Empatía/Identificación, Gratitud, Tristeza/Pena, Enfado/Desprecio/Burla e Indeterminado. Todas estas emociones pasaremos a describirlas a continuación:

- **Amor/Admiración**: Emoción muy compleja, presente en los modelos de Plutchik (2001), Fredrickson (2013) y Bisquerra (2014). Surge con otras emociones positivas y está relacionada con la admiración, la confianza y la aprobación. Algunos de los comentarios de nuestro corpus que denotan estas emociones son: *"¡Qué bonito que compartas con tanta generosidad y apertura esta experiencia!", "Eres un ejemplo a seguir", "Tú eres mucho más fuerte que todo eso, no dejes que te afecte nada de lo que digan o lo que hagan, tu vales oro".*

- **Compresión/Empatía/Identificación**: Cuando respondemos a contenido, el interés se relaciona con comprender el mensaje, empatizar y sentirnos identificados con la situación. Esta conexión puede hacernos sentir que

entendemos a la otra persona porque hemos pasado por lo mismo, como podemos ver en los modelos de Fredrickson (2013) y Plutchik (2001). Algunos ejemplos que encontramos en el corpus son: *"En la misma lucha de la ansiedad, un camino difícil de transitar", "¡Este año me ha tocado vivirlo a mí y nunca me avergüenza decirlo!, ¡Primero estoy yo por encima de todo, y nunca dudé en pedir ayuda!, ¡Nunca dudéis porque se puede vivir la vida tan feliz!", "Me siento identificado. Yo también he empezado este confinamiento con ello y ya me lo llevo para siempre porque me hace bien. Todo el mundo debe probarlo. Cambiaría el mundo."*

- **Gratitud**: Reconocida por varios autores, se trata de percibir y apreciar la amabilidad de alguien. Nos hace querer ser amables y generosos también (Baptista, 2009; Fredrickson, 2013). Esta emoción nos motiva a encontrar nuevas formas de ser atentos y cuidar de los demás (Fredrickson, 2013). Algunos ejemplos de esta emoción en el corpus son: *"Gracias por mostrar tu vulnerabilidad y HUMANIDAD, es de valientes eso, y en estas épocas, una buena forma de abrir conciencia y sensibilizar, sos una joya.", "Gracias! Qué importante hablar así de algo tan común... Qué importante volver a preguntarnos de verdad y esperando la respuesta: "cómo estás?"", "Grande tu!! Eres muy valiente! Y tus canciones hablan de ello y muchas veces han dicho lo q yo quería decir pero no sabía cómo. Gracias"*.

- **Tristeza/Pena**: Aparece cuando perdemos algo importante o nos sentimos decepcionados y sin control. Nos quita la sensación de placer y ocurre ante situaciones de insatisfacción o desagradables. En algunos modelos como Plutchik (2001) y Ekman (2004) aparecen como una emoción básica y negativa. Algunos comentarios de nuestro corpus con esta emoción son: *"¡Te quiero mucho y odio verte sufrir! Es como si una parte de mí se estuviera desmoronando. ¡Eres una de las personas más importantes para mí, y sin duda una de las que más marcó mi vida! y no te imaginas cuánto me duele verte así ...", "Miguel, verte llorar me rompe el corazón, todo estará bien", "Me duele tanto ver esto".*

- **Enfado/Desprecio/Burla**: El enfado surge cuando sentimos que nuestros derechos han sido violados, causando irritación o furia. Mientras que el desprecio es una reacción de repugnancia o rechazo hacia algo que nos desagrada. Asimismo, la burla ocurre cuando alguien ridiculiza o menosprecia a otro, a menudo implicando rechazo social. Todo esto lo podemos encontrar en el modelo de Bisquerra (2014). Algunos ejemplos que podemos encontrar en el corpus son: *"Para qué lo haces que ridículo no entiendo todos pasamos por cosas pero no andamos publicando me parece que te encanta la atención", "Deberías pasar eso sin subirlo a IG, pues igualmente aquí lo que te llegan son likes. Aprende a buscar ese ego en otro lugar que no sea esta mentira. Pese a mencionarlo en tu mensaje has vuelto a lo mismo. Que te vaya bien", "Lamentable".*

- **Indeterminado**: Abarca aquellos mensajes en los que se hace algunas preguntas, pero no expresa alguna de las emociones anteriores: *"¿y qué hacemos cuando nos pasa esto?"*. Por otro lado, también se incluyen algunos comentarios con relación a la religión: *"Somos pequeños ante la Omnipotencia De Dios. Pero quien ora, quien reflexiona y se pone a disposición De Dios recibe aquiescencia y paz de espíritu. Esa es la verdadera felicidad. Que alegría que seas ejemplo para todos nosotros y nos recuerdes que para ser felices necesitamos De Dios y lo que Su Gracia nos entrega: la capacidad de amar, de amarnos y pedir ayuda para seguir con el plan De Dios para nuestras vidas".* Además de algunos en los que no tenemos contexto suficiente para poder etiquetarlo más específicamente.

4.3 Ampliación del corpus de datos de salud mental en Instagram

4.3.1 Estrategia para la selección y obtención de publicaciones

La estrategia de selección y obtención de los posts es crucial para la ampliación del corpus inicial. Estos posts consisten en publicaciones no solo de influencers, es decir, personas conocidas única y exclusivamente en redes sociales; sino que también hemos considerados actores, cantantes, presentadores y escritores. Además, dichos post tratarán

de sus propias revelaciones acerca de la salud mental. En este caso en concreto nos centraremos en el género masculino, buscando balancear el corpus inicial con relevaciones realizadas por hombres y así poder determinar si el tipo de respuesta y reacción emocional es igual en mujeres que en hombres.

Para la selección de un post se han seguido una serie de criterios, que se detallarán a continuación:

- Deben de ser de influencers y/o famosos chicos españoles, debido a que es más sencillo el análisis por temas de lingüística. Además, debían de ser influencers masculinos para poder comparar si la reacción de las personas es distinta por el género.
- Debe tratar de la propia salud mental del influencer, es decir, este debe de hablar de sus revelaciones acerca de ella.
- Debe de estar en la cuenta oficial del influencer. No vale que sea un post de él hablando en un podcast o un programa de televisión sobre su salud mental, si este no está subido en su propia plataforma.
- Los influencers seleccionados deben de tener al menos unos 100 mil seguidores de manera que podemos garantizar que el post tenga una fuerte visibilidad y alcance.
- Cada post debe de tener mínimo unos 200 comentarios, ya que el objetivo era extraer entre 100 y 150 comentarios de cada post para poder ampliar y enriquecer nuestro corpus con nuevas muestras de manera balanceada, tanto para emociones como polaridad. Además, nos aseguramos un mínimo de respuestas para considerar que el post tenga un impacto a nivel social y haya llegado al mayor número de usuarios posible.

Con el objetivo de obtener los mejores resultados posibles hemos seguido la estrategia anteriormente descrita.

4.3.2 Descripción de las publicaciones obtenidas y procedimiento de etiquetado

En este apartado, se describirán los posts obtenidos y el procedimiento que se ha seguido para poder etiquetar los comentarios. Estos se añadieron al corpus inicial con el objetivo de tener un mayor número de datos para poder aumentar la precisión de nuestros

modelos. Las publicaciones fueron seleccionadas siguiendo los criterios descritos en el apartado anterior y, como se mencionó anteriormente, se tratan de publicaciones de hombres españoles con un gran número de seguidores en sus cuentas de Instagram. Dichas publicaciones contienen una o varias imágenes o algún video, que cuentan con un pie de foto en el que hablan de cómo está su salud mental en ese momento.

En cuanto a la selección de los comentarios, esta se hizo leyéndolos en cada publicación, descartando aquellos que eran muy repetitivos, he intentado encontrar la misma cantidad para cada una de las emociones; de tal manera que la selección fuera lo más equitativa posible y el corpus quedará balanceado. Posteriormente, estos comentarios los pasamos a un fichero .xlsx para poder etiquetarlos. La etiquetación de los comentarios se realizó de manera manual mediante la identificación de palabras clave y por el contexto de la publicación. Esta fase de etiquetado fue revisada por profesionales para asegurarnos de que los comentarios estuviesen bien etiquetados, realizando sobre el corpus un doble etiquetado por dos expertos independientes. Un tercer experto fue el encargado de desempatar en aquellos casos de discrepancia entre los dos expertos originales. En caso de no se llegue a ningún consenso entre los tres expertos, el comentario queda descartado.

Seguidamente, se pasará a describir en detalle los posts que fueron seleccionados, los autores y qué emoción predominaba en cada uno de ellos.

- **Beret (@beret1996):**

Beret es un cantante y compositor español. Las letras de sus canciones están llenas de sentimientos, con las que consigue llegar a mucha gente y las utilizan como su fuente de terapia. Actualmente cuenta con 1,2 millones de seguidores en Instagram.

El post que se seleccionó de él es una imagen en negro en la que aparecía un texto dónde explicaba que tiene ansiedad desde hace unos años. Este post lo subió el día de la Salud Mental con el objetivo de empatizar con las personas y hacerlas ver que en la lucha contra la ansiedad no están solas [30].

En total se extrajeron 123 comentarios con distintas clases de emociones como podemos ver en la Tabla 1. Cómo se puede observar las emociones que predominan son

positivas, siendo la mayoría comentarios de gratitud, amor/admiración y compresión/empatía/identificación.

Clase de emoción	N.º de comentarios
Amor/Admiración	33
Compresión/Empatía/Identificación	37
Gratitud	43
Tristeza/Pena	6
Enfado/Desprecio/Burla	2
Indeterminado	2

Tabla 1: Distribución de las emociones del post de Beret

- **Miguel Herrán (@miguel.g.herran):**

Miguel Herrán es un actor español, conocido principalmente por su interpretación de Río en la serie "*La casa de papel"* y Christian en "*Élite"*. En sus redes sociales cuenta con un gran número de seguidores, teniendo en Instagram 11,7 millones.

El post elegido en su caso se trata de un video en el que aparece llorando, y está acompañado por el siguiente texto: "*Podría subir mil fotos chulísimas que he ido recopilando para alimentar está máquina de mentir que es Instagram. Podría inflar mi ego y llenar mi vacío con likes… pero hoy no! Hoy he decidido regalaros una parte sincera de mí… no voy a entrar en detalle de que es lo que me pasa, porqué ni yo mismo lo sé. Pero esto de aquí soy yo. Sin filtros, sin edulcorante y sin mentiras."* Con ese vídeo intenta mostrarse como es, que no siempre se puede estar bien. En una entrevista anterior ya había hablado de sus problemas de salud mental.

Clase de emoción	N.º de comentarios
Amor/Admiración	34
Compresión/Empatía/Identificación	20
Gratitud	7
Tristeza/Pena	31
Enfado/Desprecio/Burla	55
Indeterminado	6

Tabla 2: Distribución de las emociones del post de Miguel Herrán

En este caso se extrajeron 153 comentarios, en la Tabla 2 observamos las distintas clases de emociones presenten en los comentarios. Principalmente cuenta con comentarios negativos, en ellos se burlan de sus lloros o hacían chiste al respecto.

Además, la credibilidad de dicho vídeo de puso en duda en varios de los comentarios. Aunque también hubo comentarios en los que se le agradecía y admiraba por mostrar su lado más humano.

- **Tomás Páramo (@tomasparamo):**

Tomás Páramo es un influencer español que cuenta con 412 mil seguidores en Instagram y cuyo contenido se basa en su día a día, principalmente las fotos que sube son con su mujer e hijos.

En este caso el post que se eligió fue uno en que aparece haciéndose una foto delante de un espejo. Pero se escogió por el texto que acompañaba la foto: "*Llevo un tiempo intentando diferenciar lo esencial de lo importante, tratando así de cuidarme a mí mismo y poniendo el foco en cuidar a los demás, a aquellas personas que multiplican mi vida. Fácil, ¿y por qué me sale al revés?*
Reconozco que llevo unos meses difíciles, incapaz de gestionar el trabajo para hacerlo compatible con mi salud mental, queriendo organizar cada punto de mi vida sin pensar que cada día es una aventura en la cual organizar y buscar una mente cuadriculada de nada sirve. Entregando a mis hijos el tiempo que el trabajo me permite y no el que el corazón me pide, valorando el tiempo con mis amigos como un capricho a cuentagotas en vez de una obligación para descansar y disipar mi mente. Cargando con Dios a mis espaldas porque egoístamente soy consciente de que sin Él no puedo pero a Él tampoco le estoy cuidando y entregando mis horas pidiéndole descansar en sus brazos huyendo al mismo tiempo de ellos. Y, sin mirar a María a los ojos buscando esa intimidad de amor, la complicidad que multiplica y el gesto que hace que cada día el árbol siga creciendo. El otro día volví a llamar a mi médico, necesitaba su ayuda para poder seguir. Me daba miedo e incluso me sentía débil al pensar que estaba "recayendo" en un lugar al que jamás quiero volver, a ese pozo negro llamado depresión. Sabía que el abismo estaba cerca y ni por asomo quería acercarme. Así que sí, me enfrenté a mi orgullo y pedí ayuda. Estoy feliz, mentiría si digo lo contrario, pero hay días en los que me invade la tristeza, días en los que me dejo llevar por un mundo en el que la tentación al dinero, a lo material, al ego, a las apariencias, al egoísmo me llevan por delante, me alzan muy arriba pero, de pronto me doy cuenta que de ahí se cae, y un vacío enorme inunda mi alma porque siento que de nuevo me olvido de lo esencial. Que ese lugar al que muchos

confunden con el cielo, está muy lejos de ser el patio de nuestra felicidad. Y de pronto, me doy cuenta de que he vuelto a fallar en mi intento de diferenciar lo esencial de lo importante. (+)". En dicho texto hace referencia a que está recayendo en una depresión, pero que aun así intenta seguir adelante y no quiere rendirse.

De aquí extrajimos 70 comentarios que fueron etiquetados en las distintas clases de emociones como observamos en la Tabla 3. En este caso, se encontró una gran cantidad de comentarios positivos, entre los que destacan los de admiración por mostrar cómo se siente, los mensajes que recibe de apoyo, la cantidad de personas que empatiza con él y se identifican con su situación.

Clase de emoción	N.º de comentarios
Amor/Admiración	21
Compresión/Empatía/Identificación	27
Gratitud	12
Tristeza/Pena	1
Enfado/Desprecio/Burla	0
Indeterminado	9

Tabla 3: Distribución de las emociones del post de Tomás Páramo

- **Jaime Lorente (@jaimelorentelo):**

Jaime Lorente es un actor y cantante español, conocido por sus interpretaciones en las series de *"La casa de papel"* y *"Élite".* Actualmente en Instagram cuenta con 12 millones de seguidores. El post seleccionado, en su caso, se trata de tres fotografías que forman parte de una entrevista que realizó en la televisión en la que se habló de la salud mental y su importancia, contando experiencias propias. Dichas fotos iban acompañadas de un texto en el que explicaba de que trataba la entrevista: "*Esta noche se emite el @salvadostv en el que tuve el placer de estar charlando y compartiendo sobre salud mental. En un ejercicio de humildad y humanidad máxima cualquier gesto suma para ponerle voz a este monstruo invisible pero no invencible"*. De dicho post se extrajeron 119 comentarios, y mediante la Tabla 4 podemos observar las emociones que predominan en ellos. En este caso, la emoción que predomina es la de gratitud, es decir, prácticamente todos los comentarios que recibió fueron positivos, le daban las gracias por sus palabras ya que estas podían llegar a ayudar a mucha gente.

Clase de emoción	N.º de comentarios
Amor/Admiración	13
Compresión/Empatía/Identificación	1
Gratitud	103
Tristeza/Pena	0
Enfado/Desprecio/Burla	1
Indeterminado	1

Tabla 4: Distribución de las emociones del post de Jaime Lorente

- **Jon Kortajarena (@jonkortajarena):**

Jon Kortajarena es un supermodelo y actor español, quien ha trabajado con marcas mundialmente conocidas como *Tom Ford* o *Versace.* Cuenta con un gran número de seguidores en sus redes sociales, destacando Instagram en la que tiene 4,2 millones.

El post seleccionado en su caso es una foto en la que sale meditando, le ayuda cuando sufre ansiedad. A esta foto la acompañó con un texto, en el que explica que es lo que hace él cuando sufre ansiedad o no se encuentra bien mentalmente. El texto de la foto es el que detallo a continuación: "*Casi todos sufrimos ansiedad, depresión y angustia en algunos momentos de la vida. No es algo de lo que haya que avergonzarse. Todo lo contrario, enfrentarse a eso, es de valientes. Yo empecé a hacer trabajos de respiración y meditación hace algo más de un mes, y aunque suene a cliché, os aseguro que ha cambiado mi vida. El poder de la respiración y de la conciencia, es enorme. Conectar con nuestro cuerpo, y desde ahí escuchar y sentir todo lo que el subconsciente trata de decirnos.. atravesar nuestras emociones de una forma honesta, valiente y sin "atajos", hace que salgamos aprendiendo lecciones muy importantes. El miedo nos hace mirar para otro lado, tiene mil trucos para que huyamos, pero si no* te enfrentas a ello y aprendes de verdad, siempre te persiguen. *Pruébalo, serás más libre*."

Clase de emoción	N.º de comentarios
Amor/Admiración	14
Compresión/Empatía/Identificación	21
Gratitud	33
Tristeza/Pena	2
Enfado/Desprecio/Burla	17
Indeterminado	6

Tabla 5: Distribución de las emociones del post de Jon Kortajarena

De este post se extrajeron 93 comentarios, cuyas emociones detallamos en la Tabla 5. Podemos observar que la mayoría de los comentarios que obtuvo fueron positivos. Destacan los mensajes de gratitud debido al consejo aportado y de identificación ya que mucha gente se veía en la misma situación que él. Pero también se han observado algún que otro comentario negativo, principalmente burlándose de él y de la situación.

- **Ángel Martín (@angel_mg):**

Ángel Martín es un cómico, actor y escritor español. En sus libros, habla en primera persona, sobre la locura y el brote psicótico que sufrió [31]. En su cuenta de Instagram cuenta actualmente con 946 mil seguidores. El post que se escogió en este caso es una foto en la que aparece el texto "*Para los que, de vez en cuando, no sabemos por dónde empezar".* Con esta frase, él se incluye en ese grupo de personas; además de comentar que publica otro libro en el que escribe consejos, que a él le ayudaron en su momento, con el objetivo de que puedan ayudar a más gente. De este post se extrajeron 46 comentarios, y mediante la Tabla 6 podemos observar la clase de emociones que destacan entre ellos. Los comentarios que recibió fueron positivos, destacando la gratitud de la gente por los consejos que va a publicar y las palabras de admiración por la valentía que tiene al hablar sobre lo que le pasó.

Clase de emoción	N.º de comentarios
Amor/Admiración	6
Compresión/Empatía/Identificación	9
Gratitud	29
Tristeza/Pena	1
Enfado/Desprecio/Burla	1
Indeterminado	0

Tabla 6: Distribución de las emociones del post de Ángel Martín.

En conclusión, como se ha podido observar de los posts seleccionados, en general, la respuesta de las personas es positiva a las publicaciones de los protagonistas a excepción de algún caso en el que los comentarios de burla superan los de gratitud, compresión y admiración.

Además, gracias a realizar el etiquetado de manera manual, se han podido descartar aquellos comentarios que no estaban bien definidos para poder tener un corpus

final lo más detallado posible. De tal manera que esto nos ayudara en la fase de construcción de nuestros modelos para que esto puedan ser más precisos.

Por último, en la Tabla 7, tenemos un breve resumen de los posts utilizados en esta sección, en la que se especifica el enlace a dicho post, el número de comentarios totales de la publicación y el número de comentarios que nos descargamos para poder ampliar el corpus. Esta información la completamos con el influencer al que pertenece el post y el número de seguidores que tiene en Instagram.

Influencer	**N.º de seguidores**	**Enlace**	**Nº comentarios post totales**	**Nº comentarios post bajados**
Beret	1,2 M	https://www.instagram.com/p/CyOXuInNElu/?igshid=MTc4MmM1YmI2Ng%3D%3D		123
Miguel Herrán	11,7 M	https://www.instagram.com/p/Bzk_vdJFPY-/?igshid=MTc4MmM1YmI2Ng%3D%3D	42260	153
Tomás Páramo	412 mil	https://www.instagram.com/p/CkeAjPfLUUJ/?igsh=MTExdDRmN2d3ZzFmYg==	233	76
Jaime Lorente	12 M	https://www.instagram.com/p/CZW-UjgMyn_/?igshid=ODh2M3FjNTNiZDJs	1014	119
Jon Kortajarena	4,2 M	https://www.instagram.com/p/CA-x_2_FQBE/	1051	93
Ángel Martín	946 mil	https://www.instagram.com/p/Cy-1ElRIgug/	307	46

Tabla 7: Tabla comparativa de todos los posts descargados.

5

Análisis de la respuesta emocional en Instagram con ChatGPT: Polaridades y Emociones

5.1 Introducción

En este capítulo, se analizará la respuesta emocional que tiene los comentarios en publicaciones de Influencers, masculinos y femeninos, en las redes sociales Instagram y TikTok. En primer lugar, se realizará una breve descripción de la base de datos de la que partimos, para poder contextualizar los datos posteriormente obtenidos. Seguidamente, se explicará el modelo gpt-3.5-turbo-0125, que será el modelo base para el entrenamiento de nuestros datos. Posteriormente, se explicará la metodología utilizada para la preparación de los datos de entrenamiento, como el código desarrollado para la realización del proceso de fine-tuning. Finalmente, se procederá al análisis de los datos obtenidos, con el objetivo de evaluar la precisión en la detección de emociones y polaridad expresadas en los comentarios de Instagram.

5.2 Descripción del corpus de datos ampliado

En este apartado, nos centraremos en el corpus ya ampliado, cuyo corpus inicial se ha descrito en secciones anteriores.

Los datos del corpus final, recogidos en formato .xlsx, constan de 4372 instancias, en las que se recogen el comentario de la publicación que se va a analizar, la polaridad y la emoción del mismo y el influencer del post al que pertenecen.

En la Figura 2, se presenta la distribución en función de la polaridad de los comentarios que hay en la base de datos, apareciendo 2751 comentarios de polaridad "Positiva", 1281 comentarios de polaridad "Negativa" y 340 comentarios "Indeterminado". Por otro lado, en la Figura 3, se puede observar la distribución del corpus en relación a la emoción de cada comentario. En este caso son, 1267 comentarios de "Amor/Admiración", 1148 comentarios de "Comprensión/Empatía/Identificación", 830 comentarios de "Enfado/Desprecio/Burla", 548 comentarios de "Gratitud", 293 comentarios de "Tristeza/Pena" y 286 comentarios "Indeterminado".

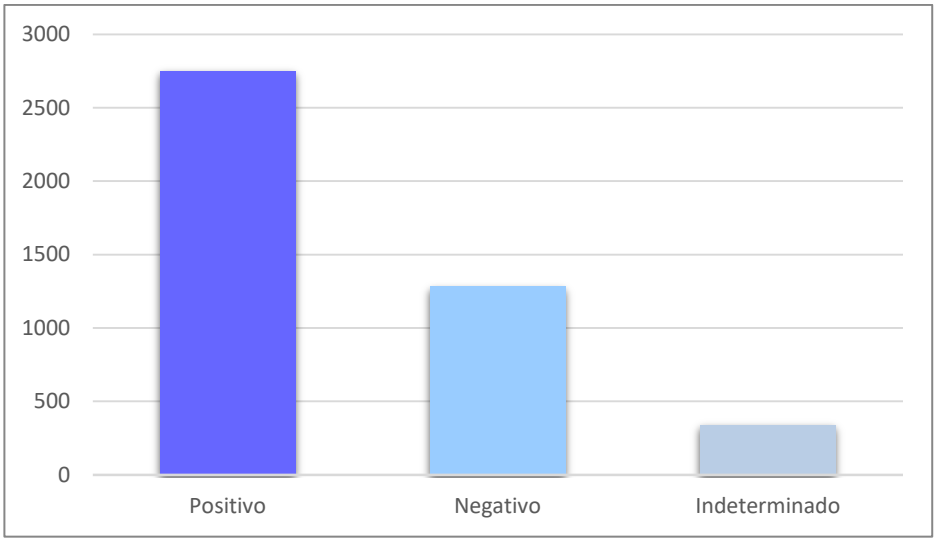


Figura 2: Distribución de polaridades en la base de datos

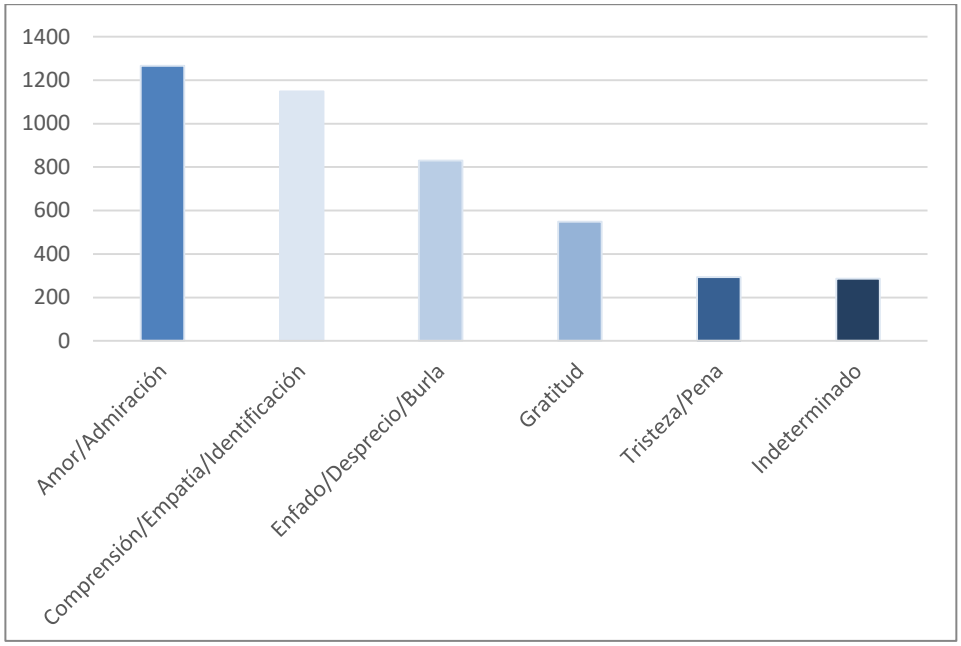


Figura 3: Distribución de emociones en la base de datos

5.3 Modelo ChatGPT utilizado: gpt-3.5-turbo

Debido a su constante evolución y desarrollo, los modelos Transformador Generativo Preentrenado (GPT, *Generative Pre-trained Transformer*) están revolucionando el campo de los chatbots. Creados por OpenAI, son modelos de lenguaje autorregresivo que emplea el aprendizaje profundo para generar textos que simulan la escritura humana. Estos modelos constan de una arquitectura basada en transformadores [32].

Nosotros, utilizaremos los modelos GPT-3.5-turbo, una subclase de los modelos GPT-3, creada en 2022 y que cuenta con 175 mil millones de parámetros. Estos modelos pueden comprender y generar texto natural con gran facilidad, pero también se pueden emplear para tareas no específicas de chat. Es decir, presentan una gran versatilidad gracias a su entrenamiento en una gran diversidad de datos [33].

Dentro de la selección de modelos que ofrece GPT-3.5-turbo, hemos optado por utilizar como modelo base para el fine-tuning el gpt-3.5-turbo-0125 debido a que presenta una mejora en la precisión y permite una mayor ventana de contexto [33]. Esto es importante porque la longitud de los datos de entrenamiento se verá limitada a ese valor y, en caso de superarlo, los datos se truncarán.

Para nuestro proyecto, hemos entrenado el modelo anterior para el análisis de sentimientos, en concreto emociones y polaridad. De esta manera, podrá enfocarse en características específicas del texto que son importantes para detectar emociones y sentimientos, mejorando así la precisión en la clasificación de emociones.

5.4 Preparación de los datos para el entrenamiento

Una vez finalizada la ampliación del corpus y antes de entrenar el modelo, es importante asegurarnos de que el conjunto de datos esté en el formato adecuado, no contenga errores y cumpla con las condiciones necesarias.

5.4.1 Formato para la base de datos

El formato que debe tener la base de datos de entrenamiento, debe ser un conjunto de conversaciones similar a las conversaciones que posteriormente se querrán obtener una vez el modelo haya sido entrenado.

En nuestro caso, como utilizaremos el modelo base gpt-3.5-turbo-0125, será necesario que nuestro conjunto de datos se encuentre en un fichero JSON de extensión .jsonl que siga la estructura que se muestra en la Figura 4. Donde debemos destacar tres roles:

- *System: M*ensaje especificado por el desarrollador para guiar la respuesta de modelo [34]. Para el caso de las emociones se indicó: *"¿Cuál es el sentimiento del siguiente texto? Responde con 'Amor/Admiración', 'Tristeza/Pena', 'Indeterminado', 'Gratitud', 'Enfado/Desprecio/Burla' o 'Comprensión/Empatía/Identificación'."* Y para el caso de polaridad: *"¿Cuál es la polaridad del siguiente texto? Responde con 'Positiva', 'Negativa' o 'Indeterminado'."*

- *User:* Es la pregunta o mensaje que envía el usuario [34]. En este caso, tanto para emociones como polaridad correspondería con el comentario de Instagram.

- *Assistant*: Mensaje que ha sido generado por el modelo entrenado como respuesta al mensaje del usuario [34].

{"messages": [{"role": "system", "content": "\u00bfCual es la polaridad del siguiente texto? Responde con 'Positiva', 'Negativa' o 'Indeterminado'."},
{"role": "user", "content": "Beret eres muy importante en mi vida te escucho desde esos d\u00edas de tristeza"},
{"role": "assistant", "content": "Positiva"}]}

Figura 4: Formato de la base de datos para el entrenamiento

Para convertir el corpus inicial que está en Excel a un fichero JSON, se desarrollaron los códigos en Python que aparecen en la Figura 5 y 6. Como podemos observar, en ambos casos, primero leeremos el archivo Excel y definimos el archivo de salida que, en caso de emociones definimos como *base_datos.jsonl* y para polaridad como *base_datos_polaridad.jsonl*. Seguidamente, abrimos el archivo de salida y escribimos en él siguiendo la estructura anteriormente explicada, donde en ambos casos el rol de usuario es el mismo *row['Text']*, que indica la columna donde se encuentran los comentarios en el corpus.

La diferencia de este proceso para emociones (Figura 5) y polaridad (Figura 6) son los mensajes en los roles de *system* y *assitant*. Para el caso de *system* escribiremos lo indicado en el apartado anterior; mientras que en el de *assitant*, para emociones, especificaremos la columna de emociones del corpus inicial mediante el código *row['Emotions']*. Para polaridad indicamos su columna correspondiente mediante *row['Polarity']*.

```
import pandas as pd
import json

# Si es un archivo Excel
df = pd.read_excel('CorpusSaludMentalCompleto.xlsx')

# Define el nombre del archivo de salida
output_filename = "base_datos.jsonl"

# Convierte al formato JSONL y escribe en el archivo
with open(output_filename, "w") as file:
    for _, row in df.iterrows():


        # Construye el objeto JSON
        data = {
                "messages": [
                    {
                        "role": "system",
                        "content": "¿Cual es el sentimiento del siguiente texto? Responde con 'Amor/Admiración', 'Tristeza/Pena',
                    },
                    {
                        "role": "user",
                        "content": row['Text']
                    },
                    {
                        "role": "assistant",
                        "content": row['Emotions']
                    }
                ]
        }

        # Escribe en el archivo JSONL
        file.write(json.dumps(data) + "\n")
```

Figura 5: Fragmento del código para pasar del formato Excel a JSON para la construcción de la base de datos para emociones

```
import pandas as pd
import json

# Si es un archivo Excel
df = pd.read_excel('CorpusSaludMentalCompleto.xlsx')

# Define el nombre del archivo de salida
output_filename = "base_datos_polaridad.jsonl"

# Convierte al formato JSONL y escribe en el archivo
with open(output_filename, "w") as file:
    for _, row in df.iterrows():


        # Construye el objeto JSON
        data = {
                "messages": [
                    {
                        "role": "system",
                        "content": "¿Cual es la polaridad del siguiente texto? Responde con 'Positiva', 'Negativa' o 'Indetermina
                    },
                    {
                        "role": "user",
                        "content": row['Text']
                    },
                    {
                        "role": "assistant",
                        "content": row['Polarity']
                    }
                ]
        }

        # Escribe en el archivo JSONL
        file.write(json.dumps(data) + "\n")
```

Figura 6: Fragmento del código para pasar de formato Excel a JSON para la construcción de la base de datos de polaridad

5.4.2 División de la base de datos

Una vez que tenemos ya los datos en el formato deseado, es hora de dividirlos en los datos de entrenamiento y los datos de prueba. Para ello, haremos una división aleatoria de la base de datos original, donde el 70% de las muestras corresponderán con los datos de entrenamiento y el 30% restante con los datos de prueba [35].

En la Figura 7 se muestra el código usado para esta división tanto para emociones como para polaridad. En el código se define la función *split_json()*, que tiene como argumentos de entrada los que a continuación se detallan:

- **Input_file**: Es el fichero que se va a dividir, en nuestro caso es la base de datos (base_datos.jsonl)
- **Output_file_1**: Fichero que se queda con el 70% de la base de datos principal. Siendo esta la base de datos para el entrenamiento (training_file.jsonl).
- **Output_file_2**: Fichero que se queda con el 30% de la base de datos, que correspondería con la base de datos para la prueba (test_file.jsonl).

La función *split_json()* abre el fichero *base_datos.jsonl* y carga en *data* línea a línea dicho fichero. Posteriormente, la línea *total_messages = len(data)* indica la longitud total que tiene *data* para saber el número total de mensajes que hay. Seguidamente, indicamos en *num_message_1* y *num_message_2* la cantidad de mensajes que deben contener, el primero el 70% del total y el segundo el resto. Mediante las líneas *shuffled_data = data[:]* y *random.shuffle(shuffled_data)* hacemos una copia de los datos y los barajamos aleatoriamente. Finalmente, dividimos los datos barajados en dos grupos mediante las líneas *data_1 = shuffled_data[:num_messages_1]* y *data_2 = shuffled_data[num_messages_1:]*. Y estos datos los agregamos a los archivos de salida.

```python
#Dividimos el la base de datos entre el training file y el test file
import json
import random

def split_json(input_file, output_file_1, output_file_2):
    with open(input_file, 'r') as f:
        data = [json.loads(line) for line in f]

    total_messages = len(data)

    # Calcula el número de mensajes para cada archivo de salida
    num_messages_1 = int(total_messages * 0.7)
    num_messages_2 = total_messages - num_messages_1

    # Hace una copia de los datos y los baraja aleatoriamente
    shuffled_data = data[:]
    random.shuffle(shuffled_data)

    # Divide los datos barajados en dos grupos
    data_1 = shuffled_data[:num_messages_1]
    data_2 = shuffled_data[num_messages_1:]

    # Escribe los datos en los archivos de salida
    with open(output_file_1, 'w') as f1:
        for item in data_1:
            json.dump(item, f1)
            f1.write('\n')  # Agrega una nueva línea después de cada objeto JSON

    with open(output_file_2, 'w') as f2:
        for item in data_2:
            json.dump(item, f2)
            f2.write('\n')  # Agrega una nueva línea después de cada objeto JSON

# Ejemplo de uso
split_json("base_datos.jsonl", "training_file.jsonl", "test_file.jsonl")
```

Figura 7: Fragmento de código para la división de la base de datos.

Podemos observar en la Figura 8 que se vuelven a pasar los datos de prueba a un archivo Excel, para poder utilizarlo posteriormente en la obtención de los resultados. En este caso, abrimos el fichero de prueba y cargamos los datos línea por línea. Como los datos que queremos extraer, son los comentarios y las emociones o polaridad del comentario, los extraemos de la lista de datos y almacenamos en las listas “textos” y “emociones”. Por último, guardamos dichas listas en un archivo Excel.

```python
# Cargar datos desde el archivo JSON
with open("test_file.jsonl", "r") as file:
    datos = [json.loads(line) for line in file]

# Extraer los datos relevantes del JSON y almacenarlos en listas
textos = []
emociones = []

for dato in datos:
    textos.append(dato["messages"][1]["content"])
    emociones.append(dato["messages"][2]["content"])

# Crear un DataFrame de pandas
df = pd.DataFrame({"Texto": textos, "Emociones": emociones})

# Guardar el DataFrame en un archivo de Excel
df.to_excel("datos_test.xlsx", index=False)
```

Figura 8: Fragmento de código para pasar de formato JSON a Excel.

5.4.3 Verificación de los datos de entrenamiento

Una vez que tenemos el fichero de datos de entrenamiento (*training_file.jsonl*), será necesario comprobar que este no contenga errores. Para ello, utilizaremos el código que se observa en la Figura 9, que nos dará, una vez ejecutado, el número de ejemplos del que consta nuestra base de datos de entrenamiento y si esta presenta algún error.

```
import json
from collections import defaultdict

data_path = "training_file_polaridad.jsonl"

# Load the dataset
with open(data_path, 'r', encoding='utf-8') as f:
  dataset = [json.loads(line) for line in f]

# Initial dataset stats
print("Num examples:", len(dataset))

# Format error checks
format_errors = defaultdict(int)

for ex in dataset:
  if not isinstance(ex, dict):
    format_errors["data_type"] += 1
    continue

  messages = ex.get("messages", None)
  if not messages:
    format_errors["missing_messages_list"] += 1
    continue

  for message in messages:
    if "role" not in message or "content" not in message:
      format_errors["message_missing_key"] += 1

    if any(k not in ("role", "content", "name", "function_call") for k in message):
      format_errors["message_unrecognized_key"] += 1

    if message.get("role", None) not in ("system", "user", "assistant", "function"):
      format_errors["unrecognized_role"] += 1

    content = message.get("content", None)
    function_call = message.get("function_call", None)

    if (not content and not function_call) or not isinstance(content, str):
      format_errors["missing_content"] += 1

  if not any(message.get("role", None) == "assistant" for message in messages):
    format_errors["example_missing_assistant_message"] += 1

if format_errors:
  print("Found errors:")
  for k, v in format_errors.items():
    print(f"{k}: {v}")
else:
  print("No errors found")
```

```
Num examples: 3060
No errors found
```

Figura 9: Fragmento de código para la comprobación de errores en los datos de entrenamiento [36].

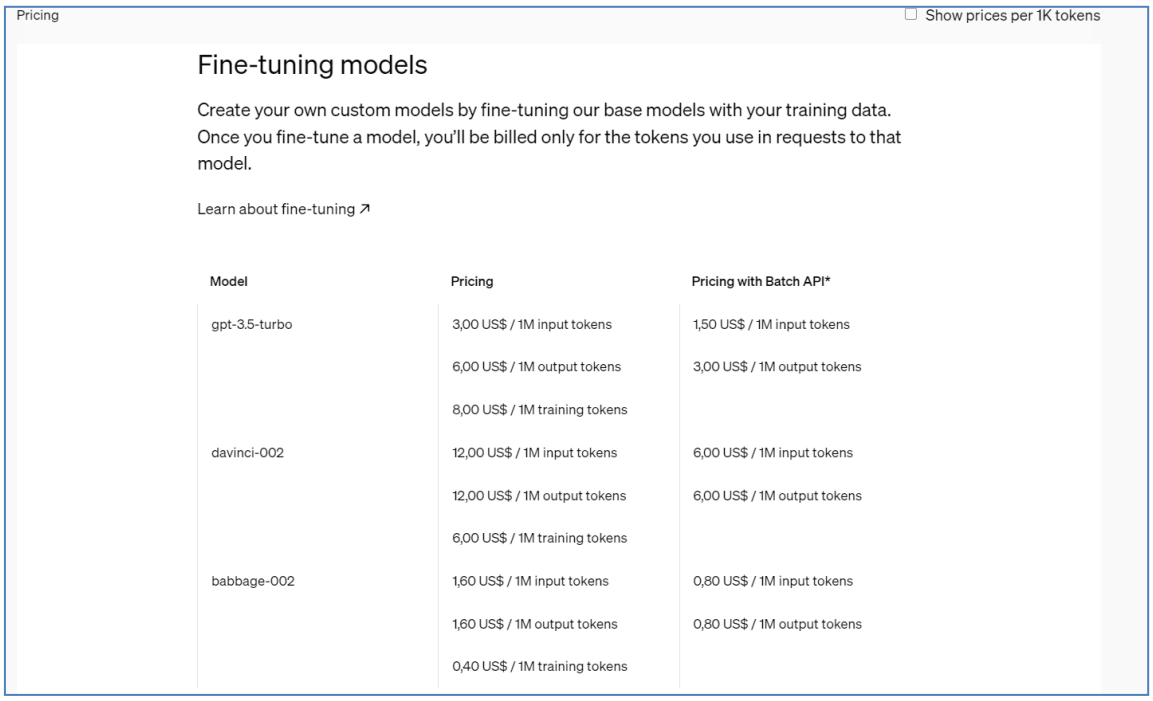
El principal objetivo del código anterior es asegurarnos de que los datos de entrenamiento cumplan el formato necesario para el proceso de fine-tuning. Se realizan varias comprobaciones, en el código, para detectar alguno de los siguientes errores:

- ***Data_type:*** Este error se produce, cuando alguna de las entradas del conjunto de datos, no es del tipo diccionario [36].
- ***Missing_messages_list*****:** Ocurre cuando no hay una lista de mensajes en alguna de las entradas del conjunto de datos [36].
- ***Message_missing_key*****:** Sucede cuando en alguno de los mensajes de la lista de mensaje, faltan las claves de rol y contenido [36].
- ***Message_unrecognized_key*****:** Se produce un mensaje que tiene una clave que no es válida o reconocida [36].
- ***Unrecognized_role:*** Es un error que sucede cuando la clave de rol es distinta de "*user*", "*system*" y "*assistan*t" [36].
- ***Missing_content:*** Aparece cuando el contenido de *content* no son datos textuales [36].
- ***Example_missing_assistant_message***: Cuando no existe al menos un mensaje del asistente [36].

En nuestro caso, observamos que, nuestra base de datos de entrenamiento cuenta con 3060 ejemplos y que no se encontraron errores en él. Sucede tanto para la base de datos de emociones como de polaridad.

5.4.4 Estimación del coste del proceso del fine-tuning con la base de datos de entrenamiento

En este apartado, se realizará una estimación del proceso del fine-tuning. Esto lo hacemos ya que para poder utilizar cualquiera de los modelos de OpenAI hay que pagar. Además, el precio a pagar variará dependiendo del uso que se vaya a hacer con el modelo y el modelo a utilizar. Los diferentes tipos de precios se muestran en la página web de OpenAI, en concreto en el apartado de *pricing* [37]. En la Figura 10, se muestra los precios que nos indica dicha página sobre el proceso de fine-tuning.



Pricing

Show prices per 1K tokens

Fine-tuning models

Create your own custom models by fine-tuning our base models with your training data. Once you fine-tune a model, you'll be billed only for the tokens you use in requests to that model.

Learn about fine-tuning ↗

Model	Pricing	Pricing with Batch API*
gpt-3.5-turbo	3,00 US$ / 1M input tokens	1,50 US$ / 1M input tokens
	6,00 US$ / 1M output tokens	3,00 US$ / 1M output tokens
	8,00 US$ / 1M training tokens	
davinci-002	12,00 US$ / 1M input tokens	6,00 US$ / 1M input tokens
	12,00 US$ / 1M output tokens	6,00 US$ / 1M output tokens
	6,00 US$ / 1M training tokens	
babbage-002	1,60 US$ / 1M input tokens	0,80 US$ / 1M input tokens
	1,60 US$ / 1M output tokens	0,80 US$ / 1M output tokens
	0,40 US$ / 1M training tokens	

Figura 10: Precios de los distintos modelos base disponibles para hacer el fine-tunig [37]

En primer lugar, utilizamos el código de las Figuras 11 y 12 para poder obtener, de manera estimada, el número de tokens en cada uno de los ejemplos de nuestra base de datos, para asegurarnos, que no se superará el número máximo que tenía el modelo que íbamos a utilizar en la fase del fine-tuning. Ya que en el caso de superarse los datos se truncarían y podría producir una pérdida de estos.

También se utilizó dicho código para poder garantizar que no haya conversaciones, en las que no aparezcan mensajes con los roles *system* y *user*, ya que estos son cruciales para el inicio de la conversación y para ver la postura que toma el *assistant*.

```python
import tiktoken
encoding = tiktoken.get_encoding("cl100k_base")

# not exact!
# simplified from https://github.com/openai/openai-cookbook/blob/main/examples/How_to_count_tokens_wit
def num_tokens_from_messages(messages, tokens_per_message=3, tokens_per_name=1):
    num_tokens = 0
    for message in messages:
        num_tokens += tokens_per_message
        for key, value in message.items():
            num_tokens += len(encoding.encode(value))
            if key == "name":
                num_tokens += tokens_per_name
    num_tokens += 3
    return num_tokens

def num_assistant_tokens_from_messages(messages):
    num_tokens = 0
    for message in messages:
        if message["role"] == "assistant":
            num_tokens += len(encoding.encode(message["content"]))
    return num_tokens

def print_distribution(values, name):
    print(f"\n#### Distribution of {name}:")
    print(f"min / max: {min(values)}, {max(values)}")
    print(f"mean / median: {np.mean(values)}, {np.median(values)}")
    print(f"p5 / p95: {np.quantile(values, 0.1)}, {np.quantile(values, 0.9)}")
```

Figura 11: Funciones utilizadas para el cálculo de tokens [36].

```python
import numpy as np
#Warnings and tokens counts
n_missing_system = 0
n_missing_user = 0
n_messages = []
convo_lens = []
assistant_message_lens = []

for ex in dataset:
    messages = ex["messages"]
    if not any(message["role"] == "system" for message in messages):
        n_missing_system += 1
    if not any(message["role"] == "user" for message in messages):
        n_missing_user += 1
    n_messages.append(len(messages))
    convo_lens.append(num_tokens_from_messages(messages))
    assistant_message_lens.append(num_assistant_tokens_from_messages(messages))

print("Num examples missing system message:", n_missing_system)
print("Num examples missing user message:", n_missing_user)
print_distribution(n_messages, "num_messages_per_example")
print_distribution(convo_lens, "num_total_tokens_per_example")
print_distribution(assistant_message_lens, "num_assistant_tokens_per_example")
n_too_long = sum(l > 4096 for l in convo_lens)
print(f"\n{n_too_long} examples may be over the 4096 token limit, they will be truncated during fine-tu
```

```
Num examples missing system message: 0
Num examples missing user message: 0

#### Distribution of num_messages_per_example:
min / max: 3, 3
mean / median: 3.0, 3.0
p5 / p95: 3.0, 3.0

#### Distribution of num_total_tokens_per_example:
min / max: 48, 470
mean / median: 76.35849673202614, 68.0
p5 / p95: 54.0, 107.0

#### Distribution of num_assistant_tokens_per_example:
min / max: 2, 3
mean / median: 2.7029411764705884, 3.0
p5 / p95: 2.0, 3.0

0 examples may be over the 4096 token limit, they will be truncated during fine-tuning
```

Figura 12: Fragmento de código para la estimación del número de ejemplos que serán truncados [36]

Por último, con el código de la Figura 13, obtenemos el número de tokens de toda nuestra base de datos de entrenamiento y poder calcular de manera estimada cuál sería el coste en el proceso de fine-tuning. Para realizar este, solo tendríamos que multiplicar el número de tokens totales por el número de épocas y el precio por token de la fase de fine-tuning.

```
 # Pricing and default n_epochs estimate
MAX_TOKENS_PER_EXAMPLE = 4096
MIN_TARGET_EXAMPLES = 100
MAX_TARGET_EXAMPLES = 25000
TARGET_EPOCHS = 3
MIN_EPOCHS = 1
MAX_EPOCHS = 25

n_epochs = TARGET_EPOCHS
n_train_examples = len(dataset)
if n_train_examples * TARGET_EPOCHS < MIN_TARGET_EXAMPLES:
     n_epochs = min(MAX_EPOCHS, MIN_TARGET_EXAMPLES // n_train_examples)
elif n_train_examples * TARGET_EPOCHS > MAX_TARGET_EXAMPLES:
    n_epochs = max(MIN_EPOCHS, MAX_TARGET_EXAMPLES // n_train_examples)

n_billing_tokens_in_dataset = sum(min(MAX_TOKENS_PER_EXAMPLE, length) for length in convo_lens)
print(f"Dataset has ~{n_billing_tokens_in_dataset} tokens that will be charged for during training")
print(f"By default, you'll train for {n_epochs} epochs on this dataset")
print(f"By default, you'll be charged for ~{n_epochs * n_billing_tokens_in_dataset} tokens")
```

```
Dataset has ~233657 tokens that will be charged for during training
By default, you'll train for 3 epochs on this dataset
By default, you'll be charged for ~700971 tokens
```

Figura 13: Fragmento de código para la estimación del número de tokens totales para la base de datos de polaridad [36].

5.5 Optimización del modelo GPT: Proceso de Fine-tuning

Una vez tenemos la base de datos de entrenamiento y que nos hemos asegurado de que no presenta ningún tipo de errores, podemos pasar a la fase de fine-tuning, es decir, a la fase de optimización del modelo.

5.5.1 Obtención de una API key

Para poder realizar la optimación de cualquier modelo perteneciente a OpenAI es necesario la obtención de una API key. Los pasos necesarios para obtenerla son los siguientes:

1. Debemos acceder la plataforma de OpenAI: https://platform.openai.com/apps y escoger la opción de API.
2. Deberemos iniciar sesión, en el caso de que tengamos ya una cuenta registrada o por el contrario registrarnos.

3. En la Figura 14, observamos lo que obtendríamos una vez estemos dentro. Pulsar la opción de *Dashboard*.

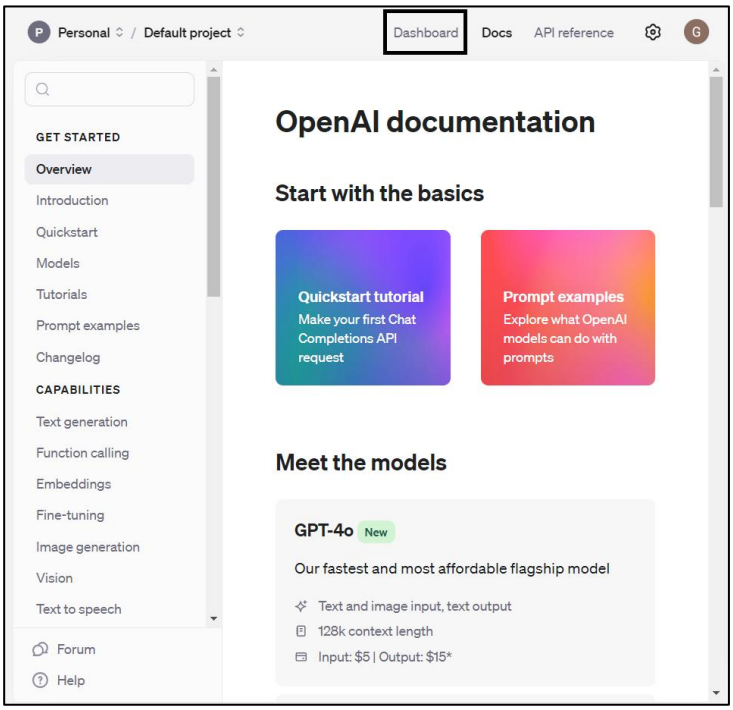


Figura 14: Captura de la página de OpenAI una vez hemos iniciado sesión.

4. Cuando estamos en *Dashboard*, en el panel de la derecha nos aparecerá la opción *API Key*, que es donde deberemos pulsar. Como vemos en la Figura 15.

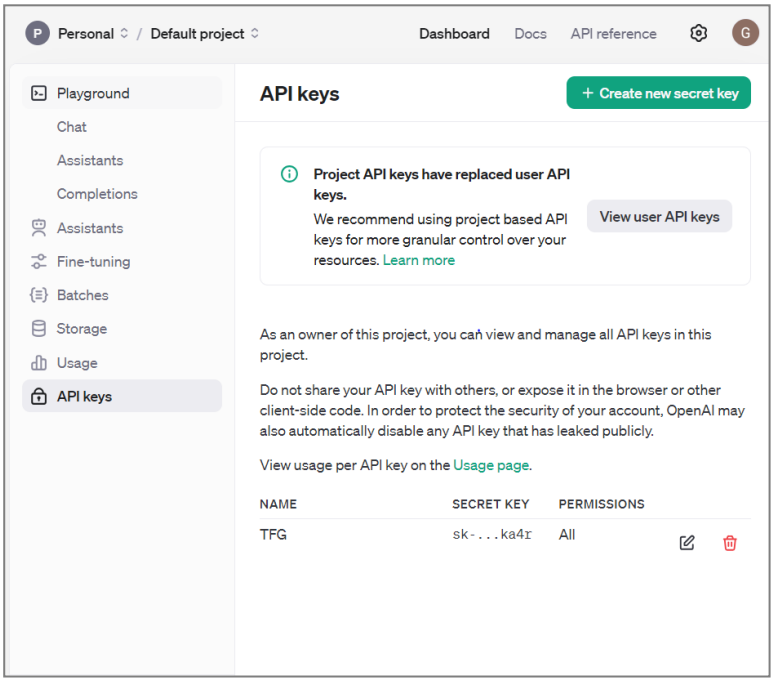


Figura 15: Captura de la página de OpenAI en la sección de API keys.

5. En *API Key*, para poder obtener una nueva clave pulsaremos el botón *Create new secret key*.
6. A continuación, aparecerá una ventana que nos pedirá que seleccionemos los permisos deseados y que opcionalmente indiquemos un título. Una vez seleccionado todo, pulsaremos el botón *Create secret key*, como vemos en la Figura 16.

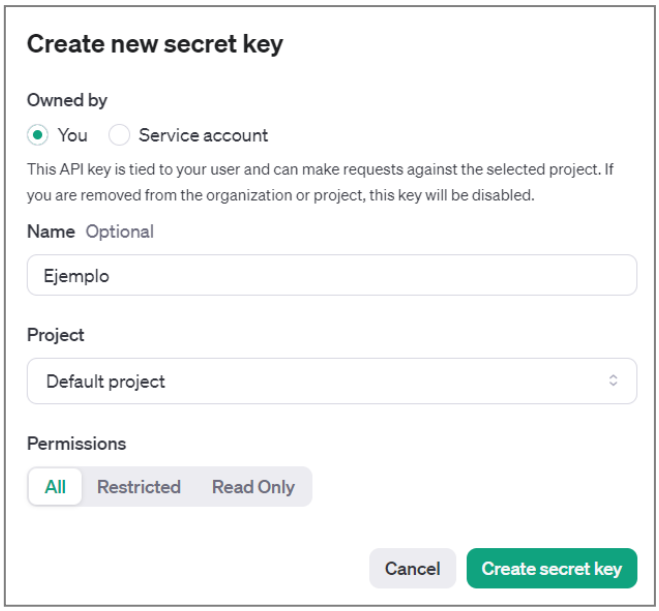


Figura 16: Captura de la ventana para crear una clave de OpenAI

7. Por último, nos aparecerá la clave que deberemos guardar para poder utilizarla posteriormente.

5.5.2 Registro en wandb

Wandb (*Weight & Biases*) [27] se trata de una plataforma que permite no solo registrar métricas y la configuración del proceso de fine-tuning, sino que también nos facilita la mejor comprensión y análisis del desempeño de los modelos optimizados.

Para que en dicha plataforma se registren los cambios durante el proceso de fine-tunig será necesario seguir los siguientes pasos para la sincronización:

1. Se accederá a la página web de wandb: https://wandb.ai/site
2. En el caso de tener ya una cuenta registrada se iniciará sesión y en caso contrario se realizará el registro.
3. La Figura 17 representa lo que se obtiene una vez hemos iniciado sesión. Posteriormente será necesario pulsar la opción *W&B Weave.*

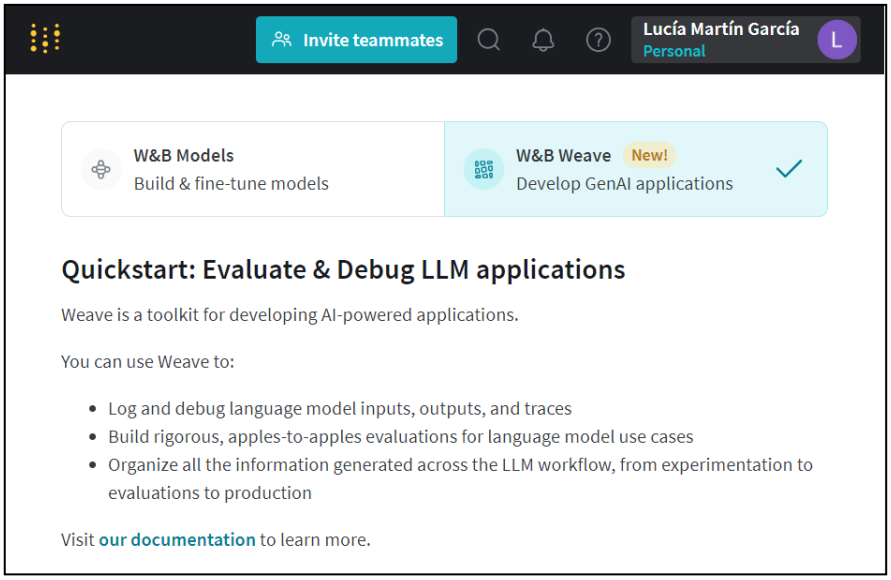


Figura 17: Captura de la página de wandb

4. A continuación, en dicha sección deberemos copiar el código *Your API key*. Como se observa en la Figura 18.

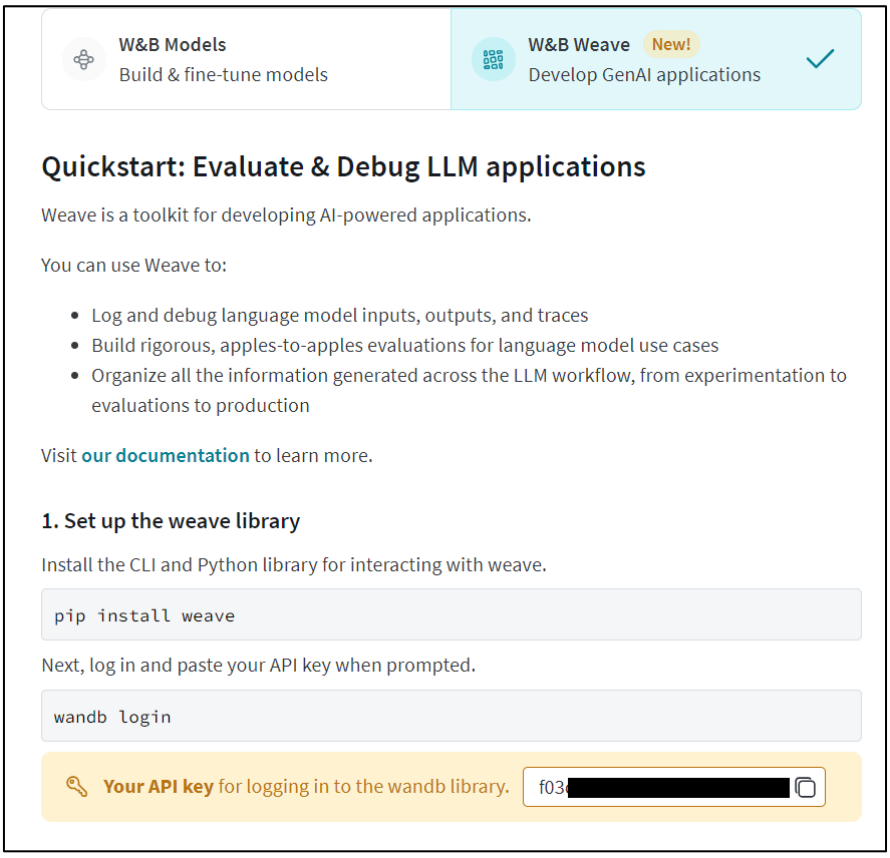


Figura 18: Captura para copiar la API key de wandb.

5. De nuevo accederemos a nuestra cuenta de OpenAI y pulsaremos en el icono de *Settings*. Ahí en la sección de *Organization, General* del menú

vertical, obtendremos lo de la Figura 19. Y copiaremos el código en *Integrations, Weights and Biases*.

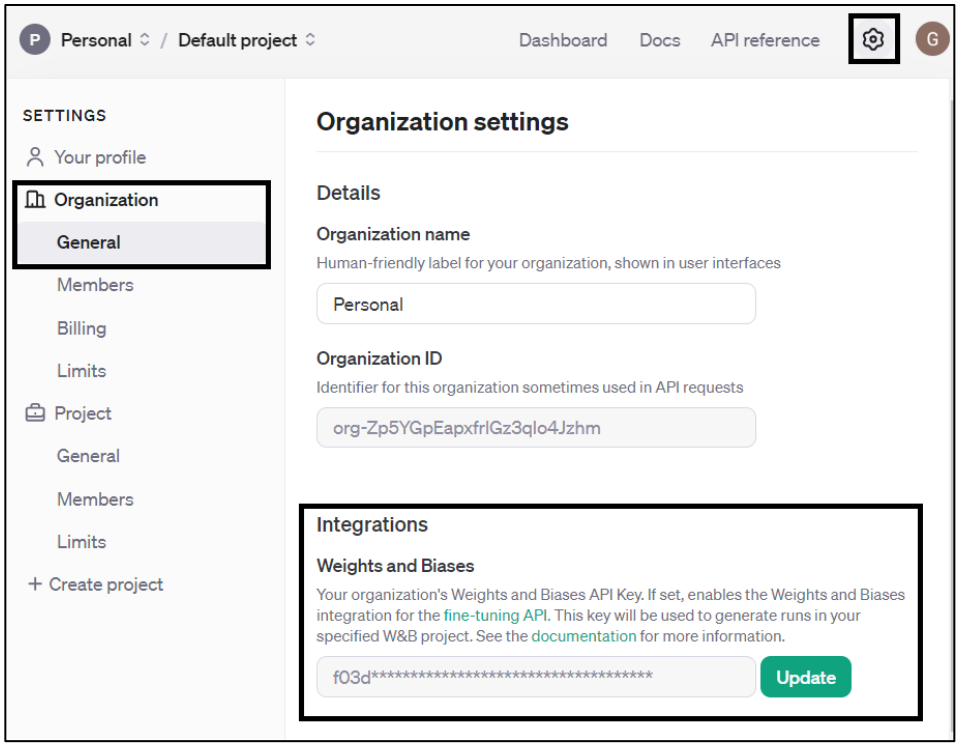


Figura 19: Captura de la interfaz de OpenAI en la que copiamos el API key de wandb

5.5.3 Proceso de optimización del modelo

En esta sección se explicarán los pasos necesarios para poder realizar la optimización del modelo gpt-3.5-turbo. En primer lugar, definiremos a nuestro cliente mediante la siguiente línea de código: *client=OpenAI(apikey="")*, donde entre comillas, indicaremos la clave obtenida en el proceso anterior.

Además, iniciaremos sesión de wandb, mediante la línea: *wandb.init(project="Analisis de sentimientos")*; para que durante el proceso de fine-tuning se agreguen las gráficas de *train_mean_token_accuracy* y *train_loss*.

En la Figura 20, tenemos el código necesario para poder cargar el fichero de entrenamiento en los servidores de OpenAI. Este fichero, será el utilizado para que nuestro modelo aprenda de él. Al ejecutarlo, se nos confirmará que el fichero se ha cargado de manera adecuada y se nos proporcionará un identificador, que será el que debemos indicar en la fase de fine-tuning [39,40].

```
#Cargamos el fichero de entrenamiento

training_file = client.files.create(
  file=open("training_file.jsonl", "rb"),
  purpose='fine-tune'
)

training_file_id = training_file.id
print("File has been uploaded to OpenAI with id ", training_file_id)
```

```
File has been uploaded to OpenAI with id  file-[redacted]
```

Figura 20: Fragmento de código para cargar el fichero de entrenamiento

A continuación, se presenta el código para realizar el fine-tuning, como vemos en la Figura 21. En este fragmento de código, es necesario indicar de manera obligatoria el fichero de datos de entrenamiento que vamos a utilizar; en nuestro caso lo haremos mediante el identificador obtenido anteriormente. Además del modelo que vamos a querer optimizar, se utilizará el gpt-3.5-turbo, que a su vez se redirige al modelo gpt-3.5-turbo-0125.

Por otro lado, de manera opcional, también hemos indicado dentro de la categoría de hiperparámetros el número de épocas, que hemos definido como tres [41]. Hemos tomado la decisión de escoger tres épocas, ya que en los ejemplos sobre este proceso que nos hemos encontrado se especificaba dicho valor para el modelo que estamos utilizando. Otros hiperparámetros que no hemos especificado son el *bacth_size* y el *learning_rate_multiplier*, en este caso no se optó por dar ningún valor, ya que el propio algoritmo de OpenAI busca lo mejores valores de ambos para la base de datos que tenemos y el número de épocas especificadas.

De esta manera, podíamos hacer un cálculo más realista del coste del proceso. Alguno de los resultados, que nos ofrece este proceso, es el *train_mean_token_accuracy* y *train_loss* [42], los cuales guardaremos en wandb, mediante la parte de código de integrations, donde indicamos que el tipo es wandb y el nombre de nuestro proyecto. Los resultados obtenidos de estas gráficas tanto para emociones y polaridad se muestran en las Figuras 22 y 23 respectivamente. Por un lado, la gráfica de *train_loss* representa la pérdida que comete el modelo durante el entrenamiento, en concreto, nos indica la discrepancia entre las predicciones del modelo y los valores reales que se espera que prediga. De esta manera un valor bajo de *train_loss* indica que el modelo está haciendo predicciones más cercanas a las reales, lo que indica una mejor precisión y desempeño. En las gráficas, se observa que la pérdida fluctúa a lo largo del entrenamiento, con

valores que oscilan entre 0 y 0.8 en el caso del modelo de emociones (Figura 22), y hasta 1.2 para el modelo de polaridad (Figura 23). Estas oscilaciones son comunes y pueden deberse a que el modelo intenta continuamente ajustarse a los diferentes lotes de datos (batches). Sin embargo, es notable que, a pesar de las fluctuaciones, las gráficas tienden a estabilizarse en un valor más bajo, lo que sugiere que el modelo está aprendiendo y mejorando en la predicción de los datos, lo que refleja un progreso en el entrenamiento. En cambio, la gráfica del *train_mean_token_accuracy* mide el porcentaje de predicciones correctas realizadas por el modelo sobre los tokens individuales durante el entrenamiento, de manera que funciona como un indicador de la precisión del modelo a nivel de token. Se trata de una métrica crucial porque una alta *train_mean_token_accuracy* indica que el modelo está aprendiendo a realizar predicciones precisas. En ambas figuras, podemos observar que la precisión es consistentemente alta, oscilando alrededor del 95%. Aunque existen pequeñas fluctuaciones, estas son relativamente menores y pueden estar relacionadas con variaciones entre los diferentes lotes de datos presentados al modelo. La alta consistencia de la precisión nos sugiere que el modelo mantiene un buen rendimiento general durante el entrenamiento.

En resumen, las gráficas de ambas figuras muestran que, aunque existen fluctuaciones en la pérdida, el modelo tiende a mejorar y estabilizarse, lo que es un buen indicio de aprendizaje. Además, la alta precisión observada en ambas figuras refuerza el hecho de que el modelo está logrando un desempeño sólido.

```
ft_job = client.fine_tuning.jobs.create(
    training_file=training_file_id,
    model="gpt-3.5-turbo",
    hyperparameters={
        "n_epochs": 3
    },
    integrations=[
        {
            "type": "wandb",
            "wandb": {
                "project": wandb.run.project,
                "name": "mental-health-model"
            }
        }
    ]
)

model_id = ft_job.id
print("Fine Tune Job has been created with id ", ft_job.id)
```

```
Fine Tune Job has been created with id  ftjob-[redacted]
```

Figura 21: Fragmento de código para la optimización de nuestro modelo

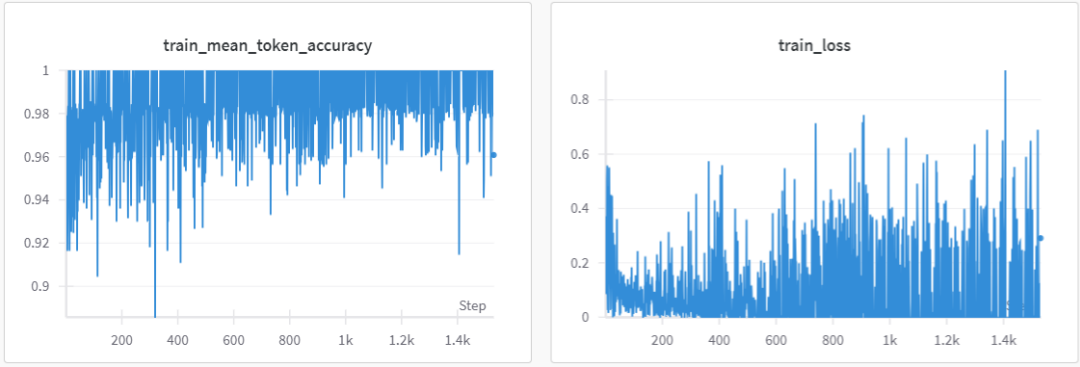


Figura 22: Gráficas del *train_mean_token_accuracy* y *train_loss* para el modelo optimizado de emociones

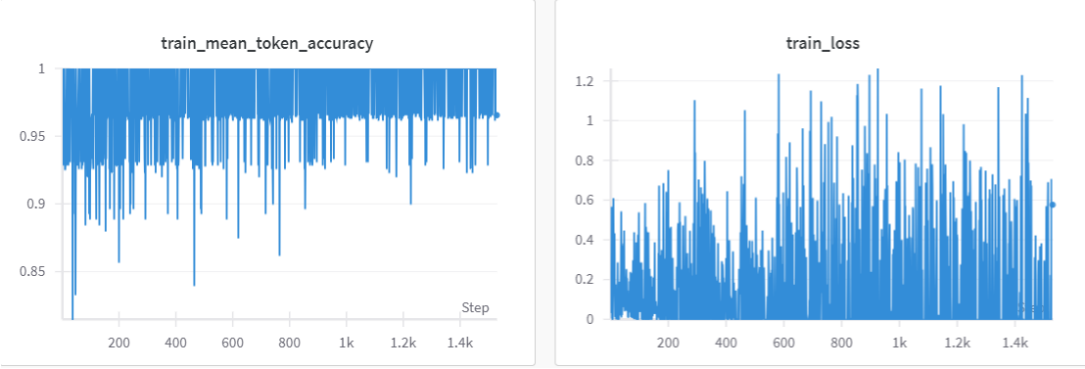


Figura 23: Gráficas del *train_mean_token_accuracy* y *train_loss* para el modelo optimizado de polaridad

Finalmente, obtendremos un identificador que nos indicará que el proceso de optimización se ha creado. Además, podremos observar todos los pasos que se realizan durante el mismo, desde la interfaz de OpenAI, en la sección de *Dashboard*, donde pone *Fine-tuning* en el menú vertical. Como vemos en la Figura 24, nos aparecerán todos los modelos que hayamos optimizado y si pulsamos en alguno de ellos nos darán su información como observamos en la Figura 25, para el caso de emociones.

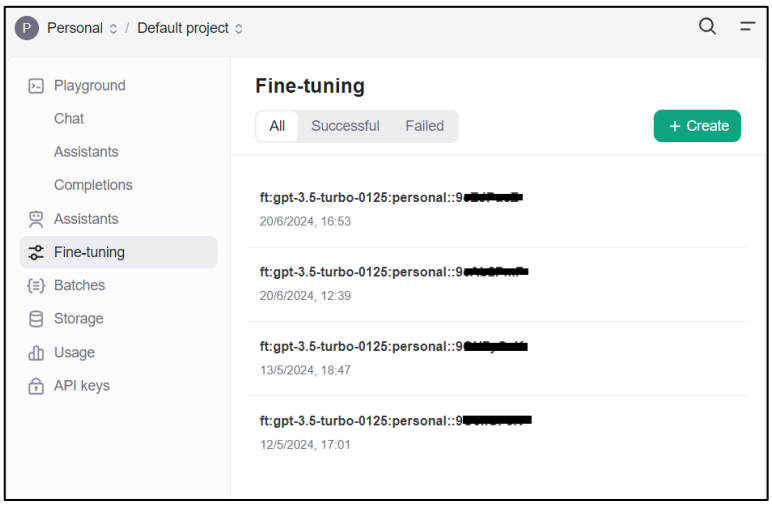


Figura 24: Captura de la interfaz de OpenAI en la que aparecen todos los modelos optimizados

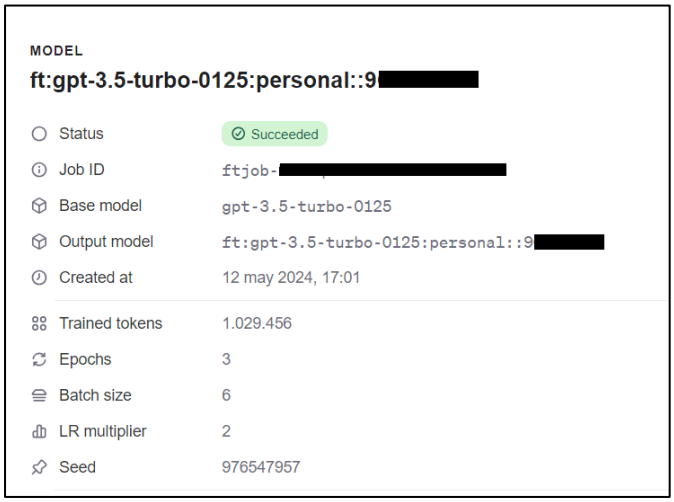


Figura 25: Captura de la información del modelo optimizado para emociones.

5.5.4 Acceso a los modelos optimizados

En esta sección, se explicarán los pasos a seguir para poder utilizar el modelo que acabamos de optimizar. Para ello, en primer lugar, deberemos acceder a nuestra cuenta en la página oficial de OpenAI, como se explicó previamente.

A continuación, una vez dentro del entorno de OpenAI, pulsaremos en *Dashboard* y en su menú desplegable en la opción de *Fine-tuning*, de esta manera dependiendo de la cantidad de modelos que se hayan optimizado, obtendremos lo mismo que en la Figura 24.

En la Figura 24, se presenta un listado de los modelos que hemos optimizado, para saber de qué modelo se trata deberemos comparar el identificador que obtenemos en

la fase de optimización con el que aparece al pulsar alguno de los modelos, el *Job ID* que aparece en la Figura 25.

Posteriormente, si queremos utilizar los modelos optimizados deberemos de copiar el enlace de dicho modelo que se corresponde con el *Output Model* de la Figura 25 y utilizarlo en el código correspondiente. El uso de los modelos se explicará en la sección 5.7.

5.6 Método para la obtención de los resultados

En esta sección, se explicará la metodología aplicada para la obtención de los resultados. Para ello, será necesario emplear la base de datos de prueba que se realizó en apartados anteriores y el modelo que acabamos de optimizar.

Además, debemos tener en cuenta que nuestro modelo optimizado es un chat, y la interfaz de OpenAI tiene unos límites de velocidad que no debemos superar. En nuestro caso, al provenir del modelo base gpt-3.5-turbo los límites eran: 3.500 RPM (solicitudes por minuto), 10.000 RPD (solicitudes por día) y 60.000 TPM (tokens por minuto). Con la base de datos que teníamos el único que se podría superar era el de TPM [43].

En la Figura 26, se presenta el fragmento de código en el que se crea una función en *Python* para que haga solicitudes a la API de chat limitando la velocidad de solicitudes mediante el parámetro de TPM. En primer lugar, definimos dicho límite a 60000 TPM y posteriormente utilizamos los decoradores *@sleep_and_retry* y *@limits(calls=RATE_LIMIT_TPM, period=60),* para controlar la tasa a la que se realizan las llamadas a la función *“realizar_solicitud”*.

Por otro lado, la función *realizar_solicitud,* es la encargada de enviar un texto, que se corresponderá con los comentarios guardados en la base de datos de test a la API, para que dependiendo del contenido del sistema y el modelo utilizado pueda analizar la polaridad o sentimiento del mismo. El ejemplo de código en la Figura 26 se corresponde con la obtención de los resultados para emociones, para el caso de polaridad se cambiaría el contenido de *system* por *"¿Cuál es la polaridad del siguiente texto? Responde con 'Positiva', 'Negativa' o 'Indeterminado'."*.

```python
#Definimos los límites de velocidad
RATE_LIMIT_TPM=60000
# Decorador para limitar la velocidad de los tokens por minuto
@sleep_and_retry
@limits(calls=RATE_LIMIT_TPM, period=60)  # 60 segundos en un minuto
def realizar_solicitud(texto):

    completion = client.chat.completions.create(
        model=model_id,
        messages=[
            {"role": "system", "content": "¿Cuál es el sentimiento del siguiente texto? Responde con 'Amor/Admiración', 'Tristeza
            {"role": "user", "content": texto},
        ]
    )
    return completion.choices[0].message.content
```

Figura 26: Fragmento de código para realizar una solicitud a la API de chat para el caso de emociones.

Por otro lado, en la Figura 27, tenemos el código en el que obtenemos las repuestas del chat y las etiquetas verdaderas de los comentarios para emociones.

En principio, cargamos el fichero .xlsx con los datos de test e indicamos el identificar del modelo optimizado que vamos a emplear. Seguidamente, definiremos las clases de emociones siendo estas: Amor/Admiración, Tristeza/Pena, Indeterminado, Gratitud, Enfado/Desprecio/Burla y Compresión/Identificación/Empatía, ya que la respuesta del chat debe corresponderse con uno de ellos.

Además, definimos dos listas para poder almacenar las predicciones que serán las respuestas que nos ofrezca el chat y las etiquetas verdaderas que serán las emociones correctas de cada uno de los comentarios proveniente de la base de datos de test.

Posteriormente, realizamos las predicciones y obtenemos las etiquetas verdaderas. Para ello, realizamos las predicciones llamando a la función *realizar_solicitud ()* introduciendo como parámetro de entrada los comentarios de las bases de datos de test. A su vez, dichas respuestas se guardarán en la lista de predicciones una vez se ha comprobado que se corresponden con una emoción de la clase de sentimientos. Por otro lado, con cada comentario se va extrayendo su emoción mediante la línea *row['Emociones']* y se almacena en la lista de etiquetas verdaderas.

```python
#Carga de los datos de test
filename = 'datos_test.xlsx'
df= pd.read_excel(filename)

#Modelo fine-tuned
model_id="ft:gpt-3.5-turbo-0125:personal::[redacted]

#Definimos las clases de sentimientos
sentimiento_clases=["Amor/Admiración", "Tristeza/Pena", "Indeterminado", "Gratitud", "Enfado/Desprecio/Burla", "Comprensión/Empat

#Definimos unas listas para almacenar las predicciones y las etiquetas verdaderas
predicciones=[]
etiquetas_verdaderas=[]

# Creamos un DataFrame para almacenar las predicciones y las etiquetas verdaderas
df_resultados = pd.DataFrame(columns=["Comentario", "Predicción", "Etiqueta Verdadera"])


#Realizamos las predicciones y obtenemos las etiquetas verdaderas
for index, row in df.iterrows():
    texto = row['Texto']

    try:

        response= realizar_solicitud(texto)
        predicciones_completas=response

        # Obtener el índice de la clase de sentimiento predicha
        etiqueta=sentimiento_clases.index(response)

        # Agregar la predicción a la lista de predicciones
        predicciones.append(sentimiento_clases[etiqueta])

        # Obtener la etiqueta verdadera del DataFrame
        true_label = row['Emociones']

        # Agregar la etiqueta verdadera a la lista de etiquetas verdaderas
        etiquetas_verdaderas.append(true_label)

        # Agregar las predicciones y etiquetas verdaderas al DataFrame
        df_resultados = pd.concat([df_resultados, pd.DataFrame({"Comentario": [texto], "Predicción": [response], "Etiqueta Verda

    except Exception as e:
        print("Error:", e)
        continue
```

Figura 27: Fragmento de código con el que obtenemos las respuestas del chat y las etiquetas verdaderas para el caso de emociones.

El ejemplo anterior, para el caso de polaridad se presenta en la Figura 28, donde hemos cambiado el modelo, por el optimizado de polaridad y la base de datos de test. Además, se han definido las clases de polaridad en vez de las clases de sentimientos y a la hora de obtener la etiqueta verdadera (*true_label*) hemos cambiado *row[‘Emociones’]* por *row[‘Polaridad’]*.

Por último, mediante la líneas de código *classification_report(etiquetas_verdaderas, predicciones)* y *confusion_matrix(etiquetas_verdaderas, predicciones, labels=sentimiento_clases),* obtenemos el reporte de clasificación y la matriz de confusión que evaluaremos en secciones posteriores.

```python
#Carga de los datos de test
filename = 'datos_test_polaridad.xlsx'
df= pd.read_excel(filename)

#Modelo fine-tuned-polaridad
model_id="ft:gpt-3.5-turbo-0125:personal::9[redacted]"

#Definimos las clases de sentimientos
polaridad_clases=["Positiva", "Negativa", "Indeterminado"]

#Definimos unas listas para almacenar las predicciones y las etiquetas verdaderas
predicciones=[]
etiquetas_verdaderas=[]

# Creamos un DataFrame para almacenar las predicciones y las etiquetas verdaderas
df_resultados = pd.DataFrame(columns=["Comentario", "Predicción", "Etiqueta Verdadera"])


#Realizamos las predicciones y obtenemos las etiquetas verdaderas
for index, row in df.iterrows():
    texto = row['Texto']

    try:

        response= realizar_solicitud(texto)
        predicciones_completas=response

        # Obtener el índice de la clase de sentimiento predicha
        etiqueta=polaridad_clases.index(response)

        # Agregar la predicción a la lista de predicciones
        predicciones.append(polaridad_clases[etiqueta])

        # Obtener la etiqueta verdadera del DataFrame
        true_label = row['Polaridad']

        # Agregar la etiqueta verdadera a la lista de etiquetas verdaderas
        etiquetas_verdaderas.append(true_label)

        # Agregar las predicciones y etiquetas verdaderas al DataFrame
        df_resultados = pd.concat([df_resultados, pd.DataFrame({"Comentario": [texto], "Predicción": [response], "Etiqueta Verdad

    except Exception as e:
        print("Error:", e)
        continue
```

Figura 28: Fragmento de código con el que obtenemos las respuestas del chat y las etiquetas verdaderas para el caso de polaridad.

5.7 Uso de los modelos optimizados

En este apartado se explicará de manera detallada, las formas en las que podemos usar nuestros modelos optimizados. Ya que existen la opción de hacerlo directamente desde la interfaz de OpenAI o mediante la utilización de código en Python (utilizado en la sección anterior).

5.7.1 Utilización del modelo optimado desde la interfaz de OpenAI

Este caso es preferible cuando el número de ejemplos que queremos comprobar es muy pequeño.

Para ello, en primer lugar, deberemos haber iniciado sesión en la interfaz de OpenAI, y pulsar en la opción de *Dashboard*. Seguidamente, en el menú vertical en la opción de *Playground* se deberá seleccionar *Chat* [43].

En la Figura 29, se observa la opción de chat en la interfaz de OpenAI, donde hemos escogido el modelo optimizado para emociones. A continuación, indicamos en la ventana de *System*: *"¿Cuál es el sentimiento del siguiente texto? Responde con*

'Amor/Admiración', 'Tristeza/Pena', 'Indeterminado', 'Gratitud', 'Enfado/Desprecio/Burla' o 'Comprensión/Empatía/Identificación'.'' Y en la ventana de *User* indicamos un comentario de nuestra base de datos de prueba, el cual es "*Ánimo eres muy valiente y piensa que hay mucha gente que te quiere todo mi apoyo un fuerte abrazo*".

Como se observa en la figura, una vez hemos introducido el comentario y pulsamos en *Run*, se genera una respuesta por parte del modelo, que en este caso corresponde con Amor/Admiración, cuya respuesta es la correcta

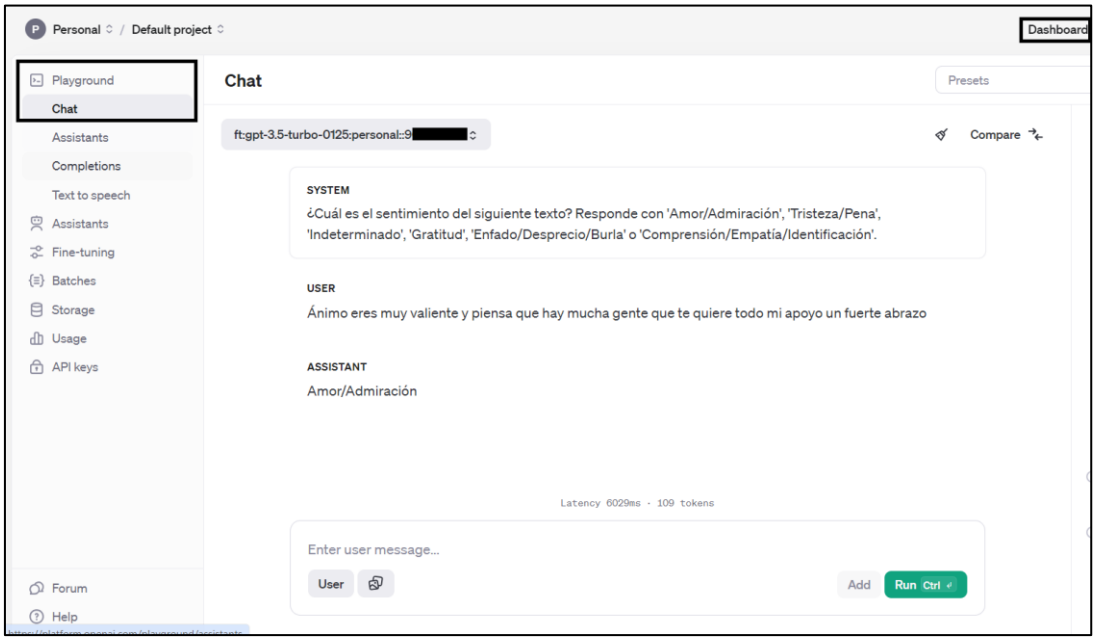


Figura 29: Captura de la interfaz de OpenAI en la que comprobamos el funcionamiento del modelo optimizado para emociones

5.7.2 Utilización del modelo optimizado mediante código en Python

Para usar el modelo optimizado en Python, se utiliza el código descrito en la sección anterior de la obtención de los resultados. En ese caso, se emplea el código para obtener las predicciones de la base de datos de prueba que está en formato Excel [39].

Dicho código se podría simplificar para un ejemplo concreto, como observamos en la Figura 30. En dicho fragmento de código, mediante la primera línea de código se especifica que se desea realizar una solicitud de chat, para posteriormente indicar el modelo que vamos a utilizar. A continuación, en *messages* indicamos tanto el mensaje

del sistema que será el mismo que en el caso anterior y un nuevo mensaje de usuario, en este caso *"Gracias por compartirlo.... Somos humanos... A veces es tan simple como reconocernos humanos.... Un beso"*. Por último, con la línea *completation.choices[0].content* se obtiene la respuesta proporcionada por el modelo, que es Gratitud la cual sería correcta.

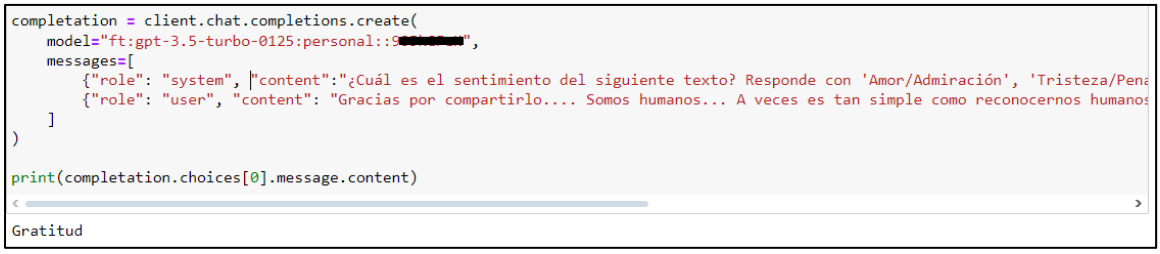


Figura 30: Fragmento de código en el que comprobamos el funcionamiento del modelo optimizado para emociones

5.8 Coste final del proceso

En esta sección se hará un desglose del coste, tanto para emociones como para polaridad, del proceso de optimización de los modelos y de la obtención de los resultados.

5.8.1 Coste para la optimización y obtención de resultados para emociones

Para poder calcular el coste del proceso de optimización, al número de tokens de nuestra base de datos de entrenamientos, debemos multiplicarle el número de épocas, que en este caso fueron 3. Teniendo en cuenta esto, la cantidad de tokens finales fueron de 1.029.456 tokens.

El precio, por un millón de tokens para realizar la optimización del modelo gpt-3.5-turbo, es de $8 [37]. Siendo de esta manera el coste de optimización del modelo de emociones de $8,24.

Por otro lado, para la obtención de los resultados, las solicitudes al chat de API suponían $3 por millón de tokens de entrada y $6 por millón de tokens de salida. Este proceso tuvo un costo de $0,48.

5.8.2 Coste para la optimización y obtención de resultados para polaridad

Para el coste en el caso de polaridad, se tenían los mismos requisitos que para el caso de emociones. Manteniendo el número de épocas a tres y utilizando la base de datos de polaridad, nuestro número de tokens totales pasa a ser de 682.611. De esta manera el coste del proceso de optimización fue de $5,46.

En cambio, el coste de la realización de las solicitudes al chat de API, para obtener los resultados, fue de $0,29.

5.9 Análisis de los resultados para polaridad con gpt-3.5-turbo-0125

La API de OpenAI, durante el proceso de fine-tuning, establece los hiperparámetros que mejor se ajustan teniendo en cuenta el tamaño de los datos de entrenamiento, para poder obtener los resultados más fiables y precisos.

En nuestro caso, como en el proceso de optimización ya establecimos el número de épocas para poder hacer una estimación del coste del proceso, el algoritmo determinó los mejores hiperparámetros considerando 3 épocas, siendo estos los siguientes:

- Gpt-3.5-turbo: num_epochs=3, batch_size= 6, learning_rate_multiplier= 2.

La Tabla 8 muestra el reporte de clasificación de los resultados obtenidos con el modelo descrito anteriormente para polaridad. A su vez, también se muestra de manera comparativa los resultados de un estudio previo realizado por Javier Estévez [2] utilizando los modelos RoBERta y RoBERTuito de Bert. En esta tabla se presenta la precisión, *recall* y *F1-score* para cada una de las clases de polaridad, además del promedio macro (*macro-avg*) y ponderado (*weighted-avg*). También, se muestra la precisión global del modelo (*Accuracy*). Todos los valores de las métricas están en formato de porcentaje y se añade la cantidad de instancias de cada una de las clases en el supuesto actual. En resumen, la tabla muestra una visión completa y detallada de los resultados para una mejor comparación y evaluación.

Polaridad	Métrica	RoBERTa (%) (estudio anterior)	RoBERTuito (%) (estudio anterior)	Gtp-3.5-turbo (%) (estudio actual)	Instancias (estudio actual)
Negativa	Precisión	86,5%	90%	87%	372
	Recall	87,9%	91,8%	92%	
	F1-score	87,1%	90,8%	89%	
Positiva	Precisión	94,3%	95,7%	94%	834
	Recall	96,8%	97,5%	95%	
	F1-score	95,5%	96,5%	94%	
Indeterminado	Precisión	76,7%	83%	70%	106
	Recall	53,2%	64,1%	47%	
	F1-score	62%	68,9%	56%	
macro-avg	Precisión	85,8%	89,6%	84%	1312
	Recall	79,3%	83,5%	78%	
	F1-score	81,5%	85,4%	80%	
weighted-avg	Precisión	-	-	90%	1312
	Recall	-	-	90%	
	F1-score	-	-	90%	
Global	Accuracy	91,2%	93,3%	90%	

Tabla 8: Resultados obtenidos en la clasificación de Polaridad

En la Tabla 9, presentamos una versión reducida de la tabla anterior en la que aparece la precisión de cada una de las clases y la precisión global de cada uno de los modelos.

Polaridad	Precisión. (%) (RoBERTa)	Precisión. (%) (RoBERTuito)	Precisión. (%) (gpt-3.5-turbo)
Negativa	86,5%	90%	87%
Positiva	94,3%	95,7%	94%
Indeterminado	76,7%	83%	70%
Accuracy	91,2%	93,3%	90%

Tabla 9: Comparación de los resultados obtenidos en la métrica de precisión en la clasificación de Polaridad

Analizando los resultados anteriores observamos que el modelo RoBERTuito del estudio del año pasado [2], es el que alcanza mejor precisión global alcanzando un 93,3%; lo que supone un 2,1% más que el modelo de RoBERTa, también del pasado año y un 3,3% más que el modelo del estudio de este año, el gpt-3.5-turbo. Sin embargo, podemos observar que no existen variaciones significativas al comparar los algoritmos para todas las clases.

En la clase "Negativa", nuestro modelo demuestra una gran capacidad para poder identificar comentarios con dicha polaridad, siendo su precisión ligeramente superior a la de RoBERta (un 0.5%), pero sigue siendo inferior a la de RoBERTuito. Si analizamos profundamente las métricas observamos que el *recall*, esta métrica es ligeramente

superior en ambos modelos siendo su valor de un 92% (RoBERTa, 87,9%; RoBERTuito, 91,8%). En cuanto al F1-score, cuyo valor es del 89% este se sitúa entre los obtenidos por RoBERTa 87,1% y RoBERTuito 90,8%.

Por otro lado, la clase "Positiva", presenta una mejoría considerable con una precisión del 94%, un *recall* 95% y un *F1-score* del 94%, siendo bastante satisfactorio. Aunque, todos estos resultados son inferiores a ambos estudios del año pasado (RoBERTa: Precisión: 94,3%, Recall: 96,8%, F1-score: 95,5% y RoBERTuito: Precisión: 95,7%, Recall: 97,5%, F1-score: 96,5%).

En clase "Inderteminada", encontramos los peores resultados, a pesar de que la precisión del 70% es aceptable, esta clase presenta un recall del 47%, siendo bastante malo y un F1-score del 56%. Todas las métricas de esta clase, al igual que en la clase Positiva son inferiores a las de los modelos del año pasado (RoBERTa: Precisión: 76,7%, Recall: 53,2%, F1-score: 62% y RoBERTuito: Precisión: 83%, Recall: 64,1%, F1-score: 68,4%). Todo esto nos sugiere que aquellos comentarios, que no tienen una polaridad bien definida, suponen una tarea compleja y costosa para los modelos ya que para estas polaridades el lenguaje es más ambiguo y no se establecen patrones claros de lenguaje que denoten esta polaridad. Es verdad que también deberíamos aclarar, que el número de ejemplos de esta clase en el corpus es mucho menor con respecto a la otras dos.

En conclusión, el modelo optimizado con gpt-3.5-turbo es bastante eficaz en la clasificación de la polaridad, principalmente de los comentarios con polaridad positiva. Pero sigue teniendo una menor precisión respecto a modelos de estudios anteriores. El mejor comportamiento de RoBERTuito se puede deber a que este modelo ha sido preentrenado con millones de Tweets de redes sociales en castellano, por lo que está muy enfocado en la detección de la respuesta emocional en redes sociales, como es nuestro caso. Mientras, los otros dos modelos, son más generalistas en el lenguaje y el contexto. Aun así, el rendimiento y precisión de los tres modelos alcanza niveles muy altos, en torno al 90%, en dicha tarea de clasificación.

Por otro lado, en la Figura 31, se presenta el análisis de la matriz de confusión para la clasificación de polaridad, en la que se puede entender de una mejor manera, los posibles resultados obtenidos en las Tablas 8 y 9. Como podemos observar en la matriz, la precisión en la clasificación de los comentarios de la clase Positiva es muy alta, donde

se han acertado un total de 791 casos. Sin embargo, hay una pequeña confusión con la clase Negativa, interpretando 30 casos erróneamente como tal.

En la clase Indeterminado, el desempeño ha sido bastante negativo, ya que de 106 casos, solo 50 han sido bien etiquetados con dicha polaridad, es decir, menos del 50%. En este caso, podemos observar que las predicciones erróneas de esta clase están bastante equilibradas entre positiva y negativa. Esto nos hace llegar a la misma conclusión que antes, de que al modelo le cuesta diferenciar entre comentarios ambiguos que no muestran una polaridad bien definida.

Por último, la clase Negativa, muestra un rendimiento favorable con un total de 344 aciertos. Por el contrario, se observa una pequeña confusión con la clase positiva, con 20 casos mal clasificados.

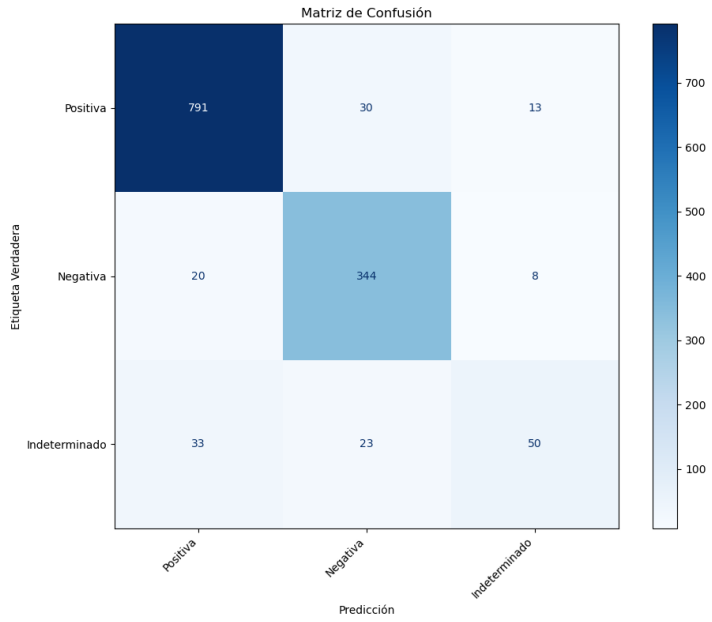


Figura 31: Matriz de confusión para la clasificación de polaridad.

En conclusión, la matriz de confusión confirma que a pesar de que nuestro modelo es bastante efectivo, todavía debería mejorarse la precisión en la clase indeterminada, para ello es necesario explorar nuevas estrategias, como por ejemplo aumentar el número de ejemplos de este en el corpus y analizar la ambigüedad de este tipo de comentarios.

5.10 Análisis de los resultados para emociones con gpt-3.5-turbo-0125.

Para la base de datos de entrenamiento de emociones y un número de épocas, previamente definido como 3 en el proceso de optimización, el algoritmo en este proceso nos proporciona los siguientes hiperparámetros:

- Gpt-3.5-turbo: num_epochs=3, batch_size= 6, learning_rate_multiplier= 2.

La Tabla 10 presenta los resultados obtenidos en la clasificación de emociones, donde podemos observar las métricas para cada una de las clases, además de los promedios macro y ponderado. También, se muestran los resultados de los modelos utilizados en un estudio anterior de Javier Estévez [2], en el que se utilizan los modelos Daveni y RoBERTuito. Para completar dicha tabla se ha añadido el número de instancias de cada clase para nuestro estudio actual con el modelo gpt-3.5-turbo.

Polaridad	**Métrica**	**Daveni (%) (estudio anterior)**	**RoBERTuito (%) (estudio anterior)**	**Gtp-3.5-turbo (%) (estudio actual)**	**Instancias (estudio actual)**
Amor/Admiración	Precisión Recall F1-score	90,6% 92,7% 91,6%	93,1% 95% 94%	93% 93% 93%	405
Gratitud	Precisión Recall F1-score	94,1% 89% 91,4%	90,8% 92,1% 91,3%	86% 98% 91%	165
Compresión/Empatía/ Identificación	Precisión Recall F1-score	84,6% 88,6% 86,3%	89,9% 88,5% 89,1%	91% 81% 85%	346
Tristeza/Pena	Precisión Recall F1-score	81,4% 79,4% 79,3%	83,7% 87,8% 85,1%	70% 88% 78%	75
Enfado/Desprecio/ Burla	Precisión Recall F1-score	84,7% 85,8% 84,9%	91% 93,2% 92%	90% 93% 91%	243
Indeterminado	Recall F1-score F1-score	70,1% 52,6% 57,4%	79,5% 65,9% 70,9%	73% 62% 67%	78
macro-avg	Precisión Recall F1-score	84,2% 81,4% 81,8%	88% 87,1% 87,1%	84% 86% 84%	1312
weighted-avg	Precisión Recall F1-score	- - -	- - -	88% 88% 88%	1312
Global	Accuracy	86%	89,9%	88%	

Tabla 10: Resultados obtenidos en la clasificación de Emociones.

Para poder observar con mayor claridad los resultados de precisión de cada uno de los modelos, así como su precisión global, se ha extraído de la tabla anterior los datos y se han presentado en la Tabla 11. Como podemos observar, la precisión de los tres modelos es bastante buena, en los tres casos muy parecidos y cercanos al 90%. Además, a excepción de algunas de las emociones en los que la precisión es más baja como en el caso Indeterminado, por lo general se alcanzan porcentajes de precisión bastante altos. Esto nos indica que los tres modelos tienen un alto rendimiento y son muy eficaces a la hora de detectar emociones.

Polaridad	**Precisión. (%) (Daveni)**	**Precisión. (%) (RoBERTuito)**	**Precisión. (%) (gpt-3.5-turbo)**
Amor/Admiración	90,6%	93,1%	93%
Gratitud	94,1%	90,8%	86%
Compresión/Empatía/Identificación	84,6%	89,9%	91%
Tristeza/Pena	81,4%	83,7%	70%
Enfado/Desprecio/Burla	84,7%	91%	90%
Indeterminado	70,1%	79,5%	73%
Accuracy	86%	89,9%	88%

Tabla 11: Comparación de los resultados obtenidos en la precisión en la clasificación de Emociones

Se empleará la Figura 32, que muestra la matriz de confusión de los resultados para emociones. Con el objetivo de analizar mejor los resultados obtenidos y comprender de mejor manera las capacidades de nuestro modelo entrenado.

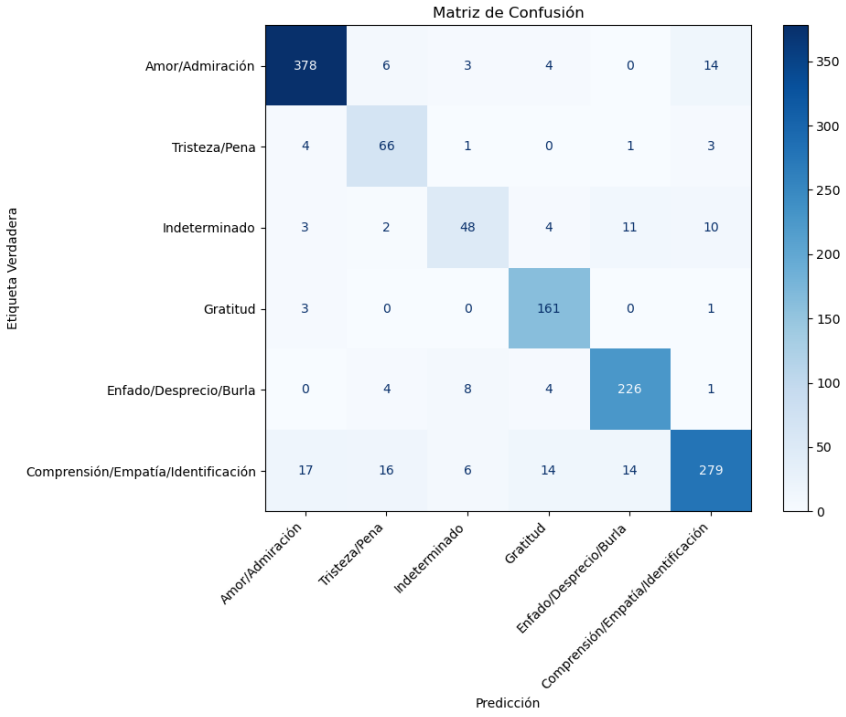


Figura 32: Matriz de confusión para la clasificación de emociones.

La clase “Amor/Admiración”, como podemos observar en la matriz de confusión y en la Tabla 11, presenta un total de 378 aciertos, siendo la emoción que más se confunde con la de “Comprensión/Identificación/Empatía” con un total de 14 fallos. Las confusiones entre ambas clases tienen sentido, pues cuando existe la emoción de amor, también vienen implícitas otras emociones como empatía o comprensión hacia la otra persona. Además, esta emoción presenta un 93% de precisión, siendo considerablemente bueno y es lo esperado teniendo en cuenta la matriz de confusión. Si esto lo comparamos con los modelos anteriores observamos que mejora con respecto al de Daveni y es muy similar al de RoBERTuito.

En relación con la clase “Gratitud”, podemos observar en las tablas que su precisión del 86% se ve reducida frente a modelos anteriores, aunque presenta un *recall* (98%) mejor y un *F1-score* (91%) muy parecidos (Daveni: Precisión:94,1%, Recall: 89%, F1-score:91,4% y RoBERTuito: Precisión:90,8%, Recall: 92,1%, F1-score:91%). Por otro lado, en la matriz de confusión observamos que ha tenido una tasa de aciertos considerablemente buena, acertando 161, siendo “Amor/Admiración” la clase de emoción con la que más se confunde, obteniendo 3 fallos. En este caso ocurre lo mismo, es decir, existe una relación entre la emoción de gratitud, donde además puede venir implícita el amor por alguien cuando también expresas gratitud.

Al contrario de lo que ocurre con la clase de “Gratitud”, la clase “Comprensión/Identificación/Empatía” presenta una precisión del 91%, mayor respecto a los modelos anteriores. Pero sus resultados de *recall* (81%) y *F1-score* (85%) son ligeramente inferiores. Sin embargo, si observamos la matriz de confusión, es una de las clases que se confunde de manera prácticamente equitativa entre el resto de las clases, interpretándose 17 como “Amor/Admiración”, 16 como “Tristeza/Pena”, 14 como “Gratitud” y 14 como “Enfado/Desprecio/Burla”. A pesar de ello, tiene una alta tasa de acierto, ya que cuenta con 279 aciertos.

Por otro lado, la clase de “Tristeza/Pena” empeora la precisión respecto a ambos modelos anteriores, mientras que antes era de 81.4% (Daveni) y el 83,7% (RoBERTuito) ahora pasa a ser el 70%. Esto puede deberse a que el número de ejemplos es bastante reducido, siendo 75, a pesar de que como observamos en la matriz de confusión cuenta con 66 aciertos. Los comentarios de esta clase se confunden principalmente con los de “Amor/Admiración” (4 fallos) y “Comprensión/Identificación/Empatía” (3 fallos).

Para la clase de “Enfado/Desprecio/Burla” los resultados obtenidos son muy parecidos a los del modelo de RoBERTuito del año pasado. En este caso, contamos con una precisión del 90%. Esto, se confirma gracias a lo obtenido en matriz de confusión, donde hay una tasa de aciertos bastante elevada (226 casos correctos), de manera que la clase “Indeterminada” es la que causa mayor confusión con 8 fallos.

Por último, la clase con peores resultados es “Indeterminado”, aunque presenta mejores resultados que el modelo de Daveni, es ligeramente inferior a los proporcionados por RoBERTuito. En este caso, se cuenta con una precisión del 73%. Si nos fijamos en la matriz de confusión, se puede observar que la tasa de aciertos es inferior frente a las otras clases teniendo solo 48 aciertos, parte de ello se debe a la ambigüedad de los comentarios pudiendo ser interpretados como diferentes emociones. Esto es una de las razones por las que presenta 10 fallos interpretando los comentarios como si fueran de “Comprensión/Identificación/Empatía” y 11 de Enfado/Desprecio/Burla”.

En resumen, el modelo actual presenta ciertas mejoras en la identificación de ciertas emociones con respecto a modelos anteriores. A pesar de ello, hay algunas como “Tristeza/Pena” e “Indeterminado” en las que la precisión se ha reducido, esto se debe principalmente al reducido número de instancias en el corpus con respecto a otras. Esto nos lleva a recalcar la importancia de tener un corpus lo más balanceado y equilibrado posible, para así contar con unos datos de entrenamiento más completos.

En conclusión, podemos afirmar que los tres algoritmos presentan un alto rendimiento en la detección de las emociones, en el que la precisión global es cercana al 90%. Cabe destacar que, aunque la precisión de mi algoritmo se encuentra entre la precisión de los otros dos modelos, en algunas de las emociones como Compresión/Empatía/Identificación muestra los mejores resultados. Aunque todos los resultados obtenidos son bastante altos, confirmando así la eficacia y robustez que tiene en la detección de emociones.

5.11 Funciones para la predicción de emociones y polaridad.

En esta sección, se explicarán detalladamente las funciones oportunas para la predicción de emociones y polaridad. Del mismo modo, se explicará tanto si se introduce un texto como un fichero en formato Excel.

5.11.1 Función para la predicción de emociones

Para el caso de predecir únicamente un texto, se ha creado la función *"predict_emotions"* que es la encargada de identificar el sentimiento predominante del texto introducido, utilizando el modelo optimizado para emociones previamente descrito.

En la Figura 33 se muestra el fragmento de código correspondiente para dicha función. En la que en primer lugar se realiza una solicitud a la API de chat de OpenAI para que interactúe con el modelo especificado, en nuestro caso se trata del modelo optimizado para emociones. Dicha solicitud consta de dos partes, una referida al mensaje del sistema donde se especifica la tarea que se solicita al modelo y la referida al mensaje del usuario que contiene el texto proporcionado por el usuario. Por último, el modelo procesa la solicitud y genera una respuesta que se retornará.

```
#Function predict emotions
def predict_emotions(text):
    completion = client.chat.completions.create(
        model="ft:gpt-3.5-turbo-0125:personal::9[redacted]",
        messages=[
            {"role": "system", "content":"¿Cuál es el sentimiento del siguiente texto? Responde con 'Amor/Admiración', 'Tristeza/
            {"role": "user", "content": text},
        ]
    )
    return completion.choices[0].message.content.strip()
```

Figura 33: Fragmento de código que define la función *predict_emotions*

A continuación, en la Figura 34, se presenta el código utilizado para la predicción del texto *"Gracias por compartirlo"*, donde se ha obtenido como respuesta *"Gratitud"*. En este código invocamos la función *predict_emotions* en la que se ha introducido el texto de entrada anterior y posteriormente se imprime por pantalla la respuesta.

```
#Example
input_text= "Gracias por compartirlo"
emocion_predicha= predict_emotions(input_text)
print(emocion_predicha)
```

Figura 34: Ejemplo de invocación de la función *predict_emotions*.

Por último, para poder predecir emociones a partir de un fichero en formato Excel, se ha desarrollado el código de la Figura 35. En este código se implementa la función *predict_emotions_excel* que permite procesar un fichero Excel y aplicar la función de predicción de emociones anteriormente descrita en cada una de las filas del fichero y generar un nuevo fichero con los resultados.

```
#Function predict emotions from an excel
def predict_emotions_excel(input_file, output_file, text_column):
    #Upload the excel input file
    df=pd.read_excel(input_file)

    df['Emotion']=df[text_column].apply(predict_emotions)
    df.to_excel(output_file, index=False)

    print(output_file)
```

Figura 35: Fragmento de código que define la función *predict_emotions_excel*

5.11.2 Función para la predicción de polaridad

En este caso para poder predecir la polaridad de un único texto se ha creado la función *"predict_polarity"*, la cual se encarga de identificar cual es la polaridad del texto introducido utilizando el modelo optimizado para polaridad.

En la Figura 36 se presenta el fragmento de código correspondiente para dicha función. En la que se realizará una solicitud a la API del chat para el modelo especifico optimizado para polaridad. Al igual que en el caso anterior dicha solicitud consta de dos partes y se ha modificado el mensaje de sistema por el adecuado en este caso, mientras que el de usuario será el texto introducido. Por último, se procesa la solicitud y se generará una respuesta.

```
#Function predict polarity
def predict_polarity(text):
    completion = client.chat.completions.create(
        model="ft:gpt-3.5-turbo-0125:personal::9[redacted]",
        messages=[
            {"role": "system", "content":"¿Cual es la polaridad del siguiente texto? Responde con 'Positiva', 'Negativa' o 'Indet
            {"role": "user", "content": text},
        ]
    )
    return completion.choices[0].message.content.strip()
```

Figura 36: Fragmento de código que define la función *predict_polarity.*

En la Figura 37, se muestra un ejemplo en el que se invoca la función para predecir la polaridad. Para ello se introduce un texto, en este caso *"Gracias por*

compartirlo", que se pasa como argumento de entrada a la función *predict_polarity* para posteriormente obtener el resultado *"Positiva"* e imprimirlo por pantalla.

```
#Example
input_text= "Gracias por compartirlo"
polaridad_predicha= predict_polarity(input_text)
print(polaridad_predicha)
```

Figura 37: Ejemplo de invocación de la función *predict_polarity*.

Para poder predecir polaridad partiendo de un fichero en formato Excel, se ha desarrollado el código presentado en la Figura 38. En este código se implementa la función *predict_polarity_excel* la cual procesa un fichero Excel, para posteriormente aplicar en cada línea del fichero la función descrita anteriormente. Por último, se generará un nuevo fichero Excel con los resultados.

```
#Function predict polarity from an excel
def predict_polarity_excel(input_file, output_file, text_column):
    #Upload the excel input file
    df=pd.read_excel(input_file)

    df['Polarity']=df[text_column].apply(predict_polarity)
    df.to_excel(output_file, index=False)

    print(output_file)
```

Figura 38: Fragmento de código que define la función *predict_polarity_excel*

5.11.3 Predicción desde la interfaz de OpenAI

Para realizar predicciones desde la interfaz de OpenAI, es necesario seguir los dos primeros pasos descritos en la Sección 5.5.1 para acceder a la plataforma. Una vez dentro, debemos dirigirnos a la pestaña *Playground* y seleccionar la opción de *Chat*, que es la que utilizaremos. A continuación, se debe especificar el modelo optimizado según el tipo de comentario que se quiera analizar; en este caso, se utiliza un modelo especializado en emociones relacionadas con la salud mental. Es crucial completar el apartado *system* con un mensaje que indique claramente la función del chat. En este caso, se debe incluir la instrucción: *"¿Cuál es el sentimiento del siguiente texto? Responde con 'Amor/Admiración', 'Tristeza/Pena', 'Indeterminado', 'Gratitud', 'Enfado/Desprecio/Burla' o 'Comprensión/Empatía/Identificación'."*. Luego, se ingresa el

comentario a analizar en el campo *user*, tal como se muestra en la Figura 29 de la sección 5.7.1.

Cabe destacar que, desde la interfaz de OpenAI, no es posible subir directamente un archivo Excel en la sección de chat del *Playground*, ya que esta está diseñada para procesar entradas de texto y generar respuestas en formato de texto, no para manejar archivos directamente. Si se desea predecir a partir de un archivo Excel, será necesario utilizar los scripts previamente descritos en los apartados anteriores.

6

Análisis de la respuesta emocional en Twitch con ChatGPT: Polaridades y Emociones

6.1 Introducción

En este capítulo, se analizará la respuesta emocional en mensajes de chat de Twitch en el entorno gamer. En primer lugar, se realizará una pequeña descripción de la base de datos que se va a utilizar, para poder entender de mejor manera los resultados obtenidos. Además, se explicarán los pequeños cambios en el código al utilizarlo para el análisis en la plataforma Twitch. Finalmente, se realizará el análisis de los datos obtenidos, y comparemos los resultados tanto para emociones como para polaridad con estudios anteriores.

6.2 Descripción del corpus utilizado

En esta sección, se explicará la base de datos con la que partimos. Esta base ha sido creada por un grupo de profesores de la Universidad de Valladolid y de la Autónoma de Madrid [44] y posteriormente ha sido ampliada por el alumno Miguel Carralero [3].

Los datos de los que consta, esta base de datos, son mensajes extraídos de chats en Twitch, constituyendo un total de 4600 instancias.

En las Figuras 39 y 40 podemos observar las distribuciones de los mensajes tanto para emociones como polaridad. Por un lado, los mensajes en función de las emociones que encontramos son 1245 de “Aprobación/Empatía/Confianza”, 970 de “Desaprobación”, 471 de “Decepción/Tristeza”, 505 de “Interés/Anticipación/Hype”,

476 de “Enfado/Ira” y 663 de “Indeterminado”. En cambio, en función de la polaridad encontramos 1694 “Positivo”, 2282 “Negativo” y 624 “Indeterminado”.

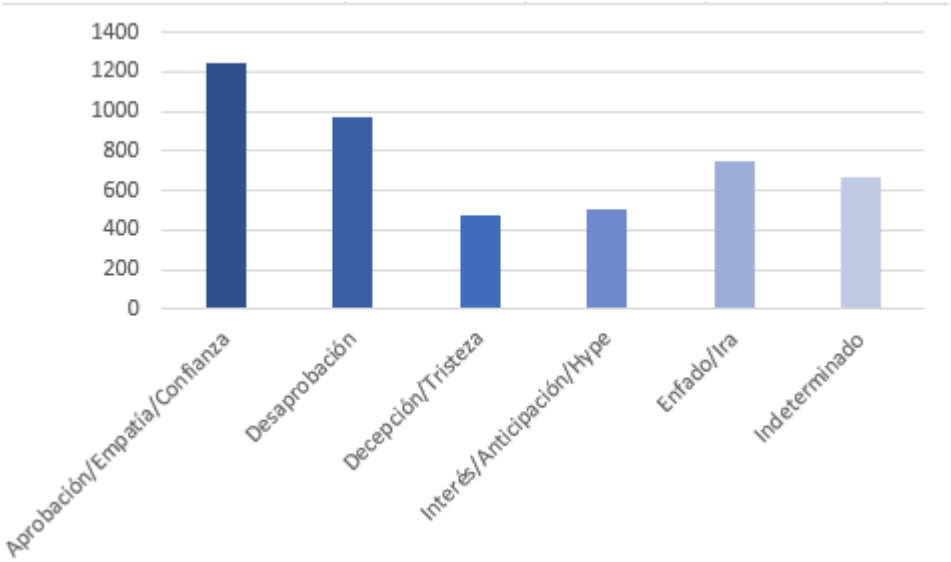


Figura 39: Distribución de emociones en la base de datos para Twitch

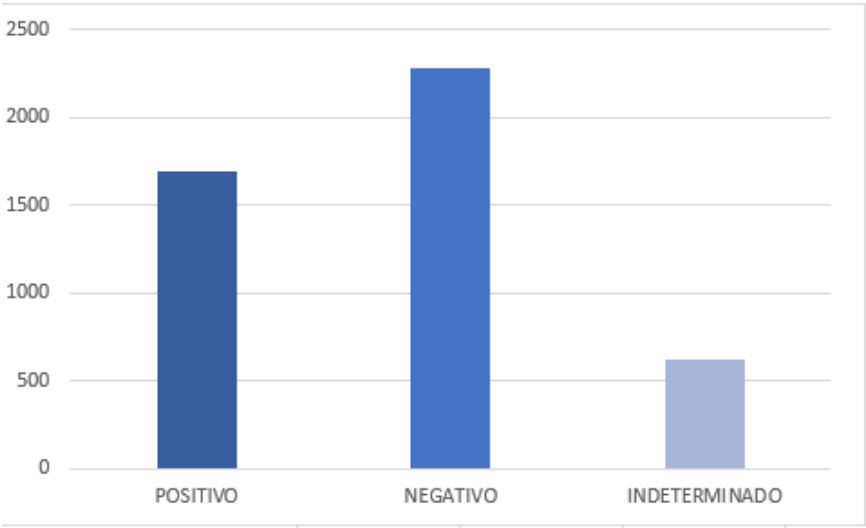


Figura 40: Distribución de polaridades en la base de datos para Twitch

6.3 Coste final del proceso

En esta sección desglosaremos el coste, tanto para emociones como para polaridad, del proceso de optimización de los modelos y de la obtención de los resultados para el conjunto de datos de Twitch.

6.3.1 Coste para la optimización y obtención de resultados para emociones

Al igual que nos ocurría para el conjunto de datos en Instagram, el cálculo del coste del proceso de optimización se obtiene al multiplicar el número de épocas definidas en el proceso de optimización, por el número de tokens con el que cuenta nuestra base de datos de entrenamiento por el precio de los tokens.

De la página de los precios de OpenAI, averiguamos que el precio por millón de tokens para realizar la optimización de modelos es de $8 [37]. Para nuestro caso, al multiplicar el número de épocas, que era 3, por los tokens de la base de entrenamiento obtenemos un total de 912.525 tokens; resultando en un coste total de $7,3.

Por otro lado, para la obtención de los resultados es necesario pagar, ya que se obtienen realizando solicitudes de chat al API, utilizando como modelo el previamente optimizado. El coste de este proceso, en el que utilizamos los datos de prueba, es bastante inferior ya que el número de tokens se reduce considerablemente. Este proceso cuesta $0,35.

6.3.2 Coste para la optimización y obtención de resultados para polaridad

Para la obtención de este coste tanto para la optimización como para las solicitudes de chat al API, hay que seguir el proceso anteriormente explicado.

En el caso de optimización, el número de épocas de mantuvo en 3, de manera que los tokens una vez multiplicado dicho valor es 521.268, resultando en un coste final de $4,18.

En cambio, el coste para la obtención de los resultados mediante solicitudes de chat al API supuso un coste de $0,21.

6.4 Proceso de optimización y obtención de los resultados

Para realizar el proceso de optimización del modelo será necesario preparar tanto los datos de entrenamiento como los de prueba al igual que como se hizo con el corpus para salud mental para Instagram. Seguidamente se seguirán los mismos pasos descritos en las Secciones 5.5.3 y 5.5.4, para los ficheros de entrenamiento de emociones y polaridad obtenidos para Twitch.

En el proceso de optimización, se obtienen las gráficas de *train_mean_token_accuracy* y *train_loss* [42] tanto para emociones (Figura 41) como para polaridad (Figura 42), las cuales quedan guardadas en wandb. En este caso, al observar la gráfica de *train_loss* para emociones se llega a la conclusión de que el modelo está mejorando la capacidad de hacer predicciones por su tendencia descendente en cambio para polaridad tenemos unos picos bastante alto los cuales pueden deberse a lotes de datos más complicados. Por otro lado, en las gráficas de *train_mean_token_accuracy* las precisiones se mantienen bastante alta a pesar de algunas caídas, por lo que podría decirse que el modelo clasifica correctamente la mayoría de los tokens.

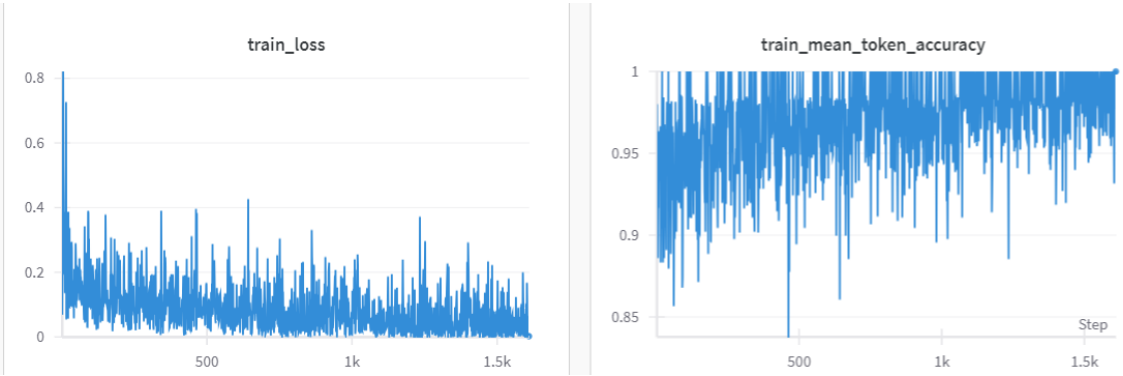


Figura 41: Gráficas del *train_mean_token_accuracy* y *train_loss* para el modelo optimizado de emociones de Twitch

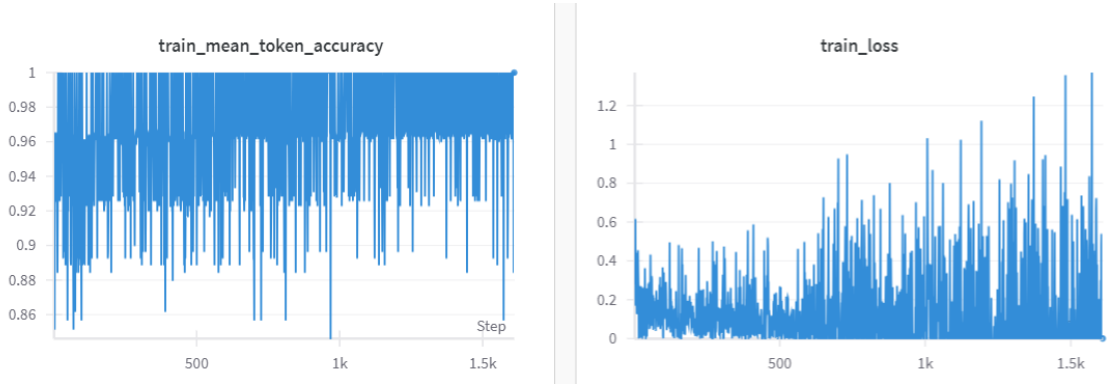


Figura 42: Gráficas del *train_mean_token_accuracy* y *train_loss* para el modelo optimizado de polaridades de Twitch

Por último, se seguirán los pasos de la Sección 5.6 para poder obtener los resultados. El código a utilizar se mantiene igual, a excepción del tipo de modelo a utilizar, que corresponderá con el modelo optimizado que le pertenezca. Además, en la definición de la función de *realizar_solicitud* en el contenido del rol de usuario nos debemos asegurar que el texto sea una cadena y no un número, para ello utilizamos la función *str()*.

6.5 Análisis de los resultados para polaridad con gpt-3.5-turbo

Como ocurría para la obtención de los resultados de comentarios en Instagram, el algoritmo de OpenAI decide los hiperparámetros que mejor se ajustan a nuestra base de datos de entrenamiento de esta manera nos aseguramos de que los resultados sean lo más precisos posibles.

En el proceso de optimización ya establecimos el número de épocas de manera que el algoritmo ha determinado los mejores hiperparámetros considerando 3 épocas, siendo estos los siguientes:

- Gpt-3.5-turbo: num_epochs=3, batch_size= 6, learning_rate_multiplier= 2.

Seguidamente, se presenta en la Tabla 12 una comparativa de los resultados obtenidos en polaridad en el estudio del año pasado de Javier [2] y de este año de Miguel [3], ambos usan RoBERTuito (un modelo basado en BERT), asimismo se presentarán los resultados obtenidos con el modelo gpt-3.5-turbo optimizado. En la tabla, podemos

observar en porcentajes los datos obtenidos en precisión, *recall* y *F1-score*, además del promedio macro y ponderado. También, se indican el número de instancias de cada una de las clases en nuestro estudio actual.

Para facilitar la comprensión de los resultados y poder comparar de mejor manera tanto, la precisión de cada uno de los modelos para las distintas polaridades, como la precisión global, se presentan en la Tabla 13 dichos resultados, extraídos de la tabla anterior.

Polaridad	**Métrica**	**RoBERTuito (%) (estudio Javier)**	**RoBERTuito (%) (estudio Miguel)**	**Gpt-3.5-turbo (%) (estudio actual)**	**Intancias (estudio actual)**
Negativo	Precisión	75%	83,3%	88%	689
	Recall	85%	87,5%	88%	
	F1-score	80%	85,3%	88%	
Positivo	Precisión	82,3%	80,4%	82%	499
	Recall	85,6%	81,7%	86%	
	F1-score	84%	80,9%	84%	
Indeterminado	Precisión	52,5%	68,7%	61%	192
	Recall	18,4%	52,2%	55%	
	F1-score	27,1%	59,1%	58%	
macro-avg	Precisión	-	77,4%	77%	1380
	Recall	-	73,8%	76%	
	F1-score	-	75,1%	77%	
weighted-avg	Precisión	-	80%	82%	1380
	Recall	-	81%	82%	
	F1-score	-	80%	82%	
Global	Accuracy	78%	81%	82%	

Tabla 12: Resultados obtenidos en la clasificación de Polaridad para Twitch

Polaridad	**Precisión (%) (estudio Javier)**	**Precisión (%) (estudio Miguel)**	**Precisión (%) (estudio actual)**
Negativo	75%	83,3%	88%
Positivo	82,3%	80,4%	82%
Indeterminado	52,5%	68,7%	61%
Accuracy	78%	81%	82%

Tabla 13: Comparación de los resultados obtenidos en la métrica de precisión en la clasificación de Polaridad para mensajes de Twitch

Como se puede observar de los resultados anteriores, el modelo actual realizado con gpt-3.5-turbo presenta una precisión global ligeramente superior respecto a los otros dos modelos. Todos ellos presentan una precisión en torno al 80%, lo cual implica que

los modelos son bastante robustos. No obstante, se realizará una comparación más exhaustiva, evidenciando así los cambios en el rendimiento de las distintas clases de polaridad.

En la clase "Negativo", se puede observar que el modelo gpt-3.5-turbo muestra una gran capacidad para identificar aquellos mensajes con esta polaridad, alcanzando una precisión del 88%, *recall* del 88% y *F1-score* del 88%. Como podemos ver los valores alcanzado son muy buenos, cercanos al 90%, de manera que podemos afirmar que el gpt-3.5-turbo es muy robusto y eficaz. Todos estos resultados superan los alcanzados en el estudio de Javier [2] (Precisión: 75%, Recall: 85%, F1-score: 80%) y el de Miguel [3] (Precisión: 83,3%, Recall: 87,5%, F1-score: 85,3%), indicando que la detección de emociones negativas es bastante fiable.

Por otro lado, en la clase "Positiva" seguimos manteniendo valores bastante elevados todos ellos ligeramente superiores al 80%. En este caso, nuestro modelo presenta una precisión ligeramente inferior con respecto al modelo de estudio de Javier [2] (82,3%), siendo esta diferencia de solo un 0,3%. Con respecto al estudio de Miguel [3] su modelo presenta una precisión ligeramente inferior a los otros dos modelos. En cambio, si nos referimos a las métricas de *Recall* y *F1-score* el modelo basado en gpt-3.5-turbo presenta mejores valores siendo estos un 86% y 84% ligeramente superiores respecto a los otros modelos (RoBERTuito Javier: Recall: 85,6%, F1-score: 84% y RoBERTuito Miguel: Recall: 81,7%, F1-score: 80,9%).

Por último, la clase "Indeterminado" es aquella que presenta los peores resultados. A pesar de que los resultados obtenidos en el estudio actual (Precisión: 61%, Recall: 55%, F1-score: 58%) son superiores con respecto al estudio de Javier (Precisión: 52,5%, Recall: 18,4%, F1-score: 27%), estos son parecidos e incluso ligeramente inferiores al estudio de Miguel (Precisión: 68,7%, Recall: 52,2%, F1-score: 59,1%). Todo esto nos sugiere que la detección de la polaridad no es sencilla para ninguno de los modelos, lo que puede deberse a la ambigüedad que presentan dichos mensajes, en los que no hay una clara polaridad definida. Otro factor que ha podido influir en los resultados es la cantidad de instancias de dicha polaridad en el corpus, la cual es bastante inferior con respecto al resto de clases.

Con el objetivo de entender mejor los resultados obtenidos y el posible motivo de estos, en la Figura 43 se presenta la matriz de confusión obtenida este año. De esta manera, podremos identificar qué clases son las que más se confunden entre ellas.

La matriz de confusión refuerza en este caso, los resultados que se observan en las Tablas 12 y 13. De manera que la polaridad Negativa es la que tiene un mayor número de aciertos siendo estos 603, y se confunde con un porcentaje parecido con mensajes que en verdad tienen polaridad positiva o indeterminada. En este caso se han detectado erróneamente 46 mensajes con polaridad positiva y 40 con polaridad indeterminada.

Por otro lado, en la clase Indeterminada, el modelo gpt-3.5-turbo presenta un desempeño moderado, en el que se han obtenido 105 aciertos y los errores están distribuidos de manera prácticamente equitativa entre las polaridades negativa y positiva, siendo los fallos 40 y 47 respectivamente. Esto coincide con los resultados obtenidos en las tablas, en las que esta clase mostraba una precisión inferior al resto. Como ya mencioné anteriormente esto puede ser debido a que estos mensajes suelen ser bastante ambiguos o incluso que haya dobles connotaciones de las otras dos clases, de manera que al algoritmo le resulte confuso y pueda interpretar la clase que esta más marcada. Además, cabe resaltar que el número de instancias referidas a esta clase en el corpus es bastante inferior con respecto a las clases positiva y negativa.

Por último, en el caso de la clase Positiva también se obtiene un alto rendimiento en el que tenemos un total de 430 aciertos. En cambio, se puede observar que hay una ligera confusión con respecto a la clase negativa, dando lugar a 43 fallos.

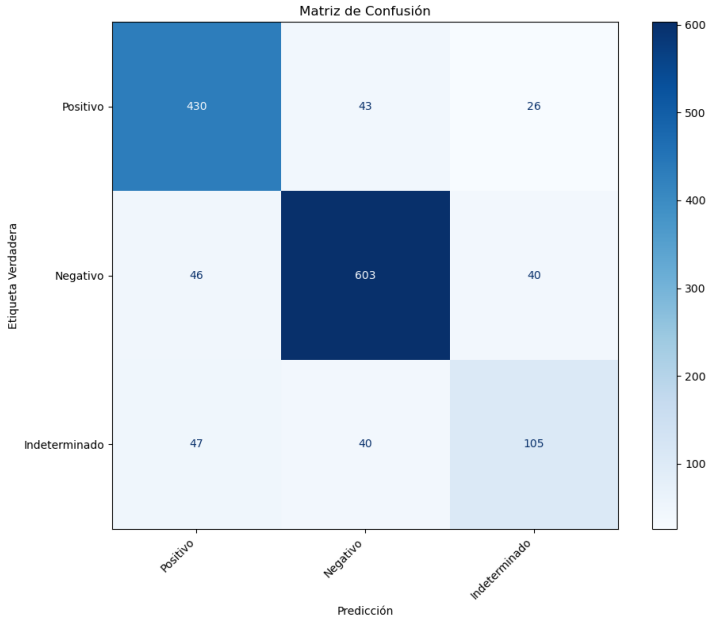


Figura 43: Matriz de confusión para la clasificación de polaridades en Twitch en el modelo gpt-3.5-turbo

En conclusión, la matriz de confusión respalda los resultados obtenidos, en los que se destaca el buen rendimiento del modelo para la detección de las clases negativas y positivas, pero se evidencia una deficiencia a la hora de detectar aquellos mensajes con polaridad indeterminada.

6.6 Análisis de los resultados para emociones con gpt-3.5-turbo

Teniendo en cuenta la base de datos de entrenamiento para emociones de Twitch y con un número de épocas definido previamente en el proceso de optimización como 3, el algoritmo de gpt-3.5-turbo proporcionó los siguientes hiperparámetros:

- Gpt-3.5-turbo: num_epochs=3, batch_size= 6, learning_rate_multiplier= 2.

La Tabla 14 presenta los resultados obtenidos en la clasificación de emociones para Twitch, en ella observamos las métricas para cada una de las clases, además de los promedios macro y ponderado. También, se muestran los resultados de los modelos utilizados en dos estudios, uno realizado por Javier [2] y otro por Miguel [3], en ambos estudios se utiliza el modelo de RoBERTuito. Además, completamos la tabla añadiendo el número de instancias para cada clase en el modelo actual con gtp-3.5-turbo.

Emoción	Métrica	RoBERTuito (%) (estudio Javier)	RoBERTuito (%) (estudio Miguel)	Gpt-3.5-turbo (%) (estudio actual)	Instancias (estudio actual)
Aprobación/Empatía /Confianza	Precisión Recall F1-score	76% 76,5% 75,7%	71% 77,1% 74%	73% 80% 77%	363
Desaprobación	Precisión Recall F1-score	59,7% 65,3% 62%	57,1% 65,3% 60,6%	63% 61% 62%	288
Decepción/Tristeza	Precisión Recall F1-score	85% 93% 89%	80% 73% 76,1%	74% 76% 75%	146
Interés/Anticipación /Hype	Precisión Recall F1-score	74,7% 67,5% 70%	71,7% 63,7% 67,3%	74% 79% 76%	144
Enfado/Ira	Precisión Recall F1-score	59% 56,5% 56%	76,4% 71,3% 73,6%	77% 76% 76%	232
Indeterminado	Precisión Recall F1-score	52,3% 39,7% 44%	64,1% 51,2% 56,6%	74% 79% 76%	205
macro-avg	Precisión Recall F1-score	- - -	70% 67% 68%	70% 70% 70%	1378
Weighted-avg	Precisión Recall F1-score	- - -	68,9% 68,1% 68%	70% 70% 70%	1378
Global	Accuracy	68%	68.1%	70%	

Tabla 14: Resultados obtenidos en la clasificación de Emociones para Twitch

Con el objetivo de facilitar la comparación y observar con mayor claridad los resultados de precisión de cada uno de los modelos para las distintas emociones, así como su precisión global, se ha extraído de la tabla anterior dichos resultados y se han presentado en la Tabla 15.

En dicha tabla, podemos observar que las presiones globales de los modelos se encuentran en torno al 70%. Esto no indica que, aunque los modelos son buenos, todavía tienen margen de mejora.

Para el análisis de cada una de las emociones, nos basaremos tanto en los resultados obtenidos en las Tablas 14 y 15 como en la matriz de confusión (Figura 44). De tal manera que podemos estudiar la capacidad de nuestro modelo optimizado. La clase "Aprobación/Empatía/Confianza", como podemos observar en la matriz de confusión y en la Tabla 14, presenta un total de 291 aciertos, siendo "Indeterminado" y "Interés/Anticipación/Hype" las emociones que más se confunden con un total de 20 y 21 fallos respectivamente. La precisión de esta emoción en el modelo actual es de un

73%, algo superior que lo obtenido en el modelo de Miguel, pero inferior a lo que obtuvo Javier en su estudio. En cambio, para las métricas de Recall y F1-score se obtienen los mejores resultados siendo estos 80% y 77%, en comparación con los estudios anteriores (RoBERTuito Javier: Recall: 76,5%, F1-score:75,7% y RoBERTuito Miguel: Recall: 77,1%, F1-score: 74%).

Polaridad	**Precisión (%) (estudio Javier)**	**Precisión (%) (estudio Miguel)**	**Presisión (%) (estudio actual)**
Aprobación/Empatía/Confianza	76%	71%	73%
Desaprobación	59,7%	57,1%	63%
Decepción/Tristeza	85%	80%	74%
Interés/Anticipación/Hype	74,7%	71,7%	74%
Enfado/Ira	59%	76,4%	77%
Indeterminado	52,3%	64,1%	60%
Accuracy	68%	68,1%	70%

Tabla 15: Comparación de los resultados obtenidos en la métrica de precisión en la clasificación de Emociones para mensajes de Twitch

En relación con la clase “Desaprobación”, observamos en las tablas que su precisión del 63% ha aumentado en relación con los otros dos modelos, aunque presenta un *F1-score* (62%) similar a los otros modelos, pero un *Recall* (61%) inferior (RoBERTuito Javier: Precisión:59,7%, Recall: 65,3%, F1-score:62% y RoBERTuito Miguel: Precisión:57,1%, Recall: 65,3%, F1-score:60,6%). En cambio, si nos fijamos en la matriz de confusión se ha obtenido un total de 175 aciertos. Pero se trata de la emoción que proporciona un mayor número de confusiones entre las emociones, detectándose 29 confusiones con “Aprobación/Empatía/Confianza”, 22 con “Decepción/Tristeza”, 32 con “Enfado/Ira” y 25 con “Indeterminado”. El mayor número de confusión se detecta con enfado e ira, lo cual puede deberse a una posible connotación negativa del mensaje.

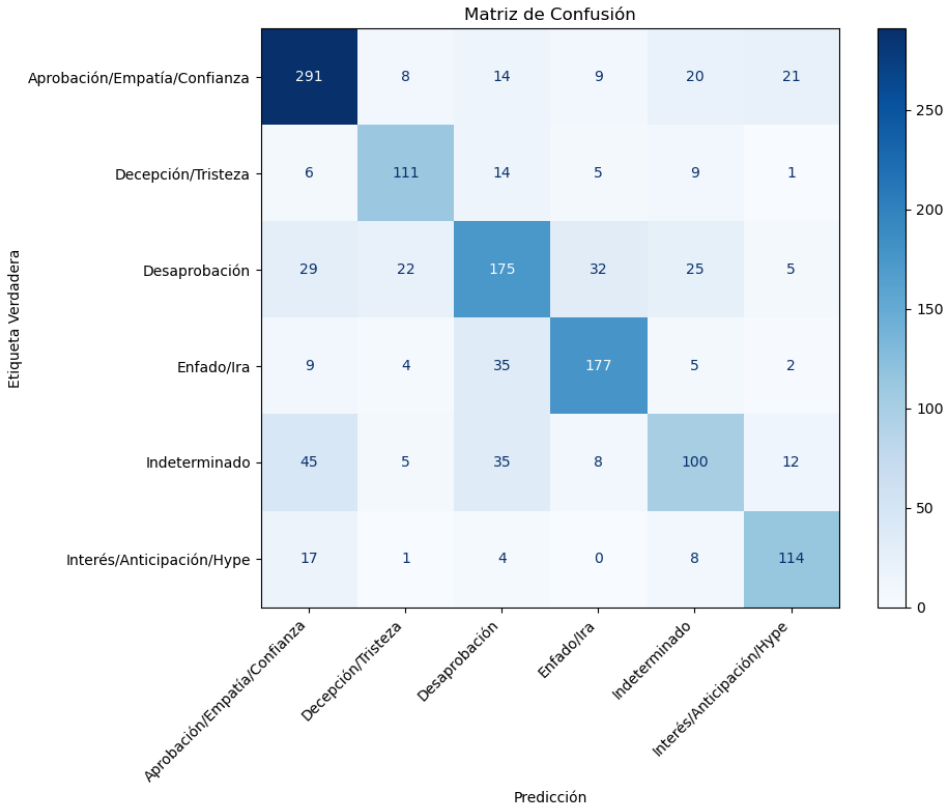


Figura 44: Matriz de confusión para la clasificación de emociones en Twitch

Al contrario de lo que ocurre con la clase de “Desaprobación”, la clase “Decepción/Tristeza” presenta una precisión del 74%, inferior respecto a los modelos anteriores. Para esta clase se observan que los resultados obtenidos por el estudio de Javier son los mejores, todos ellos cercanos al 90%. Sin embargo, al observar la matriz de confusión, se puede comprobar que presenta una alta tasa de acierto, contando con 111 aciertos de 146 instancias totales. Por otro lado, se obtiene que las clases que más se confunden con ella son “Desaprobación” con 14 fallos e “Indeterminado” con 9 fallos. El hecho de que la desaprobación se confunda con la tristeza puede deberse a que un sentimiento puede desembocar en el otro.

Por otro lado, la clase de “Enfado/Ira” mejora la precisión respecto a ambos modelos anteriores, en el que obtenemos los mejores resultados de precisión. Mientras que en los modelos anteriores obteníamos una precisión del 59% (RoBERTuito Javier) y 71,7% (RoBERTuito Miguel) ahora pasamos a tener el 77%. Estos resultados se respaldan con lo obtenido en la matriz de confusión, donde la tasa de acierto es considerablemente buena, con 177 aciertos. Por otro lado, se observa que la emoción con la que más se confunde es “Desaprobación” con un total de 35 fallos.

Para la clase de "Interés/Anticipación/Hype" los resultados de precisión obtenidos, siendo este 74%, se encuentran entre los obtenidos en el modelo del estudio de Javier (74,7%) y el de Miguel (71,7%). A pesar de ello, los resultados de las métricas de *Recall* (79%) y *F1-score* (76%) son superiores a ambos modelos anteriores (RoBERTuito Javier: Recall: 67,5%, F1-score:70% y RoBERTuito Miguel: Recall: 63,7%, F1-score: 67,3%). Si observamos la matriz de confusión vemos que la tasa de acierto es bastante buena, ya que hay 114 aciertos de las 144 instancias de dicha clase. Por otro lado, se observa que dicha emoción se confunde principalmente con "Aprobación/Empatía/Confianza", presentándose 17 fallos.

Por último, la clase "Indeterminado" presenta los mejores resultados en todas las métricas respecto a los modelos anteriores, teniendo una precisión del 74%. En cambio, si nos fijamos en la matriz de confusión su tasa de acierto no es tan alta, teniendo 100 aciertos de las 205 instancias que teníamos en la base de datos de prueba. Una posible razón por la que esto sucede puede ser debido a que los mensajes no muestran una emoción clara, donde los textos son bastante ambiguos. Esto lo observamos en la matriz al ver que se presentan 45 fallos interpretando los mensajes como "Aprobación/Empatía/Confianza" y 35 de "Desaprobación".

En resumen, como hemos podido observar el modelo actual presenta una precisión global algo superior a los modelos anteriores. Aunque, si bien es cierto que en alguna emoción se presenta alguna precisión ligeramente inferior, por lo general se puede observar cierta mejora respecto a los otros modelos. Esto confirma la eficacia y robustez del modelo actual ante la detección de emociones. Asimismo, todavía queda espacio para mejorar dicho rendimiento, ya sea ampliando el número de instancias de aquellas emociones que presentan una menor precisión o detectando mejor los patrones de lenguaje en entornos de videojuegos, el cual tiene una gran complejidad.

7

Conclusiones y líneas futuras

7.1 Conclusiones

Este Trabajo de Fin de Grado ha conseguido expandir y mejorar un corpus lingüístico existente, centrado en el análisis de sentimientos y polaridad de comentarios en publicaciones de Instagram relacionadas con la salud mental. La estrategia seguida, que se centró en la selección de influencers masculinos españoles y en la identificación de las emociones con una menor representación en el corpus inicial, ha demostrado ser eficaz para diversificar y equilibrar el conjunto de datos. Además, para lograr un balance de género más equitativo, amplié el corpus incorporando publicaciones de hombres que abordan temas de salud mental, ya que previamente estas estaban subrepresentadas.

De este modo, los resultados obtenidos tanto para polaridad como emociones en lo que se ha utilizado un modelo optimizado de gpt-3.5-turbo son bastante buenos. Principalmente se ha notado una mejora en la identificación de comentarios de emociones como “Compresión/Empatía/Identificación” y en el resto de los casos se han mantenido resultados parecidos a los de estudios anteriores, siendo estos bastante buenos. Aunque si bien es cierto, la identificación de la clase “Indeterminada” sigue siendo un reto ya que es la que menor precisión presenta.

Por otro lado, los resultados obtenidos con el mismo algoritmo y estrategia en el ámbito de videojuegos dentro de la plataforma Twitch, utilizando los chats de transmisiones, muestran una mejora significativa en comparación con estudios previos tanto en polaridad como en emociones, especialmente en las clases de "Enfado/Ira" e "Indeterminado". Sin embargo, aunque la clase "Desaprobación" ha mejorado en precisión, sigue siendo la que presenta los resultados más bajos.

En resumen, este trabajo ha establecido un punto de partida para nuevas investigaciones de análisis de sentimientos en redes sociales y en el contexto de videojuegos. Mediante la optimización de un modelo de OpenaAI, se han obtenido resultados significativos y se ha desarrollado una herramienta que ayuda en el avance en este campo, ofreciendo nuevas perspectivas para comprender la respuesta emocional en los entornos digitales.

7.2 Líneas futuras

Algunas líneas futuras que se proponen, considerando que los resultados obtenidos son bastante positivos, aunque existen clases de emociones en las que la precisión se podría mejorarse, son las siguientes:

- Optimización del modelo para clases con baja precisión: Se podría refinar los modelos para mejorar la precisión en las clases de emociones que han demostrados que son más difíciles de identificar. Para ello, se podría investigar en técnicas para adecuar los hiperparámetros.
- Ampliación del corpus: A pesar de que ya se ha hecho una ampliación de este, sería conveniente incluir más datos y aumentar la presencia de aquellas clases que sigan con una menor representación. Por otro lado, sería importante aumentar el número de publicaciones de influencers masculinos para poder observar de una mejor manera si las reacciones son distintas en función del género.
- Personalización del modelo: Sería interesante desarrollar versiones personalizadas del modelo para que se puedan adaptar a diferentes tipos de contenido o comunidades.
- Estudio de otros modelos: Aunque en este trabajo nos hemos centrado en la optimización del modelo gpt-3.5-turbo de OpenAI, sería conveniente realizar el estudio en los nuevos modelos disponibles como el gpt-4.

8

Bibliografía

[1] F. Neri, C. Aliprandi, F. Capeci, M. Cuadros and T. By, "Sentiment Analysis on Social Media," *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Istanbul, Turkey, 2012, pp. 919-926, doi: 10.1109/ASONAM.2012.16

[2] J. E. Asensio, N. M. Alvarez, y J. M. V. Hernández, "Análisis emocional en redes sociales basados en modelos de aprendizaje automático transformers BERT", Universidad de Valladolid, 2023. Disponible en: https://uvadoc.uva.es/handle/10324/62911.

[3] M. C. Lanchares y N. M. Álvarez, "Aplicación de aprendizaje automático para evaluar lenguaje de videojuegos en Twitch", Universidad de Valladolid, 2024.

[4] OpenAI. Disponible en: https://openai.com/

[5] Morcela, O. A. (2022). *ChatGPT: la IA está aquí y nos desafía. AACINI-Revista Internacional de Ingeniería Industrial, (6).* Disponible en: *https://www3.fi.mdp.edu.ar/otec/revista/index.php/AACINI-RIII/article/view/67*

[6] Zhang, M., & Li, J. (2021). A commentary of GPT-3 in MIT Technology Review 2021. *Fundamental Research*, *1*(6), 831-833. Dispnible en: https://www.sciencedirect.com/science/article/pii/S2667325821002193

[7] Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., ... & Huang, X. (2023). *A comprehensive capability analysis of gpt-3 and gpt-3.5 series models*. *arXiv preprint arXiv:2303.10420*. Disponible en: https://arxiv.org/pdf/2303.10420

[8] Roumeliotis, K. I., & Tselikas, N. D. (2023). Chatgpt and open-ai models: A preliminary review. *Future Internet*, *15*(6), 192. Disponible en: https://www.mdpi.com/1999-5903/15/6/192

[9] Kemp, S. (2024, January 31). *Digital 2024: Global Overview Report — DataReportal – Global Digital Insights*. DataReportal. Disponible en: https://datareportal.com/reports/digital-2024-global-overview-report

[10] Kemp, S. (2020, July 21). *More than half of the people on Earth now use social media — DataReportal – Global Digital Insights*. DataReportal. Disponible en: https://datareportal.com/reports/more-than-half-the-world-now-uses-social-media?rq=Social%20media

[11] Bashir, H., & Bhat, S. A. (2017). Effects of social media on mental health: A review. *International Journal of Indian Psychology*, *4*(3), 125-131. Disponible en: https://www.researchgate.net/profile/Shabir-Bhat/publication/323018957_Effects_of_Social_Media_on_Mental_Health_A_Review/links/5a7c9e97aca272341aeb7472/Effects-of-Social-Media-on-Mental-Health-A-Review.pdf

[12] Welcome to Python.org. Disponible en: https://www.python.org.

[13] Aurora. (2023, July 18). *¿Qué son las librerías de Python? - ID Bootcamps*. ID Digital School. Disponible en: https://iddigitalschool.com/bootcamps/que-son-las-librerias-de-python/

[14] Os. Disponible en: https://docs.python.org/es/3.10/library/os.html

[15] Libreía OpenAI. Disponible en: https://pypi.org/project/openai/0.26.5/

[16] Librería Wandb. Disponible en: https://pypi.org/project/wandb/

[17] Pandas. Disponible en: https://pandas.pydata.org/

[18] Json. Disponible en: https://docs.python.org/es/3/library/json.html

[19] Sklearn.metrics. Disponible en: https://scikit-learn.org/stable/

[20] Ratelimit. Disponible en: https://pypi.org/project/ratelimit/

[21] Matplotlib.pyplot. Disponible en: https://matplotlib.org/

[22] Numpy. Disponible en: https://numpy.org/

[23] Ramdom. Disponible en: https://docs.python.org/es/3/library/random.html

[24] Collections. Disponible en: https://docs.python.org/es/3/library/collections.html

[25] Tiktoken. Disponible en: https://pypi.org/project/tiktoken-async/

[26] Jupyter Notebook. Disponible en: https://jupyter.org/

[27] Wandb. Disponible en: https://wandb.ai/site

[28] Instagram. Disponible en: https://www.instagram.com/

[29] Luque, A., Carrasco, A., Martín, A., & de Las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, *91*, 216-231. Disponible en: https://www.sciencedirect.com/science/article/pii/S0031320319300950

[30] Otero, C. (2022, February 21). *Beret narra su lucha contra la ansiedad: "Lleva conmigo muchos años, pero nunca he parado de vivir"*. Divinity. Disponible en: https://www.divinity.es/actualidad/beret-ansiedad_18_3286774421.html

[31] Carbonero, S., & Taboada, N. (2023, July 11). *Ángel Martín habla del miedo a la locura y de cómo acabó en un centro psiquiátrico*. El Periódico. Disponible en: https://www.elperiodico.com/cuore/famosos/2023/07/11/angel-martin-brote-psicotico-centro-psiquiatrico-89739317

[32] Sharples, M. (n.d.). *GPT-3*. Wikipedia. Disponible en: https://es.wikipedia.org/wiki/GPT-3

[33] *Models*. (n.d.). OpenAI API. Disponible en: https://platform.openai.com/docs/models/gpt-3-5-turbo

[34] *Chat completions with GPT | OpenAI*. (n.d.). DataCamp. Retrieved August 22, 2024, from https://campus.datacamp.com/courses/working-with-the-openai-api/openais-text-and-chat-capabilities?ex=8

[35] Gillis, A. S. (n.d.). *What is data splitting and why is it important?* TechTarget. Disponible en: https://www.techtarget.com/searchenterpriseai/definition/data-splitting

[36] *Data preparation and analysis for chat model fine-tuning*. (2023, August 21). OpenAI Cookbook. Disponible en: https://cookbook.openai.com/examples/chat_finetuning_data_prep

[37] *Pricing*. (n.d.). OpenAI. Disponible en: https://openai.com/api/pricing/

[38] *Fine-tuning*. (n.d.). OpenAI API. Retrieved August 22, 2024, from https://platform.openai.com/docs/guides/fine-tuning/fine-tuning-integrations

[39] Ibrahim, M. (2023, November 22). *Fine-Tuning ChatGPT for Sentiment Analysis With W&B*. Wandb. Disponible en: https://wandb.ai/mostafaibrahim17/ml-articles/reports/Fine-Tuning-ChatGPT-for-Sentiment-Analysis-With-W-B--Vmlldzo1NjMzMjQx

[40] Ibrahim, M. (2023, November 22). *Fine-Tuning ChatGPT for Question Answering With W&B*. Wandb. Disponible en: https://wandb.ai/mostafaibrahim17/ml-

articles/reports/Fine-Tuning-ChatGPT-for-Question-Answering-With-W-B--Vmlldzo1NTEzNjU2

[41] *Create fine-tuning job*. (n.d.). API Reference OpenAI. https://platform.openai.com/docs/api-reference/fine-tuning/create

[42] Lad, J. (2024, April 20). *Balancing Training Loss and train_mean_token_accuracy in OpenAI API Fine-Tuning*. Medium. Disponible en: https://medium.com/@lad.jai/balancing-training-loss-and-train-mean-token-accuracy-in-openai-api-fine-tuning-d92346bb8bc9

[43] Bae, E. (2023, October 25). *Learn Fine Tuning: Making GPT-3.5 Better at Recognizing Sentiment*. Medium. Retrieved August 22, 2024, from https://medium.com/bertcode/learn-fine-tuning-making-gpt-3-5-better-at-recognizing-sentiment-9dc7a36966ae

[44] N. Merayo, R. Cotelo, R. Carratalá-Sáez, y F. J. Andújar, "Applying machine learning to assess emotional reactions to video game content streamed on Spanish Twitch channels", Comput. Speech Lang., vol. 88, núm. 101651, 2017. Disponible en: https://www.sciencedirect.com/science/article/pii/S0885230824000342?dgcid=rss_sd_all.