UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Detección y Clasificación de Emociones en Diálogos de Videojuegos mediante Procesamiento del Lenguaje Natural

Autor:
Víctor Sánchez Valencia
Tutora:
Noemí Merayo Álvarez

Valladolid, julio de 2025

TÍTULO:	**Detección y Clasificación de Emociones en Diálogos de Videojuegos mediante Procesamiento del Lenguaje Natural**
AUTOR:	**D. Víctor Sánchez Valencia**
TUTORA:	**Dra. Dña. Noemí Merayo Álvarez**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dra. Dña. Noemí Merayo Álvarez**
SECRETARIA:	**Dra. Dña. Lara del Val Puente**
VOCAL:	**Dr. D. Alfonso Bahillo Martínez**
SUPLENTE:	**Dr. D. Ramón de la Rosa Steinz**
SUPLENTE:	**Dr. D. Juan Carlos Aguado Manzano**

FECHA:
CALIFICACIÓN:

Resumen

Este Trabajo Fin de Grado desarrolla un modelo de análisis de sentimientos en diálogos del videojuego The Last of Us mediante Procesamiento del Lenguaje Natural (PLN) con modelos de OpenAI, específicamente GPT-4o-mini y GPT-4.1-mini. Se construyó desde cero un corpus lingüístico etiquetado manualmente con emociones y polaridad, aplicando en su parte final técnicas de balanceo para garantizar su representatividad. Se llevó a cabo el *fine-tuning* de los modelos seleccionados evaluando precisión y coste, destacando GPT-4.1-mini por su equilibrio entre rendimiento y coste. El estudio incluye un análisis comparativo de ambos modelos, estimación de costes asociados al entrenamiento y uso, así como una evaluación de las métricas obtenidas, como la matriz de confusión y el reporte de clasificación. Los resultados reflejan que el análisis lingüístico en videojuegos mediante Inteligencia Artificial (IA) puede contribuir a la comprensión del impacto social y cultural del lenguaje en este ámbito. Además, el trabajo propone líneas futuras de investigación para mejorar el análisis emocional en otros videojuegos, extendiendo la aplicación de estas técnicas a campos como la psicología, la sociología o la economía.

Palabras Clave: Procesamiento del Lenguaje Natural, Análisis de sentimientos, Inteligencia Artificial, Videojuego, The Last of Us, OpenAI, GPT, Fine-tuning, Emociones, Polaridad

Abstract

This Final Degree Project develops a model for detecting and classifying emotions in dialogues from the video game The Last of Us using Natural Language Processing (NLP) with OpenAI models, specifically GPT-4o-mini and GPT-4.1-mini. A linguistic corpus was built from scratch, manually labeled with emotions and polarity, applying balancing techniques to ensure its representativeness. The selected models were fine-tuned, evaluating precision and cost, with GPT-4.1-mini standing out for its balance between performance and cost. The study includes a comparative analysis of both models, an estimation of costs associated with training and usage, as well as an evaluation of the obtained metrics, such as the confusion matrix and classification report. The results show that linguistic analysis in video games through Artificial Intelligence (AI) can contribute to understanding the social and cultural impact of language in this field. Additionally, the project proposes future research lines to improve emotional analysis in other video games, extending the application of these techniques to fields such as psychology, sociology, or economics.

Keywords: Natural Language Processing, Sentiment Analysis, Artificial Intelligence, Video Game, The Last of Us, OpenAI, GPT, Fine-tuning, Emotions, Polarity

Agradecimientos

A todas aquellas personas que me han permitido acabar esta carrera que concluyo con este Trabajo Fin de Grado. En primer lugar, agradecer a mi tutora, Noemi, por haberme permitido realizar el trabajo con ella y abrirme un mundo más en las telecomunicaciones. Extiendo este agradecimiento a la profesora Rosalía Cotelo García de la Universidad Autónoma de Madrid y a la profesora Rocío Carratalá Sáez de la Universitat Jaume I que han estado en todo momento ayudándome con el trabajo, infinitas gracias. Quiero agradecer a mi familia, mi pilar más importante, vuestro apoyo constante ha sido mi motor para llegar hasta aquí. No me puedo olvidar de dar las gracias a todos los compañeros de la carrera por todo el soporte durante estos años. Gracias.

Abuelo,

"El esfuerzo tiene mucho más sentido cuando lo motiva algo que realmente nos importa, como por los valores y enseñanzas de personas que nos han marcado. Esas son las que nos impulsan a seguir adelante."

Índice

Índice de Figuras

Índice de Tablas

Índice de Ecuaciones

1 Introducción

1.1 Motivación

El presente Trabajo Fin de Grado (TFG) se centra en el análisis del lenguaje del videojuego *The Last of Us,* un fenómeno cultural y económico de gran relevancia en la sociedad actual. Los videojuegos han visto un crecimiento exponencial en su uso en las últimas décadas, consolidándose como una de las principales formas de entretenimiento, con un impacto económico en la industria que incluso supera a otras industrias como es la del cine o la música. Este auge ha llevado a que los videojuegos tengan una influencia en la sociedad cada vez mayor, especialmente en los adolescentes, siendo el segmento de la población que más consume estos recursos digitales.

El videojuego *The Last of Us,* aparte de su éxito comercial, destaca por su combinación perfecta de guion, narrativa y personajes completos y fascinantes, además de su capacidad para para generar experiencias emocionales complejas en los jugadores del videojuego (Vandal Team, 2022). Esto lo convierte en un objeto de estudio interesante para analizar el uso y evolución del lenguaje en el mundo digital.

La influencia de los videojuegos ha sido desde hace un tiempo motivo de debate social, destacando en muchas ocasiones con mayor peso las consecuencias negativas de estos, como el comportamiento agresivo, el aislamiento social o los problemas de salud que pueden sufrir los jugadores, frente a las positivas como la mejora de habilidades cognitivas.

En este contexto, es fundamental comprender cómo el lenguaje de los videojuegos interactúa con los jugadores y cómo se traslada a la sociedad. La industria del videojuego representa un motor económico en la actualidad, generando millones de euros cada día, lo que hace que se potencie el impacto de sus contenidos en la sociedad.

Este trabajo se motiva en la capacidad que ofrecen las técnicas de procesamiento de lenguaje natural y los modelos de inteligencia artificial de última generación como GPT-4.0 y GPT-4.1 de la empresa OpenAI, para analizar las emociones del lenguaje en el videojuego. El análisis comparativo de los modelos más actuales permitirá entender cómo se interpretan y

como responden a las estructuras del lenguaje "*gamer*", abriendo la posibilidad de mejora por parte de las empresas creadoras de videojuegos en la interacción jugador-juego y permitiendo entender el impacto cultural que tiene.

Por último, este trabajo busca aportar un conocimiento más profundo sobre la relación entre los videojuegos y la sociedad, en concreto sobre la comunicación y la transmisión de emociones, contribuyendo a un mejor entendimiento de un fenómeno que cada día tiene más calado en la vida de los adolescentes.

1.2 Objetivos

1.2.1 Objetivo principal

Este objetivo principal es emplear técnicas avanzadas de PLN mediante modelos de inteligencia artificial de última generación, en concreto los modelos de la empresa OpenAI, GPT-4.0 y GPT-4.1, para analizar el lenguaje y, en concreto, las emociones que genera el dialogo del videojuego *The Last of Us.* Se busca poder evaluar y comparar la capacidad de ambos modelos para interpretar, clasificar y extraer características del leguaje en el corpus extraído del juego.

Además, este TFG se basa en la creación y etiquetado de un corpus representativo de las oraciones y diálogos del juego, elaborado desde cero, con el fin de identificar la polaridad y la emoción que expresan. Además, este corpus servirá para llevar a cabo el entrenamiento de los modelos supervisados de IA. Por otra parte, el análisis comparativo de los modelos permitirá determinar cuál de ellos ofrece mejores prestaciones para adaptarse a datos relacionados con otros videojuegos diferentes de habla inglesa.

El objetivo final es desarrollar un modelo supervisado de inteligencia artificial, capaz de realizar un análisis preciso del lenguaje en videojuegos narrativos. El resultado puede ser de interés en ramas como la investigación cultural, la psicología, la economía o la sociología.

1.2.1 Objetivos específicos

Para alcanzar el objetivo principal de este trabajo, se establecen un listado de objetivos específicos:

- Recolectar y depurar el corpus de las oraciones y diálogos del videojuego *The Last of Us,* asegurando su calidad y entendimiento.

- Etiquetar el corpus para facilitar el entrenamiento de los modelos supervisados. Para ello se debe garantizar la representatividad de todas las emociones y las polaridades, y el equilibrado.
- Optimizar los modelos más adecuados para el análisis de los sentimientos, usando las técnicas de *fine-tuning*.
- Evaluar y comparar los resultados obtenidos para los distintos modelos en la clasificación de sentimientos del corpus, incluyendo la emoción y la polaridad.
- Estimar los costes asociados a todo el proceso de optimizado y uso de los modelos de la empresa OpenAI.
- Explorar las implicaciones sociales y culturales del análisis lingüístico en videojuegos, enfocándose en la contribución al bienestar emocional, la educación de calidad y la innovación tecnológica, alineando el trabajo con los Objetivos de Desarrollo Sostenible.
- Proponer recomendaciones para futuras investigaciones basadas en los resultados obtenidos del análisis comparativo y su impacto en la industria del videojuego y la sociedad.

1.3 Metodología

En esta sección se describe la metodología empleada para alcanzar los objetivos establecidos previamente. Para ello, el proceso se ha estructurado en distintas fases, las cuales se detallan a continuación.

1.3.1 Fase de documentación

En esta primera fase del proceso se ha realizado una toma de contacto con el PLN, utilizando modelos de OpenAI. El objetivo principal ha sido identificar herramientas capaces de analizar emociones en el ámbito de los videojuegos, así como conceptos clave, lenguajes de programación adecuados y librerías necesarias para su implementación.

En primer lugar, se llevó a cabo un análisis de los distintos modelos de inteligencia artificial disponibles para el procesamiento emocional, evaluando sus características técnicas, costes y requisitos de acceso. A continuación, se compararon estos modelos con el fin de seleccionar el más adecuado para los objetivos del proyecto.

Posteriormente, se eligieron las librerías más convenientes y se desarrolló código de alto nivel orientado a la creación de un modelo optimizado para el análisis de sentimientos y polaridad. Finalmente, se seleccionaron herramientas adicionales que facilitasen la obtención e interpretación de resultados.

Por otro lado, también se realizó un proceso de documentación y formación sobre el mundo de las emociones, las diferentes teorías y el análisis emocional, especialmente en el contexto de la inteligencia artificial y en la aplicación de técnicas de procesamiento de lenguaje natural.

1.3.2 Fase de creación del corpus The Last of Us

Esta fase representa una de las etapas más relevantes del procedimiento. En ella se ha recopilado el diálogo completo del videojuego *The Last of Us* y en una segunda fase del etiquetado (que se detallará a lo largo de la memoria) se tomó el diálogo completo de *The Last of Us II*, con el objetivo de etiquetar manualmente las oraciones según la emoción y la polaridad expresados por los personajes del videojuego.

El etiquetado se ha realizado de forma manual, clasificando cada intervención en función de la emoción predominante y su polaridad (positiva, negativa o neutra). Una vez completado este proceso, los datos han sido sometidos a una comprobación por pares para garantizar la coherencia y fiabilidad del corpus generado.

Posteriormente, se ha llevado a cabo un intento manual de balanceo del conjunto de datos, asegurando una representación más equitativa de cada categoría emocional. De este modo, se obtiene el corpus final, que será empleado en la fase de entrenamiento supervisado del modelo.

1.3.3 Fase de análisis con modelos GPT

En esta fase se ha llevado a cabo el entrenamiento de los modelos. Para ello, el corpus previamente etiquetado ha sido dividido en tres conjuntos: entrenamiento, validación y prueba (test), siguiendo las buenas prácticas en el aprendizaje supervisado.

Una vez optimizados los modelos para la tarea de análisis de sentimientos en el contexto del videojuego *The Last of Us,* estos han sido utilizados para obtener resultados que permitan evaluar su rendimiento y determinar si se han alcanzado los objetivos planteados inicialmente.

En una primera iteración del proceso, los resultados obtenidos en términos de precisión del modelo optimizado para el análisis de sentimientos fueron moderados. Ante esta situación, fue

necesario repetir la fase de etiquetado del corpus, ampliado su contenido y reetiquetando alguna de las emociones, con el objetivo de mejorar la calidad de los datos y, en consecuencia, obtener un mejor rendimiento de los modelos.

1.3.4 Fase de comparación entre modelos

En esta etapa se analizan los resultados obtenidos en la fase anterior con el fin de evaluar el grado de cumplimiento de los objetivos planteados inicialmente. Asimismo, se realiza una comparación entre los modelos seleccionados para su optimización, con el objetivo de determinar cuál de ellos ofrece un mejor rendimiento en la tarea de análisis de sentimientos.

1.3.5 Fase de redacción

En la última de las fases, se redacta el presente documento con el fin de que quede reflejo de todas las fases llevadas a cabo para el logro de los objetivos marcados. La finalidad de este documento no es otro que el de hacer comprender que herramientas se han usado, los pasos llevados a cabo y las conclusiones obtenidas en relación del análisis de las emociones en el videojuego *The Last of Us.*

1.4 Estructura de la Memoria

Esta memoria se encuentra estructurada en nueve capítulos. Algunos de ellos están dedicados a la definición de conceptos clave y a la descripción de la metodología utilizada, mientras que otros recogen de forma detallada los objetivos planteados, el desarrollo del proyecto y los resultados obtenidos.

- Capítulo 1: Introducción. En este capítulo se abordan los objetivos de este Trabajo Fin de Grado, las fases seguidas para obtener los resultados y la estructura general de esta memoria.
- Capítulo 2: Marco teórico y estado del arte. En esta sección se da una explicación del propio videojuego objeto de estudio, asimismo del impacto social del lenguaje de videojuegos. Además, se realiza una comparación de los modelos optimizables para escoger el, o los más adecuados para la tarea de análisis de sentimientos y se explica el potencial área de investigación en este ámbito.

- Capítulo 3: Herramientas y recursos utilizados. Aquí se exponen las herramientas empleadas para este trabajo, las definiciones más relevantes y las métricas empleadas para interpretar los resultados.
- Capítulo 4: Corpus lingüístico: *The Last of Us.* En esta sección se va a detallar la metodología seguida para la obtención del corpus desde cero, que recoge las oraciones extraídas del videojuego *The Last of Us*. Además, se explica los pasos seguidos para su depurado, comprobación por pares y para dar el formato requerido por los modelos de OpenAI.
- Capítulo 5: Evaluación de los costes de implementación de modelos de OpenAI con el corpus *The Last of Us*. En este capítulo se describen los pasos seguidos para estimar el coste asociado al entrenamiento y uso de los modelos de la empresa OpenAI. Además, se realiza una comparación entre la estimación inicial y el coste total final, con el fin de analizar posibles desviaciones y justificar los recursos empleados.
- Capítulo 6: Evaluación del modelo GPT-4o-mini en el contexto del estudio . En esta sección se detallan los pasos seguidos para la optimización del modelo GPT4o-mini orientado al análisis de emociones y polaridad, así como la obtención y el análisis de los resultados obtenidos.
- Capítulo 7: Análisis del corpus con GPT-4.1-mini. Al igual que en el capítulo anterior, en este se detallan los cambios realizados para obtener los modelos optimizados de las versiones GPT4.1-mini, así como los resultados obtenidos y su posterior análisis y comparación con el modelo GPT4o-mini.
- Capítulo 8: Conclusiones y líneas futuras. En este capítulo final se exponen las conclusiones principales y se sugieren nuevas líneas de actuación.

2
Marco teórico y Estado del arte

2.1 Introducción

En estos últimos años, el análisis de las emociones en los videojuegos se ha establecido como un campo de estudio multidisciplinar sólido, que implica otras ramas de la ciencia como la psicología, la sociología, las telecomunicaciones o la economía. Los videojuegos han evolucionado a gran velocidad. Han pasado de ser mero juego para el entretenimiento a proporcionar experiencias culturales complejas, en gran parte por la evolución de las tecnologías.

Este capítulo muestra un repaso de cómo se construyen las emociones a través del lenguaje verbal, no verbal en los videojuegos, el impacto social que tienen y el uso de la inteligencia artificial más avanzada para estudiar las reacciones de los jugadores.

2.2 El Lenguaje en los videojuegos. Impacto Social

El lenguaje en los videojuegos ha experimentado cambios importantes, puesto que no solo cumple una labor informativa-narrativa como venía haciendo, sino que también tiene una profunda carga emocional en sus diálogos y textos. A medida que esta industria ha crecido, también ha cambiado la manera de comunicar a través del juego, creando un medio de transmisión de valores, ideologías y normas sociales que moldean la forma de pensar y de actuar de los jugadores (Gee, 2003).

A través de los videojuegos, y en especial de su lenguaje, se suelen normalizar determinados comportamientos y actitudes. Por un lado, los videojuegos pueden reforzar comportamientos positivos, como el pensamiento crítico, el uso inclusivo del lenguaje o la representación a través de personajes de minorías sociales. Pero, por otra parte, y más frecuentemente en juegos de acción o supervivencia se fomentan actitudes negativas, como la agresividad, el lenguaje ofensivo o la violencia.

Casos como el de *The Last of Us* destacan por incluir un lenguaje muy realista y crudo, que busca profundizar en la narrativa del juego puede también contribuir a desensibilizar la violencia o normalizar actitudes negativas en la sociedad.

Por tanto, se hace necesario estudiar el lenguaje en los videojuegos desde la perspectiva lingüística y social para evaluar el impacto positivo y consecuencias negativas que puede tener sobre la sociedad, y los jóvenes en especial.

2.3 Descripción del videojuego *The Last of Us*

The Last of Us (TLoU) es un videojuego de género acción y s*urvival horror* desarrollado por Naughty Dog y puesto en marcha por Sony Interactive Entertainment en 2013 para la consola Play Station 3 (Fandom, s.f. para *The Last of Us*).

Sus directores Bruce Straley, Neil Druckmann tomaron la idea principal del videojuego de una serie de la cadena inglesa BBC llamada "Planet Earth". Pero también pudieron tener influencia la novela *The Road, The World Without Us* o películas como *No Country for Old Men (2007)* y *Gravity (Wikipedia, 2025).*

El videojuego se desarrolla por la vida de Joel, un hombre endurecido por la pérdida que, en un mundo postapocalíptico devastado por una infección del hongo Cordyceps que convierte a las personas en criaturas violentas, debe acompañar y proteger a Ellie, una joven clave para la cura de la infección. A lo largo de su viaje por ciudades abandonadas, Joel enfrenta tanto a infectados (*clickers*) como a supervivientes desesperados, mientras su relación con Ellie evoluciona de la desconfianza al vínculo paternal, obligándolo a tomar decisiones que pondrán a prueba su moral y el destino de la humanidad (Fandom, s.f. para The Last of Us).

La productora HBO tomó la historia del videojuego para realizar la serie televisiva: *The Last of Us* que fue lanzada a la plataforma en enero de 2023. Cuenta con dos temporadas, la primera de ellas, de nueve episodios, está basada en *The Last of Us I (2013)* y la segunda temporada, de siete capítulos, en la segunda parte del videojuego *The Last of Us II (2020) (Wikipedia, 2025).* La producción de la serie ocasionó un impacto económico en la ciudad de grabación, Alberta (Canadá) de alrededor 282 millones de dólares (Motion Picture Association – Canada, 2023).

El lanzamiento del videojuego causó gran expectativa en los jugadores, muchos también seguidores de otras sagas del mismo desarrollador como la saga *Uncharted*, llegando

a un gran número de jugadores y alcanzando volúmenes de ventas elevados para la industria del videojuego en 2013. Su impacto se disparó con el lanzamiento de la versión remasterizada del juego *The Last of Us Remastered* (2014) y el lanzamiento de la segunda parte *The Last of Us II*, que vendió más de diez millones de unidades para la PS4 según Lloria, 2025. A estas ventas se añadieron más de dos millones de copias más tras la salida de la serie de HBO. En concreto, 20 millones de usuarios de la plataforma HBO visualizaron la segunda temporada de la serie, que supusieron 72 millones de dólares en suscripciones y 17,5 millones en publicidad, por no hablar del impacto en redes sociales: más de 10.000 millones de visualizaciones del *hashtag #TheLastOfUs* en la red social Tiktok *(Lloria, 2025).* Como se ha recogido, podemos decir que estamos hablando de una trama de gran alcance que refuerza la relevancia cultural y económica.

El impacto social que ha tenido el videojuego en su lanzamiento y después la adaptación de la serie de HBO, sigue las líneas comentadas anteriormente. TLoU es conocido por abordar algunas cuestiones sociales como el trauma psicológico, dar visibilidad al colectivo LGTB que se reflejan en emociones como la empatía y la confianza. Por otro lado, la gran parte del lenguaje contiene emociones asociadas con la violencia, la agresividad, y que incluyen amenazas e insultos. Si bien no debemos olvidar que se va a analizar los textos del videojuego, de manera que también tienen una gran parte narrativa que explican el contexto y los acontecimientos que suceden en la historia del juego.

El alcance de esta trama es tan amplio que son varios los estudios investigación que se han realizado sobre el juego. Algunos han destacado que *The Last of Us* no solo impacta por su narrativa o gráficos, sino también por el efecto emocional y moral que provoca en los jugadores. Según Anderson (2022), la estructura narrativa fija del juego, junto con el realismo de sus personajes, genera en los jugadores un tipo de "angustia moral", al verse obligados a participar en actos violentos sin poder tomar decisiones éticas dentro del juego. Esta situación lleva a los jugadores a buscar formas cognitivas de afrontamiento, como la mentalización, es decir, intentar comprender las emociones y motivaciones de los personajes para reconciliar sus propias creencias morales con las acciones del juego, lo que amplía su agencia moral y refuerza valores prosociales como la empatía.

Algunos comentarios de los jugadores en (Redditor u/OP, 2023) indican que TLoU, en especial su segunda parte, presenta tal violencia que puede llegar a insensibilizar al jugador y a justificar la violencia, tal y como se observa en comentarios como: "*By the end I felt numb*

from all the killing. It stopped feeling impactful and just became repetitive violence. También mostraban su inquietud a las relaciones tóxicas y ataques personales como se muestra en las frases: *"The amount of hate Neil Druckmann got was disgusting. People can't separate fiction from reality"* o a la gran carga emocional del juego: *"It really drained me mentally. I wasn't ready for how dark it gets."*.

2.4 Evolución de la IA y Consolidación de los modelos LLMs

La Inteligencia Artificial ha evolucionado significativamente desde su origen en 1956, cuando aparecen los primeros programas capaces de realizar tareas cognitivas básicas. Con el tiempo, el avance de la tecnología permitió el surgimiento del *Machine Learning*, que marcó el hito de superar al ser humano en juegos como el ajedrez (Deep Blue). Posteriormente, el desarrollo del *Deep Learning* permitió nuevas capacidades en reconocimiento de voz, imágenes y PLN. Esta evolución llegó al culmen con la publicación del artículo "*Attention is All You Need*", donde se presenta el modelo de *Transformers,* que pueden rastrear dónde aparece cada palabra o frase en una secuencia y entender el significado contextual de las palabras, y se asientan las bases de Inteligencia Artificial Generativa (GAI, *Generative Artificial Intelligence*) (Rico Sanguino, Perona Jiménez y Sánchez Piñeiro, 2025).

La GAI es un tipo de inteligencia artificial que está orientada a la creación autónoma de datos, imágenes, textos, música y otros tipos de contenidos, aprendiendo de ejemplos previos para generar resultados que imitan la creatividad humana. Dentro de este ámbito, destacan los *Large Language Models* (LLMs), modelos de aprendizaje automático basados en redes neuronales profundas entrenadas con grandes volúmenes de datos textuales. Estos modelos son capaces de generar textos coherentes y contextualmente adecuados, asemejándose al lenguaje humano, debido a su capacidad para aprender las relaciones complejas entre palabras, conceptos y emociones (Rico Sanguino, Perona Jiménez y Sánchez Piñeiro, 2025).

Entre los LLMs más importantes se encuentran BERT (*Biderectional Encoder Representations from Transformers*) y GPT (*Generative Pre-trained Transforemer*). BERT puede ser más conocido por su capacidad para entender el contexto bidireccional de las palabras en una oración, siendo una herramienta brillante para la clasificación de texto y análisis de sentimientos (Devlin et al., 2019). Por otra parte, están los modelos GPT que destacan por su capacidad generativa, creando textos coherentes y con contexto a partir de una

petición (*prompt*) inicial. Estos modelos son flexibles y de gran capacidad para la generación de lenguaje natural (Brown et al., 2020).

Los dos modelos marcaron un hito en el PLN, con los modelos de *Transformer*s, siendo de gran utilidad para este trabajo al facilitar el análisis de los diálogos del videojuego. En concreto, este TFG se va a centrar en el uso de modelos GPT, tal y como se analizará a partir de las siguientes secciones del capítulo.

2.5 Comparativa de modelos de OpenAI: GPT-3.5, GPT-4, GPT-4o y GPT-4.1

OpenAI es una organización dedicada a la investigación y desarrollo de inteligencia artificial avanzada. Entre sus tecnologías más conocidas está GPT un modelo de lenguaje capaz de generar texto, responder preguntas, crear contenido y asistir en tareas diversas mediante el procesamiento de lenguaje natural (OpenAI, 2024).

En este apartado de la memoria, se va a realizar una comparativa de los modelos que, en abril de 2025, fecha en las que se elabora este Trabajo Fin de Grado, son optimizables por la plataforma OpenAI. Los modelos de la empresa OpenAI que son válidos (y liberados) para ser optimizados son (OpenAI, 16 abril. 2025. para *Models documentation*):

- GPT-4.1-mini-2025-04-14
- GPT-4.1 -2025-04-14
- GPT-4o-2024-08-06
- GPT-4o-mini-2024-07-18
- GPT-4-0613
- GPT-3.5-turbo-0125
- GPT-3.5-turbo-1106
- GPT-3.5-turbo-0613

Sin embargo y en aras de obtener los mejores resultados, se emplearán los modelos de las versiones 4, 4o, 4o-mini, 4.1 y 4.1-mini, siendo las últimas versiones de GPT y dejando atrás las versiones 3.5. En concreto, se elabora en la Tabla 1 una comparativa de los modelos con las principales características:

Característica/Modelo	GPT-4	GPT-4o	GPT-4o-mini	GPT-4.1	GPT-4.1-mini
Inteligencia [1-4]	2	3	2	4	3
Velocidad [1-4]	3	3	4	3	4
Entrada	TEXTO	TEXTO, IMAGEN	TEXTO, IMAGEN	TEXTO, IMAGEN	TEXTO, IMAGEN
Salida	TEXTO	TEXTO	TEXTO	TEXTO	TEXTO
Precio de entrenamiento /1M de tokens	-	25$	3$	25$	5$
Precio entrada /1M Token	30$	2.50$	0.15$	2$	0.40$
Ventana de contexto	8.192 tokens	128.000 tokens	128.000 tokens	1.047.576 tokens	1.047.576 tokens
Max Tokens salida	8.192 tokens	16.384 tokens	16.384 tokens	32.768 tokens	32.768 tokens
Limites TPM	10.000	30.000	200.000	30.000	200.000
Límite de conocimiento	1 diciembre 2023	1 octubre 2023	1 octubre 2023	1 junio 2024	1 junio 2024
Coste estimado	-	**13.86$**	**1.66$**	**13.84$**	**2.76$**

Tabla 1. Comparación de los modelos GPT válidos para la optimización. Parte de (OpenAI, 16 abril. 2025. Para Models Documentation y OpenAI, 16 mayo. 2025. Para Organization usage limits)

El coste estimado ha sido calculado en apartados posteriores para la primera versión del corpus, como se explicará. Los factores más importantes a la hora de elegir un modelo son (Rico Sanguino, Perona Jiménez y Sánchez Piñeiro, 2025):

1. Precisión, usando modelos de alta precisión a partir de una referencia que se establezca.
2. Coste, optimizando el coste usando varios modelos de diferentes tamaños, inteligencia, rapidez…
3. Latencia, pudiendo usar modelos en tiempo real o por lotes.
4. Seguridad, modelos de datos públicos o privados.
5. El fácil manejo de su entorno, iteraciones e implementación rápida.

Durante la elección de uno de los modelos se deberá tener en cuenta un equilibrio entre precisión, latencia y coste. Se optimizará en orden la precisión, el coste y la latencia (OpenAI, 19 marzo. 2025. *Para Model selection guide).*

En este sentido, se deberá fijar un criterio de precisión relativo a los datos de entrenamiento que se van a usar y que van a ser clasificados de forma correcta en una primera interacción (70-80% en procesamiento de lenguaje natural puede parecer aceptable). Para ello es necesario

tener un conjunto de datos bien clasificados de forma que se pueda ver si se cumple o no el criterio de precisión.

Se emplearán distintos modelos empezando por el más capaz para alcanzar la referencia de precisión y se irá probando con otros modelos hasta alcanzar la precisión deseada.

Se deberán recopilar todas las respuestas para comparar modelos y usar aquel que mejor se ajuste. Esta práctica es conocida como *prompt baking* y para ello se usará una herramienta que recogerá todos estos datos, los analizará y emitirá resultados visuales muy útiles.

Una vez que se ha logrado la referencia de precisión con un modelo que se adapta a su caso de uso, el siguiente criterio es considerar un modelo más "pequeño", como lo son los mini, normalmente más económicos. Se probará si mantiene la precisión con el menor coste y latencia deseada.

Por otro lado, el coste y la latencia están altamente relacionados; una reducción de los *tokens* y las peticiones que se hacen, llevan a un menor coste y un procesamiento más rápido y con ello menor latencia. Con ello se debe considerar las siguiente dos premisas:

- Reducir el número de solicitudes: con un conjunto de datos preparados, agrupados y las llamadas por lotes de datos y no realizarlo de forma individual.
- Minimizar *tokens*: para ello el *prompt engineering* puede ser de ayuda para hacer peticiones más concisas.

Como se ve en la Figura 1 , al cambiar de GPT-4o a GPT-4o-mini con optimización, se logra un rendimiento equivalente (91,5%) por mucho menos coste. Por lo que se usaría el modelo GPT-4o-mini con 1000 ejemplos.

ID	METHOD	ACCURACY	ACCURACY TARGET	COST	COST TARGET	AVG. LATENCY	LATENCY TARGET
1	gpt-4o zero-shot	84.5%		$1.72		< 1s	
2	gpt-4o few-shot (n=5)	91.5%	✓	$11.92		< 1s	✓
3	gpt-4o-mini fine-tuned w/ 1000 examples	91.5%	✓	$0.21	✓	< 1s	✓

Figura 1. Ejemplo para elegir un modelo adecuado que se adapte al caso de uso. (OpenAI, 19 marzo. 2025. Para Model selection guide)

Cuando se habla de modelos de *"Zero-shot"* (ZSL), estos llevan a cabo la orden únicamente con la información que han sido entrenados, no tiene por qué ser de la misma temática y que

puede hacer que generen problemas futuros si aparecen. Para ello requieren de una base amplia de datos etiquetados donde se imponga como será la salida de datos y la respuesta correcta, y para ello se deben ajustar los parámetros del sistema para su calibración. De forma que "aprenden" únicamente de lo que les enseñamos. Los modelos *"One-shot"* se les proporciona un ejemplo antes de ejecutar la orden, o los "*Few-shot/n-shot*" donde se proporcionan más de un ejemplo /n-ejemplos para proporcionar un contexto y mejorar la salida proporcionada.

Si se realiza una comparación de los 5 modelos compatibles con el *fine-tuning*, teniendo en cuenta que el modelo GPT4o en su versión de noviembre (nov'24) tiene unas características similares al modelo optimizable GPT4o (ago'24). Respecto a factores considerados como relevantes a la hora de escoger un modelo como se ha visto antes, dos a dos, se obtiene la Figura 2. Si se pone el foco en los recuadros verdes, considerados como el "cuadrante más atractivo" en cada una de las gráficas, puede verse como los modelos más atractivos son el GTP4o, GPT4.1, el modelo GPT4o-mini y el GPT4.1-mini. El modelo GPT 4o destaca cuando lo que se busca es inteligencia puesto que para un mismo tamaño de ventana (128K *tokens*), la inteligencia de GPT 4o es un punto mayor a 4o-mini. Este modelo y también se encuentra dentro del cuadrante de interés cuando se enfrentan la inteligencia con la velocidad de salida, aunque no será de mucha relevancia para este trabajo. Si lo que se busca es una ventana más amplia (1M *tokens*) las versiones 4.1 tienen similares características, pero con mayor nivel de inteligencia.

El modelo GPT 4o-mini destaca en la comparación latencia-velocidad de salida por su baja latencia (0,3 segundos), pero su versión 4.1-mini teniendo una latencia algo mayor (0,4 segundos), tiene una mayor velocidad de salida que le hace mejor modelo. El modelo 4o-mini

destaca cuando se compara la latencia con el precio, pues GPT 4o-mini es aquel de menor coste por cada millón de *tokens* como recogíamos en la Tabla 1 y el de menor latencia.

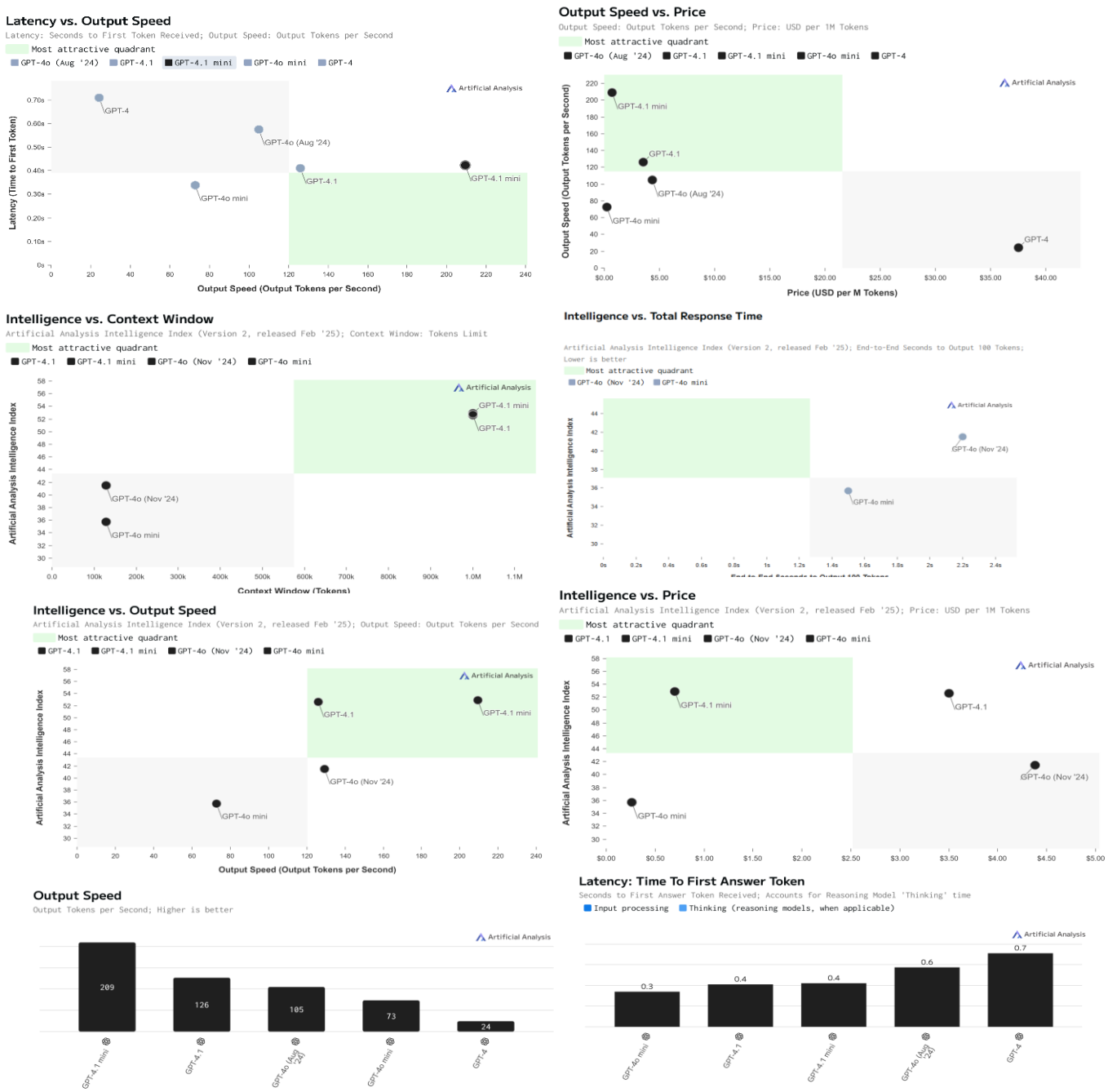

Figura 2. Comparación de modelos GPT4. (Artificial Analysis, 16 abril. 2025.)

Si se fija en la inteligencia versus tiempo de respuesta total, con un nivel de inteligencia seis puntos por debajo, GPT4o-mini presenta un tiempo de respuesta global menor que GPT 4o. En inteligencia versus precio (la comparación más interesante para este trabajo), a pesar de

una mayor inteligencia de GPT 4o, el menor precio de GPT4o-mini hace que se acerque al cuadrante verde y sea más atractivo. En las versiones 4.1, el modelo mini se encuentra en el recuadro más atractivo por su coste similar a la versión 4o-mini y su mayor inteligencia.

Todo parece apuntar a un uso del modelo GPT-4o mini (o de "omni"). Este modelo es un modelo pequeño, rápido y asequible para tareas específicas. Acepta entradas de texto e imagen y genera salidas de texto. Es ideal para la optimización. Las salidas de un modelo más grande como GPT-4o se pueden destinar a GPT-4o-mini para obtener resultados similares con menor coste y latencia. (OpenAI, 19 marzo. 2025. *Para Model selection guide*). Como se indica en la propia web de OpenAI: "Esperamos que GPT-4o-mini sea el modelo adecuado para la mayoría de los usuarios en términos de rendimiento, coste y facilidad de uso."

Con la salida de los modelos versión 4.1 el 14 de abril de 2025, se podría decir que la versión 4.1-mini mejora considerablemente las prestaciones de su versión 4o-mini a un precio muy similar. Lo que lo hace muy atractivo para el desarrollo de este Trabajo de Fin de Grado, ya que permite aprovechar un modelo más eficiente, actualizado y potente sin comprometer el coste computacional.

Es por ello por lo que para la elaboración de este Trabajo Fin de Grado se ha considerado realizar una comparativa entre el modelo GPT-4o-mini y GPT-4.1-mini por su similar coste y amplias prestaciones.

2.6 Estudios sobre análisis lingüísticos y emocionales con LLMS en entornos de videojuegos

El impacto social, que ya se ha comentado, de los videojuegos es enorme, no solo por el peso en la economía o el entretenimiento, sino también por el papel que juegan en la cultura digital. Plataformas como Twitch son un buen ejemplo, ya que por ejemplo en junio de 2024 había más de siete millones de *streamers*, y el español es el segundo idioma con más canales y espectadores después del inglés (Merayo et al., s.f.). Estos mensajes que los mismos jugadores generan, miles cada día, en las plataformas también forman parte del impacto social que tienen los videojuegos en los más jóvenes.

Aquí es donde juega el papel el PLN, un campo que combina la informática y la lingüística para analizar los miles de mensajes generados por los jugadores de manera automática. Gracias al PLN y a los LLMs, varios estudios han investigado el contenido que

generan los seguidores de plataformas de *streaming.* Otros estudios se han dedicado a la detección de diferentes comportamientos como el "*trolling",* analizando el uso de emoticonos, otros a estudiar las emociones en reseñas y comentarios como el estudio de Carralero Lanchares (2025), cuya lectura es muy recomendada.

Pero, aunque existen estudios previos que aplican técnicas de PNL en los videojuegos, la mayoría se centra en reseñas, resúmenes o conversaciones de jugadores en chats de plataformas de *streaming.* No hay muchos estudios que analizan de forma específica el análisis de emociones en los diálogos propios del juego. Solo algunos como *Fallout New Vegas (*Hämäläinen, Alnajjar, & Poibeau, 2022) ha usado diálogos etiquetados de emociones para crear el corpus y estrenar modelos, pero se tratan de casos muy específicos y limitados. En concreto, la investigación se centró en extraer un conjunto de datos multilingüe anotado con sentimientos de dicho videojuego, cuyas líneas de diálogo están preclasificadas en ocho emociones (como enojo, miedo y felicidad). Se realizaron experimentos de análisis de sentimientos multilingüe y *multilabel* usando modelos BERT específicos por idioma. Los resultados muestran que la mayor precisión alcanzada fue 54%, evidenciando que el conjunto de datos es un reto para el análisis de sentimientos. Por todo esto, se puede concluir que existe un área de investigación potencial en el análisis de emociones en diálogos de videojuegos (Merayo et al., s.f.).

3
Herramientas y Recursos utilizados

3.1 Introducción

En este capítulo se detallan todas las herramientas que han sido empleadas durante la elaboración de este Trabajo Fin Grado para lograr exponer los resultados. Además de algunas métricas y fórmulas necesarias para la interpretación y cálculo de los resultados.

3.2 Python

Python es un lenguaje de programación de código abierto y de alto nivel, diseñado para ser fácil de leer y escribir, lo que lo hace ideal tanto para programadores *juniors* como *seniors*. Es el lenguaje de programación más usado en 2025 según el informe *PYPL PopularitY of Programming Language Index* y el *TIOBE Index*.

Su popularidad se debe a su extensa biblioteca de módulos, lo que permite desarrollar aplicaciones muy diversas, entre la que destacan la ciencia de datos, la IA y, en particular, el procesamiento del lenguaje natural (Python Software Foundation, s. f.).

En este Trabajo Fin Grado, Python es utilizado como herramienta principal a lo largo de todo el proceso. Entre otros usos:

- Preparación del corpus para su uso. Usando una estructura prefijada con roles, división aleatoria de los datos en tres ficheros, conversión de Excel a JSONL y viceversa.
- Llamadas a la API de OpenAI. Subida de ficheros, creación de trabajos, obtener los modelos disponibles, ajuste del modelo, uso de modelos optimizados.
- Tokenizado del corpus para el cálculo de costes.
- Empleo de otras herramientas para análisis de resultados. Integración de la herramienta Weights & Biases.

Su uso ha sido clave para automatizar procesos, ya que ha facilitado repetir de forma sistemática los mismos pasos en múltiples oraciones del corpus, permitiendo la lectura de éstas

de un fichero Excel y agilizando así el proceso. Además, ha permitido analizar y obtener resultados de forma clara y sencilla.

3.3 Librerías

En este proyecto se ha hecho uso de diversas bibliotecas de Python que facilitan el procesamiento del lenguaje natural, la interacción con APIs, el análisis de datos y la visualización de resultados. A continuación, se describen las principales librerías utilizadas y su propósito (Python Software Foundation, 2024):

- Json: Permite el manejo de datos en formato JSONL, formato necesario para la optimización del modelo.
- Random: Genera valores pseudoaleatorios de diferentes distribuciones. Se empleará para la selección de forma aleatoria de las oraciones cuando se hace una división del corpus en tres ficheros.
- Pandas: Es la biblioteca empleada para leer y escribir datos entre estructuras de datos, a través del objeto *DataFrame* (Pandas development team, s.f.). Se empleará para la lectura de ficheros Excel, preparar el corpus o estructurar las respuestas en un fichero por columnas.
- Collections.defaultdict: Es una biblioteca que permite gestionar datos de contenedores y en concreto diccionarios con la subclase de *dic: defaultdict*.
- Tiktoken: Permite tokenizar el texto como lo hacen los modelos de la empresa OpenAI. Se empleará para la estimación de los costes del proyecto.
- Numpy: Biblioteca esencial para realizar operaciones numéricas básicas, científicas y estadísticas. Uso para datos unidimensionales como n-dimensionales.
- Openai: Permite el uso de los paquetes de la empresa OpenAI para interactuar con su API como alternativa al uso de la plataforma disponible. Se usará para subir los ficheros de training, test y validación a la plataforma de OpenAI, ajustar el modelo o usar el modelo optimizado.
- Wandb: Una librería que permite enlazar en Pyhton la herramienta de Weights & Biases. Permite monitorizar y almacenar métricas de la creación de modelos de OpenAI.

- Scikit-learn: Es la biblioteca empleada para obtener la evaluación del modelo a través de las métricas de rendimiento. Se empleará para obtener el reporte de clasificación o la matriz de confusión (Pedregosa et al., 2011).
- Ratelimit: Se empleará para limitar la velocidad a la que se realizan peticiones a la API de OpenAI. Evitando así sobrepasar los límites, que existan interrupciones o incurrir en un coste.
- Matplotlib.pyplot: Permite analizar de forma más visual los resultados con la creación de tablas y gráficos. Se empleará para generar el reporte de clasificación o la matriz de confusión.

Para tener disponibles estas librerías será necesario instalarlas, como se hace en la Figura 3, empleado la siguiente línea de código en Python como ejemplo para instalar la librería openai: *pip install openai*. También es posible actualizar algunas librerías para ello se debe emplear el código: *pip install—upgrade tiktoken*, como ejemplo para actualizar la librería tiktoken.

```
#Importing libraries
import tiktoken # for token counting
import numpy as np
import json
import random
from collections import defaultdict
import openai
from openai.types.fine_tuning import SupervisedMethod, SupervisedHyperparameters
from openai import OpenAI
import wandb
import pandas as pd
import sklearn.metrics
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
from ratelimit import limits, sleep_and_retry
import matplotlib.pyplot as plt
```

Figura 3. Código para importar todas las librerías necesarias en este proyecto.

3.4 Jupyter Notebook

Jupyter Notebook es un entorno de desarrollo de código abierto, que permite crear documentos con código, en más de 40 lenguajes, texto, ecuaciones y figuras visuales variadas. Además, permite compartir estos cuadernos vía correo electrónico, Dropbox, GitHub y Jupyter Notebook Viewer (Project Jupyter, s.f.).

Jupyter Notebook ha sido utilizado como entorno principal de desarrollo durante la realización de este Trabajo Fin de Grado por su fácil manejo del código en módulos y su salida interactiva de los resultados. Lo que ha permitido obtener no solo resultados de texto, si no

tablas, como el reporte de clasificación o imágenes como la matriz de confusión. Al mismo tiempo, su interfaz flexible facilita su uso para el procesamiento de lenguaje natural a través de la IA.

Para poder acceder a Jupyter Notebook será necesario tener instalado el paquete, bien desde su web o con la llamada: *pip install* `notebook`. Para usar el cuaderno se deberá realizar la llamada desde la terminal del ordenador: *python -m notebook*.

3.5 Weights & Biases

Weights & Biases es una herramienta para desarrolladores de Inteligencia Artificial (Weights and Biases, s. f. para *About Weights and Biases*) que ofrecerá estadísticas, gráficas y una mejor compresión de los datos para optimizar de la mejor manera el modelo.

Para poder hacer pleno uso de la herramienta será necesario hacer la integración a través de Python. De forma que a la hora de realizar el ajuste del modelo pueda obtener todas las métricas. Entre las funcionalidades más destacadas para este proyecto se encuentran:

- Registrar los parámetros del entrenamiento.
- Visualizar métricas de rendimiento en tiempo real.
- Comparar distintos modelos y ejecuciones.
- Almacenar los logs y gráficas del proceso de entrenamiento.
- Descarga de los resultados de forma sencilla en varios formatos.

OpenAI respalda el uso de esta herramienta de análisis en sus modelos optimizados, al permitir su integración directa en las llamadas a la API, lo que facilita la evaluación y seguimiento del rendimiento de los modelos.

3.6 OpenAI: Obtención API Key y Configuración

Para la ejecución, gestión y evaluación de los modelos de lenguaje utilizados en este proyecto se ha empleado la plataforma de la empresa OpenAI: https://platform.openai.com/. Esta plataforma ofrece una interfaz gráfica completa, junto con acceso mediante API, para gestionar todas las fases del proyecto.

Esta plataforma a mayores incluye información detallada relativa a los modelos (características, costes, límites, …) en la sección *Docs*, al uso de la API (llamadas, objetos de

respuesta, parámetros, …) en la sección *API reference.* Permite el acceso a los modelos optimizados, a los ficheros subidos, a información de uso y coste, en la sección de *Dashboard.* Igualmente permite usar estos modelos ajustados mediante *prompts* en la sección *Playground.*

A la hora de realizar el *fine-tuning* con uno de los modelos de GPT que se haya elegido, es necesario contar con una *API-Key*, una clave de acceso para poder autenticarse y usar el *back-end* de OpenAI. A continuación, se detallan los pasos para la obtención de la *API-key:*

1. Acceder a la plataforma a Open AI: http://platform.openai.com/apps, donde se pedirá elegir entre "ChatGPT" o "*API*" y se optará esta última opción.
2. Clicar en el botón superior derecho: *"Log in"* para iniciar sesión si ya se dispone de una cuenta o por el contrario en *"Sign up"* para poder hacer el registro.
    a. En caso del registro se seguirá los pasos que se indiquen desde la plataforma para poder tener una cuenta en la plataforma de OpenAI.
3. Ya se ha accedido a la plataforma para desarrolladores de OpenAI. Ahora se pulsará sobre el botón *"Start building"* que se encuentra en la esquina superior derecha y que se muestra en la Figura 4.

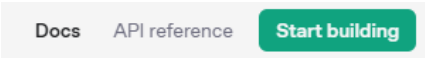

Figura 4. Botón para poder empezar a usar la plataforma de OpenAI como desarrollador de IA.

4. Rellenar los datos que se solicitan sobre el nombre de la organización o la consideración de la organización (con conocimientos técnicos o sin ellos).
5. Observar en la vista general de la plataforma se ve algo similar a la Figura 5.

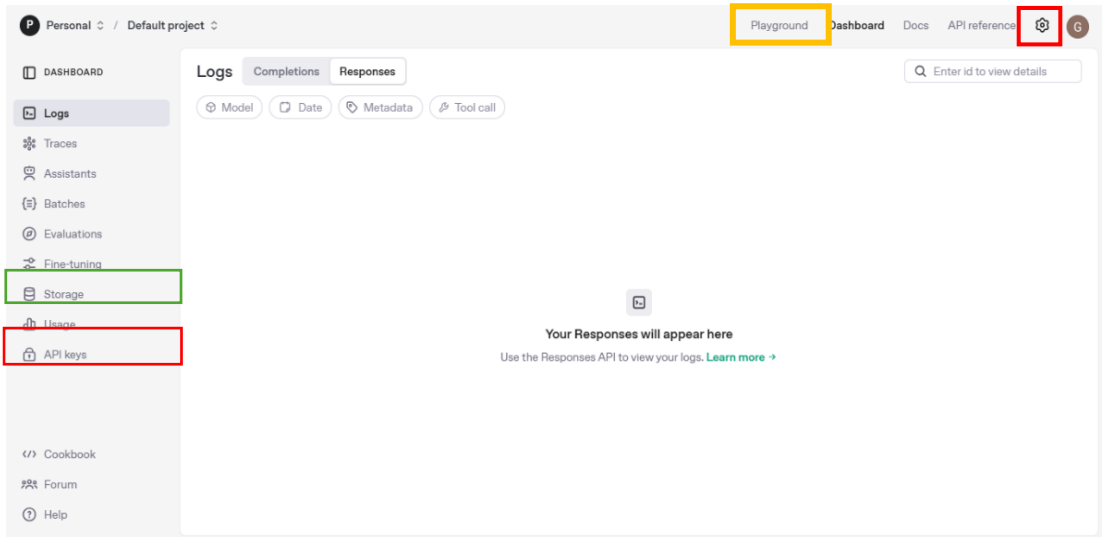

Figura 5. Vista general de la dashboard de OpenAI

6. Acceder a la sección “API keys”, clicando en el botón de crear una nueva clave secreta como se ve en la Figura 6, “*Create new secret key*”, y completar los datos sobre nombre proyecto y los permisos que se van a dar. Al crearla se podrá copiar en al portapapeles.

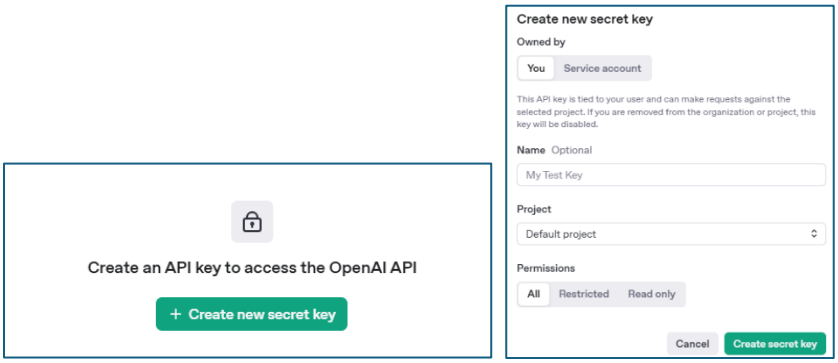

Figura 6. Crear una nueva API key para el uso de modelos GPT.

7. Obtener un resultado como el que se muestra en la Figura 7.

View usage per API key on the Usage page.

NAME	SECRET KEY	CREATED	LAST USED ⓘ	PROJECT ACCESS	CREATED BY	PERMISSIONS
TFG-VICTOR	sk-...UFEA	3 abr 2025	Never	Default project	GCO Uva	All

Figura 7. Obtención de la clave secreta.

Para poder obtener los datos que puede arrojar el modelo en cada una de las ejecuciones se empleará Weights and Biases. Para que este instrumento pueda leer los datos del modelo se deberá seguir lo siguientes pasos:

1. Acceder a la plataforma de WandB: https://wandb.ai/site
2. En la esquina superior derecha, clicar en “Log in” para iniciar sesión si ya se dispone de una cuenta o por el contrario en “Sign up” para poder registrarnos.
   a) En caso del registro hay que seguir los pasos que se indiquen desde la plataforma para poder tener una cuenta en la plataforma de OpenAI.
3. Una vez accedido a la cuenta personal, aparecerá una interfaz como la Figura 8.

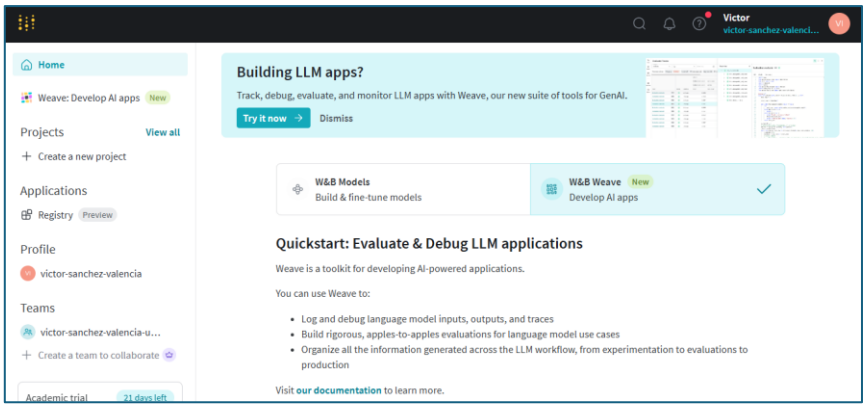

Figura 8. Interfaz de W&B.

Y se pulsará sobre la opción *"W&B Weave"*

4. Deslizando hacia abajo en la interfaz, se parará en la sección *"Set up the weave library"* como se ve en la Figura 9 y se podrá copiar nuestra API-Key.

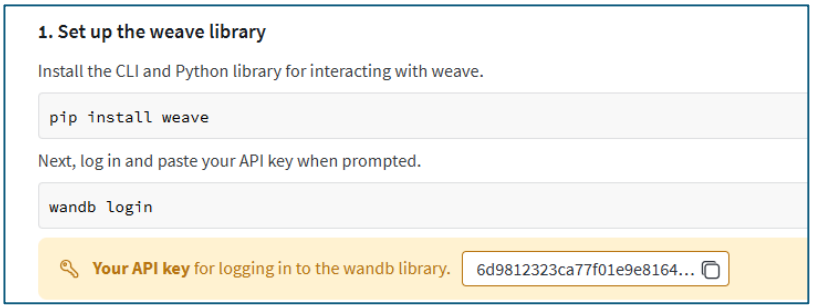

Figura 9. Sección "Set up the weave library" con la API Key de W&B.

5. Acceder de nuevo en la plataforma de OpenAI como se recoge en el punto anterior y pulsar en el botón de *"Settings"* identificado con un icono de una rueda dentada y colocada en la esquina superior derecha, recuadrada de rojo en la Figura 5.
6. En el menú del lateral izquierdo de la Figura 10 entrar en la sección "*Organization/General"*. Dónde se muestra la opción *"Integrations/ Weights and Biases"* y se coloca nuestra *API Key* de W&B.

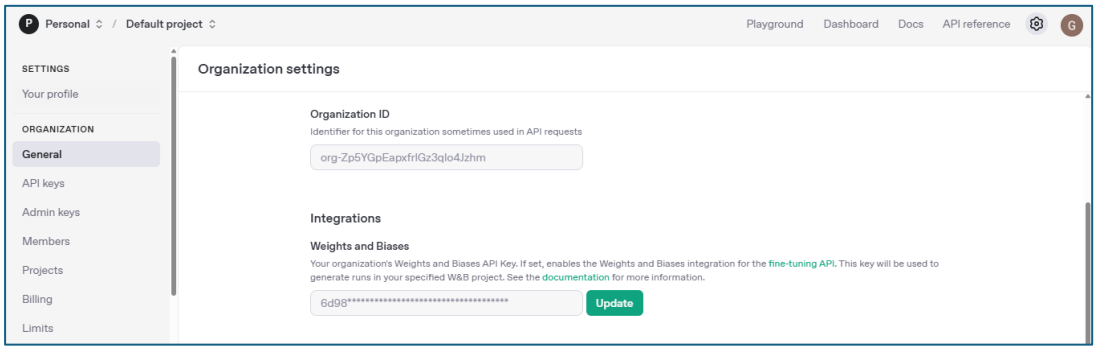

Figura 10. Dashboard de OpenAI con API key de W&B.

3.7 Definiciones y Métricas

3.7.1 Matriz de confusión

La matriz de confusión compara las predicciones del modelo con las etiquetas reales del conjunto de prueba, dividiendo los resultados en verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos para cada clase, tal y como se muestra en la Tabla 2. A continuación, se definen dichos conceptos de forma detallada:

- VP (Verdadero Positivo): Casos positivos correctamente clasificados.
- VN (Verdadero Negativo): Casos negativos correctamente clasificados.

- FP (Falso Positivo): Casos negativos que el modelo clasificó como positivos.
- FN (Falso Negativo): Casos positivos que el modelo clasificó como negativos.

	Predicción: Positivo	**Predicción: Negativo**
Real: Positivo	Verdadero Positivo (VP)	Falso Negativo (FN)
Real: Negativo	Falso Positivo (FP)	Verdadero Negativo (VN)

Tabla 2. Definiciones de la Matriz de Confusión. (Bagnato, s.f.)

3.7.2 Reporte de clasificación

Es un informe detallado sobre el rendimiento del modelo optimizado. Entre las métricas que presenta para la valoración están (Scikit-learn developers, 2025), (Bagnato, s.f.):

- ***Precision*:** la proporción de predicciones positivas correctas frente falsos positivos. Evalúa la capacidad del modelo para no atribuir incorrectamente una clase que no corresponde (Ecuación 1).

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

Ecuación 1. Función de precisión

- ***Recall* (exhaustividad)**: proporción de veces que el modelo identifica correctamente una clase cuando en realidad está presente, frente a las veces que no logra detectarla. Es la capacidad del modelo para no pasar por alto una emoción cuando sí debería reconocerla (Ecuación 2).

$$recall = \frac{VP}{VP + FN}$$

Ecuación 2. Función de recall

- ***F1-score***: media armónica entre precisión y recall (Ecuación 3).

$$F1 - score = 2 * \frac{Precision * recall}{precision + recall}$$

Ecuación 3. Función de F1-score

- ***Support***: Número real de ejemplos de esa clase en los datos verdaderos.
- ***Macro avg***: promedio simple entre clases.
- ***Weighted avg*:** promedio ponderado según el número de ejemplos por clase.
- ***Accuracy***: proporción de predicciones correctas totales (Ecuación 4).

$$accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

Ecuación 4.Ecuación de accuracy

3.7.3 Gráficas del proceso de Fine-tuning

La herramienta Weights & Biases presenta durante el proceso de *fine-tuning* seis gráficas que muestran el rendimiento que está llevando el modelo durante su aprendizaje. Para que estas gráficas sean visibles es necesario realizar la integración de WandB al trabajo de ajuste del modelo. Se mostrará como realizar la integración en secciones posteriores de este trabajo.

Entre las seis graficas que ofrece, se describe aquí su representación, su objetivo y cual sería un resultado óptimo para cada una de ellas, en concreto:

1. **train-loss**. mide la diferencia entre las predicciones del modelo y las etiquetas verdaderas, sobre el conjunto de datos de entrenamiento. Mide cuánto se equivoca el modelo durante el entrenamiento. Su disminución indica aprendizaje. Debe disminuir de forma continua durante el entrenamiento.
2. **valid_loss:** mide la diferencia entre las predicciones del modelo y las etiquetas verdaderas sobre el conjunto de validación. Su disminución indica aprendizaje. Debe disminuir de forma continua durante el entrenamiento.
3. **full-valid-loss:** mide la evolución del *valid_loss* al final de cada época sobre el conjunto de validación.
4. **train-mean-token-accuracy:** mide cuántos tokens individuales predice correctamente el modelo con el fichero de *training*. Sirve para evaluar el rendimiento token a token. Lo óptimo sería valores cercanos a 1. Cuanto más cortas sean las barras más acertado será el modelo.
5. **valid-mean-token-accuracy:** Similar a la anterior, mide la predicción correcta del modelo token a token, en esta ocasión para los datos del fichero de *validation.*
6. **full_mean_token_accuracy:** mide la precisión media por token, evaluada sobre todo el conjunto de validación y al final de cada época. Mide qué tan bien el modelo generaliza a datos nunca vistos. El valor óptimo sería de 1/100% que implicaría que acierta todos los nuevos datos que predice.

4

Corpus Lingüístico: *The Last of Us*

4.1 Introducción

En este capítulo se va a detallar la metodología seguida para la creación y etiquetado del corpus, que recoge las oraciones extraídas del videojuego *The Last of Us*. Se describe el proceso de recopilación, preparación y organización del corpus de estudio, que será utilizado posteriormente para el entrenamiento los modelos supervisados de IA.

Este corpus se ha elaborado desde cero, ya que no existía uno previo del que partir. De manera que lo que se buscará en este corpus inicial será un grado de precisión que resulte adecuado para interpretar resultados correctamente y un balance de las clases para obtener una representación completa.

4.2 Elaboración del Corpus basado en *The Last of Us*

4.2.1. Recopilación del material de análisis y definición de clases

El material con el que se va a trabajar corresponde a los diálogos y a todo texto incluido en la versión original del videojuego *The Last of Us,* lanzado en 2013. En una segunda iteración realizada para la mejora del corpus (explicado en secciones posteriores), debido a resultados no muy favorables se amplió el corpus con parte de los diálogos de *The Last of Us II*. A mayores, se aplicaron técnicas de creación de oraciones sintéticas (muestras sintéticas) con IA como mejora del balanceo del corpus. Para la obtención de este material, se ha recurrido a su descarga desde (Shotgunnova, 2013) para la primera parte del videojuego y de (Game Scripts Wiki, 2020) para su segunda parte, donde se obtuvo no solo los diálogos de los personajes, sino también información contextual, como la escena donde ocurre, el personaje que realiza el comentario y su género. Todo el corpus está en el idioma inglés, por lo que todo el proceso se realizó en el mismo idioma.

Figura 11. Captura de la web que ha permitido la descarga del contenido de análisis. (Shotgunnova, 2013)

La descarga del contenido se realizó en un formato texto (.txt) como puede verse en la Figura 11, por lo que fue necesario ordenar las oraciones y llevar a cabo un proceso de adecuación para su uso posterior. Para proceder a la estructuración del corpus inicial, todo el material descargado fue dividido en oraciones individuales, cada una acompañada de la escena donde se realiza, el personaje o ente que lo expresa y su género. Con esta información se creó una base de datos en un fichero Excel de extensión .xlsx, que contenía como columnas principales las que se muestran en la Tabla 3.

Chapter	*Character*	*Gender*	*Message*	*Polarity*	*Emotion*

Tabla 3. Títulos de columna en el fichero Excel del corpus inicial.

A este Excel se añadieron las columnas de *Polarity* y *Emotion* para en la fase de etiquetado se pudieran indicar para cada una de las oraciones.

La columna *Chapter,* hace referencia a la escena donde se produce la acción. Es una variable independiente que representa una entrada de texto. El videojuego cuenta con las siguientes escenas:

The Last of Us I

1. *Prologue – Hometown*
2. *Summer - The Quarantine Zone*
3. *Summer - The Outskirts*
7. *Fall - Tommy's Dam*
8. *Fall - The University*
9. *Winter - Lakeside Resort*

4. *Summer - Bill's Town*
5. *Summer – Pittsburgh*
6. *Summer - The Suburbs*
10. *Spring - Bus Depot*
11. *The Firefly Lab*
12. *Epilogue – Jackson*

The Last of Us II

1. *JACKSON*
2. *4 YEARS LATER*
3. *SEATTLE DAY 1*
4. *3 YEARS EARLIER*
5. *SEATTLE DAY 2*
6. *2 YEARS EARLIER*
7. *SEATTLE DAY 3*
8. *4 YEARS EARLIER*
9. *4 MONTHS EARLIER*
10. *THE FARM*
11. *SANTA BARBARA*

La columna *Character,* es una entrada de texto que hace referencia al personaje del juego que ha realizado el comentario. Entre ellos, podemos encontrar los siguientes personajes de la primera parte:

1. Bill
2. Clip
3. Cop
4. David
5. Doctor
6. Dying Woman
7. Earl
8. Ellie
9. Firefly
10. Girl
11. Goon
12. Guard
13. Henry
14. Hunter
15. Infected
16. James
17. Joel
18. Kid
19. Loudspeaker
20. Malick
21. Man
22. Maria
23. Marlene
24. Motorist
25. Nurse
26. Passerby
27. Patrol
28. Radio
29. Reporter
30. Robert
31. Sam
32. Sarah
33. Sentry
34. Soldier
35. System
36. Terence
37. Tess
38. Thing
39. Tommy
40. Voice
41. Woman

A continuación, se muestran algunos de los personajes de la segunda parte, *The Last of Us II:*

1. Abby
2. Bill
3. Dina
4. Isaac
5. Jerry
6. Jesse
7. Jordan
8. Lev
9. Manny
10. Mel
11. Owen
12. Rattler
13. WLF Soldier
14. Yara

Algunos de los personajes sí se les puede atribuir un género, sin embargo, otros como *Voice*, *Firefly, WLF Soldier* o *Motorist*, que hacen referencia a personajes genéricos que pueden no conocerse y entonces no se les puede atribuir.

La columna *Gender* hace referencia al género del interlocutor, y será "M": *Male,* "F": *Female*, o *Unknown*: desconocido.

La columna *Message,* es la oración que se pretende analizar. Cuenta con el mensaje en su formato original en habla inglesa, con expresiones, ortografía y gramática propia del idioma. Algunas de las oraciones se modificaron y depuraron al contener caracteres erróneos en su descarga, que se explicaran posteriormente.

La columna añadida de *Polarity,* indicará, en la fase de etiquetado, la polaridad general de la oración, siendo las posibles opciones:

1. *Positive (*Positiva*):* denotará aquellos diálogos que muestran emociones, actitudes o estados que generan bienestar o sentimiento favorable.
2. *Negative (*Negativa*)*: denotará aquellas expresiones que muestran emociones o estados emocionales de malestar, enfado, incomodidad o sufrimiento.
3. *Neutral (*Neutral): son aquellas expresiones o comentarios que generan cierta indiferencia. No generan bienestar o malestar de forma clara.

Por último, la columna *Emotion,* recogerá la emoción que expresa cada oración. De entre las que se recogieron en el corpus inicial, encontramos las primarias que se enuncian en la rueda de las emociones de Plutchink (Karimova, 2025). En concreto en nuestro corpus se seleccionaron las siguientes:

1. *Happy (*felicidad): Esta emoción es una de las emociones primarias que encuentran en la rueda de Plutchik, Figura 12. Se empleará para aquellas oraciones que muestren bienestar, satisfacción, logros o noticias alentadoras. Por ejemplo, en la oración del corpus: "*Alright...that's Good" o "I think it's adorable.".*
2. Sad (tristeza): se empleará para los diálogos que muestren insatisfacción, pérdidas, nostalgia, decepción, o a través de palabras concretas como: triste, llorar, soledad. También es una de las emociones primarias de Plutchik y es la emoción opuesta a felicidad. Por ejemplo, en la oración *"Babe... I'm really tired." o "Horrible".*
3. *Pained* (dolor): el dolor a diferencia de la tristeza se empleará cuando se expresen emociones de dolor emocional o angustia. Ambos sentimientos son muy similares, tienen una connotación negativa. Tan similares son que veremos como avanzado el proceso fue necesario juntarlas en la misma categoría.

4. *Anger* (enfado): Emoción primaria que se empleará ante emociones de injusticia, agresión, frustración o conflicto. Suele ser empleado con tono fuerte, y expresado con interrogaciones o insultos. Alguna de las oraciones podría ser: *"Agh, fuck that" o "Bullshit! I'm not going with him".*
5. *Fear* (miedo): mostrará oraciones en las que se exprese peligro por parte de los personajes, incertidumbre, se perciba incertidumbre o los personajes consideren una amenaza. También se indica como emoción primaria en la Figura 12. Oraciones calificadas como *fear* podrían ser: *"Get off! Ellie, help! Shit, watch out!" o "Joel, behind ya!".*
6. *Surprised* (sorpresa): Otra de las emociones primarias que se empleará para etiquetar expresiones que muestren asombro, algo inesperado o fuera de lo común. Suelen ir acompañadas de interrogaciones y a veces de exclamaciones.
7. *Neutral* (neutral): esta emoción se empleará cuando la oración no muestre una carga emocional. Puede ser informativo o ambiguo. Ejemplos de esta clase tan compleja serán: *"Do what?" o "I can't believe".*
8. *Empathy/Trust/Confidence* (empatía/confianza): Esta emoción más compleja fue incorporada avanzado el proceso de etiquetado. Será empleada para expresar compresión, apoyo emocional, precaución por la otra persona, cercanía, admiración o reconocimiento. Esta emoción podríamos verla en *"Alright. We did it" o "Come on, baby girl. I gotcha...".*
9. *Disgust* (asco): esta emoción primaria según Plutchik, se etiquetará cuando los diálogos de los personajes muestren una repulsión hacia algo. Avanzado la fase esta emoción será eliminada como veremos.

Figura 12. La rueda de las emociones de Plutchik (Karimova, 2025).

Una vez se ha completado el Excel con el formato indicado, se consigue el corpus inicial de los diálogos de la primera parte del juego. Este corpus inicial cuenta con 3249 instancias, y con ellas se puede comenzar el proceso de análisis.

4.2.2 Depurado inicial del corpus

Para lograr obtener el corpus final para entrenar los modelos, debemos en primer lugar realizar un depurado manual inicial de los mensajes. Este depurado consiste en realizar una lectura rápida de todas las instancias del corpus inicial. El objetivo es detectar simbología rara que se haya podido mantener a la hora de descargar el contenido. Símbolos como *"%%%"*, *"_____"* o *"$$$$"* fueron los detectados en alguna de las oraciones del corpus, como en *"Sarah... Baby... Don't do this to me, baby. Don't do this to me, baby girl. Come on... No, no... Oh no, no, no... Please. Oh, God. Please, please, don't do this. Please, God... _____ ¬Ø¬Ø¬Ø¬Ø¬Ø¬Ø¬Ø¬Ø¬Ø"*

Otros símbolos detectados fueron guiones al comienzo y final de las oraciones como en *"—she can shove it right up—"* que no aportaban información al sentimiento o polaridad, pudiendo indicar el comienzo o final de la intervención de un personaje y por tanto fueron eliminados en esta fase.

Otro de los aspectos detectados en la primera lectura fue la falta de la alguno de los campos en el corpus inicial. Generalmente eran los datos relativos al género de los personajes. Ante esta situación tuvimos dos casos, aquellos personajes que sus nombres ya nos indicaban el género como la etiqueta *Woman,* y otros casos en los que se tuvo que recurrir a (Fandom, s.f.

para *Categoría: Personajes*) para poder indicar el género de forma correcta, este fue el caso del personaje Ellie o Terence. Además, en esta etapa se procedió a eliminar los puntos finales, pues como puede verse en la Figura 13 añadían un token más y carecían de información relevante.

```
texto="Here."
encoding = tiktoken. get_encoding (o200k_base)
encoding.encode(texto)
>> [12253, 13]
[encoding.decode_single_token_bytes(token) for token in [12253, 13]]
>>[b'Here', b'.']

texto="Here"
encoding = tiktoken. get_encoding (o200k_base)
encoding.encode(texto)
>> [12253]
[encoding.decode_single_token_bytes(token) for token in [12253]]
>>[b'Here']
```

Figura 13. Código y respuesta para hacer ver que el punto final añade un token que carece de información.

4.2.3 Proceso etiquetado del corpus por pares

Con el corpus inicial ordenado y depurado de errores gramaticales que pudieran afectar al rendimiento y precisión del modelo, se comienza a realizar el etiquetado de la polaridad y emociones de cada una de las oraciones.

Esta fase se llevó a cabo por mí, autor de este trabajo fin de grado, en colaboración con el grupo de investigación, formando un equipo de trabajo paralelo. El objetivo es que todas las instancias del corpus inicial sean categorizadas en alguna emoción y polaridad mencionadas en apartados anteriores. Para que el etiquetado no se convirtiese en un proceso tedioso y largo, se dividieron las instancias entre los integrantes del equipo para que cada uno etiquetase únicamente su parte.

El proceso es manual y sencillo. Cada uno de los integrantes debe leer oración a oración la parte del corpus que debía etiquetar. Se debe interpretar esa oración y colocar en las columnas correspondientes de polaridad y emoción, la categoría que le suscita. Para evitar cualquier tipo de sesgo en el que se pudiera incurrir debido al contexto del videojuego, las oraciones se colocaron por orden alfabético, de manera que se rompía el orden cronológico de la historia del videojuego. A mayores, se creó una hoja diferenciada para cada una de las emociones, donde se copiaba la oración etiquetada y con su color resaltado.

Algunos de los problemas con los que se ha encontrado el equipo a la hora de etiquetar el dataset son: frases muy genéricas, o muy cortas y que el modelo puede no interpretar bien, o bien la aparición de múltiples emociones en las oraciones que podrían generar confusión entre ellas. Finalmente, otro problema ha sido la complejidad del concepto de emoción *neutral* y su falta de patrones concretos para su detección.

4.2.4 Modificaciones en el etiquetado del corpus

A partir del paso anterior, se lleva a cabo una revisión y corrección del etiquetado por parte del equipo de investigación, expertos en el procesado del lenguaje natural (una de ellas lingüista) y en el lenguaje de videojuegos. Además, se lleva a cabo una selección de las oraciones para cada una de las emociones, de manera que consigamos el mayor número de ejemplos claros de cada una de las emociones y pueda ser una base de datos balanceada. Esta revisión corresponde con una corrección por pares, en la que dos expertos han etiquetado la misma sección de diálogos y un tercer experto debe decidir sobre cualquier discordancia existente entre las emociones y/o polaridad de las oraciones.

Durante esta etapa también se realizaron cambios en algunas de las categorías de las emociones, en concreto:

- Se suprimió la emoción *Disgust,* ya que presentaba muy pocas instancias lo que complicaría el entrenamiento del modelo.
- Se juntaron las emociones de *Sad* y *Pained*. Existe un límite muy próximo entre ambas emociones, lo que hace que su clasificación sea a veces compleja de diferenciar. Además, se contaba con muy pocas oraciones etiquetadas como *Pained*, de manera que el algoritmo no sería capaz de aprender de manera precisa esta emoción.
- Se añadió la emoción *Empathy/trust/confidence*, que hizo que la categoría *Happy* fuera reetiquetada. Se encontraron muchas oraciones que expresaban este tipo de emoción, que indicaba algo a mayores que *Happy*, y que implicaban confianza o empatía entre personajes. Esta emoción además está muy presente en este videojuego por su contexto y temática.
- Se reetiquetaron algunas emociones que presentaban correlación como *Fear* y *Surprised*, *Happy* y *Empathy*.

- Se revisó la emoción *Neutral*, pues es una categoría amplia y en muchas ocasiones se mezclan emociones en esta.

También en esta etapa se realizó una verificación de oraciones etiquetadas repetidas. Para ello, se colocó el corpus por orden alfabético y con funciones de Excel se realizó la verificación de oraciones repetidas. Finalmente, se obtuvo el corpus final, un fichero (*CorpusDialoguesTLoUv1*) en formato (.xlxs) con las oraciones etiquetadas en polaridad y emoción que cuenta con 1969 instancias.

4.3 Refinamiento y Expansión del Corpus *The Last of Us*

Tras el entrenamiento del modelo con la primera de las versiones del corpus, se obtuvieron unos resultados moderados que se expondrán en apartados siguientes. Estos resultados no mostraban una precisión adecuada para varias de las categorías, especialmente en aquellas con menor representación en los datos originales: *Happy, Sad y Fear* como se verá en la distribución de la Tabla 4.

La baja precisión, que se mostrará en los resultados, indicó una necesidad de ampliar y modificar la primera versión del corpus. Los cambios realizados fueron:

- Ampliar el corpus con nuevos fragmentos de *The Last of Us II*, incorporando de manera independiente a como se realizó en la primera ocasión nuevas oraciones de aquellas emociones con peores resultados en la primera iteración del modelo.

La matriz de confusión también reveló que el modelo confundía algunas de las emociones, como las parejas *Neutral-Empathy, Neutral-Sad, Fear-Empathy o Neutral-Fear,* como se verá en capítulos posteriores. Este comportamiento nos llevó a realizar el siguiente proceso en el etiquetado del corpus:

- Reetiquetado de las emociones *Empathy, Sad, Fear y Neutral.* Con el objetivo de afinar la precisión de las etiquetas y evitar la confusión del modelo.

El proceso de etiquetado por pares se realizado bajo la misma metodología que el descrito con anterioridad. De manera que con los cambios realizados se obtiene una segunda versión del corpus que denominamos *CorpusDialoguesTLoUv2* (Anexo 2) y que cuenta con 2077 instancias, 108 más incorporadas a su primera versión mejorada.

4.4 Preparación del Corpus para su análisis con OpenAI

Una vez que se ha realizado el etiquetado del corpus y se ha obtenido una versión final para llevar a cabo un aprendizaje automático supervisado con nuestro modelo de OpenAI, se debe dar el formato correcto a estos datos. Se debe transformar el corpus en el que se ha trabajado en formato Excel al formato necesario para la optimización del modelo, es decir, formato *JavaScript Object Notation* (JSON).

El formato debe ser una especie de conversación con el modelo, una lista de mensajes como se muestra en el ejemplo más abajo, similar a la conversación que se tendrá una vez optimizado. Con la presencia del rol "system" que realiza la petición "analizar la emoción/polaridad", y el rol "user" que indica el mensaje que debe ser analizado. Además, en la fase de entrenamiento se usará el rol "assistant" para indicar cuál es la etiqueta verdadera de emoción/polaridad (Open AI, s.f. para Fine-tuning). Este proceso se puede ver en el siguiente código:

{"messages":

[{"role": "system", "content": "What is the emotion of the following text? Respond with 'Anger', 'Fear', 'Happy', 'Sad', 'Neutral', 'Surprised' or 'Empathy'."},

{"role": "user", "content": "Wait! Let me go. Please"},

{"role": "assistant", "content": "Sad"}]}

Para realizar la transformación del formato se ha empleado el código de la Figura 14. En primer lugar, se realiza la lectura del fichero Excel y después, línea a línea, se realiza la lectura de las columnas para incluirlas en el formato exigido.

```
#This script reads messages and emotions from an Excel file and writes them as structured
#prompt-response pairs in a JSONL format for emotion classification.
df = pd.read_excel("CorpusDialoguesTLoU.xlsx") #data input file
name=CorpusDialoguesTLoU.xlsx
examples=0 #used to know number of rows
# Iterate over each row of the DataFrame
with open("corpus_EMOCION.jsonl", "w", encoding="utf-8") as f: #data output file
name=corpus_EMOCION.jsonl
for _, row in df.iterrows():
# Add each row as a separate message
data = {
"messages": [
                {
"role": "system", #cuestion ask to the model
"content": "What is the emotion of the following text? Respond with 'Anger', 'Fear',
'Happy', 'Sad', 'Neutral', 'Surprised' or 'Empathy'."
                },
                {
```

```
“role”: “user”,
“content”: row[”Text”] #column title where the message is located
                },
                {
“role”: “assistant”,
“content”: row[”Emotions”] #column title where the emotion is located
                }
            ]
        }
f.write(json.dumps(data)+”\n”)
examples+=1
# to resume
print(“Num examples:”, int(examples))
print(“File saved as corpus_EMOCION.jsonl”)
```

Figura 14. Fragmento de código para convertir el corpus a JSON.

Si bien se puede ver que este fragmento permite convertir el corpus de Excel a JSON, lo hará para las emociones, pues en la pregunta del sistema se pide que identifique la emoción de la oración y que diga cual es de las indicadas: *‘Anger’, ‘Fear’, ‘Happy’, ‘Sad’, ‘Neutral’, ‘Surprised’ or ‘Empathy’*. Además, después del diálogo propiamente dicho del videojuego se añade en este caso la columna del Excel donde se encuentra la emoción: *row[“Emotions”]*. A continuación, el JSON se guarda en un fichero con nombre *corpus_EMOCION.jsonl* en la carpeta de trabajo. Así pues, se obtendría una salida, cuando se convierte el corpus en su totalidad, similar a la Figura 15.

```
Num examples: 1969
File saved as corpus_EMOCION.jsonl
```

Figura 15. Salida tras convertir el corpus de Excel a JSON.

Los cambios necesarios para realizar la misma acción con la polaridad son:

1. Cambio en la formulación de la pregunta del sistema tal y como se muestra en el código:

```
{
“role”: “system”,
“content”: “What is the polarity of the following text? Respond with
‘Positive’, ‘Negative’ or Neutral’.”
            }
```

2. Cambio en el nombre de la columna para incorporar la polaridad y no la emoción, tal y como se muestra en el siguiente código:

```
{
“role”: “assistant”,
“content”: row[“POLARITY”]
            }
```

3. Cambio en el nombre del fichero de salida, tal y como se observa a continuación:

```
with open(“corpus_POLARIDAD.jsonl”, “w”, encoding=”utf-8”) as f:
```

Una vez que se tiene el corpus con el formato adecuado, se carga el fichero JSON y se lleva a cabo una serie de comprobaciones de forma que se eviten algunos errores comunes como

error de tipo, mensajes en blanco, de rol o de contenido entre los que veremos más adelante (Figura 17).

```
#This line sets the file path for the emotion-labeled dataset stored in JSONL format.
data_path = "corpus_EMOCION.jsonl"
# This code reads a JSONL dataset file into a list of JSON objects and prints the total
number of examples.
with open(data_path, 'r', encoding='utf-8') as f:
dataset = [json.loads(line) for line in f if line.strip()]
# Initial dataset stats
print("Num examples:", len(dataset))
```

Figura 16. Carga del fichero JSON. Modificado a partir de (Wu, M. y Fishman, S, s. f)

Se empleará el código de las Figuras 16 y 17 para cargar (Figura 16) y llevar a cabo la comprobación de errores (Figura 17) del corpus de emociones, y si por el contrario se busca cargar el fichero de polaridad se cambiará la primera línea a: *data_path = "corpus_POLARIDAD.jsonl"*.

Entre las comprobaciones que se hacen en Wu, M. y Fishman, S, (s. f) para que los datos se ajusten al proceso de *fine-tuning* hemos usado las siguientes (código de la Figura 17):

1. Tipo de dato: se verifica que cada entrada es de tipo diccionario (*dict*) y si no muestra como error *"data_type"* seguido del número de veces que ocurre.
2. Listado en los mensajes: comprueba que haya al menos una conversación, si no muestra el error: *"missing_messages_list"*.
3. Claves de cada mensaje: se verifica que cada mensaje incluya *role y content,* si alguna no lo hace muestra error: *"message_missing_key" y* el número de veces ocurrido.
4. Claves no reconocidas por el modelo: si alguna clave escrita en las conversaciones es distinta de las prefijadas: "*role, content, weight, function_cally name"* muestra el error *"message_unrecognized_key"*.
5. Rol adecuado: verifica que el rol contenga al menos una de las siguientes claves *"system, user, o assistant"* de lo contrario muestra error *"unrecognized_role"*.
6. Contenido adecuado: que el contenido sea una cadena de texto si no muestra junto con un recuento: *"missing_content"*.
7. Asistente válido: que exista el rol de *assistant* en caso contrario muestra *"example_missing_assistant_message"*.

Como salida se presentará el tipo de error y el número de veces que este ocurre en el fichero cargado o por el contrario *"No errors founds"*.

```
# This script validates the structure of each JSONL example by checking required fields and
roles, and reports formatting errors if found.
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”, “weight”) 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”)
```

Figura 17. Código para la comprobación de errores. (Wu, M. y Fishman, S, s. f)

Una vez dado el formato y limpio de errores, se debe llevar a cabo la división en tres partes de los datos: datos de entrenamiento, datos de validación y datos de test. Este paso es crucial para evitar errores de precisión en el modelo. Se debe tener en cuenta que el etiquetado se ha llevado a cabo por humanos, con conocimientos específicos sobre la temática a tratar, pero no quita que se deban considerar imprecisiones.

Los datos de entrenamiento se emplearán para optimizar el modelo. Aprenderá para que conjunto de diálogos, expresiones, palabras se le ha dado una etiqueta y replicará el comportamiento. Los datos de validación son usados también durante el proceso de optimizado. Son usados por el modelo para comprobar el rendimiento del entrenamiento, al finalizar cada época se realiza una evaluación con los datos de validación. Estos datos son muy útiles, pues proporcionan información al modelo para ajustar los hiperparámetros. Por último, lo datos de test son empleados al tener ya el modelo optimizado y servirán para evaluar la precisión y otras métricas sobre datos no vistos antes por el sistema de inteligencia artificial, lo que dará más robustez a los resultados de nuestros modelos. Se solicitará al modelo optimizado que evalúe los datos de test y teniendo su predicción y las etiquetas verdaderas se

podrá ver las desviaciones del modelo. Cabría esperar que para tener una mayor precisión se tendría que usar la totalidad de los datos etiquetados, pero entonces se caería en el problema por excelencia del aprendizaje automático supervisado: el sobreajuste o en inglés *overfitting*. El sobreajuste implica querer ser "demasiado listo". Al entrenar el modelo con todos los datos disponibles, éste se ajusta perfectamente a ellos de forma que la precisión de acierto sobre datos futuros mengua.

Por ello se debe seleccionar un porcentaje de los datos etiquetados para entrenar el modelo, para la validación y lo restante para la realización de test. No existe un porcentaje óptimo para dividir los datos, pero un porcentaje del 70% para los datos de entrenamiento es adecuado. De forma que se dividirá los datos en 70% entrenamiento - 15% validación – 15% test (V7 Labs, 2024). El código implementado para llevar a cabo la división aleatoria de los datos se ha obtenido a partir de una modificación de Martín García (2024), tal y como se muestra en la Figura 18.

```
#This function splits a JSONL dataset into training (70%), validation (15%), and test (15%)
sets,
#shuffles the data, and writes each split to separate files.
#Imputs: (data file, training_file, validation_file, test_file)
def split_json(input_file, output_file_Training, output_file_validation, output_file_test):
with open(input_file, 'r') as f:
data= [json.loads(line) for line in f]
total_messages =len(data)

# Calculate the split sizes
n_train = int(total_messages * 0.70)
n_val = int(total_messages * 0.15)
n_test = total_messages - n_train - n_val  # ensures total is preserved

#makes a copy of the data, sorts it randomly
shuffled_data=data[:]
random.shuffle(shuffled_data)

# Split the data
data_train = shuffled_data[:n_train]
data_val = shuffled_data[n_train:n_train + n_val]
data_test = shuffled_data[n_train + n_val:]

# Write to output files
with open(output_file_Training, 'w') as f_train:
for item in data_train:
json.dump(item, f_train)
f_train.write('\n') #add a new line after each json object
with open(output_file_validation, 'w') as f_val:
for item in data_val:
json.dump(item, f_val)
f_val.write('\n') #add a new line after each json object
with open(output_file_test, 'w') as f_test:
for item in data_test:
json.dump(item, f_test)
f_test.write('\n') #add a new line after each json object
```

```
#This line calls the ‘split_json‘ function to divide the emotion-labeled dataset into
training, validation, and test files.
split_json(“corpus_EMOCION.jsonl”, “training_Emocion.jsonl”,
“validation_Emocion.jsonl”,”test_Emocion.jsonl”)
#This line calls the ‘split_json‘ function to divide the polarity-labeled dataset into
training, validation, and test files.
split_json(“corpus_POLARIDAD.jsonl”, “training_Polaridad.jsonl”,
“validation_Polaridad.jsonl”,”test_Polaridad.jsonl”)
```

Figura 18. Código para la división de datos en training-validation-test.

Hasta este punto se dispone de ocho ficheros en formato JSONL, en concreto:

- *corpus_EMOCION,jsonl:* corpus en su totalidad para el análisis de las emociones de los diálogos del videojuego.
- *corpus_POLARIDAD.jsonl:* corpus en su totalidad para el análisis de la polaridad de los diálogos del videojuego.
- *training_Emocion.jsonl:* datos de entramiento para el análisis de las emociones en los diálogos del videojuego.
- *training_Polaridad.jsonl*: datos de entramiento para el análisis de la polaridad en los diálogos del videojuego.
- *validation_Emocion.jsonl*: datos de validación para el análisis de las emociones en los diálogos del videojuego.
- *validation_Polaridad.jsonl*: datos de validación para el análisis de la polaridad en los diálogos del videojuego.
- *test_Polaridad.jsonl:* datos de test para el análisis de la polaridad en los diálogos del videojuego.
- *test_Emocion.jsonl:* datos de test para el análisis de la emoción en los diálogos del videojuego.

Estos últimos ficheros, los datos de test, son empleados para hacer pruebas con el modelo ya optimizado y se deben lanzar peticiones al modelo con cada una de las oraciones como se realizará más adelante. De esta forma, y para llevar a cabo de forma más sencilla la ejecución de las ordenes, se pasarán dichos ficheros al formato en Excel para poder obtener los diálogos de las columnas de forma más sencilla usando el código de la Figura 19.

De igual forma se podría hacer este mismo proceso para el fichero *test_Polaridad.jsonl*. para ello se debe modificar el fichero de entrada, la variable *emotion=[]* por *polaridad=[]*, y la columna donde se guardarán las emociones: *df = pd.DataFrame({”MESSAGE”: textos, “ POLARITY “: polaridad })*.

```
#This code reads an emotion test JSONL file, extracts messages and emotions, then exports
the data to an Excel spreadsheet.
with open("test_Emocion.jsonl", "r") as file:
data = [json.loads(line) for line in file]
texts = [] #empty list
emotion = [] #empty list

for dato in data: #read only the messages and emotion in the jsonl file
texts.append(dato["messages"][1]["content"])
emotion.append(dato["messages"][2]["content"])


df = pd.DataFrame({"MESSAGE": texts, "EMOTION": emotion}) #save de data in column form
df.to_excel("test_Emocion.xlsx", index=False) #save into excel
```

Figura 19. Código para pasar el fichero de test de formato JSONL a Excel.

El proceso para la preparación de los datos de la versión dos del corpus, *CorpusDialoguesTLoUv2,* es idéntica a la descrita en este apartado. Con la salvedad del nombre del fichero de donde se recogen los datos.

4.5 Análisis descriptivo de Corpus

Realizada la división del corpus en los tres ficheros, se debe tener en cuenta la distribución de las emociones en cada uno de los ficheros. El objetivo es que cada una de las emociones tenga una cantidad adecuada de instancias para que el modelo sea capaz de aprender de cada una de ellas. En caso de que se considere que no existe una buena distribución de las emociones o polaridad en los diferentes ficheros, se deberá ejecutar de nuevo el código, llamando a la función *split_json* de la Figura 18 correspondiente para obtener una nueva distribución.

Aun así, puede ser el caso que una de las clases este poco representada, hablamos de corpus desbalanceado. Un corpus desbalanceado crea modelos sesgados que tienden a predecir la clase mayoritaria y llevan a evaluaciones engañosas. Sin embargo, suele ser difícil encontrarse ante un corpus con todas sus clases balanceadas, lo normal es que no lo estén (Bagnato, s.f.).

En la primera de las iteraciones, es decir, con la primera versión del dataset, se obtuvo, a través de la ejecución de código del Anexo 1, una distribución en los ficheros de emociones como la que se muestra en la Tabla 4. Como se comentó, la escasa presencia (<10% del total) de etiquetas en alguna de las emociones, pudo ocasionar la baja precisión del modelo para predecir las emociones (análisis realizado en los siguientes capítulos).

Emoción	Nº instancias- *Training* (%Total)	Nº instancias- *Validation* (%Total)	Nº instancias- Test (%Total)
Happy	**42(3%)**	**17(5,8%)**	12(40,1%)
Empathy/ Confidance /Trust	294(21,3%)	67(22,7%)	67(22,6%)
Fear	141(10,2%)	33(11,2%)	33(11,0%)
Neutral	496(36%)	95(32,2%)	87(29,4%)
Surprised	**122(8,9%)**	**27(9,1%)**	32(10,8%)
Anger	196(14,3%)	36(12,2%)	43(14,5%)
Sad	**87(6,3%)**	**20(6,8%)**	**22(7,4%)**

Tabla 4. Distribución de emociones en la primera iteración.

Con la segunda versión del corpus, se buscó contribuir en aquellas emociones con menor representación. De manera que tras la ampliación del corpus se obtuvo una distribución como la que se muestra en la Tabla 5.

Emoción	Nº instancias- *Training* (%Total)	Nº instancias- *Validation* (%Total)	Nº instancias- Test (%Total)
Happy	**56(3,9%%)**	**12(3,9%)**	**9(2,9%)**
Empathy/ Confidance /Trust	319(22%)	74(23,8%)	62(19,8%)
Fear	**136(9,4%)**	31(10%)	31(9,9%)
Neutral	448(30,8%)	94(30,2%)	112(35,8%)
Surprised	**144(9,9%)**	**26(8,4%)**	33(10,5%)
Anger	218(15%)	52(16,7%)	43(13,7%)
Sad	**132(9,1%)**	**22(7,1%)**	**23(7,3%)**

Polaridad	Nº instancias- *Training* (%Total)	Nº instancias- *Validation* (%Total)	Nº instancias- Test (%Total)
Positive	322(22,2%)	79(25,4%)	80(25,6%)
Negative	612(42,1%)	125(40,2%)	129(41,2%)
Neutral	519(35,7%)	107(34,4%)	104(33,2%)

Tabla 5. Distribución de emociones y polaridad en la segunda versión del corpus.

A pesar de mejorar el número de instancias de las clases que se encontraban muy escasas en la primera versión de corpus, las clases de *Happy y Sad* siguen las dos clases de emociones que se encuentran en todos los ficheros por debajo de un 10% de representación respecto del total. Como se mostrará posteriormente en capítulos posteriores, en los resultados, su precisión mejoró, pero aún se mantuvo baja.

Es por ello por lo que se decidió ampliar esta segunda versión del corpus con técnicas de mejora de balanceo, que se contarán en apartados posteriores. El etiquetado de estas nuevas oraciones sintéticas siguió el mismo proceso que las anteriores y que ya ha sido descrito con anterioridad. Finalizando el proceso con una nueva versión del corpus: *CorpusDialoguesTLoUv2_sintetic*, con 2159 instancias (Anexo 2) y que tras la división en ficheros se obtuvo la distribución de la Tabla 6.

Emoción	Nº instancias- *Training* (%Total)	Nº instancias- *Validation* (%Total)	Nº instancias- Test (%Total)
Happy	**81(5,4%)**	**18(5,6%)**	**17(5,2%)**
Empathy/ Confidance /Trust	320(21,2%)	69(21,4%)	66(20,3%)
Fear	**142(9,4%)**	**28(8,7%)**	**38(8,6%)**
Neutral	456(30,2%)	105(32,5%)	93(28,6%)
Surprised	**136(9,0%)**	**30(9,3%)**	37(11,4%)
Anger	228(15,1%)	42(13%)	43(13,2%)
Sad	**148(9,8%)**	**31(9,6%)**	41(12,6%)

Polaridad	Nº instancias- *Training* (%Total)	Nº instancias- *Validation* (%Total)	Nº instancias- Test (%Total)
Positive	375(24,8%)	76(23,5%)	69(21,2%)
Negative	639(42,3%)	133(41,2%)	137(42,2%)
Neutral	497(32,9%)	114(35,3%)	119(36,6%)

Tabla 6. Distribución de emociones y polaridad en la segunda versión del corpus con muestras sintéticas.

Se puede observar como el número de instancias de las clases que se incorporaron, aumentó. Llegando a mejorar los porcentajes sobre el total e incluso superando el 10% en la clase *Sad* para el fichero de test. Sin embargo, aún son escasas las muestras y se encuentran desbalanceadas con respecto del resto de clases. En los resultados, que se mostrarán posteriormente en los siguientes capítulos, se observara una mejoría de las clases al aplicar esta estratégica de datos sintéticos.

5

Evaluación de los costes de implementación de modelos de OpenAI con el corpus *The Last of Us*

5.1 Introducción

En este capítulo se analizará el coste de implementación y entrenamiento de los modelos de OpenAI. Con el objetivo de poder calcular cual será el coste que se incurrirá durante el proceso de ajuste del modelo, se deberá trocear el corpus imitando el comportamiento de los modelos. Para ello empleamos la librería tiktoken de OpenAI. Se estima que 1 *token* ≃ 0,75 palabras de habla inglesa (Rico Sanguino, Perona Jiménez y Sánchez Piñeiro, 2025).

5.2 Proceso de tokenización del Corpus

Para el troceado en *tokens* del corpus, se partirá del fichero JSON, que deberá cargarse de nuevo. En primer lugar, se hace para el fichero de emociones y después para la polaridad y para ello se puede emplear el siguiente código de la Figura 20.

```
#This code loads a polarity training JSONL file into a list and prints the total number of
examples.
data_path = “training_Polaridad.jsonl” #training file url
# Load the dataset
with open(data_path, ‘r’, encoding=’utf-8’) as f:
dataset = [json.loads(line) for line in f if line.strip()]
# Initial dataset stats
print(“Num examples:”, len(dataset))

#This code sets the text encoding using the “o200k\_base” tokenizer for tokenizing text
with the tiktoken library.
#encoding = tiktoken.encoding_for_model(‘gpt-4o-mini’)#in case of ignorance of the encoding
encoding=tiktoken.get_encoding(“o200k_base”)
# These functions calculate token counts for message data using a tokenizer and print
statistical distributions of given values.
# not exact!
# simplified from https://github.com/openai/openai-
cookbook/blob/main/examples/How_to_count_tokens_with_tiktoken.ipynb
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 20. Funciones necesarias para el tokenizado del corpus. (Wu, M. y Fishman, S, s. f)

Este código puede emplearse para cualquier fichero en formato JSONL. Es importante tener en cuenta que, según la información de OpenAI (2023) y una consulta reciente del usuario Tarekgh (2025) sobre el soporte para GPT-4.1 en la biblioteca tiktoken, el *encoding* utilizado varía entre modelos. Para los modelos GPT-4o, GPT-4o-mini, GPT-4.1 y GPT-4.1-mini, el *encoding* es *o200k_base*, mientras que para GPT-4 se utiliza *cl100k_base*. Por lo tanto, la cantidad de tokens puede diferir entre estos modelos.

A continuación, se realiza una nueva comprobación de los ficheros JSONL, en este caso con carácter estadístico, que permitirá saber si los datos se truncarán en el proceso de entrenamiento. Esto podría producir la pérdida de información relevante y haría que nuestro modelo no fuera apto. La salida tras la ejecución del código de la Figura 21 es (Wu, M. y Fishman, S, s. f):

1. Número de conversaciones que no cuenta con mensajes de *System.*
2. Número de conversaciones que no cuenta con mensajes de *User.*
3. La distribución de los mensajes por cada una de las conversaciones (*System, User, Assistant.*
4. La distribución de los tokens por cada conversación.
5. La distribución de los tokens por cada conversación referidos únicamente al rol *Assistant*.
6. Número de conversaciones que superan el límite del modelo. Es la salida más importante, pues indicará si hay datos truncados y si se puede estar cometiendo error por el truncado.

Se debe tener en cuenta que el valor límite de 16.384 tokens que se encuentra en el código de la Figura 21 es el que corresponde con el valor de la Tabla 1: "*Max Tokens salida*" y por ello debe modificarse para cada modelo.

```
#This code analyzes the dataset for missing roles,
#calculates token counts per message and conversation, and reports examples that exceed the
16,384 token limit.
#Statistical check. This will determine whether the data will be truncated, resulting in
the loss of relevant information and the
#model being unsuitable.
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 > 16384  for l in convo_lens)  # Limit value of the model, GPT4o-mini=
16384 max token output
print(f”\n{n_too_long} examples may be over the 16.384  token limit, they will be truncated
during fine-tuning”)
```

Figura 21. Comprobación estadística de los tokens. (Wu, M. y Fishman, S, s. f)

En el último de los códigos empleados (Figura 22), se realiza el conteo de los *tokens* del fichero de emociones o polaridad que hemos cargado en la Figura 16. Se debe indicar el número de épocas perseguidas. El número de épocas hace referencia a cuántas veces el modelo recorre todo el conjunto de datos de entrenamiento durante el proceso de aprendizaje. Se ha indicado trabajar con tres épocas, pues para los modelos de OpenAI, se suele sugerir este número de épocas (para no incurrir en gastos excesivos en el proceso de entrenamiento), como también se recoge en el trabajo Martín García (2024). Se optó por fijar únicamente el número de *epochs* para controlar la duración del entrenamiento y el coste.

```
#This code calculates the optimal number of training epochs based on dataset size and token
limits,
#and estimates total token usage for billing during model training.
MAX_TOKENS_PER_EXAMPLE = 16384  # Limit value of the model, GPT4o-mini= 16384 max token
output
MIN_TARGET_EXAMPLES = 100
MAX_TARGET_EXAMPLES = 25000
TARGET_EPOCHS = 3   #epochs estimated 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")
```

Figura 22. Cálculo de los tokens con un número de épocas establecido (Wu, M. y Fishman, S, s. f)

5.3 Cálculo de costes

Los modelos que se van a emplear requieren de un pago por su uso, como se indicó en la Tabla 1. Para poder estimar cuánto puede ser el coste en el que se puede incurrir, se calcula el coste de entrenamiento/uso del corpus. Una forma de poder estimar el posible precio será a través de la fórmula de la Ecuación 5.

$$\frac{Precio\ entrenamiento}{1.000.000\ tokens} * nº\ de\ tockens\ en\ los\ que\ se\ divide\ el\ corpus * nº\ épocas$$

Ecuación 5. Fórmula para el cálculo del coste.

El precio de entramiento por cada millón de *tokens* se encuentra en la Tabla 1 y es diferente en cada modelo. Es necesario calcular el nº de *tokens* en los que troceará el corpus cada modelo. Además, se analizará el coste de entrenamiento para el conjunto de las emociones y para la polaridad.

5.4 Estimación de coste del Corpus etiquetado

Con el objetivo de poder comparar los diferentes modelos disponibles, descritos en el apartado 2.5 Comparativa de modelos de OpenAI: GPT-3.5, GPT-4, GPT-4o y GPT-4.1, se precisa calcular el coste estimado que se puede incurrir. Para ello se analizará el coste de entrenar y probar de corpus.

El proceso de tokenizado es el que se describe en una sección anterior. Se debe realizar para el fichero JSON del corpus en su totalidad tanto en polaridad como en emociones. Además, se debe tener en cuenta la separación del corpus en los ficheros de entrenamiento, validación y test. De esta forma, los ficheros a usar son los seis mencionados, tres para emociones y tres para polaridad. Tras la ejecución del código para el conteo de los *tokens*

(Figura 22) de los ficheros de emociones y polaridad para la primera versión del corpus y para cada uno de los modelos se obtuvieron los resultados que se reflejan en la Tabla 7. La diferencia que podemos encontrar en los *tokens* de emociones y polaridad para cada modelo es debida principalmente al mensaje utilizado en el rol de "System", siendo el mensaje en emociones más largo que el empleado por polaridad.

	EMOCIONES Training+ Validation	**POLARIDAD Training+ Validation**	**SUMA**	**X ÉPOCAS**	**EMOCIONES Test**	**POLARIDAD Test**	**SUMA TEST**
ÉPOCAS	3	3	-	-	1	1	-
cl100k_base (GPT4)	107.406	78.189	185.595	**556.785**	19.060	13.739	**32.799**
o200k_base (GPT4.1) (GPT4.1-mini) (GPT4o) (GPT4o-mini)	106.488	77.282	183.770	**551.310**	18.911	13.579	**32.490**

Tabla 7. Tokens estimados del corpus utilizado.

Así pues, los costes estimados, tomando los precios de la Tabla 1 y siguiendo la fórmula de la Ecuación 5, son los que aparecen en la Tabla 8.

	TOKENS ENTRENAMIENTO	**PRECIO DE ENTRENAMIENTO /1M tokens**	**COSTE ENTRENAMIENTO**	**TOKENS TEST**	**PRECIO INPUT/1M tokens**	**COSTE TEST**	**COSTE ESTIMADO**
GPT4	556.785	-	**-**	32.799	30$	**-**	**-**
GTP4o	551.310	25$	**13.78$**	32.490	2.50$	**0.08$**	**13.86$**
GPT4o-mini	551.310	3$	**1.65$**	32.490	0.15$	**0.005$**	**1.66$**
GPT4.1	551.310	25$	**13.78$**	32.490	2$	**0.06$**	**13.84$**
GPT4.1-mini	551.310	5$	**2.75$**	32.490	0.40$	**0.01$**	**2.76$**

Tabla 8. Cálculo de coste con el uso de la primera versión del corpus.

En esta ocasión se seguirán los mismos pasos anteriores, pero el corpus que analizaremos será el corpus final en esta ocasión la segunda versión. Como se vio en el apartado anterior, los dos modelos que vamos a analizar (GPT4o-mini y su versión 4.1) emplean el mismo tokenizador de forma que sus *tokens* serán los mismos y se reflejan en la Tabla 9 (arriba) y el coste para cada modelo en la Tabla 9 (abajo).

	EMOCIONES	POLARIDAD	SUMA
Épocas	3	3	
Tokens-Training	93.564	67.928	
Tokens-Validation	19.949	14.354	
SUMA	113.513	82.282	
XÉPOCAS	340.539	246.846	587.385

	TOKENS	PRECIO DE ENTRENAMIENTO /1M tokens	COSTE ESTIMADO FINE-TUNING
GPT4o-mini	587.385	3$	**1.76$**
GPT4.1-mini	587.385	5$	**2.94 $**

Tabla 9. Suma de tokens (arriba) y coste estimado para fine-tuning del corpus etiquetado v2 (abajo).

Si tenemos en cuenta el coste que se incurrirá al usar el modelo con las oraciones del fichero de test, se obtienen los costes de test de la Tabla 10:

	EMOCIONES	POLARIDAD	SUMA	PRECIO DE ENTRADA /1M tokens	COSTE ESTIMADO TEST
Tokens-Test (GPT4o-mini)	19.872	14.620	34.492	0.15$	**0.005$**
Tokens-Test (GPT4.1-mini)	19.872	14.620	34.492	0.40$	**0.014$**

Tabla 10. Coste por uso del modelo con el fichero de test v2.

Se debe tener en cuenta que en esta ocasión no se debe multiplicar el número de *tokens* por el número de épocas, pues es un proceso que solo ocurre en el ajuste del modelo. Además, se debe tener en cuenta que el precio por millón de *tokens* no es el mismo al precio de entrenamiento. Los valores de entrada han sido tomados de la Tabla 1.

Con todo ello, el coste estimado de todo el proceso será de 1.77$ para el modelo de GPT4o-mini y de 2.95$ para el modelo GPT4.1-mini, sumando 4.72$. Así pues, el coste es de 1.66$ para la primera versión, (ya que, debido a los resultados previstos, no se realizó el entrenamiento con GPT4.1-mini) y 4.72$ para la segunda versión del corpus.

5.5 Coste real del proyecto

Una vez llevado a cabo el proceso de ajuste de los modelos, podemos ver cuál fue el coste real de este trabajo. La propia plataforma de OpenAI proporciona la información de uso, en la

sección de ajustes. El coste real que se tubo con la primera iteración, el corpus *CorpusDialoguesTLoUv1,* fue de 1.32$, que se desglosa en la Tabla 11.

	Emociones Training	**Emociones Test Input**	**Polaridad Training**	**Polaridad Test Input**	**Test Output**	**Total**
GPT-4o-mini	0.763$	0.005$	0.547$	0.004$	0.001$	**1.32$**

Tabla 11. Gasto real por la optimización y uso del modelo GPT4o-mini. Primera iteración.

El coste con la segunda versión del corpus dónde se amplió el corpus con oraciones de la segunda parte del videojuego fue de 1.41$, una pequeña cantidad más que la versión primera ya que se añadió más contenido, como se ve desglosado en la Tabla 12.

	Emociones Training	**Emociones Test Input**	**Polaridad Training**	**Polaridad Test Input**	**Test Output**	**Total**
GPT-4o-mini	0.816$	0.005$	0.585 $	0.004$	0.001$	**1.41$**

Tabla 12.Gasto real por la optimización y uso del modelo GPT40-mini. Segunda iteración.

A continuación, el coste del corpus final con los datos sintéticos añadidos a la segunda versión del corpus, esto es *CorpusDialoguesTLoUv2_sintetic*, fue de 1.47$ para en entrenamiento y obtención de resultados con el modelo GPT4o-mini, y de 2.44$ para el modelo GPT4.1-mini, como se ve desglosa en la Tabla 13.

	Emociones Training	**Emociones Test Input**	**Polaridad Training**	**Polaridad Test Input**	**Test Output**	**Total**
GPT-4o-mini	0.846$	0.007 $	0.61$	0.004$	0.0005 $	**1.47$**
GPT-4.1-mini	1.41$	0.012$	1.016$	0.004$	0.002$	**2.44$**

Tabla 13. Gasto real por la optimización y uso de los modelos GPT con técnica de mejora de balanceo.

Por último, el coste incurrido para llevar a cabo una última prueba de entrenar con cinco épocas el modelo con mejores prestaciones, esto es GPT4o-mini para la versión final del dataset de emociones, tuvo unos costes de 1.42$ como los mostrados en la Tabla 14.

	Emociones Training	Emociones Test Input	Test Output	Total
GPT-4o-mini	1.41$ $	0.004$ $	0.0005 $	**1.42$**

Tabla 14. Costes para el entramiento con cinco épocas del corpus final.

En suma, el coste total para la realización completa del TFG de 8.06$.

5.6 Consideraciones finales

Debido a los resultados que se fueron alcanzando con las diferentes versiones del corpus y a las continuas mejoras de éste, fue necesario entrenar más modelos de los que estaban previstos y se realizaron diversas pruebas de su rendimiento. De manera que el coste final incurrido en este Trabajo Fin de Grado, 8.06$, es superior a la previsión realizada de 6.38$.

Si se observan las estimaciones realizadas para la versión primera del dataset, para el modelo GPT4o se estimaba un coste de 1.66$ que resultó ser de 1.32$. Igual pasó para la segunda versión, 1.77$ frente a 1.41$ de coste real. Estas diferencias pudieron ser ocasionadas por:

- Diferencias en el troceado de las oraciones: puede existir diferencias en la manera que *tiktoken* trocea los diálogos que difieran de cómo se realiza en los modelos de OpenAI.
- Eficiencia de los modelos ante mensajes repetitivos: al realizar la verificación de los modelos con *chat completions*, la estructura repetitiva en JSON donde se realiza la pregunta y se da formato a la respuesta puede ser optimizada por el modelo.
- Aplicación de precios por cada un millón de *tokens* y recuento manual, hace que el precio pueda estimarse en alza.

6

Evaluación del modelo GPT-4o-mini con el Corpus *The Last of Us*

6.1 Introducción

En este capítulo de la memoria se va a proceder con el ajuste del modelo supervisado de GPT4o-mini. Se explicará paso a pasos como conseguir obtener el modelo optimizado, como usarlo y como obtener los resultados necesarios para el análisis de las emociones y polaridad del videojuego *The Last of Us*. Finalmente, se hará un análisis profundo de los resultados.

6.2 Proceso de entrenamiento y optimización de hiperparámetros del modelo GPT4o-mini

Para comenzar se debe iniciar nuestro cliente de OpenAI. Con la clave *API Key* se llama a la función OpenAI y se le pasa la *API Key,* obtenida en la Figura 7, *(Diego C., 2024)*, tal y como se muestra en la Figura 23.

```
#This code sets OpenAI API keys, initializes a Weights & Biases project for tracking, and
creates an OpenAI client instance.

#API key for GCO account

openai.api_key='sk-proj-p**********mGUisUFjwkEoj_MvHcOXxLzq-
4rrMqaQpldy1vgAzbmR**********PtaW-dXGSZrD-
LyzJ3E2gmG1iphBL2m8wonxaqPpm**********TdFdI7EdAA0A'

#Start wandb

#will guide you to add your API key in order to login

wandb.init(project="Emotion analysis 4o-mini")

#Definition of the client

client = OpenAI(api_key='sk-proj-p**********mGUisUFjwkEoj_MvHcOXxLzq-
4rrMqaQpldy1vgAzbmR**********PtaW-dXGSZrD-
LyzJ3E2gmG1iphBL2m8wonxaqPpm**********TdFdI7EdAA0A')
```

Figura 23. Inicio del cliente de OpenAI y de la herramienta Wandb.

Para iniciar un proyecto de Wandb se ejecuta *wandb.init(project="Emotion analysis 4o-mini"*) con el nombre que se desee, en nuestro caso *Emotion/Polarity anaysis* seguido del modelo que se esté usando (Weights & Biases para Quickstart).

Cuando se ejecute la primera vez se abrirá un recuadro rojo debajo de nuestro código, ahí se solicita la clave secreta de Wandb y Biases. Se deberá copiar la *API Key* de la Figura 9 para iniciar sesión. En este punto del proceso, ya se ha autenticado y creado un nuevo proyecto con el nombre indicado en la plataforma de Wand & Biases como se ve la Figura 24.

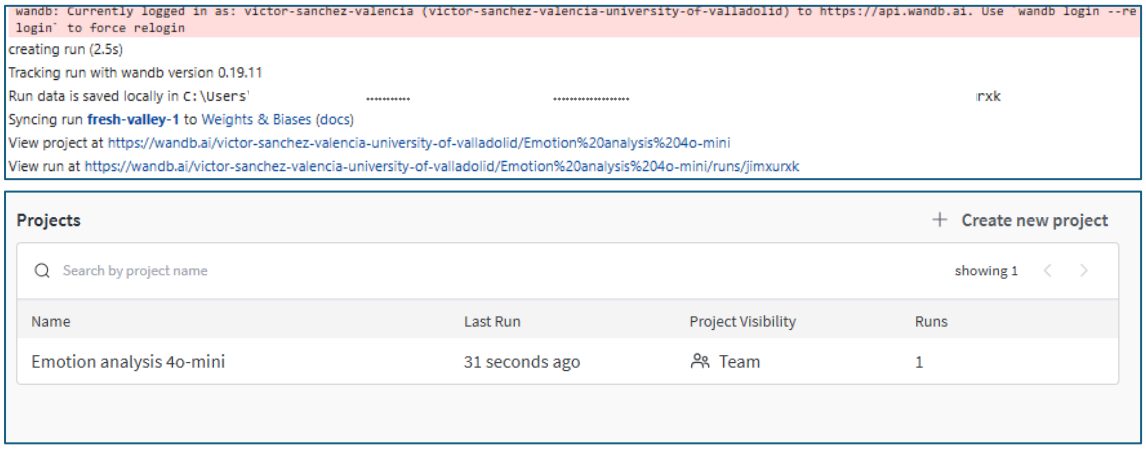

Figura 24. Inicio en W&B y creación de un proyecto. Arriba: Captura del código de salida donde se indica la ejecución y el proyecto actual. Abajo captura de la plataforma wandb.ai donde se ve que se ha creado el proyecto.

El siguiente paso es subir a la plataforma de OpenAI los diferentes ficheros de entrenamiento y validación, que serán usados para entrenar el modelo. Para subir los ficheros, se podrá realizar bien a través de la plataforma de OpenAI o bien a través de código, en concreto desde la sección *Dashboard* que se muestra en la Figura 5. Se buscará la sección "Vista general", y después se deberá entrar en *Storage,* marcado con un recuadro verde en la Figura 5. Después pulsando en *"Upload"* se cargará el fichero, indicando un nombre del fichero y el propósito que en el caso que nos ocupa será "*fine-tune"* como se muestra en la Figura 25.

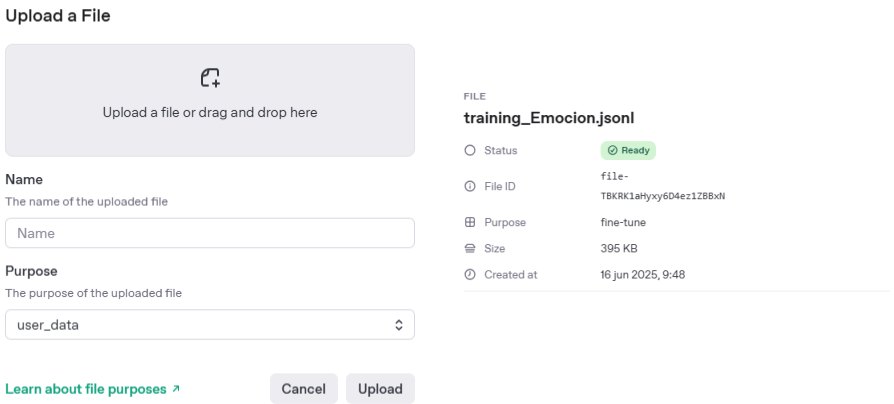

Figura 25. Subir un fichero desde la plataforma OpenAI. Izquierda: vista generada tras pulsar "upload" donde se piden los datos del fichero a subir. Derecha: parte del menú general y los detalles de un fichero subido.

Usando el código de la Figura 26 se suben los ficheros *training_Emocion.jsonl* y *validation_Emocion.jsonl* (OpenAI. s.f. para API-*reference*). Como respuesta a la llamada *client.files.create* se obtiene un objeto para uno de los ficheros, como el que se muestra a continuación:

```
{
FileObject(id='file-TBKR*******4ez1ZBBxN',
bytes=404094, created_at=1750060125,
filename='training_Emocion.jsonl',
object='file',
purpose='fine-tune',
status='processed',
expires_at=None,
status_details=None
)}
```

```
# This code uploads training and validation JSONL files to OpenAI for fine-tuning and
prints their assigned file IDs.
training_file = client.files.create(
file=open("training_Emocion.jsonl", "rb"),
purpose='fine-tune'
)
validate_file = client.files.create(
file=open("validation_Emocion.jsonl", "rb"),
purpose="fine-tune"
)

training_file_id = training_file.id
validate_file_id = validate_file.id
print("Training_File has been uploaded to OpenAI with id ", training_file_id)
print("Validate_File has been uploaded to OpenAI with id ", validate_file_id)
```

Figura 26. Subida de los ficheros de training y validation a la plataforma de OpenAI.

Se deberá guardar el identificador del fichero en una variable que llamaremos *training_file_id* y *validate_file_id*, para que pueda usarse. Se realizará de igual manera para subir los ficheros relativos a la polaridad: *training_Polaridad.jsonl y validation_Polaridad.jsonl*.

Con los ficheros subidos se va a crear un nuevo trabajo que inicie el proceso de optimización del modelo. Se indican los ficheros de entrenamiento y validación con sus respectivos identificadores (guardados en las variables *training_file_id y validate_file_id*), se especifica el modelo que se quiere emplear y las épocas. A continuación, se integra la herramienta de wandb indicando a que proyecto queremos que se manden las métricas, el que se ejecuta actualmente, y un nombre (OpenAI. s.f. para *API-reference*).

El número de épocas se establecerá en 3, como ya se expuso con anterioridad. Otros hiperparámetros que se podrían indicar, como *bacth_size* y el *learning_rate_multiplier*, se dejan para que sea el modelo quien los optimice.

Para poder saber cuál es el identificador del modelo se puede emplear el código que se muestra al comienzo de la Figura 28 donde se hace una llamada que devuelve un listado (Figura 27) con todos los modelos disponibles de OpenAI. Después se hace un filtrado de coincidencias con el texto escrito en negrita. Si en la lista existen coincidencias se listarán los modelos y se escogerá el que se busca. Tras la ejecución se obtiene el listado de modelos de la Figura 27.

```
[
Model(id='gpt-4o-mini-audio-preview', created=1743878424, object='model', owned_by='system'),
Model(id='gpt-4o-mini-realtime-preview', created=1743873880, object='model', owned_by='system'),
Model(id='gpt-4o-mini-realtime-preview-2024-12-17', created=1743112601, object='model', owned_by='system')
Model(id='gpt-4o-mini-search-preview', created=1741931161, object='model', owned_by='system'),
Model(id='gpt-4o-mini-search-preview-2025-03-11', created=1741398858, object='model', owned_by='system'),
Model(id='gpt-4o-mini-tts', created=1742403959, object='model', owned_by='system'),
Model(id='gpt-4o-mini-2024-07-18', created=1712172177, object='model', owned_by='system'),
Model(id='gpt-4o-mini', created=1712172174, object='model', owned_by='system'),
Model(id='gpt-4o-mini-audio-preview-2024-12-17', created=1743115920, object='model', owned_by='system'),
Model(id='gpt-4o-mini-transcribe', created=1742088596, object='model', owned_by='system')
]
```

Figura 27. Listado de los modelos que coinciden con "gpt-4o-mini"

```
#This code retrieves all available models from the OpenAI client and filters the list to
show only those with "gpt-4o-mini" in their model ID.
models=client.models.list()
gpt4o_models = [m for m in models if "gpt-4o-mini" in m.id]
gpt4o_models

#This code initiates a fine-tuning job on the "gpt-4o-mini" model
#using uploaded training and validation files, sets hyperparameters,
#integrates with Weights & Biases, and prints job details and ID.
ft_job = client.fine_tuning.jobs.create(
training_file=training_file_id,
validation_file=validate_file_id,
model="gpt-4o-mini-2024-07-18",
hyperparameters={
„n_epochs": 3
    },
integrations=[
        {
„type": „wandb",
„wandb": {
"project": wandb.run.project,
"name": "4o-mini-model-emotion"
            }
        }
    ]
)
model_id=ft_job.fine_tuned_model
print("Fine Tune Job has been created with id ", ft_job.id)
```

Figura 28. Creación de un trabajo para la optimización del modelo con ficheros de validación y test e integración de wandb.

De igual forma se realizará para la creación del modelo para análisis de la polaridad de emociones, con los cambios pertinentes en *training_file_id, validate_file_id* y el nombre del proyecto de wandb: *"4o-mini-model-polarity"*.

El modelo optimizado ya se ha creado. En la plataforma de OpenAI (Figura 5) se puede ver el modelo que se ha ajustado, para ello se debe entrar en la sección "Fine-tuning", donde se listan todos los modelos creados, y sus características.

6.2.1 Características del modelo optimizado

La herramienta Weights & Biases permite obtener las gráficas comentada en la sección 3.7.3 Gráficas del proceso de Fine-tuning. Para presentar y analizar el comportamiento del modelo de forma clara y concisa, se optó por centrarnos en las gráficas *train loss* y *valid mean token accuracy* que se recogen en las Figuras 29 y 30.

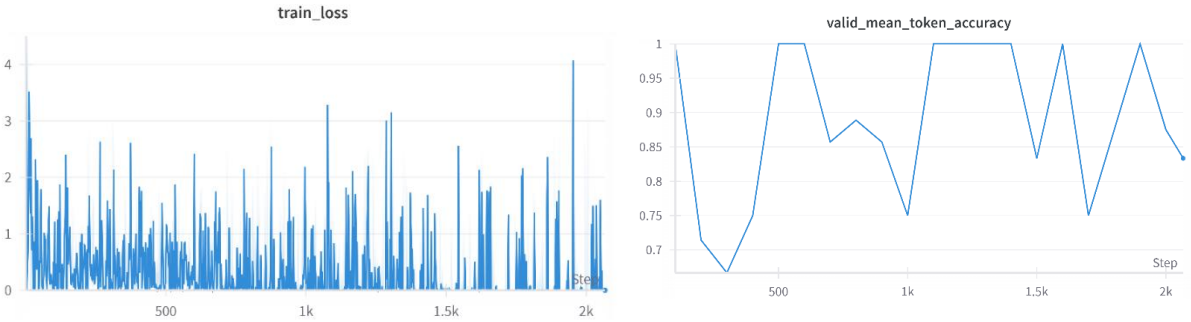

Figura 29. Gráficas de rendimiento del modelo de emociones en el proceso de aprendizaje.

Tanto en la Figura 29 (emociones), como en la Figura 30 (polaridad), la gráfica de *train_loss* refleja una tendencia a disminuir según se van entrenando más *tokens*. Esto refleja como el modelo va aprendiendo de forma que la diferencia entre las etiquetas verdaderas y la predicción del modelo es cada vez menor. Si embargo se observan picos que muestran inestabilidad del modelo, en parte debido al alto *learning rate* y que pueden señalar como el modelo cambia los pesos para ajustarse a los datos (*batches)* (Martín García, 2024).

En la gráfica referida al modelo de emociones (Figura 29), la gráfica *valid_mean_token_accurancy,* muestra fluctuaciones a lo largo del entrenamiento que implican inestabilidad como hemos comentado. Importante es que se alcanzan valores de 1 (100%), que muestran que todos los *tokens* fueron predichos de manera correcta conforme a las etiquetas verdaderas. Sin embargo, no se aprecia que los valores se estabilicen al alza según se realiza el

entrenamiento. Una combinación de baja precisión con una caída de las perdidas en el proceso de ajuste puede indicar *overfitting.*

En la Figura 30 se muestran las gráficas para el modelo de polaridad, a la derecha se encuentra la gráfica *valid_mean_token_accuracy*. En comparación con el modelo de emociones podemos ver cómo se consigue en más ocasiones valores cercanos a uno, pero también se ve una mayor inestabilidad. En el resto de los modelos entrenados las gráficas presentaban valores similares.

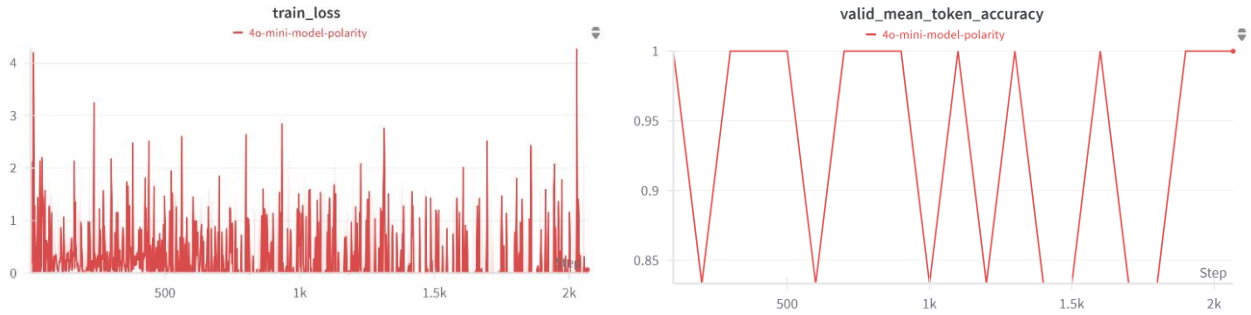

Figura 30. Gráficas de rendimiento del modelo de polaridad en el proceso de aprendizaje.

6.3 Proceso de validación del modelo GPT4o-mini con datos de test

Con el modelo optimizado (hiperparámetros óptimos), se podrá medir su eficiencia. Para hacer esta evaluación se va a recurrir a los datos de test que se encuentran en el fichero Excel *test_Emocion.xlsx*, y que corresponden con datos que los modelos no han visto antes durante el proceso de entrenamiento.

Para realizar este proceso de evaluación final, se realizará una petición al modelo con cada una de las líneas de diálogo de este fichero Excel, para que dé como respuesta la etiqueta que predice de emoción. Con la etiqueta predicha por el modelo y la etiqueta verdadera, que se impuso en el proceso de elaboración del corpus, se elabora una tabla comparativa que se sube a la herramienta wandb, una matriz de confusión y un reporte de clasificación.

En primer lugar, se define una función que hará la llamada al modelo con la pregunta del rol "*System*": "*What is the emotion of the following text? Respond with 'Anger', 'Fear', 'Happy', 'Sad', 'Neutral', 'Surprised' or 'Empathy'*". Y el texto del diálogo que debe analizar.

En esta función se debe establecer los límites de velocidad a la hora de realizar las peticiones. Estos límites vienen establecidos por los modelos, esto es, 500 peticiones/minuto, 10.000 peticiones/día y 200.000 *tokens*/minuto (TPM) para GPT-4o-mini (OpenAI, 16 mayo.

2025. Para *Organization usage limits*). Siendo este último el único que se podría traspasar y por ello que se deba establecer en la llamada. Además, en la llamada se debe indicar cual es el modelo contra el que se va a realizar la petición. La función devolverá únicamente el valor de la respuesta que se encuentra en *completion.choices[0].message.content* (Figura 31).

```
#This function sends a text to the fine-tuned GPT-4o-mini model to get its emotion
prediction, respecting rate limits to avoid API throttling.
RATE_LIMIT_TPM=200000 #gpt4o-mini TPM limit Tier1
@sleep_and_retry
@limits(calls=RATE_LIMIT_TPM, period=60)
def make_request(text):
completion = client.chat.completions.create(
model=model_id,
messages=[
{"role": "system", "content": "What is the emotion of the following text? Respond with
'Anger', 'Fear', 'Happy', 'Sad', 'Neutral', 'Surprised' or 'Empathy'."},
{"role": "user", "content": text},
        ]
    )
return completion.choices[0].message.content
```

```
#This code reads test data from Excel, sends texts to the model for emotions predictions,
stores results with true labels,
#and logs them to Weights & Biases for analysis.
filename = 'test_Emocion.xlsx'
df= pd.read_excel(filename)

model_id="ft:gpt-4o-mini-2024-07-18:personal::Bi***Wn"
class_emotion=["Anger", "Fear", "Happy", "Sad", "Neutral", "Surprised", "Empathy"]
predictions=[]
true_labels=[]

df_results = pd.DataFrame(columns=["Comment", "Prediction", "True label"])
for index, row in df.iterrows():
text = row['MESSAGE']
try:

response=make_request(text)
complete_predictions=response

label=class_emotion.index(response)
predictions.append(class_emotion[label])
true_label = row['EMOTION']
true_labels.append(true_label)
df_results = pd.concat([df_results, pd.DataFrame({"Commnent": [text], "Prediction":
[response], "True Label": [true_label]})], ignore_index=True)
except Exception as e:
print("Error:", e)
continue

wandb.log({"predictions_vs_true_label_emotion-4o-mini":
wandb.Table(dataframe=df_results)})
```

Figura 31. Uso del modelo optimizado para obtener etiquetas de emociones y medir la eficiencia. Modificado de Martín García (2024).

Tal como se observa en la parte inferior de la Figura 31, se realizan las llamadas a la función que da comienzo a la ejecución del flujo principal. Se establecen tres listas: la primera, *class_emotion,* donde se recogen las emociones con las que se ido trabajando; *predictions,*

que recogerá las emociones que genere el modelo para cada texto y finalmente *true_labels*, que recogerá cual fue la etiqueta que se impuso en el proceso de elaboración del corpus. En el bucle *for* que recorre cada una de las filas del fichero, se recoge el texto que se analizará y realiza la llamada a la función *make_request* pasando el texto. Esta función devolverá la respuesta del modelo. Para buscar que etiqueta es, iteramos por la lista que contiene todas las emociones para ver si existe coincidencia y ver en que índice se encuentra. Después guardamos en la lista *predictions* la emoción predicha por el modelo. En la lista *true_label* se almacena el valor de la columna del Excel "*EMOTION"* que contiene la emoción dada en el proceso de etiquetado. Con las listas completas, tras finalizar el bucle, se obtendrá una variable de tipo *DataFrame df_results* que tendrá un formato matriz con columnas: *Comment, Prediction, True label* y tantas filas como diálogos en el fichero de test. Esta tabla se subirá a la herramienta Wandb.

De igual manera se deberá realizar los pasos anteriores para el caso de uso de polaridad, con los siguientes cambios:

- Cambio en el modelo: *model_id*
- Uso del fichero: *test_Polaridad.xlsx*
- Cambio en la cuestión a preguntar: *"What is the polarity of the following text? Respond with 'Positive', 'Negative' or Neutral'."*
- Uso de la variable *polarity_classes=["Positive", "Negative", "Neutral"]*
- Cambio en la selección de la *true_label = row['POLARITY']*
- Cambio en el nombre para subir la tabla a wandb: *predictions_vs_true_labels_polarity-4o-mini*

Con este proceso de validación se obtiene el reporte de clasificación y la matriz de confusión que se describen en las siguiente secciones.

6.3.1 El reporte de clasificación

Como veíamos en el Apartado 3.7.2 Reporte de clasificación, el reporte de clasificación nos proporcionará el rendimiento del modelo ajustado. Para obtener el reporte de clasificación se hace uso del código de la Figura 32, donde también se sube a la herramienta de Wandb como tabla.

```
#This code generates and prints a classification report comparing the model's predicted
emotions to the true labels,
#including precision, recall, and F1-score metrics.
report = classification_report(true_labels, predictions)
```

```
print(“Clasification Report:”)
print(report)

#Parses the ‘classification_report‘ output into a structured table and logs it to Weights &
Biases for easy performance visualization.
lines = report.split(‘\n’)
lines = [line for line in lines if line.strip()]
table_data = []
for line in lines:
parts = line.split()
parts = [part for part in parts if part]

if len(parts) == 5 and parts[0] != ‘accuracy’:
class_name = parts[0]
precision = parts[1]
recall = parts[2]
f1_score = parts[3]
support = parts[4]

table_data.append([class_name, precision, recall, f1_score, support])

columns = [”Class”, “Precision”, “Recall”, “F1-score”, “Support”]
wandb.log({”classification_report_emotion-4o-mini”: wandb.Table(data=table_data,
columns=columns)})
```

Figura 32. *Reporte de clasificación y subida a wandb.ai. De la sección de Tables de (Weights & Biases para Quickstart).*

Para obtener el reporte de clasificación relativo al análisis de la polaridad se deberá ejecutar el mismo código con el único cambio en el nombre de la tabla a subir a la herramienta wandb: *classification_report_polarity-4o-mini.*

6.3.2 La matriz de confusión.

Como se había visto en el Apartado 3.7.1 Matriz de confusión, la matriz de confusión comparará las predicciones con las etiquetas verdaderas para ofrecer en formato visual una tabla. Para obtener la matriz de confusión se hace uso del código de la Figura 33, donde también se sube a la herramienta de Wandb como una imagen.

```
#Generates and visualizes the confusion matrix of model predictions versus true labels,
#saves it as an image, and logs it to Weights & Biases for tracking and analysis.
#Confusion matrix
matrix= confusion_matrix(true_labels, predictions, labels=class_emotion)
# Confusion matrix printing
print(“Confusion matrix:”)
print(matrix)

vis=ConfusionMatrixDisplay(matrix, display_labels=class_emotion)
fig, ax = plt.subplots(figsize=(10, 8))
vis.plot(ax=ax, cmap=plt.cm.Blues)

# Adjust the spacing between names on the x-axis
ax.set_xticklabels(class_emotion, rotation=45, ha=”right”)
plt.title(“Confusion matrix”)
plt.xlabel(“Predicion”)
plt.ylabel(“True Labels”)
plt.tight_layout()  #Adjust the layout to avoid overlaps

# Save the figure as an image file
```

```
img_path = “confusion_matrix-emotion-4o-mini.png”
plt.savefig(img_path)

# Upload the image to wandb as an artifact
wandb.log({”confusion_matrix_emotion-4o-mini”: wandb.Image(img_path)})
plt.show()
wandb.finish()
```

Figura 33. Matriz de confusión y subida a wandb. De la sección Images de (Weights & Biases para Quickstart).

Para obtener la matriz de confusión del análisis de la polaridad se deberán realizar algunos cambios en el código proporcionado:

- Cambio en las etiquetas: *labels=polarity_classes*
- Cambio en los nombres de la imagen creada y de la que se va a subir a la herramienta wandb: *confusion_matrix-polarity-4o-mini.png* y *confusion_matrix_polarity-4o-mini*.

6.4 Análisis de resultados con GPT4o-mini: emociones y polaridad

En esta sección se analizan los resultados del modelo GPT4o-mini en las tareas de clasificación de emociones y polaridad, comparando las dos primeras versiones del corpus. A partir de esta comparativa, se identifica la necesidad de ampliar y balancear el corpus para mejorar el rendimiento. Posteriormente, se describe la técnica de balanceo aplicada y se presentan los resultados obtenidos. Finalmente, se evalúa el impacto de aumentar el número de épocas de entrenamiento como ajuste final del proceso.

6.4.1 Análisis comparativo de resultados en la detección de emociones

Antes de presentar los resultados obtenidos, es importante detallar la configuración de los hiperparámetros utilizada durante el entrenamiento del modelo. Los hiperparámetros pueden encontrarse entrando en cada uno de los modelos optimizados.

Para los modelos creados para emociones y polaridad con las dos primeras versiones del corpus se obtuvieron los hiperparámetros de la Tabla 15. El número de épocas es el que indicamos anteriormente: 3, un *batch size* de 2 y un *learning rate multiplier* de 1.8. Estos valores no fueron establecidos manualmente, sino que corresponden a configuraciones internas por defecto, ajustadas de forma automática según las condiciones del entorno de entrenamiento. El *batch size*, implica cuantas muestras se procesan antes de cambiar los pesos del modelo. Este valor implica actualizaciones muy frecuentes del modelo. *LR multiplier* es el valor por el que se multiplica la base de aprendizaje, para acelerar la convergencia del modelo. El valor obtenido sugiere un cambio rápido de los pesos.

Hiperparameter	***Value***
Epochs	3
Batch size	2
LR multiplier	1.8

Tabla 15. Hiperparámetros del modelo emociones GPT4o-mini.

En la Tabla 16 se encuentran los resultados resumidos del reporte de clasificación obtenidos con el modelo GPT4o-mini para el caso de las emociones cuando comparamos los resultados de las dos primeras versiones del corpus. En dicha tabla podemos encontrar, en porcentaje, para cada una de las emociones, su precisión, *recall y F1-score.* También se incluye el promedio macro (*macro-avg)* y ponderado (*weighted-avg).*

En primer lugar, se debe enfocar en los resultados de la primera versión del corpus, para dar respuesta a la necesidad de ampliar y modificar el corpus. Estos resultados arrojan una precisión global del 64%. Si nos fijamos en los resultados por clases, *Happy* es la clase con peores resultados (*F1-score* de 38%, 33% *recall* y 60% precisión) lo que implica que esta clase no se predice correctamente, y en muchos casos se clasifica con *Empathy o Neutral*, tal y como se muestra en la matriz de confusión de la Figura 34 (izquierda). Por otro lado, la clase *Sad* es la segunda peor clase, con un *F1-score* del 55%, 59% de *recall* y 52% de precisión. Ambos casos muestran un desajuste entre *recall* y precisión. El caso de *Empathy* (59% F1, 61% *recall* y 57% de precisión) indica que hay otras clases que se están prediciendo como *Empathy* y que no lo son, como *Fear, Neutral, o Happy*, tal y como también se observa en la matriz de confusión de la Figura 34 (izquierda). El resto de las clases tienen resultados algo más moderados, pero por debajo de lo esperado para obtener un modelo preciso.

Con los resultados vistos y lo comentado en el Apartado 4.5 Análisis descriptivo de Corpus, se llevó a cabo el proceso de ampliación del corpus descrito en el Apartado 4.3 Refinamiento y Expansión del Corpus *The Last of Us*. Si ahora se observa en la Tabla 16 los resultados de la ampliación realizada en el dataset, estos han mejorado en nueve puntos porcentuales la precisión global (de 64% a 73%), lo que es buena señal de que la ampliación y mejora del corpus ha ayudado al progreso de resultados. Todas las emociones, excepto *Anger* (que presenta una caída de un punto porcentual), mejoran su *F1-score*. Destacan especialmente las mejoras en las clases con peores resultados de las clases previas, esto es, *Happy* (de 38 % a 60 %) y *Sad* (de 55 % a 71 %). Sin embargo, a pesar de estas mejoras respecto a la versión

anterior, ambas siguen siendo las clases con menor rendimiento, en parte porque son también las más minoritarias. Destaca el bajo valor de *recall* de *Neutral* (64%, que se mantiene), con un *F1-score* de74% y precisión 88%, lo que implica que se están prediciendo mal muchas instancias de *Neutral,* en mayoría confundidas con otras emociones como *Empathy, Surprised o Sad* (Figura 36, derecha). Finalmente, El *F1-score* de *macro-avg*, con valor del 71% implica que hay un buen equilibrio entre las clases, pero sigue existiendo desbalanceo de las clases *Happy* y *Sad,* que cuentan con bajas instancias y los peores resultados.

Emotion	Metric	CorpusDialogues TLoUv1 (%)	Support	CorpusDialogues TLoUv2 (%)	Support
Anger	Precision Recall F1-score	85% 77% 80%	43	80% 77% 79%	43
Empathy/Trust/ Confidence	Precision Recall F1-score	57% 61% 59%	67	64% 75% 69%	61*
Fear	Precision Recall F1-score	60% 64% 62%	33	65% 77% 71%	31
Happy	Precision Recall F1-score	44% 33% 38%	12	55% 67% 60%	9
Neutral	Precision Recall F1-score	63% 64% 64%	87	88% 64% 74%	112
Sad	Precision Recall F1-score	52% 59% 55%	22	64% 78% 71%	23
Surprised	Precision Recall F1-score	78% 66% 71%	32	68% 85% 76%	33
macro-avg	Precision Recall F1-score	63% 61% 61%	296	69% 75% 71%	312
weighted-avg	Precision Recall F1-score	64% 64% 64%	296	75% 73% 73%	312
Global	Accuracy	64%		73%	

** Una de las instancias dio error, de manera que se pasó de 62 a 61 instancias*

Tabla 16. Comparación de resultados en las primeras versiones del corpus para GPT4o-mini.

En la matriz de confusión de la Figura 34, observamos una mejoría generalizada de las confusiones entre las clases. La diagonal principal (clases bien etiquetadas) se observa una clara mejoría en clases destacando *Sad* (pasa de 13 instancias bien clasificadas a 18) y *Empathy* mejora 5 instancias. *Neutral* parece aumentar el número de instancias bien calificadas, pero no se debe olvidar que se ha aumentado el número de instancias respecto del corpus anterior. Como se ha visto, *neutral* bajó su *recall*, lo que implica que se confunde con otras clases, lo que puede verse por ejemplo con *Empathy,* (15 confusiones), *Surprised* (9 confusiones) o

Happy (3 confusiones). Además, se baja la confusión entre clases como *Happy,* que reduce la confusión con las categorías *Empathy o Neutral.* Destaca también en la matriz de confusión la clase *Surprised,* que reduce las confusiones con todo el conjunto de clases y más pronunciado con *Neutral.*

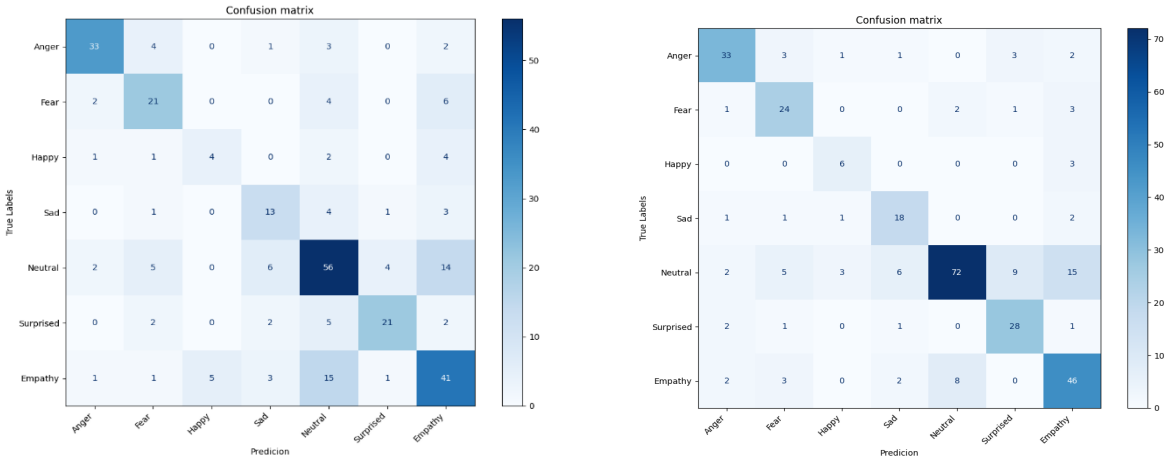

Figura 34. *Matriz de confusión para emociones v1 (izquierda), v2 (derecha)*

Para el caso la polaridad, se obtuvieron unos reportes de clasificación que se resumen en la Tabla 17. La precisión global en la polaridad aumentó en 5 puntos porcentuales (75% a 80%), implicando gran mejoría. La clase *Negative,* mejoró considerablemente, aumentando en gran medida las métricas de *recall* y *F1-score*. La clase *Positive*, tiene una mejoría de 10 puntos porcentuales en *recall* y de 8 puntos en *F1-score*. Por el contrario, la clase *Neutral,* emporó en dos puntos porcentuales, destacando la caída de *recall* (72% a 67%). El modelo ha priorizado la mejora en las clases *Positive* y *Negative*, que tienen mayor carga emocional y un impacto directo en la polaridad general. Como consecuencia, se observa una leve pérdida de rendimiento en la clase *Neutral*. Sin embargo, esta orientación sugiere un avance prometedor hacia una detección más precisa de emociones explícitas, lo que podría beneficiar significativamente tareas donde la polaridad es clave.

Polarity	**Metric**	**Corpus Dialogues TLoUv1 (%)**	**Support**	**CorpusDialogues TLoUv2 (%)**	**Support**
Negative	Precisión Recall F1-score	75% 82% 78%	101	82% 91% 86%	129
Neutral	Precisión Recall F1-score	77% 72% 74%	128*	78% 67% 72%	103**
Positive	Precisión Recall F1-score	72% 70% 71%	66	78% 80% 79%	79***
macro-avg	Precisión	74%		80%	311

	Recall F1-score	75% 74%	295	79% 79%	
weighted-avg	Precisión Recall F1-score	75% 75% 75%	295	80% 80% 80%	311
Global	Accuracy	75%		80%	

** Una de las instancias dio error, de manera que se pasó de 129 a 128 instancias*
*** Una de las instancias dio error, de manera que se pasó de 104 a 103 instancias*
**** Una de las instancias dio error, de manera que se pasó de 80 a 79 instancias*

Tabla 17. Resumen comparativo del reporte de clasificación para la polaridad de las primeras versiones y modelo GPT4o.

La matriz de confusión de la Figura 35, derecha, para el modelo de polaridad, ha ganado en precisión para las clases de *Positive y Negative,* pero se ha reducido para *Neutral* de (de 92 instancias bien clasificadas a 69). Por otra parte, se han reducido las confusiones entre clases para *Positive* y *Negative,* también para la pareja *Neutral-Negative* (de 22 a 19 confusiones) pero ha aumentado para la pareja *Neutral-Positive*.

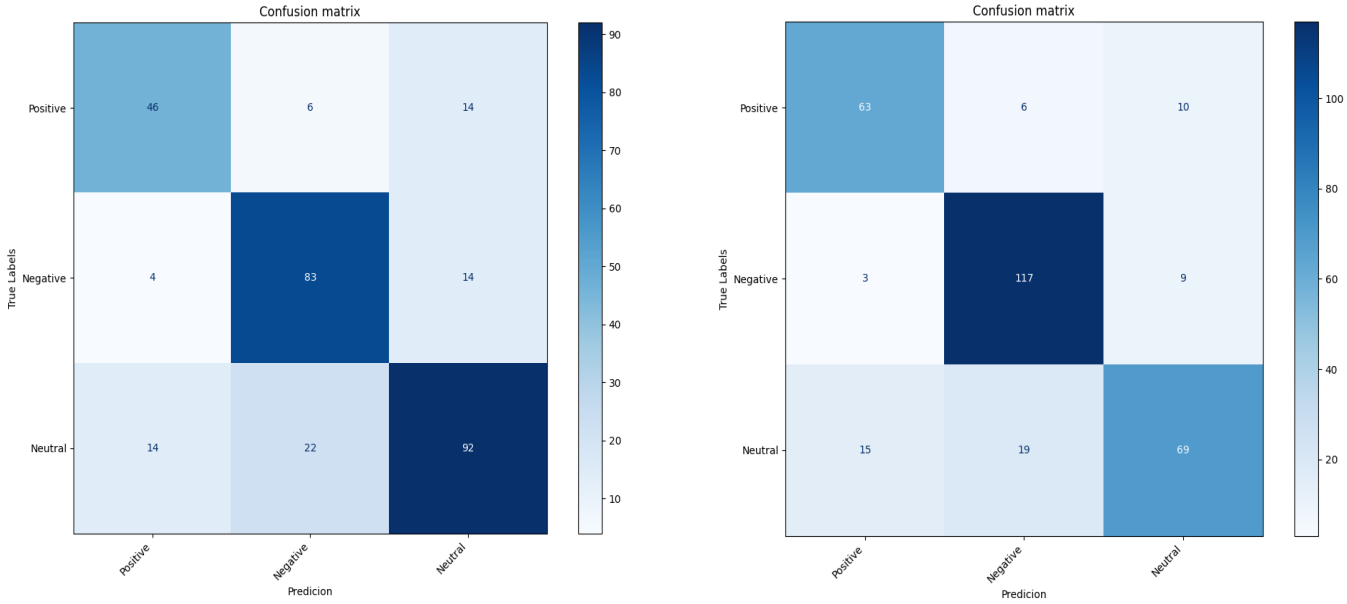

Figura 35. Matriz de confusión para polaridad v1 (izquierda), v2 (derecha)

A la vista de estos resultados, como se ha comentado en otra sección de este trabajo, se observa que persiste cierto desbalance en algunas clases, especialmente en emociones como *Happy* y *Sad.* Para mejorar la capacidad del modelo, se implementarán técnicas de balanceo de datos en la siguiente sección. En cuanto a la polaridad, las clases *Positive y Negative* mejoran a costa de empeorar la clase *Neutral.*

6.4.2 Balanceo de Datos: estrategias de integración de muestras sintéticas

De entre las posibles técnicas o estrategias de balanceo, tal como ajuste de los parámetros del modelo, eliminar muestras de las clases mayoritarias, métodos de ensamble de modelos o crear muestras artificiales, se decidió usar esta última estrategia (Bagnato, s.f.). La técnica empleada, *Synthetic Minority Oversampling Technique (SMOTE),* permitió generar muestras sintéticas de las clases minoritarias, para aumentar su proporción y que el modelo pueda aprender y replicar el comportamiento de esa clase (Brownlee, 2020).

Estas muestras generadas fueron de las clases más minoritarias *Happy y Sad,* pues tenían menor presencia y peores resultados. Las muestras fueron generadas con chatGPT, un modelo de la empresa OpenAI de uso generalista y de acceso público. Además, se crearon en base a otras muestras reales del corpus y todas ellas bajo el contexto del videojuego *TLoU,* de manera que el contenido debía ser del mismo tipo. Tras este proceso de generación de muestras sintéticas, se siguió la metodología de etiquetado por pares, al igual que en etapas previas de etiquetado del corpus.

Los hiperparámetros que optimizaban el modelo para esta nueva versión del corpus fueron: *epochs=3; Batch size=3* y *LR multiplier=1.8.* Con respecto a los modelos anteriores solo se ha producido un cambio en el valor de *Batch size* pasando de 2 a 3. Este cambio implica que se toman más muestras antes de ajustar los pesos del modelo, lo que le da una mayor estabilidad y elimina el ruido al usar un promedio de muestras más representativo.

Los resultados de las distintas métricas utilizadas para comparar el rendimiento del modelo GPT-4o-mini en la tarea de clasificación de emociones —entre la versión 2 del corpus original y su versión ampliada con datos sintéticos— se resumen en la Tabla 18. Tal y como se observa en los resultados, la incorporación de muestras sintéticas en las clases minoritarias *Happy* y *Sad* ha mejorado notablemente su rendimiento, con incrementos sustanciales en el *F1-score* (de 60 % a 84 % en *Happy* y de 71 % a 81 % en *Sad*), así como en precisión y *recall*. Estas mejoras reflejan el impacto positivo del balanceo sobre clases con baja representación. No obstante, se observa una ligera pérdida en el rendimiento de algunas clases previamente fuertes, como *Neutral* (*F1-score* de 74 % a 72 %) y *Anger* (de 79 % a 75 %), posiblemente debido al reajuste del modelo para adaptarse a una distribución más equilibrada. A nivel global, el F1 macro-promedio mejora (de 71 % a 73 %), lo que indica un rendimiento más equitativo entre clases, mientras que el F1 ponderado y la precisión global muestran caídas mínimas, lo que sugiere un leve coste en las clases dominantes a favor de una clasificación más justa y balanceada.

Emotion	Metric	CorpusDialogues TLoUv2 (%)	Support	CorpusDialogues TLoUv2_sintetic (%)	Support
Anger	Precisión Recall F1-score	80% 77% 79%	43	85% 67% 75%	43
Empathy/Trust/ Confidence	Precisión Recall F1-score	64% 75% 69%	61	58% 74% 65%	66
Fear	Precisión Recall F1-score	65% 77% 71%	31	62% 75% 68%	28
Happy	Precisión Recall F1-score	55% 67% 60%	9	93% 76% 84%	17
Neutral	Precisión Recall F1-score	88% 64% 74%	112	78% 67% 72%	93
Sad	Precisión Recall F1-score	64% 78% 71%	23	84% 78% 81%	41
Surprised	Precision Recall F1-score	68% 85% 76%	33	66% 73% 69%	37
macro-avg	Precisión Recall F1-score	69% 75% 71%	312	75% 73% 73%	325
weighted-avg	Precisión Recall F1-score	75% 73% 73%	312	74% 72% 72%	325
Global	Accuracy	73%		72%	

Tabla 18. Comparación de resultados al introducir muestras sintéticas para GPT4o-mini.

Este comportamiento puede también observarse en la comparativa de las matrices de confusión de la Figura 36. Se observa que las clases que mejoran en la versión con sintéticos son *Happy* (de 6 instancias bien clasificadas a 13) y *Sad* (de 18 a 32 bien predichas). Otras clases pierden precisión como *Anger* (pasa de 33 acertadas a 29) *y Neutral* (de 72 a 62 bien predichas). Además, se observa que en ambas matrices se siguen confundiendo la clase de *Neutral* con otras emociones como *Empathy*, *Surprised.*

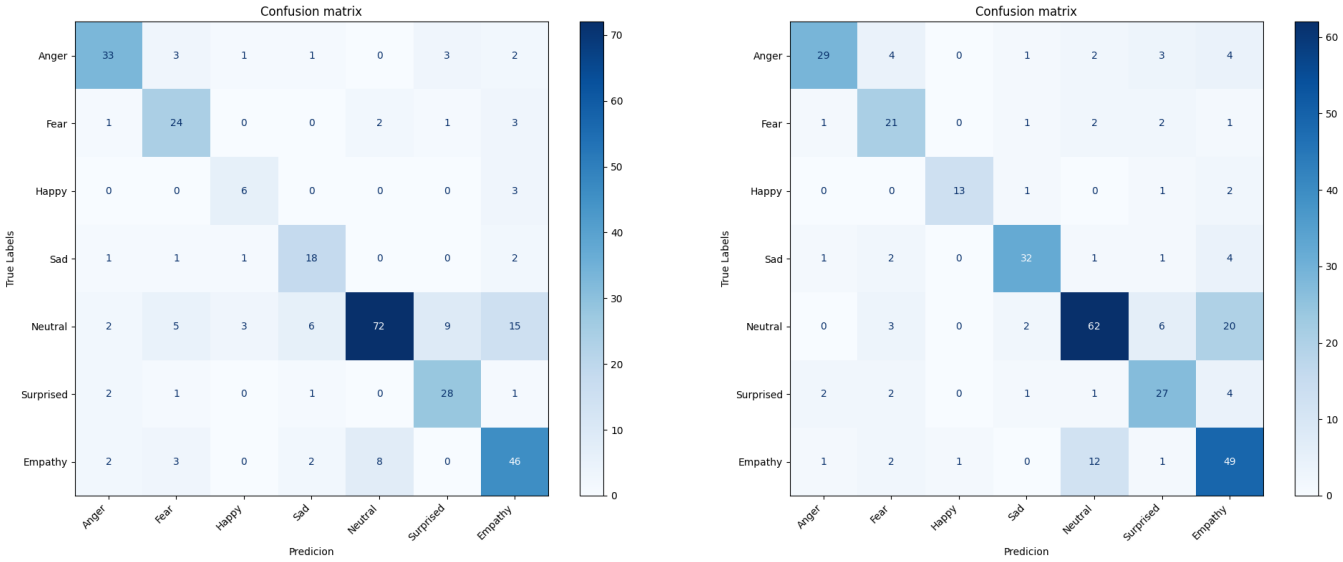

Figura 36. Matriz de confusión para polaridad v2 (izquierda), v2_sintetic (derecha)

En la tarea de clasificación de polaridad, Tabla 19, la incorporación de datos sintéticos ha producido efectos mixtos. La clase *Negative* mantiene un *F1-score* estable (86 %), con una mejora en precisión (de 82 % a 85 %) y una ligera caída en *recall* (de 91 % a 86 %), lo que sugiere un enfoque más conservador del modelo. *Positive* presenta un leve descenso en el *F1-score* (de 79 % a 76 %), pese a una mejora en *recall* (de 80 % a 83 %), y *Neutral* se mantiene estable en *recall* (67 %) pero pierde algo de precisión y *F1-score*. A nivel general, tanto el F1 macro como el F1 ponderado descienden ligeramente (de 79–80 % a 78 %), al igual que la *accuracy* global (de 80 % a 78 %). En conjunto, estos resultados indican que, aunque el impacto de los datos sintéticos sobre polaridad es más limitado que en emociones, el modelo conserva un rendimiento sólido con una ligera tendencia hacia el equilibrio en el *recall* entre clases.

Polarity	**Metric**	**Corpus Dialogues TLoUv2 (%)**	**Support**	**CorpusDialogues TLoUv2_sintetic (%)**	**Support**
Negative	Precisión Recall F1-score	82% 91% 86%	129	85% 86% 86%	137
Neutral	Precisión Recall F1-score	78% 67% 72%	103	76% 67% 71%	119
Positive	Precisión Recall F1-score	78% 80% 79%	79	70% 83% 76%	69
macro-avg	Precisión Recall F1-score	80% 79% 79%	311	77% 79% 78%	325
weighted-avg	Precisión Recall F1-score	80% 80% 80%	311	79% 78% 78%	325
Global	Accuracy	80%		78%	

Tabla 19. Resumen comparativo del reporte de clasificación para la polaridad al incluir muestras sintéticas y modelo GPT4o.

Si nos fijamos en las matrices de la Figura 37, podemos observar como la clase *Neutral* se confunde más con otras clases (*Positive,* 18 confusiones *y Negative,* 21 confusiones*)*, y que otras clases se predicen más con esta clase. En cuanto a la clase *Positive,* se confunde menos con la clase *Negative,* pero más con *Neutral,* además se generan más FP de las clases *Negative y Neutral.* La clase *Negative,* se sigue confundiendo con las otras dos clases.

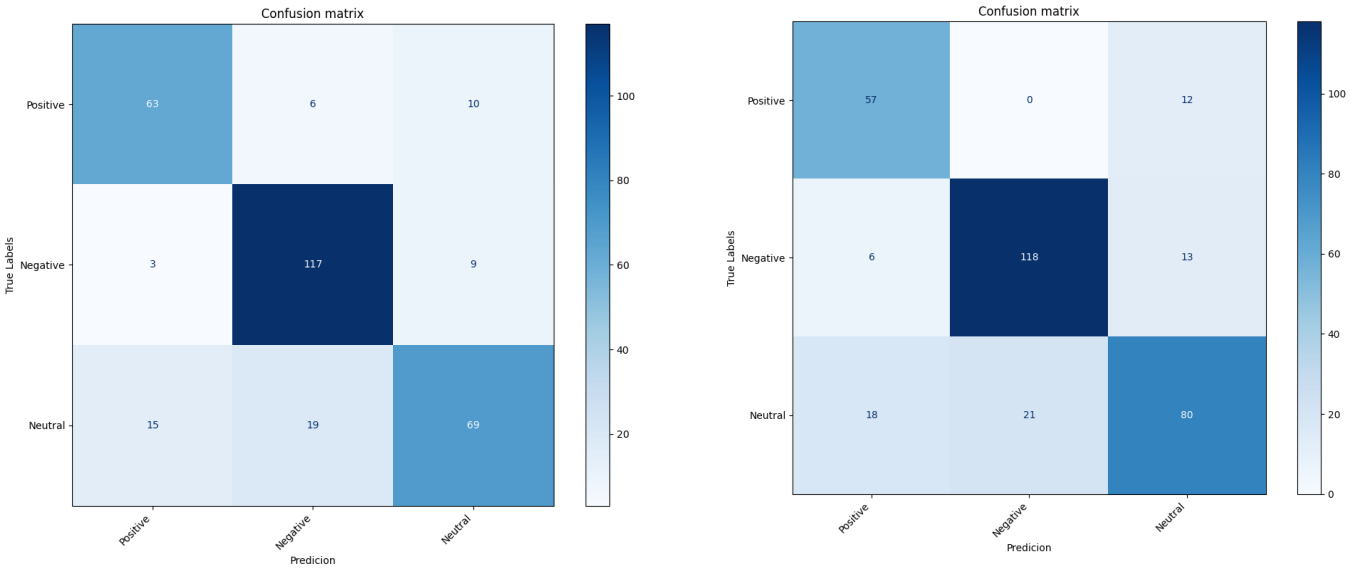

Figura 37. Matriz de confusión para polaridad v2 (izquierda), v2_sintetic (derecha)

A modo de conclusión, la integración de datos sintéticos al corpus mejoró el equilibrio general del modelo, elevando el macro *F1-score* y fortaleciendo las clases minoritarias como *Happy* y *Sad*. Aun así, introduce una ligera pérdida de precisión en algunas clases mayoritarias y una pequeña reducción de la exactitud global. Para evitar esa caída de las clases mayoritarias un posible idea es volver a analizar el reetiquetado de alguna de las clases más afectadas (*Empathy, Fear o Anger*), pues cuentan con un número de muestras suficientes para aplicar balanceo.

6.4.3 Evaluación con un mayor número de épocas

Con el fin de comprobar una posible falta de entrenamiento por parte del modelo, y ratificar o no la necesidad de reetiquetar alguna de las clases de emociones, se lanzó otra simulación empleando la última versión del corpus (*CorpusDialoguesTLoUv2_sintetic*) al que se le modificó el hiperparámetro de número de épocas (*epochs*), incrementado dicho valor a 5 épocas. Los hiperparámetros que optimizaban este modelo fueron: *epochs*=5; *Batch size*=5 y *LR multiplier*=1.8. El aumento del número de épocas suponía que el modelo vería el conjunto de datos más veces, de manera que podrá aprender las estructura de los datos mejor. Un valor de *Batch size,* al igual que modelos previos, un aumento mayor en este hiperparámetro da

estabilidad a los resultados. El resto del proceso seguido fue el mismo que los anteriores y se obtuvieron los resultados que se recogen en la Tabla 20. Aumentar de 3 a 5 épocas no mejoró el rendimiento, ya que el accuracy global bajo de 72% al 70%, la *macro-avg* de 73% a 60% y la *weighted* de 72% a 70%. En concreto, en alguna clase crítica como *Sad o Happy,* se empeoró, de 81% a 70% y de 84% al 65% respectivamente.

Emotion	Metric	CorpusDialogues TLoUv2_sintetic (%)	Support	CorpusDialogues TLoUv2_sintetic 5 epochs (%)	Support
Anger	Precisión Recall F1-score	85% 67% 75%	43	81% 81% 81%	43
Empathy/Trust/ Confidence	Precisión Recall F1-score	58% 74% 65%	66	61% 68% 64%	66
Fear	Precisión Recall F1-score	62% 75% 68%	28	62% 64% 63%	28
Happy	Precisión Recall F1-score	93% 76% 84%	17	71% 59% 65%	17
Neutral	Precisión Recall F1-score	78% 67% 72%	93	69% 75% 72%	93
Sad	Precisión Recall F1-score	84% 78% 81%	41	86% 59% 70%	41
Surprised	Precision Recall F1-score	66% 73% 69%	37	71% 68% 69%	37
macro-avg	Precisión Recall F1-score	75% 73% 73%	325	72% 68% 69%	325
weighted-avg	Precisión Recall F1-score	74% 72% 72%	325	71% 70% 70%	325
Global	Accuracy	72%		70%	

Tabla 20. Comparación de resultados al introducir cinco épocas con muestras sintéticas al modelo GPT4o-mini.

In Incluso alguna clase se confundía más con otras clases que como lo hacía con 3 épocas, tal y como se observa en la matriz de confusión de la Figura 38. Por ejemplo, *Sad*, que se confunde más con *Neutral* o *Empathy*. Muchas de las clases se predicen como *Neutral,* lo que hace que disminuya su precisión. *Empathy* sigue confundiéndose con *Neutral,* incluso se ha agravado (de 12 instancias confundidas a 13).

Con estos resultados se confirmó que la baja precisión no venía ocasionada por una falta de entrenamiento, el modelo probablemente ya había aprendido lo esencial en las primeras 3 épocas y con 5 épocas aparece un sobreentrenamiento de los datos. Al llegar a estas conclusiones no se realizó prueba con la polaridad y se ratificó la necesidad de cara al futuro de mejorar el etiquetado de las clases que no habían sido reetiquetadas en la segunda versión.

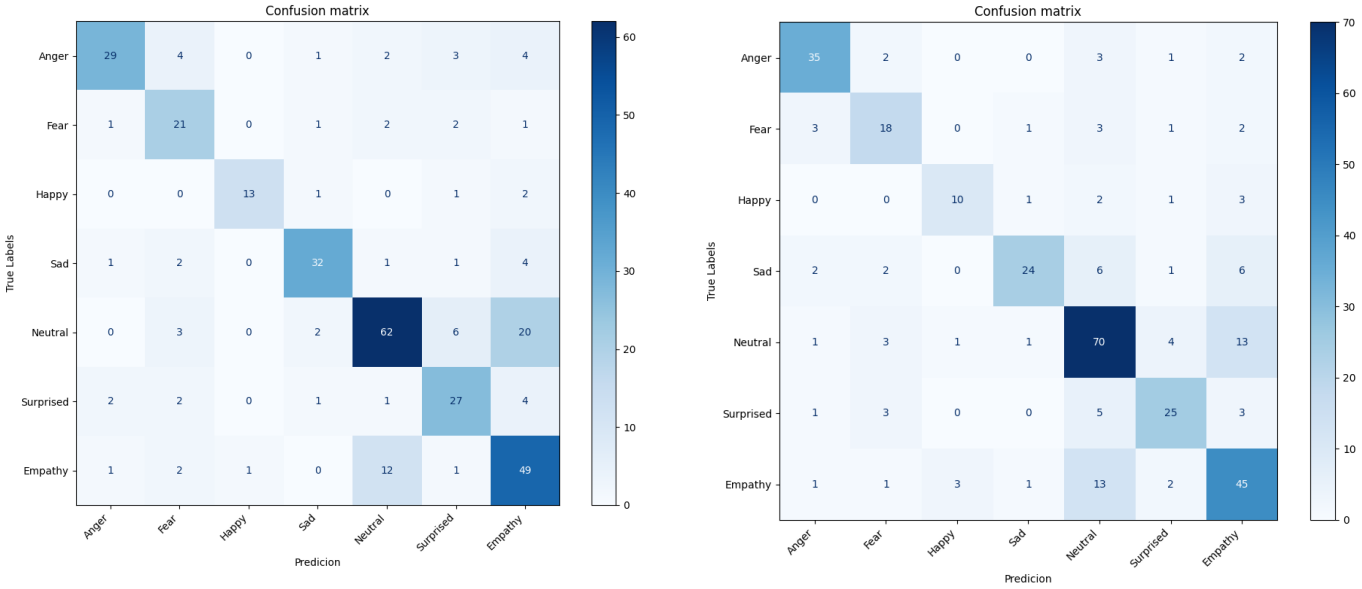

Figura 38. Matriz de confusión para polaridad v2_sintetic (izquierda), v2_sintetic_5_epcohs (derecha)

6.5 Uso del modelo para interactuar con oraciones o ficheros Excel

Una vez optimizado el moddelo GPT4o-mini para el análisis de sentimientos, se puede interactuar con este de manera similar a como se hace con chatGPT. Para interactuar con el modelo desde la plataforma de OpenAI, en la sección de "*Playground"*(recuadrado de color naranja en la Figura 5), se debe entrar en la barra lateral izquierda en la sección *"Prompts"*. Después se debe elegir qué modelo emplear, y para ello donde se indica *Model,* recuadrado de color rojo en la Figura 39, y se usará el *fine-tuned* según queramos analizar emociones o polaridad.

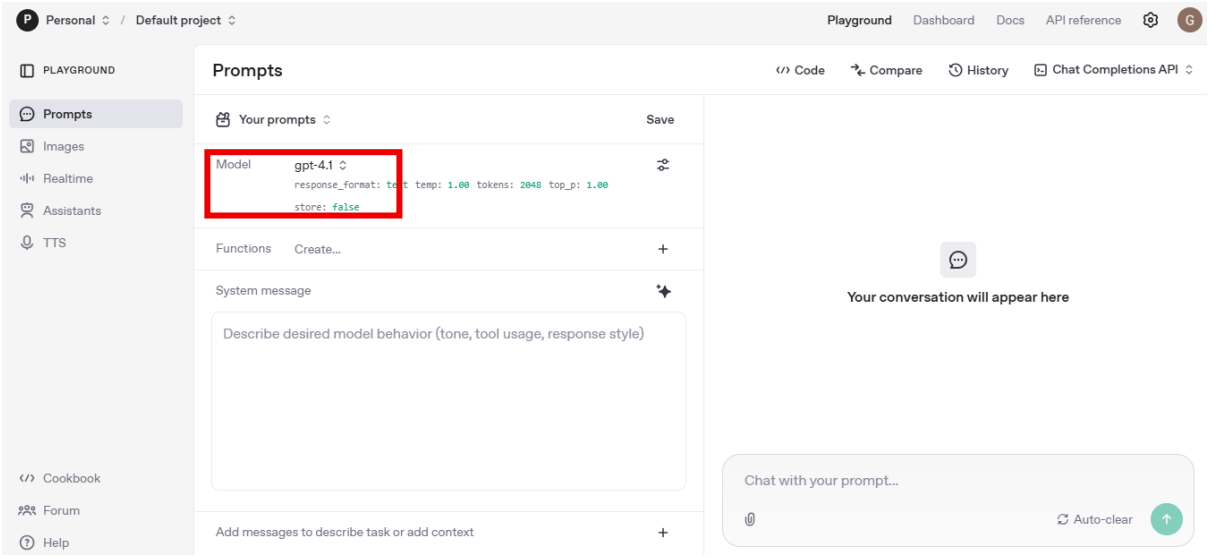

Figura 39. Uso del modelo desde la plataforma.

En la parte derecha de la Figura 39 se aprecia la sección donde ya puede interactuar con el modelo, realizado preguntas y obteniendo las respuestas pertinentes. Se debe recordar que el uso del modelo tiene un coste que hemos recogido en la fila *precio entrada /1M tokens* de la Tabla 1.

Por otro lado, también se puede usar el modelo a través de código de Python. Para el uso del modelo a través de Python se empleará el código de la Figura 40, donde se puede realizar las peticiones de análisis bien sobre emociones, escribiendo la oración manualmente o incorporando un fichero Excel, e indicando las columnas del texto y de la emoción/polaridad.

```
#Defines a function that queries a fine-tuned GPT model to predict the sentiment emotions
#('Anger', 'Fear', 'Happy', 'Sad', 'Neutral', 'Surprised' or 'Empathy') of a given input
text, then demonstrates it with an example sentence.
def predict_emotion(text):
completion = client.chat.completions.create(
model="ft:gpt-4o-mini-2024-07-18:personal::Bi***Wn"
messages=[
{"role": "system", "content":"What is the emotion of the following text? Respond with
'Anger', 'Fear', 'Happy', 'Sad', 'Neutral', 'Surprised' or 'Empathy'."},
{"role": "user", "content": text},
        ]
    )
return completion.choices[0].message.content.strip()
#Example
input_text= "Shit, Ellie died because of the fireflies."
predicted_emotion= predict_emotion(input_text)
print(predicted_emotion)

#####

#Defines a function that reads text data from an Excel file, applies a emotions prediction
to each text entry,
#and writes the results (including predictions) to a new Excel file. Ready for batch
inference with a specified column.
def predict_emotion_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_emotion)
df.to_excel(output_file, index=False)

print(output_file)
#Example
input_file="" #Write an input file
output_file="" #Write an output file where do you what all the predictions
text_column="" #Write the name of the column where all the text you want to predict are.

predict_emotion_excel(input_file, output_file, text_column)
```

Figura 40. Uso del modelo a través de Python.

De igual manera podrá emplearse este código para el análisis de la polaridad en el modelo optimizado para ello. Los cambios que deben realizarse son:

- Cambio en el modelo, para que muestre el *model_id* del modelo optimizado para la polaridad.
- Cambio en la pregunta: *"What is the polarity of the following text? Respond with 'Positive', 'Negative' or Neutral'."*.
- Cambio en la columna del Excel que se genera con las respuestas: *df['Polarity']=df[text_column].apply(predict_polarity).*

6.6 Conclusiones

El corpus ha pasado por varias etapas: desde una versión inicial con desbalanceo evidente entre clases (TLoUv1), hasta la ampliación y ajuste con TLoUv2, incluyendo técnicas de balanceo mediante datos sintéticos (TLoUv2_sintetic). Estas mejoras beneficiaron principalmente la clasificación de emociones, especialmente en clases minoritarias, *Happy y Sad.*

Posteriormente, se ajustó el número de épocas de entrenamiento de 3 a 5, observándose que no aportaba mejoras globales, lo que puede resultar que aumentar el número de épocas y alargar el entrenamiento lleve al *overfitting* o sobreentrenamiento. Con los resultados vistos, se recomienda mantener el hiperparámetro *epcohs* en el valor de 3. Como opción se podría entrenar un modelo con *early stopping,* que implica que el modelo detiene el entrenamiento automáticamente cuando el deja de mejorar en un conjunto de validación. Pero se debe recordar que nos encontramos en un modelo de OpenAI de pago, y aumentar el número de épocas implicaría un mayor coste.

También se debe de analizar en profundidad algunos de los casos con menor precisión o *recall* para comprobar si parte del problema se debe a etiquetas mal clasificadas. Estas clases serían aquellas que no se reetiquetaron de nuevo durante este TFG, en concreto *Empathy, Fear y Anger.* Además, sería recomendable explorar ajustes adicionales en polaridad, donde el impacto del balanceo fue limitado, y plantear la ampliación del corpus con nuevos datos reales para mejorar la generalización del modelo.

7
Evaluación del modelo GPT-4.1-mini con el Corpus *The Last of Us*

7.1 Introducción

De la misma manera que se ha realizado el análisis del corpus con instancias sintéticas para el modelo GPT4o-mini, ahora se va a realizar el mismo análisis del corpus para el modelo más reciente de OpenAI, esto es GPT4.1-mini, y una comparación con dicho modelo.

7.2 Cambios metodológicos o técnicos respecto al análisis anterior

El procedimiento para realizar el *fine-tuning* del modelo 4.1-mini sigue esencialmente las mismas etapas que las descritas para el modelo 4o-mini. Sin embargo, se deben considerar ciertas particularidades propias del modelo 4.1, las cuales se exponen a continuación. En concreto se han realizado los siguientes cambios:

- Cambios en nombres de proyectos de la herramienta Wandb: por ejemplo: *wandb.init(project="Polarity analysis 4.1-mini")* o *wandb.log({"predictions_vs_true_labels_polarity-4.1-mini": wandb.Table(dataframe=df_results)}).*
- Cambio en el modelo a entrenar. Para ello podemos emplear el buscador de modelos que se recoge en la Figura 41, una vez ejecutado se mostrará un listado de modelos como el de la Figura 42.

```
#This code retrieves all available models from the OpenAI client and filters
the list to show only those with "gpt-4.1" in their model ID.
models=client.models.list()
gpt4_1_models = [m for m in models if "gpt-4.1" in m.id]
gpt4_1_models
```

Figura 41. Uso del fltro de modelos para observar el identificador del modelo GPT4.1.

Si indicamos que queremos buscar aquellos modelos que coincidan con los modelos 4.1. se obtiene el listado que se muestra en la Figura 42.

```
[
Model(id='gpt-4.1-nano', created=1744321707, object='model', owned_by='system'),
Model(id='gpt-4.1-2025-04-14', created=1744315746, object='model', owned_by='system'),
Model(id='gpt-4.1', created=1744316542, object='model', owned_by='system'),
Model(id='gpt-4.1-mini-2025-04-14', created=1744317547, object='model', owned_by='system'),
Model(id='gpt-4.1-mini', created=1744318173, object='model', owned_by='system'),
Model(id='gpt-4.1-nano-2025-04-14', created=1744321025, object='model', owned_by='system')
]
```

Figura 42. Listado de modelos de GPT4.1.

Usando finalmente el modelo 'gpt-4.1-mini-2025-04-14'. Podemos lazar la orden de ajustar el modelo para los datos de entrenamiento y validación, a través del código de la Figura 43.

```
#This code initiates a fine-tuning job on the "gpt-4.1-mini" model
#using uploaded training and validation files, sets hyperparameters,
#integrates with Weights & Biases, and prints job details and ID.
ft_job = client.fine_tuning.jobs.create(
training_file=training_file_id,
validation_file=validate_file_id,
model="gpt-4.1-mini-2025-04-14",
hyperparameters={
„n_epochs": 3
    },
integrations=[
        {
„type": „wandb",
„wandb": {
"project": wandb.run.project,
"name": "4.1-mini-model-polarity"
            }
        }
    ]
)

model_id=ft_job.fine_tuned_model
print("object of the created model ", ft_job)
print("Fine Tune Job has been created with id ", ft_job.id)
```

Figura 43. Ajuste del modelo GPT4.1-mini.

- Límites de velocidad TPM, en esta ocasión y como se recogían en la Tabla 1, el modelo GPT4.1-mini tiene la misma limitación de velocidad: 200.000 TPM.
- Cambio en las variables del modelo a emplear una vez ajustado. Se debe modificar las variables del modelo a usar, una vez que ya ha obtenido la variable *model_id*.

7.3 Evaluación comparativa del desempeño de GPT-4.1-mini frente a GPT-4o-mini

Llevado a cabo el mismo procedimiento que para el modelo de GPT4o-mini y con el corpus final con datos sintéticos, se obtiene un modelo con hiperparámetros: *epochs*=3; *Batch size*=3 y *LR multiplier*=2. Al igual que lo comentado con anterioridad, un aumento de *Batch size* supone una mayor estabilidad del modelo, respecto de los iniciales (*epochs*=3; *Batch size*=2 y *LR multiplier*=1.8). Además, se ha incrementado el *learning rate multiplier* a un valor de 2, lo que implica que el modelo actualizará los pesos más rápido. El efecto de *batch size* puede compensar la posible inestabilidad de entrenar más rápido del LR.

El reporte de clasificación se ha resumido de manera comparativa en las Tablas 21 y 22. En la Tabla 21, para emociones, se puede observar como el rendimiento global se ha reducido en dos puntos porcentuales (de 72% al 70%), de igual manera lo ha hecho la *macro-avg* (del 73% al 70%) y *weighted-avg* (de 72% al 70%), por lo que este modelo de GPT no predice mejor las emociones del corpus cuando se han incluido instancias sintéticas. En cuanto al comportamiento de cada emoción, muchas de ellas mantienen los mismos resultados, en concreto *Empathy, Surprised o Fear.* Destaca la caída de precisión de *Anger* (85% a 75%), aunque conlleva un aumento de la métrica de *recall* (de 67% a 77%), lo que implica que otras clases son calificadas como *Anger,* y se generan más falsos positivos.

Emotion	Metric	CorpusDialogues TLoUv2_sintetic GPT4o-mini (%)	Suport	CorpusDialogues TLoUv2_sintetic GPT4.1-mini (%)	Support
Anger	Precisión Recall F1-score	85% 67% 75%	43	75% 77% 76%	43
Empathy/Trust/ Confidence	Precisión Recall F1-score	58% 74% 65%	66	63% 67% 65%	66
Fear	Precisión Recall F1-score	62% 75% 68%	28	70% 68% 69%	28
Happy	Precisión Recall F1-score	93% 76% 84%	17	71% 71% 71%	17
Neutral	Precisión Recall F1-score	78% 67% 72%	93	71% 69% 70%	93
Sad	Precisión Recall F1-score	84% 78% 81%	41	74% 71% 72%	41
Surprised	Precision Recall	66% 73%	37	68% 70%	37

	F1-score	69%		69%	
macro-avg	Precisión Recall F1-score	75% 73% 73%	325	70% 70% 70%	325
weighted-avg	Precisión Recall F1-score	74% 72% 72%	325	70% 70% 70%	325
Global	Accuracy	72%		70%	

Tabla 21. Comparación de resultados en emociones GPT4o-mini y GPT4.1, con muestras sintéticas.

En la comparación de matrices de confusión de la Figura 44, la diagonal principal ha empeorado de manera general. Se detecta un aumento puntual en las confusiones entre clases, concretamente en la clasificación de la clase *Neutral*, que anteriormente no se predecía como *Anger* y que ahora aparece confundida en tres ocasiones. Aun así, se ha logrado mejorar la identificación de la clase *Anger*, aumentando su *recall* y reduciendo los errores con el resto de las categorías.

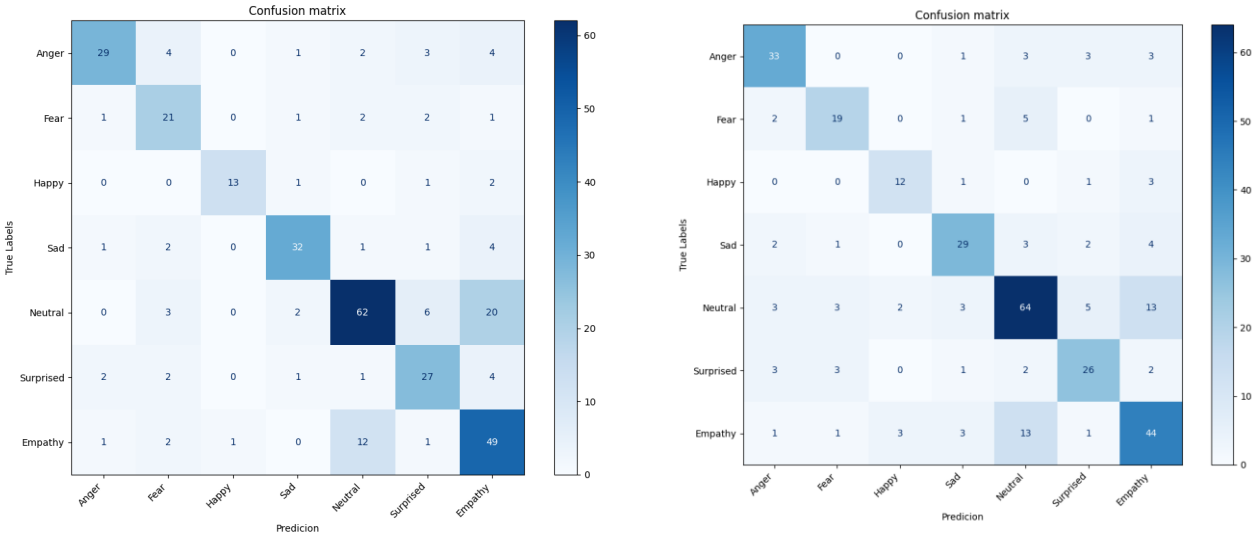

Figura 44. Matriz de confusión para polaridad, GPT4o-mini(izquierda), GPT4.1-mini(derecha).

Si se analiza los resultados comparativos de la Tabla 22 de ambos modelos para la polaridad, se observa una mejoría global en la precisión (de 78% a 82%), de igual manera lo hace *weighted-avg* y la métrica *macro-avg* aumentó de 78% a 81%.

Polarity	**Metric**	**Corpus Dialogues TLoUv2 (%)**	**Support**	**CorpusDialogues TLoUv2_sintetic (%)**	**Support**
Negative	Precisión Recall F1-score	85% 86% 86%	137	86% 88% 87%	137
Neutral	Precisión Recall F1-score	76% 67% 71%	119	79% 74% 77%	119
Positive	Precisión Recall F1-score	70% 83% 76%	69	77% 83% 80%	69

macro-avg	Precisión Recall F1-score	77% 79% 78%	325	81% 82% 81%	325
weighted-avg	Precisión Recall F1-score	79% 78% 78%	325	82% 82% 82%	325
Global	Accuracy	78%		82%	

Tabla 22. Resumen comparativo del reporte de clasificación para GPT4o-mini y GPT4.1-mini para la polaridad.

Todas las clases mejoran, pero especialmente la clase *Negative* se refuerza como clase fuerte al aumentar su *F1-score* de 86% a 87%, reduciendo las confusiones con otras clases y reduciendo las clases que se confunden con *Negative*, tal y como se observa en la matriz de confusión de la Figura 45. La mayor subida es para la clase *Neutral* que aumenta su *F1-score* en seis puntos porcentuales, debido a un aumento desde el 67% al 74% de *recall,* que supone que más instancias son clasificadas correctamente como *Neutral* (118 a 121 instancias) (Figura 45). *Positive* ha mejorado, las otras clases ya no se predicen erróneamente con la clase *Positive.* En resumen, todas las clases mejoran y existe un aumento generalizado de la diagonal principal.

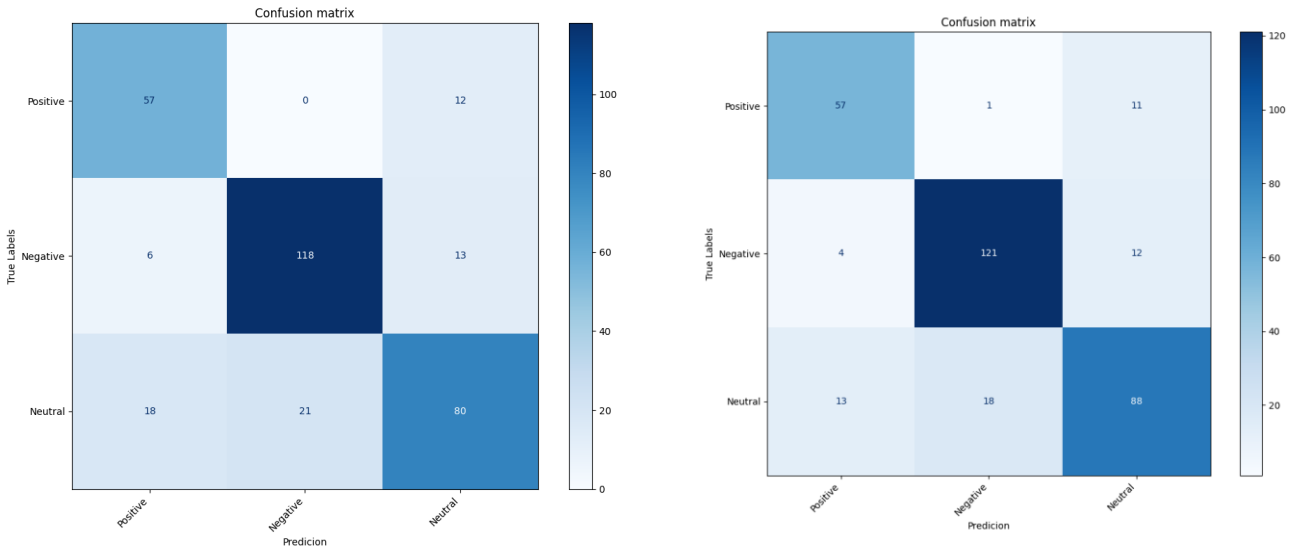

Figura 45. Matriz de confusión para polaridad GPT4o-mini (izquierda), GPT4.1-mini (derecha)

7.4 Conclusiones

Al evaluar el corpus con las versiones más actuales de los modelos GPT (GPT4.1-mini), se observa una disminución ligera de las métricas de emociones respecto de GPT4o-mini. Esta caída, afecta por completo a las clases minoritarias, *Happy* y *Sad,* que fueron reforzadas con muestras sintéticas. Los resultados muestran que GPT-4.0-mini obtiene mejores métricas de precisión y *F1-score.* Por el contrario, el modelo se comporta mejor con la polaridad, mejorando todas las clases.

Esto sugiere que el modelo GPT4.1-mini pueda ser empleado para la predicción de polaridades, ofreciendo un mejor rendimiento. Sin embargo, para el análisis de emociones más complejas y clases desbalanceadas es preferible entrenar modelos anteriores. Según el informe técnico de OpenAI (2025, 14 de abril. *Para GPT-4.1 technical report*), GPT-4.1-mini fue optimizado para tareas que requieren procesamiento de contexto largo, razonamiento estructurado y codificación, lo que puede afectar al desempeño en interacciones sobre diálogos naturales, donde modelos como GPT-4o-mini, orientados a lenguaje casual y multimodalidad, resultan más adecuados.

8
Conclusiones y Líneas futuras

8.1 Conclusiones

El proceso de elaboración del corpus de diálogos de *The Last of Us* ha ido evolucionado durante este Trabajo Fin de Grado, desde un corpus inicial desbalanceado, que dificultaba el entrenamiento de un modelo preciso, en concreto para las clases *Happy* y *Sad.* Para pasar a su mejora, se realizaron ampliaciones del corpus y balanceo de las clases minoritarias con datos sintéticos. Estas técnicas han resultado ser útiles para aumentar la representatividad de las clases minoritarias y todo ello sin hacer que el resto de las clases se vea muy perjudicada. Sin embargo, clases como *Neutral*, *Fear* o *Empathy* muestran resultados mejorables.

Los entrenamientos del modelo GPT4o-mini han ido mostrando alguna mejora progresiva tras la ampliación del corpus. Inicialmente el modelo presentaba problemas con las clases *Happy* y *Sad,* pero se logró mejorar los resultados (*F1-score* de *Happy* mejoró 24 puntos porcentuales y de *Sad* 10 puntos porcentuales). Sin embargo, la incorporación de muestras sintéticas hizo que la precisión global del modelo bajara 1 punto porcentual para emociones y 2 para la polaridad, a cambio de la mejora en las clases minoritarias.

A lo largo del proyecto se ha realizado diferentes pruebas con cambios en el hiperparámetro *epochs* y uso de los modelos más actuales (GPT4.1-mini). Esto nos ha permitido realizar comprobaciones sobre las posibles causas de los resultados no tan favorables de los primeros modelos. Al probar con más épocas se percató de un problema de *overfitting*, y al usar nuevas generaciones del modelo GPT4.1-mini, se encontró el modelo ajustado con mejores resultados hasta ese momento para la polaridad.

Estos resultados deben entenderse dentro del contexto del videojuego *The Last of Us*. Los mensajes el videojuego tienen una característica especial y diferente de los diálogos reales o los diálogos que realizan los jugadores en otras plataformas o redes sociales. Estas conversaciones e interacciones reales tienen como finalidad transmitir información, expresar emociones o entablar relaciones afectivas; sin embargo, el diálogo de los videojuegos tiene un

enfoque a la muestra de indicaciones, pistas, pasos a seguir, contexto y relaciones entre personajes. Ejemplos como *"Get that door open"* o *"Go around to the other side"* muestran este comportamiento que afecta al balanceo de clases y complican la clasificación al existir instancias mayoritariamente informativas neutras muy genéricas y carecer de diálogos cargados de emociones. Una complicación añadida es el contexto apocalíptico en el que se enmarca el videojuego que hace que emociones como la ira o el miedo sean las más abundantes, mientras que emociones como la tristeza o la felicidad aparezcan menos.

Este contexto afecta tanto a la interpretación lingüística como a las implicaciones sociales. La violencia puede insensibilizar y moldear la forma de pensar, primero, y de actuar, después, de los jugadores, especialmente jóvenes. Es por este motivo por el que es importante el estudio del lenguaje en videojuegos a través del PLN y la IA. Con ello se logra entender que el lenguaje del juego también repercute en la sociedad a través del comportamiento de los jugadores y permite ofrecer herramientas a los desarrolladores de videojuegos para mitigar los posibles efectos negativos. El fin es potenciar la industria para que sea una herramienta de cohesión social, educación, que apoye la salud mental y la innovación tecnológica responsable.

8.2 Líneas futuras

Como líneas futuras se plantea utilizar técnicas para mejorar los resultados obtenidos en este estudio y ampliar el dataset:

- Revisar el etiquetado de algunas clases como *Empathy, Fear y Anger*, que no fueron revisadas en la segunda versión del corpus y que después han mostrado resultados no muy favorables.
- Ampliar el corpus con más datos de la segunda parte del videojuego o incluso de la serie de HBO, para mejorar las clases con peores resultados y balancear el resto de las clases.
- Mejorar el rendimiento de los modelos con el uso de *early stopping*, teniendo en cuenta el posible coste que supondría. Con la evolución de los modelos es posible que el coste de entrenamiento y de uso minoren y sea posible emplear *early stpping*.
- Explorar otros modelos de procesamiento de lenguaje natural, como BERT que se puedan ajustar mejor a contextos narrativos y ofrecer una mayor capacidad para predecir datos nunca vistos.

Referencias

Anderson, K. A. (2022). Moral distress in The Last of Us: Moral agency, character realism, and navigating fixed gaming narratives. Computers in Human Behavior Reports, 5, 100163. https://doi.org/10.1016/j.chbr.2021.100163

Artificial Analysis. (16 abril. 2025.). Model & API providers analysis. Disponible en https://artificialanalysis.ai

Bagnato, J. I. (s.f.). Clasificación con datos desbalanceados. Aprende Machine Learning. https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. In Advances in Neural Information Processing Systems, 33, 1877–1901. https://doi.org/10.48550/arXiv.2005.14165

Brownlee, J. (2020, 6 de marzo). SMOTE for imbalanced classification with Python. Machine Learning Mastery. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Carralero Lanchares, D. M. (2025). Aplicación de aprendizaje automático para evaluar lenguaje de videojuegos en Twitch (Trabajo Fin de Grado). Universidad de Valladolid. Disponible en https://uvadoc.uva.es/handle/10324/71264

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of NAACL-HLT 2019 (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.48550/arXiv.1810.04805

Diego C. (2024, 21 de enero). Introducción a la API de OpenAI y GPT con Python. Medium. https://medium.com/@diego.coder/introducci%C3%B3n-a-la-api-de-openai-y-gpt-con-python-fbcc219a44f4

Fandom. (s.f.). Categoría: Personajes – The Last of Us Wiki. https://thelastofus.fandom.com/es/wiki/Categor%C3%ADa:Personajes

Fandom. (s.f.). The Last of Us. https://thelastofus.fandom.com/es/wiki/The_Last_of_Us

Game Scripts Wiki. (2020, octubre). The Last of Us Part II full transcript [Transcripción]. https://game-scripts-wiki.blogspot.com/2020/10/the-last-of-us-part-ii-full-transcript.html

Gee, J.P(2003). What video games have to teach us about learning and literacy. Palgrave Macmillan. Extracto disponible en https://sobrief.com/books/what-video-games-have-to-teach-us-about-learning-and-literacy

Hämäläinen, M., Alnajjar, K., & Poibeau, T. (2022). Video games as a corpus: Sentiment analysis using Fallout New Vegas dialog. En Proceedings of the 17th International Conference on the Foundations of Digital Games (FDG '22). Association for Computing Machinery. https://doi.org/10.1145/3555858.3555930

Karimova, H. (2025, 9 de junio). The Emotion Wheel: What It Is and How to Use It. PositivePsychology.com. https://positivepsychology.com/emotion-wheel/

Lloria, A. (2025, 29 de mayo). No es muy habitual que un juego de hace 5 años sume ventas de seis cifras en tres semanas, 3DJuegos. https://goo.su/cCOtU3

Martín García, L. (2024). Análisis de sentimientos en Instagram usando ChatGPT (Trabajo de fin de grado, Universidad de Valladolid). Universidad de Valladolid. Disponible en https://uvadoc.uva.es/handle/10324/71261

Merayo, N., Cotelo, R., Carralero, M., Pinto-Ríos, J., & Carratalá-Sáez, R. (s.f.). Cutting-edge sentiment analysis in gaming: BERT models for Twitch chats. [Manuscrito no publicado].

Motion Picture Association – Canada. (2023, noviembre). Economic impacts of The Last of Us season one in Alberta: A report for the Motion Picture Association – Canada.

OpenAI. (2023). model.py (Versión 4560ª88) [Archivo fuente]. GitHub. Disponible en https://github.com/openai/tiktoken/blob/4560a8896f5fb1d35c6f8fd6eee0399f9a1a27ca/tiktoken/model.py#L26-L30

OpenAI. (2024). GPT: Generative Pre-trained Transformer. Recuperado de https://openai.com/research/gpt

OpenAI. (2025, 16 abril). *Models' documentation*. OpenAI. https://platform.openai.com/docs/models

OpenAI. (2025, 16 mayo). Organization usage limits. OpenAI Platform. Disponible en https://platform.openai.com/settings/organization/limits

OpenAI. (2025, 19 marzo). Model selection guide. OpenAI. https://platform.openai.com/docs/guides/model-selection

OpenAI. (2025, abril 14). GPT-4.1 technical report.OpenAI. https://openai.com/index/gpt-4-1

OpenAI. (s. f.). Fine-tuning. OpenAI Platform. Disponible en https://platform.openai.com/docs/guides/fine-tuning

OpenAI. (s.f.). API-reference. OpenAI Platform. Recuperado el 17 de mayo de 2025 Disponible en https://platform.openai.com/docs/api-reference/introduction

Pandas development team. (s.f.). About pandas. https://pandas.pydata.org/about/index.html

Pedregosa, F., Varoquaux, G., Gramfort. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12.

Project Jupyter. (s.f.). About Project Jupyter. https://jupyter.org/about

Python Software Foundation. (2024). Documentación de la biblioteca estándar de Python 3.13 (versión 3.13). https://docs.python.org/es/3.13/library/

Python Software Foundation. (s. f.). About Python. https://www.python.org/about/

Redditor u/OP. (2023, 7 de noviembre). What impact did The Last of Us have on you? [Mensaje en un foro]. Reddit. https://www.reddit.com/r/thelastofus/comments/17pzffa/what_impact_did_the_last_of_us_have_on_you/?tl=es-419

Rico Sanguino, A., Perona Jiménez, J., y Sánchez Piñeiro, M. (28-29 de enero de 2025). Aprovechando el poder de la IA generativa [Presentación de curso]. NTTDATA.

Scikit-learn developers. (2025). sklearn.metrics.precision_score. Scikit-learn 1.6.1 documentation. Disponible en https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html .

Shotgunnova. (2013). The Last of Us – Game Script [Guía]. GameFAQs. https://gamefaqs.gamespot.com/ps3/652686-the-last-of-us/faqs/68485

Tarekgh. (2025, 14 de abril). Add GPT-4.1 support [Incidencia nº 395]. GitHub. https://github.com/openai/tiktoken/issues/395

V7 Labs. (2024). Train Test Validation Split: How To & Best Practices. https://www.v7labs.com/blog/train-validation-test-set

Vandal Team. (2022, 2 de septiembre). Análisis de The Last of Us Parte I para PS5. Vandal. https://vandal.elespanol.com/analisis/ps5/the-last-of-us-parte-i/120665

Weights & Biases. (s.f.). Quickstart. Recuperado el 17 de mayo de 2025. Disponible en https://docs.wandb.ai/quickstart/

Weights and Biases. (s. f.). About Weights and Biases. https://wandb.ai/site/company/about-us/

Wikipedia. (2025, 11 de julio). The Last of Us. https://es.wikipedia.org/wiki/The_Last_of_Us

Wu, M. y Fishman, S. (s. f.). Data preparation and analysis for chat model fine-tuning. OpenAI Cookbook. Disponible en https://cookbook.openai.com/examples/chat_finetuning_data_prep

Anexo 1: Código empleado

Código empleado para obtener las estadísticas de polaridad y emoción en la separación en ficheros de entrenamiento, validación y test.

```
#This code computes and prints polarity label counts for training, validation, and test JSONL files.
filenames = [
“training_Polaridad.jsonl”,
“validation_Polaridad.jsonl”,
“test_Polaridad.jsonl”
]
for filename in filenames:
counts = {
“Positive”: 0,
“Negative”: 0,
“Neutral”: 0
    }
total = 0
with open(filename, “r”, encoding=”utf-8”) as f:
for line in f:
line = line.strip()
if not line:
continue
data = json.loads(line)
messages = data.get(“messages”, [])
assistant_msgs = [m for m in messages if m.get(“role”) == “assistant”]
if assistant_msgs:
polarity = assistant_msgs[-1].get(“content”)
if polarity in counts:
counts[polarity] += 1
total += 1

print(f”Polarity statistics for {filename}:”)
print(f”Total examples: {total}”)
for polarity, count in counts.items():
print(f”{polarity}: {count}”)
print(“-“ * 40)


#This script counts and displays the distribution of emotion labels in training, validation, and test
JSONL files.
filenames = [
“training_Emocion.jsonl”, “validation_Emocion.jsonl”,”test_Emocion.jsonl”
]
for filename in filenames:
counts = defaultdict(int)
total = 0

with open(filename, “r”, encoding=”utf-8”) as f:
for line in f:
line = line.strip()
if not line:
continue
data = json.loads(line)
messages = data.get(“messages”, [])
assistant_msgs = [m for m in messages if m.get(“role”) == “assistant”]
if assistant_msgs:
emotion = assistant_msgs[-1].get(“content”)
if emotion:
counts[emotion] += 1
total += 1

print(f”Emotion statistics for {filename}:”)
print(f”Total examples: {total}”)
for emotion, count in counts.items():
print(f”{emotion}: {count}”)
print(“-“ * 40)
```

Anexo 2: Parte del corpus analizado

CHAPTER	CHARACTER	GENDER	TEXT	Polarity	Emotions
Winter - Lakeside Resort	Hunter 2	M	I just want to finish up and go home. I'm freezing my ass off	Negative	Anger
Winter - Lakeside Resort	Hunter 2	M	What are you waiting for? Shoot her!	Negative	Anger
Prologue - Hometown	SARAH	F	"Where the hell are you? Call me!," "On my way"...	Negative	Anger
Summer - The Suburbs	SAM	M	"Will shoot on sight." Lots of friendly people lived here	Negative	Sad
SEATTLE DAY 1	Ellie	F	(finds a drawing) Damn... That's pretty good. (making a note) Jotting this down. (finds a FEDRA scanner) Oof. Haven't seen one of these in a while. Looks like that's everything over here.	Positive	Happy
THE FARM	Ellie	F	(hugs him) Hey. It's good to see you.	Positive	Happy
4 YEARS LATER	Ellie	F	(sighs) Smells good.	Positive	Happy
Winter - Lakeside Resort	ELLIE	F	...and so are you. Right there. Roll up my sleeve. Look at it!	Neutral	Neutral
Fall - The University	VOICE	Unknown	...fucking thing was a giant waste of ti—	Negative	Anger
Fall - The University	VOICE	Unknown	...looking for the others, they've all returned to Saint Mary's Hospital in Salt Lake City. You'll find them there, still trying to save the world. Good luck with that	Negative	Sad
Summer - Pittsburgh	SAM	M	A blueberry hurt you?	Neutral	Surprised
Summer - The Suburbs	ELLIE	F	A what?	Neutral	Surprised
Winter - Lakeside Resort	DAVID	M	A what?!	Neutral	Surprised
Fall - Tommy's Dam	ELLIE	F	A...hydra who?	Neutral	Surprised
	Sintetic		A sunny day and no clickers? Best. Day. Ever	Positive	Happy
Fall - Tommy's Dam	MARIA	F	Absolutely not. You tell him to go find somebody else	Negative	Anger
Epilogue - Jackson	JOEL	M	Actually kinda pretty, ain't it?	Positive	Neutral
Summer - Bill's Town	JOEL	M	Actually might work	Positive	Empathy
Fall - The University	JOEL	M	Adios, little brother. C'mon	Positive	Empathy
	Sintetic		All that talk about rebuilding — it's just noise now, empty and tired	Negative	Sad