UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Análisis de Emociones en la Salud Mental mediante Modelos BERT: Un Enfoque Basado en Procesamiento del Lenguaje Natural

Autor:
Dña. Sofía González Hurtado
Tutor:
Dña. Noemí Merayo Álvarez

TÍTULO: Análisis de Emociones en la Salud Mental mediante Modelos BERT: Un Enfoque Basado en Procesamiento del Lenguaje Natural
AUTOR: **Dña. Sofía González Hurtado**
TUTOR: **Dña. Noemí Merayo Álvarez**
DEPARTAMENTO: **Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL
PRESIDENTE: Noemí Merayo Álvarez
SECRETARIO:
VOCAL:
SUPLENTE:
SUPLENTE:

FECHA:
CALIFICACIÓN:

Resumen de TFG

Este trabajo aborda el análisis automático de polaridad y emociones en textos extraóídos de redes sociales relacionados con la salud mental. Para ello, se ha utilizado un modelo preentrenado basado en la arquitectura DistilBERT, una versión simplificada y más eficiente de BERT. Su integración con la librería SimpleTransformers ha permitido una configuración más accesible y organizada del proceso de entrenamiento y evaluación. El objetivo principal fue adaptar el modelo al dominio de la salud mental y examinar su rendimiento en los procesos de clasificación de polaridad (positiva, negativa e indeterminada) y de emociones (amor/admiración, gratitud, comprensión/empatía/ identificación, tristeza/pena, enfado/desprecio/burla e indeterminado).

Para ello, se utilizó un corpus desequilibrado, lo que ha motivado la aplicación de varias técnicas de ponderación de clases para ajustar esta distribución desigual. También se ha llevado a cabo una búsqueda sistemática de hiperparámetros mediante barridos aleatorios (*random sweeps*) informando configuraciones clave para la tasa de aprendizaje, el tamaño del lote y el número de épocas para cada experimento.

Los resultados obtenidos indican un rendimiento competitivo del modelo, destacando la efectividad de las técnicas de balanceo en la tarea de emociones, donde las puntuaciones F1 en clases minoritarias aumentan considerablemente. En polaridad, el balanceo no siempre es beneficioso, especialmente en clases ambiguas como "*Indeterminado*". En general, el trabajo demuestra que los modelos de lenguaje preentrenados pueden usarse para identificar emociones en textos de salud mental y destaca la importancia de la selección de técnicas de ajuste en contextos con fuerte desbalance de clases.

Palabras clave

Salud mental, análisis emocional, polaridad, redes sociales, DistilBERT, SimpleTransformers, Procesamiento del Lenguaje Natural, balanceo de clases, Weights & Biases.

Abstract

This work focuses on the automatic analysis of polarity and emotions in texts extracted from social media related to mental health. To this end, a pre-trained model based on the DistilBERT architecture — a simplified and more efficient version of BERT — was used. Its integration with the SimpleTransformers library enabled a more accessible and organized configuration of the training and evaluation process. The main goal was to adapt the model to the mental health domain and assess its performance in polarity classification (positive, negative, and neutral) and emotion classification (love/admiration, gratitude, understanding/empathy/identification, sadness/grief, anger/contempt/mockery, and neutral).

An imbalanced corpus was used, prompting the application of various class weighting techniques to compensate for the unequal distribution. Additionally, a systematic hyperparameter search was conducted through random sweeps, allowing for the identification of optimal learning rates, batch sizes, and number of epochs for each experiment.

The results show competitive performance of the model, highlighting the effectiveness of the balancing techniques in the emotion classification task, where F1-scores for minority classes improved significantly. In the polarity task, however, the benefits of class weighting were less evident, especially for ambiguous classes such as “Neutral.” Overall, this work demonstrates the potential of pre-trained language models for identifying emotions in mental health texts and emphasizes the importance of selecting appropriate adjustment techniques when dealing with highly imbalanced classes.

Keywords

Mental health, emotion analysis, polarity, social media, DistilBERT, SimpleTransformers, Natural Language Processing, class balancing, Weights & Biases.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que me han acompañado y apoyado a lo largo de este camino.

En primer lugar, a mi tutora, Noemí Merayo Álvarez, por su implicación, su orientación constante y su confianza en este proyecto desde el inicio. Su experiencia y disponibilidad han sido clave para poder llevar este trabajo a buen término.

Agradezco profundamente a mis padres, por ser siempre mi mayor apoyo, por enseñarme el valor del esfuerzo y por estar presentes en cada etapa, incluso en las más difíciles. También a Anto, mi pareja, por su paciencia, ánimo incondicional y por creer en mí incluso cuando yo no lo hacía.

A mis amigos y compañeros de carrera, por la ayuda y el esfuerzo mutuo, que han facilitado el camino y hecho de estos años una experiencia inolvidable. En especial, quiero destacar a Felipe, por ser una guía constante y un ejemplo durante toda la carrera. Gracias por tu generosidad, tu claridad y por estar siempre dispuesto a ayudar.

A todos, gracias por formar parte de este proceso y por hacerlo más llevadero, más enriquecedor y, sobre todo, más humano.

Índice

Agradecimientos ..... **v**

Índice ..... **vi**

Índice de figuras ..... **viii**

Índice de tablas ..... **x**

Índice de ecuaciones ..... **xi**

1 Introducción **1**

1.1 Motivación 1

1.2 Objetivos 2

1.2.1 Objetivo principal 2

1.2.2 Objetivos específicos 2

1.3 Metodología 3

1.3.1 Fase de documentación 3

1.3.2 Fase de análisis 4

1.3.3 Fase de prueba 4

1.3.4 Fase de escritura 4

1.4 Estructura de la memoria 5

2 Estado del arte **7**

2.1 Introducción 7

2.2 Marco teórico sobre SimpleTransformers 7

2.3 La salud mental en redes sociales 8

3 Herramientas utilizadas **10**

3.1 Introducción 10

3.2 Python 10

3.3 Librerías empleadas 11
3.4 Google Colab 12
3.5 Wandb 12
3.6 Métricas de rendimiento 13
4 Análisis de la respuesta emocional con DistilBERT 15
4.1 Introducción 15
4.2 Análisis descriptivo del dataset 16
4.3 Modelo SimpleTransformers utilizado: DistilBERT 19
4.4 Preprocesamiento de datos 21
4.5 Balanceo de clases 24
4.5.1 Desbalanceo en el dataset: distribución de clases y problemas asociados 24
4.5.2 Técnicas aplicadas 24
4.5.3 Otras técnicas alternativas no aplicadas 29
4.6 Tokenización con DistilBERT 30
4.7 Elección de hiperparámetros del modelo 30
4.7.1 Configuración del barrido de hiperparámetros 31
4.7.2 Visualización y selección de los parámetros óptimos en W&B 34
4.7.3 Ejecuciones de sweep realizadas y resumen de configuraciones óptimas 37
4.8 Entrenamiento y evaluación del modelo 42
4.8.1 Análisis de los resultados para emociones 45
4.8.2 Análisis de los resultados para polaridad 52
4.9 Comparativa con otros modelos BERT (RoBERTuito) 57
4.10 Coste del proyecto 60
5 Conclusiones y líneas futuras 62
5.1 Conclusiones 62
5.2 Líneas futuras 63
6 Bibliografía 64

Índice de figuras

Figura 1: Conjunto de datos. 16

Figura 2: Distribución de las clases de emociones. 18

Figura 3: Distribución de las clases de polaridad. 19

Figura 4: Fragmento de código encargado de la preparación del conjunto de datos y la importación de las librerías necesarias. 21

Figura 5: Fragmento de código que carga y codifica las etiquetas de emociones en formato numérico. 22

Figura 6: Fragmento de código para la división estratificada del conjunto de datos. 22

Figura 7: Código de limpieza de texto aplicado a los comentarios. 23

Figura 8: Configuración e inicialización del sweep aleatorio. 33

Figura 9: Código para lanzar el sweep que indica los hiperparámetros óptimos. 34

Figura 10: Pestaña Home de Wandb. 35

Figura 11: Pestaña de la simulación “bs3zsg3l”, dentro del proyecto de “EMOCIONES”. 36

Figura 12: Gráficas de la simulación “bs3zsg3l”, dentro del proyecto de “EMOCIONES”. 36

Figura 13: Gráfica de hiperparámetros para el proyecto "EMOCIONES". 37

Figura 14: Hiperparámetros óptimos para el proyecto "EMOCIONES". 38

Figura 15: Hiperparámetros óptimos para emociones con balanceo 1. 39

Figura 16: Hiperparámetros óptimos para emociones con balanceo 2. 39

Figura 17: Hiperparámetros óptimos para polaridad sin balanceo. 40

Figura 18: Hiperparámetros óptimos para polaridad con balanceo 1. 40

Figura 19: Hiperparámetros óptimos para polaridad con balanceo 2. 41

Figura 20: Código que define los mejores parámetros para emociones sin class weighting. 43

Figura 21: Función llamada compute_metrics. 43

Figura 22: Código que inicializa y entrena el modelo. 44

Figura 23: Función EvaluateModel. 45

Figura 24: Matriz de confusión para emociones sin balanceo. 46

Figura 25: Métricas de evaluación e informe de clasificación. 46

Figura 26: Matriz de confusión para emociones con balanceo 1. 48

Figura 27: Métricas de evaluación e informe de clasificación para emociones con balanceo 1. 48

Figura 28:Matriz de confusión para emociones con balanceo 2. 50

Figura 29: Métricas de evaluación e informe de clasificación para emociones con balanceo 2. 50

Figura 30: Matriz de confusión para polaridad sin balanceo. 53

Figura 31: Métricas de evaluación e informe de clasificación para polaridad sin balanceo de clases. 53

Figura 32: Matriz de confusión para polaridad con balanceo 1. 54

Figura 33: Métricas de evaluación e informe de clasificación para polaridad con balanceo 1. 55

Figura 34: Matriz de confusión para polaridad con balanceo 2. 56

Figura 35: Resultados para polaridad con balanceo 2. 56

Figura 36: Google Colab Pro. Figura 37: Entornos de ejecución disponibles. 60

Índice de tablas

Tabla 1: Resultados tras aplicar la ponderación de clases con la primera técnica. 26

Tabla 2: Resultados tras aplicar la ponderación de clases con la segunda técnica. 27

Tabla 3: Resultados tras aplicar la ponderación de clases con la primera técnica para polaridad. 28

Tabla 4: Resultados tras aplicar la ponderación de clases con la segunda técnica para polaridad. 28

Tabla 5: Resumen de los hiperparámetros óptimos. 41

Tabla 6: Resultados modelos de análisis de emociones. 51

Tabla 7: Resultados por clase para cada modelo de análisis de emociones. 52

Tabla 8: Resultados generales para los modelos de análisis de polaridad. 56

Tabla 9: Resultados por clase para cada modelo de análisis de polaridad. 57

Tabla 10: Comparativa de resultados entre DistilBERT (Balanceo 2) y RoBERTuito para emociones. 58

Tabla 11: Comparativa de resultados entre DistilBERT (Balanceo 1) y RoBERTuito para polaridad. 59

Índice de ecuaciones

Ecuación 1: Ponderación de clases con la primera técnica. 25

Ecuación 2: Ponderación de clases con la segunda técnica. 26

1

Introducción

1.1 Motivación

Este Trabajo de Fin de Grado (TFG) se desarrolla en el ámbito del Procesamiento del Lenguaje Natural (PLN) y está enfocado en el análisis emocional y de polaridad en textos publicados en redes sociales.

Plataformas como Instagram han transformado la manera en que las personas expresan y comparten sus emociones, especialmente cuando se trata de temas sensibles como la salud mental. Cada día se producen miles de comentarios que expresan opiniones, emociones y reacciones ante publicaciones de este tipo [1]. Por la gran cantidad de mensajes, la carga subjetiva que contienen y la informalidad del lenguaje utilizado, analizar estos contenidos de manera manual se vuelve prácticamente inviable, por lo que cada vez es más necesario contar con herramientas automatizadas que faciliten esta tarea [2].

En este contexto, el presente trabajo se motiva por el potencial que ofrecen los modelos de lenguaje basados en la arquitectura Transformer [3], y concretamente el uso de DistilBERT [4] a través de la librería SimpleTransformers [5], para abordar tareas complejas de clasificación emocional. Se ha empleado un modelo previamente optimizado para clasificación de texto en español, lo que ha permitido centrar el esfuerzo en su evaluación y adaptación a un nuevo dominio temático: el de la salud mental.

Más allá del interés técnico, este trabajo persigue un objetivo social. Podría ser muy interesante aplicar el análisis emocional en el contexto de la salud mental, tanto en entornos clínicos como sociales, para permitir la detección de tendencias emocionales y posibles “alarmas” o “necesidades” de intervención.

Finalmente, este proyecto también quiere investigar cómo los desequilibrios de clase, característicos de este tipo de corpus, impactan en el rendimiento del modelo y también evaluar diferentes formas de ponderar estos desequilibrios [6]. Todo ello se enmarca en una línea de trabajo aplicada y actual, que conjuga la utilidad social con el desarrollo de competencias en PLN y evaluación de modelos de aprendizaje automático.

1.2 Objetivos

1.2.1 Objetivo principal

El principal objetivo de este trabajo es optimizar y evaluar modelos de clasificación basados en la arquitectura DistilBERT para detectar emociones y polaridad en comentarios en español relacionados con la salud mental en redes sociales. Para lograrlo, se ha utilizado un modelo previamente entrenado para clasificar textos en español, concretamente, el modelo *francisco-perez-sorrosal/dccuchile-distilbert-base-spanish-uncased-finetuned-with-spanish-tweets-clf-cleaned-ds* [7], ya entrenado con tweets y se ha adaptado para responder a las necesidades específicas de este proyecto.

A lo largo del trabajo se estudia el impacto de diferentes combinaciones de hiperparámetros y técnicas de balanceo de clases en el rendimiento del sistema. Además, se realiza una comparativa de los resultados obtenidos con las distintas configuraciones óptimas e incluso con los resultados tras el uso de otros modelos. El propósito final es identificar la configuración más eficaz que permita clasificar de forma precisa y equilibrada tanto la polaridad (positiva, negativa o indeterminada) como las emociones expresadas en los textos, prestando especial atención al comportamiento del modelo frente a clases minoritarias [8].

1.2.2 Objetivos específicos

Para alcanzar este objetivo general, se han definido los siguientes objetivos específicos:

1. Preparación y análisis del corpus: organizar y preprocesar un conjunto de datos de comentarios en español vinculados a temas de salud mental, adaptándolo al

formato necesario para su uso con modelos Transformer mediante SimpleTransformers [5].

2. Entrenamiento de modelos de clasificación: entrenar dos modelos independientes de DistilBERT, uno para clasificación de emociones (6 clases) y otro para polaridad (3 clases), evaluando el rendimiento en cada tarea por separado [4].
3. Aplicación de técnicas de balanceo de clases: aplicar dos estrategias distintas de ponderación de clases durante el entrenamiento (por intervalos y ponderación inversa a la frecuencia), y comparar su efecto en el rendimiento de los modelos [6].
4. Optimización de hiperparámetros: llevar a cabo búsquedas aleatorias (*random sweeps*) para identificar las configuraciones más eficaces de hiperparámetros (learning rate, batch size y número de épocas) mediante la plataforma Weights & Biases (Wandb) [9].
5. Evaluación comparativa del rendimiento: analizar a fondo diferentes métricas de evaluación (como la *accuracy*, la precision, el *recall* y la F1-score), tanto a nivel general como para cada clase, con el objetivo de identificar qué modelo y qué técnica funcionan mejor, prestando especial atención a aquellas clases menos representadas [10].
6. Documentación del proceso experimental: redactar una memoria clara y detallada sobre la metodología, los resultados obtenidos y las conclusiones extraídas, para dejar constancia tanto del valor técnico como de la utilidad práctica del trabajo realizado.

1.3 Metodología

La metodología de este trabajo consta de cuatro fases claramente diferenciadas, que abarcan desde la revisión teórica inicial hasta la evaluación de los modelos entrenados. Cada etapa ha sido diseñada para responder a los objetivos planteados y garantizar la solidez técnica del proyecto.

1.3.1 Fase de documentación

Durante esta fase inicial, se llevó a cabo una revisión profunda sobre la clasificación de emociones y polaridad en el ámbito del Procesamiento de Lenguaje Natural, así como sobre el uso de modelos basados en la arquitectura Transformer. En

concreto, se estudió el funcionamiento de DistilBERT y la integración de modelos preentrenados mediante la librería SimpleTransformers [4][5].

Una parte importante de la comprensión del problema vino del análisis de trabajos previos, destacando especialmente el realizado por Andrea Chaves Villota, autora del proyecto *EmoSPeech* [11], cuya investigación sobre estrategias de balanceo y ponderación de clases resultó especialmente útil como referencia y guía para el diseño experimental de este trabajo.

1.3.2 Fase de análisis

Con la base establecida en el marco teórico, se procedió a la comprensión del corpus textual y al preprocesamiento de los datos. Se transformaron las etiquetas a formato numérico, se realizó una división estratificada en subconjuntos de entrenamiento, validación y prueba, y se aplicaron técnicas de limpieza textual [12]. Simultáneamente, se analizaron las distribuciones de clase y se aplicaron dos estrategias de ponderación para compensar el desbalance observado [6].

1.3.3 Fase de prueba

Durante esta etapa se entrenaron seis modelos (tres para emociones y tres para polaridad), utilizando el modelo preentrenado en español *francisco-perez-sorrosal/dccuchile-distilbert-base-spanish-uncased-finetuned-with-spanish-tweets-clf-cleaned-ds*, disponible en Hugging Face y la librería SimpleTransformers [7]. Para cada uno, se buscaron muchas combinaciones de hiperparámetros mediante barridos aleatorios utilizando Weights & Biases (Wandb) [9]. Los mejores conjuntos de configuraciones se eligieron a partir de estos resultados y se entrenaron los modelos finales. Luego, se evaluaron diferentes medidas de rendimiento como precisión, puntuación F1, precisión, recall, resultados detallados de clases y matrices de confusión para evaluar el rendimiento de los modelos [10].

1.3.4 Fase de escritura

La última fase se centró en la elaboración del documento que recoge todo el trabajo realizado. Todas las fases anteriores han sido documentadas mostrando el proceso técnico desarrollado, racionalizando las selecciones técnicas y presentando los resultados de

manera ordenada y eficiente. Además, se ofrece un comentario final sobre las conclusiones alcanzadas y el posible trabajo futuro.

1.4 Estructura de la memoria

La presente memoria está organizada en seis capítulos principales que recogen de forma estructurada el desarrollo del trabajo realizado, desde su planteamiento inicial hasta las conclusiones extraídas.

Capítulo 1: Introducción. Esta sección incluye tanto la motivación del proyecto, los objetivos principales y específicos, así como la metodología seguida para su desarrollo. También se presenta una visión general de la estructura del documento.

Capítulo 2: Estado del arte. Este capítulo contextualiza el trabajo dentro del ámbito del Procesamiento del Lenguaje Natural y la clasificación de emociones y polaridad en redes sociales. Además, se revisan los conceptos clave de los modelos basados en Transformer, con especial énfasis en DistilBERT y la librería SimpleTransformers, y se analiza cómo influye el lenguaje emocional en temas de salud mental en redes sociales.

Capítulo 3: Herramientas utilizadas. Se presentan todas las herramientas y tecnologías que han hecho posible el desarrollo del proyecto, como Python, Google Colab y Weights & Biases (Wandb), junto con las métricas utilizadas para evaluar el rendimiento de los modelos entrenados.

Capítulo 4: Análisis de la respuesta emocional con DistilBERT. Este es el núcleo del trabajo, en el cual se presenta el corpus de estudio, se describe el modelo utilizado y se detallan todas las fases del proceso: preprocesamiento de los datos, aplicación de técnicas de balanceo de clases, configuración del entrenamiento, elección de hiperparámetros mediante barridos aleatorios (sweeps) y análisis de resultados. También se evalúa comparativamente el efecto de las distintas técnicas de ponderación sobre el rendimiento del modelo, y se realiza una comparativa final de resultados óptimos alcanzados por DistilBERT y los obtenidos en un estudio previo con otro modelo BERT, con el fin de evaluar fortalezas y limitaciones de cada enfoque.

Capítulo 5: Conclusiones y líneas futuras. Se exponen las principales conclusiones extraídas del estudio y se proponen posibles líneas de trabajo para continuar y mejorar esta investigación.

Capítulo 6: Bibliografía. Incluye todas las referencias bibliográficas utilizadas a lo largo del trabajo, tanto académicas como técnicas.

2

Estado del arte

2.1 Introducción

Este capítulo discute el estado actual del uso de las tecnologías de Procesamiento de Lenguaje Natural (PLN) en el ámbito de la salud mental, centrándose particularmente en el análisis emocional y de polaridad en textos generados por usuarios. Para este propósito, describimos brevemente los fundamentos de la biblioteca SimpleTransformers y la estrategia seguida en otras investigaciones para explorar el bienestar psicológico a partir de la información recopilada de las redes sociales. Todas las herramientas y métodos disponibles aplicados en el TFG actual están integrados en este enfoque.

2.2 Marco teórico SimpleTransformers

La introducción de métodos de Inteligencia Artificial, en particular las técnicas de Aprendizaje Automático y Aprendizaje Profundo, ha cambiado drásticamente la forma en que las máquinas procesan expresiones escritas y habladas. En el Procesamiento de Lenguaje Natural, se aplican modelos y algoritmos para analizar de manera automática el significado de un texto, interpretar su intención e incluso generar contenidos coherentes y adaptados al contexto. [2]. Gracias a estos avances, hoy en día es posible llevar a cabo de forma automatizada tareas tan complejas como detectar el sentimiento de un mensaje, identificar expresiones emocionales o generar textos coherentes, actividades que antes requerían inevitablemente la intervención humana

Un detonante importante para este avance fue el desarrollo de la arquitectura Transformer [3] que planteó el mecanismo de atención como una forma natural de capturar el contexto de las palabras dentro de una secuencia.

De esta arquitectura surgieron modelos preentrenados como BERT (*Bidirectional Encoder Representations from Transformers*) [13] o GPT (*Generative Pretrained*

Transformer)[14] que introdujeron un "antes y después" en el área del PLN. Sin embargo, aplicar estos modelos desde cero puede ser muy intensivo técnicamente y consumir mucho tiempo. Para hacerlos fáciles de usar, han aparecido bibliotecas de alto nivel como SimpleTransformers [5], una biblioteca construida sobre Hugging Face Transformers y PyTorch. El propósito de esta es simplificar el uso de estos modelos a través de un entorno de trabajo fácil de usar y altamente automatizado.

SimpleTransformers hace que sea muy fácil y conveniente resolver tareas típicas como clasificación de texto o análisis de sentimientos, con muy pocas líneas de código. Realiza automáticamente muchos procesos internos tediosos, como la tokenización, la preparación de datos del modelo o la construcción de la arquitectura, y así actúa para liberar a los usuarios de la carga técnica. También elimina las dificultades de trabajos como el entrenamiento, la evaluación o la validación, y aunque no son puramente automatizados, se pueden realizar utilizando una sintaxis simple y sin tener que especificar todo en el flujo desde cero.

En este trabajo se ha utilizado DistilBERT [4], un modelo Transformer que es una versión simplificada de BERT, diseñada para reducir su tamaño y necesidad de recursos, pero sin perder apenas rendimiento. Esto lo hace especialmente útil para entornos con recursos limitados, como Google Colab. Gracias a su integración con SimpleTransformers, se ha podido entrenar y adaptar el modelo de manera práctica para clasificar la polaridad y las emociones en textos relacionados con la salud mental.

En general, SimpleTransformers es una herramienta del ecosistema de inteligencia artificial aplicada a los lenguajes, siendo un instrumento fundamental para el desarrollo de este trabajo de manera ágil y dirigiendo el esfuerzo de trabajo al diseño experimental, análisis de resultados y su impacto, sin hacer necesario programar desde conceptos técnicos avanzados.

2.3 La salud mental en redes sociales

En los últimos años, las redes sociales han crecido enormemente y se han convertido en una de las principales formas en que las personas se comunican e interactúan con otras personas y el mundo que las rodea, especialmente los jóvenes. Estas plataformas nos permiten compartir fotos, vídeos, opiniones y experiencias al instante, creando cada día una enorme cantidad de contenidos [1]. Esta conexión permanente no solo ha

transformado la manera en la que nos relacionamos, sino que también ha abierto el debate sobre su impacto en la salud mental de las personas. De hecho, muchos estudios sugieren que existe una relación compleja entre el uso de redes sociales y el bienestar emocional.

El uso excesivo también puede causar efectos secundarios, incluyendo ansiedad, depresión, estrés o dudas sobre uno mismo. Preocupaciones como la presión por mostrar una imagen ideal, compararse con otros y buscar validación, indicada a través de "me gusta" y comentarios, pueden causar inseguridad, frustración o insatisfacción [15]. Además, pasar una cantidad considerable de horas conectado puede perturbar el sueño, la atención y las relaciones orales, y por lo tanto generar cierto aislamiento [16].

De hecho, la exposición a contenido negativo, noticias impactantes o mensajes agresivos también puede afectar el estado de ánimo, resultando en fatiga emocional o sobrecarga. En algunos casos, el anonimato en estos espacios puede potenciar el acoso u ofensa, e influir en la salud mental [17].

Pero eso no significa que todos los efectos de las redes sociales sean malos. Las personas a menudo las utilizan para la expresión emocional, el compromiso con otros y un sentido de compañerismo. Existen comunidades positivas que apoyan el autocuidado, la empatía y el diálogo sobre la salud mental [18]. Además, estos sitios también pueden ayudar a reducir el estigma sobre las enfermedades mentales al aumentar la conciencia en la sociedad en general.

En otras palabras, los efectos de las redes sociales en la salud mental tienden a depender de cómo se empleen. Puede ser un uso positivo y equilibrado, y también puede ser un uso perjudicial, excesivo y mal manejado. Por lo tanto, es clave fomentar hábitos digitales saludables, especialmente entre adolescentes y jóvenes.

3

Herramientas utilizadas

3.1 Introducción

La elección del entorno de trabajo, lenguaje de programación y bibliotecas se ha realizado con el objetivo de proporcionar la implementación de modelos de procesamiento de lenguaje natural de manera fácil y comprensible, con flexibilidad y facilidad para experimentar. A continuación, se presenta una lista de herramientas utilizadas para el desarrollo.

3.2 Python

Este proyecto ha sido programado con Python [19], uno de los lenguajes de programación más utilizados para inteligencia artificial y análisis de datos. Su popularidad cada vez es mayor ya que destaca por su sencillez, flexibilidad y por contar con una gran variedad de librerías especializadas.

Gracias a que su sintaxis es clara y fácil de entender, resulta ideal para proyectos como este, que incluyen varias etapas y requieren un desarrollo organizado, una depuración eficaz y un fácil mantenimiento a lo largo del tiempo. Esta característica es especialmente útil en entornos académicos donde el tiempo de implementación, así como la legibilidad del código, son cruciales [20].

Además de eso, Python tiene una comunidad fuerte y muchas bibliotecas dedicadas al procesamiento de lenguaje natural, aprendizaje automático y visualización de datos. Estas herramientas permiten trabajar en todas las etapas del proyecto, desde la carga y preprocesamiento de datos hasta el entrenamiento, evaluación e interpretación del modelo. Algunas de ellas incluyen pandas, scikit-learn, transformers y SimpleTransformers que se utilizaron en este documento [5].

Otra gran ventaja es el hecho de que soporta plataformas de desarrollo en la nube como Google Colab, donde el cálculo se puede realizar en un entorno GPU sin configurar un entorno local elaborado [21]. Esta flexibilidad ha permitido la creación y entrenamiento de modelos con recursos escasos que funcionan bien.

En conclusión, Python ha sido fundamental para la realización de este trabajo, proporcionando un buen compromiso entre potencia, facilidad de uso y recursos. Su capacidad para combinar cualquier etapa de procesamiento de texto es otra gran característica de un lenguaje para proyectos como análisis de sentimientos y clasificación de polaridad del lenguaje natural.

3.3 Librerías Python empleadas

A lo largo del desarrollo del proyecto se han utilizado diversas librerías de Python. A continuación, se describen las principales librerías externas empleadas y su función dentro del proyecto:

- Pandas: utilizada para la lectura de archivos CSV y la manipulación de datos tabulares mediante estructuras como DataFrame [22].
- Numpy: empleada para realizar operaciones numéricas y trabajar con arrays de manera rápida y eficaz. Aunque su uso en este proyecto ha sido puntual, ha resultado especialmente útil en tareas de apoyo [23].
- Datasets (Hugging Face): facilita el manejo de datos en NLP y ha permitido transformar etiquetas categóricas a través de la clase *ClassLabel* y organizar la información para adaptarla al modelo [24].
- Scikit-learn (sklearn): ha sido una de las herramientas más relevantes en la fase de evaluación. Se ha utilizado para dividir el dataset (*train_test_split*), calcular métricas de rendimiento, generar informes (*classification_report*) y construir matrices de confusión (*confusion_matrix*) [12].
- Simpletransformers: librería de alto nivel que ha simplificado el entrenamiento de modelos de clasificación basados en arquitecturas Transformer, como DistilBERT. Su sintaxis accesible ha permitido acelerar el desarrollo del modelo manteniendo un alto rendimiento [5].
- Torch (PyTorch): aunque no se ha utilizado directamente en el código, funciona como *backend* del modelo. También ha permitido detectar y

utilizar aceleración por GPU mediante la función *torch.cuda.is_available*() [25].

- Matplotlib.pyplot: utilizada para representar gráficamente los resultados del modelo, especialmente la matriz de confusión [26].
- Seaborn: complementa a matplotlib y ha sido usada para generar visualizaciones más estilizadas, como mapas de calor (*heatmap*) aplicados a la matriz de confusión [27].

El uso conjunto de estas librerías ha permitido implementar, entrenar y evaluar el modelo de forma estructurada, facilitando el análisis automático de emociones y polaridad en textos con herramientas de procesamiento de lenguaje natural.

3.4 Google Colab

Google Colab es una plataforma gratuita proporcionada por Google para escribir y ejecutar Python directamente en el navegador. Está basada en cuadernos Jupyter y te permite ejecutar código basado en GPU o TPU. Proporciona una solución flexible para proyectos basados en requisitos y recursos disponibles [21].

La plataforma principal de desarrollo en este trabajo ha sido Google Colab. Su interfaz amigable ha facilitado la escritura del código, la integración con explicaciones teóricas y la visualización de los resultados en un solo documento. También ha sido posible almacenar automáticamente modelos, gráficos y archivos intermedios, gracias a su integración con Google Drive.

Una de las mayores ventajas es que ha permitido habilitar la aceleración de GPU cuando lo necesitaba: ha reducido el tiempo de entrenamiento de modelos como DistilBERT, pero también ha estado en condiciones de ejecutar algunas de sus partes utilizando CPU, ya sea los pasos de preprocesamiento o el análisis de resultados.

3.5 Wandb

Wandb (Weights & Biases) [9] es una herramienta para rastrear y visualizar experimentos de aprendizaje automático que es capaz de registrar métricas en tiempo real, comparar ejecuciones y analizar los resultados de manera organizada. Funciona sin problemas con los marcos más populares como PyTorch o TensorFlow para gestionar fácilmente el entrenamiento y la evaluación del modelo bajo control.

Wandb se ha empleado para rastrear el rendimiento del modelo en cada época de entrenamiento a lo largo del desarrollo de este proyecto. Se han realizado medidas como pérdida, precisión o f1-score para identificar rápidamente problemas de sobreajuste o saturación en el proceso de aprendizaje.

Además, su interfaz gráfica ha simplificado la comparación de varias configuraciones de hiperparámetros y el estudio de los resultados obtenidos en las diferentes pruebas realizadas.

Una de las características más poderosas ha sido la capacidad de iniciar barridos automáticos que toman una variedad de hiperparámetros (por ejemplo, tasa de aprendizaje, tamaño de lote, número de épocas) y prueban varias combinaciones de estos y tiempos el mejor conjunto basado en las métricas capturadas. Todo esto junto hace que sea más eficiente y sistemático optimizar el modelo.

En resumen, Wandb ha sido una herramienta importante para monitorear el curso de los experimentos, evaluar el desarrollo del modelo y obtener información sobre el proceso de entrenamiento.

3.6 Métricas de rendimiento

Para evaluar el rendimiento de los modelos en tareas de clasificación de polaridad y emoción, se han adoptado métricas ampliamente reconocidas en el aprendizaje automático. Estas medidas permiten el análisis de la calidad de la predicción desde un conjunto diverso de enfoques y han sido esenciales para comparar configuraciones y ajustar durante el desarrollo del proyecto.

Las métricas más comunes empleadas que se han utilizado son:

• Precisión: proporción del número de predicciones correctas sobre el número total de predicciones. Es útil tener el conteo de cuántos de los elementos positivos u otras emociones fueron realmente positivos u otra emoción [28].

• Recall: Es la capacidad del modelo para identificar todos los miembros de una clase. Es decir, qué proporción de los casos reales encuentra el modelo [28].

• Puntaje F1: agrega precisión y recall en un solo parámetro usando su media armónica. Es particularmente útil en caso de clases desbalanceadas ya que ofrece una visión equilibrada del rendimiento [10].

• Precisión: indica la proporción general de predicciones correctas entre el número total de casos evaluados. Es una medida general, que podría ser menos informativa cuando algunas clases están fuertemente subrepresentadas en comparación con otras clases en los conjuntos de datos [29].

• Matriz de confusión: se utiliza para obtener una imagen clara de los aciertos y errores del modelo en cada clase. También permite establecer con qué otras clases se confunde una etiqueta, útil tanto para adaptar el modelo como para la interpretación de resultados [30].

• Coeficiente de correlación de Matthews (MCC): Tiende a ser una medida equilibrada incluso si las clases son de tamaños muy diferentes. Presenta una evaluación más justa sobre el rendimiento del modelo, lo cual es especialmente necesario para el conjunto de datos desbalanceado [31].

Todas estas métricas se han calculado con métodos de scikit-learn y han sido cruciales para probar la calidad del modelo, elegir la mejor configuración y comprender su comportamiento.

4

Análisis de la respuesta emocional con DistilBERT

4.1 Introducción

Este capítulo describe todo el proceso de entrenamiento de modelos de clasificación utilizando DistilBERT. A lo largo del proceso, se ha trabajado con un conjunto de datos etiquetados y se han aplicado técnicas de preprocesamiento, división del dataset, entrenamiento y evaluación mediante métricas específicas.

Primero, se da una visión general del corpus para analizar la estructura de los datos, la distribución de clases y los posibles desequilibrios. Luego, se explica la configuración del modelo DistilBERT utilizando la biblioteca SimpleTransformers, junto con las elecciones para la configuración de entrenamiento y la estrategia para mejorar el rendimiento, incluyendo el equilibrio de clases.

Finalmente, se muestran los resultados obtenidos por el modelo en las dos tareas: clasificación de polaridad y de emociones, con visualizaciones, análisis de las métricas alcanzadas y la comparación de resultados utilizando las distintas técnicas de balanceo, junto a una comparativa con trabajos anteriores utilizando otro modelo BERT.

Por todo ello, esta sección representa la parte experimental del proyecto, donde ponemos a prueba la capacidad del modelo para detectar de manera automática el contenido emocional en los datos, e investigamos qué aspectos afectan de manera clave a su rendimiento.

4.2 Análisis descriptivo del dataset

El conjunto de datos para este proyecto es una colección de comentarios sobre salud mental y se basa principalmente en comentarios extraídos de redes sociales (Instagram y TikTok). Estos datos se proporcionan en un archivo con tres columnas principales: el texto, la polaridad y la emoción detallada (ver Figura 1, columnas A, C y D).

A	B	C	D	E	F
Text	**Emoticons**	**Polarity**	**Emotions**	**Stigma**	**Influencer**
¡Animo guapa! Aquí tienes a tu familia virtual que te sostiene ♡😍	♡😍	Positiva	Amor/Admira	No	TamaraGorro
¡Nos alegra tenerte nuevamente por aquí!.Te echábamos de menos. Ánimos preci	♡♡♡♡	Positiva	Amor/Admira	No	LauraEscanes
@dulceida Puedes con todo, pero no todo al mismo tiempo ♡♡♡♡ tomate	♡♡♡♡	Positiva	Amor/Admira	No	Dulceida
@mpombor mucho ánimo y estamos contigo muchos besos		Positiva	Amor/Admira	No	MariaPombo
@tamara_gorro GUAPA,no te conozco pero me transmites cosas muy positivas!	😘😘😘	Positiva	Amor/Admira	No	TamaraGorro
♡♡... se nota que eres un amor. Cada tanto hay que desconectar por nuestra s	♡♡	Positiva	Amor/Admira	No	LauraEscanes
♡♡♡ ánimos bonita. Poco a poco. Un abrazo	♡♡♡	Positiva	Amor/Admira	No	LauraEscanes
♡♡♡ eres enorme	♡♡♡	Positiva	Amor/Admira	No	LauraEscanes
♡♡♡ eres la mejor	♡♡♡	Positiva	Amor/Admira	No	MariaPombo
♡♡♡ eres un amor.	♡♡♡	Positiva	Amor/Admira	No	LauraEscanes
♡♡♡ eres un ejemplo	♡♡♡	Positiva	Amor/Admira	No	MariaPombo
♡♡♡ mucho amor, ánimo Marta!!	♡♡♡	Positiva	Amor/Admira	No	MariaPombo
♡♡♡ Mucho ánimo!!!!	♡♡♡	Positiva	Amor/Admira	No	LauraEscanes
♡♡♡ Todo pasará. Mucho ánimo.Me gusta mucho verte porque aportas luz y	♡♡♡	Positiva	Amor/Admira	No	MariaPombo
♡♡♡♡ ánimo eres una campeona	♡♡♡♡	Positiva	Amor/Admira	No	LauraEscanes
♡♡♡♡ eres maravillosa y te echamos mucho de menos, pero como dices, l	♡♡♡♡	Positiva	Amor/Admira	No	MariaPombo

Figura 1: Conjunto de datos.

En la columna de texto, hay 4,372 comentarios que han sido escritos por los usuarios de manera más espontánea. Las menciones sobre personas externas han sido anonimizadas para proteger la privacidad. Las notas son típicamente breves y describen experiencias, reflexiones o respuestas emocionales a personas/eventos.

Cada uno de estos comentarios está doblemente etiquetado, es decir, el corpus se basa en dos niveles de clasificación: una etiqueta de polaridad que muestra si el mensaje es positivo, negativo o neutral, y una etiqueta de emoción, que también especifica qué emoción dominante (como la más relevante) se percibe (aplicada al texto de introducción de un mensaje) [8].

En este sentido, el conjunto de datos puede considerarse un caso de clasificación estructurada de múltiples etiquetas, pero el problema se ha abordado como dos problemas de clasificación diferentes: uno para la detección de polaridad y otro para la clasificación de emociones.

La polaridad es el sentimiento general manifestado en los mensajes. En este corpus, se han clasificado en tres categorías:

- Positiva: comentarios con un tono afectuoso, de admiración, aliento o valoración positiva. Reflejan emociones agradables o de apoyo: *"¡Ánimo guapa! Aquí tienes a tu familia virtual que te sostiene"*.
- Negativa: comentarios que expresan rechazo, enfado, crítica o burla. Presentan un tono desfavorable o conflictivo: *"Eres una ridícula, que nunca te toque de verdad. Qué vergüenza."*
- Indeterminada (Neutral): casos donde el contenido del mensaje es ambiguo o no se puede clasificar con claridad como positivo o negativo, ya sea por falta de contexto o neutralidad en la expresión*: "Hay que estar siempre alerta para no dejar que el foco de luz a lo importante"*.

En cuanto a las emociones, estas se han organizado en seis categorías principales, que reflejan distintos matices emocionales detectados en los comentarios [32]. A continuación, se describen siguiendo el mismo orden empleado en el dataset:

- Amor/Admiración: incluye expresiones de afecto, cariño, respeto o reconocimiento hacia otra persona. Son mensajes que muestran apoyo, valoración positiva y cercanía emocional: *"Eres única, toda una chica valiente, te admiro"*.
- Gratitud: son aquellos comentarios que expresan agradecimiento por un gesto, una experiencia compartida o una acción significativa. Este tipo de emoción suelen transmitir cariño y aprecio hacia alguien: *"Grande Tania, gracias por reflejarnos la vida tal cuál, unas veces se ríe, otras se lloran...no somos robots. Besitos bonita"*
- Tristeza/Pena: comprende mensajes en los que se expresa dolor, sufrimiento o preocupación: *"Que pena verte así…"*
- Enfado/Desprecio/Burla: agrupa comentarios con un tono crítico, sarcástico u hostil. Reflejan desacuerdo, rechazo o ridiculización hacia la persona destinataria o hacia el contenido publicado: *"Yo creo que tiene un problema psicológico importante. Siempre llamando la atención. No debería de estar en las redes así."*
- Comprensión/Empatía/Identificación: engloba mensajes en los que el emisor demuestra una conexión emocional con lo que se ha expresado, ya sea por experiencia compartida o por una actitud empática hacia la situación

descrita: *"Me encanta tu sinceridad, tu humanidad y naturalidad, entiendo perfectamente cómo te sientes"*

- Indeterminada (Neutral): se refiere a comentarios en los que no se identifica con claridad una emoción específica. Esto puede deberse a que son neutros, reflexivos, de carácter religioso o simplemente no contienen información emocional suficiente: *"Oren para que aquellos que no conocen el significado de la salud mental lo sientan algún día"*.

Durante la fase de análisis exploratorio inicial, se analizaron y visualizaron las distribuciones de clase mediante gráficos, lo que permitió detectar un desequilibrio significativo tanto en polaridad como en emociones. Como puede observarse en las Figuras 2 y 3, algunas clases como "*Amor/Admiración*" o la polaridad "*Positiva*" están altamente representadas, mientras que otras como "*Tristeza/Pena*" o "*Indeterminado*" aparecen con una frecuencia mucho menor.

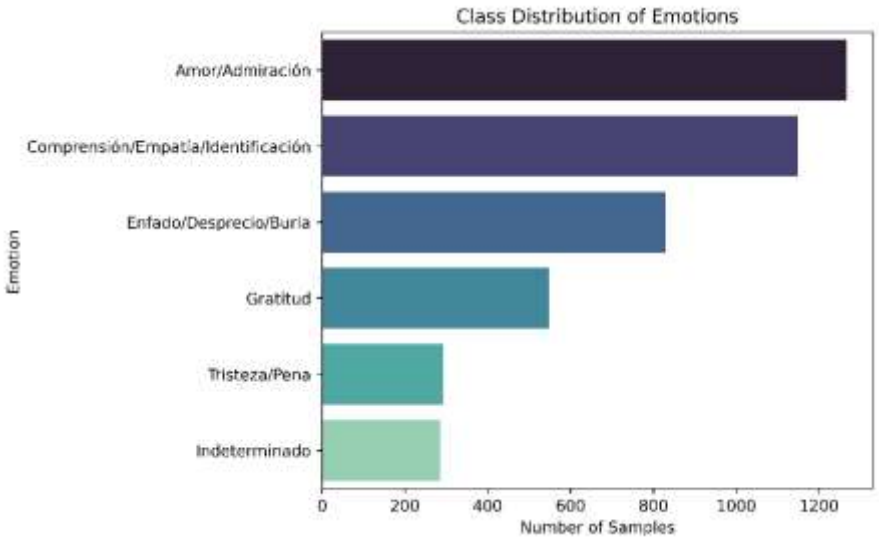

Figura 2: Distribución de las clases de emociones.

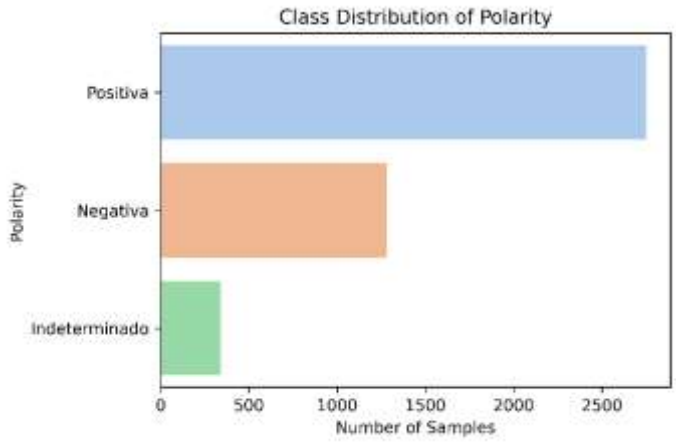

Figura 3: Distribución de las clases de polaridad.

Este tipo de desbalance puede afectar el rendimiento del modelo porque tiende a dar prioridad a las clases mayoritarias, dificultando la capacidad de detectar correctamente las clases minoritarias.

Este análisis ha sido crucial para comprender la estructura del corpus así como para tener una idea de cuáles son las decisiones importantes que deben tomarse durante el entrenamiento. Esto ha resaltado la importancia de utilizar estrategias adecuadas de equilibrio de clases, algo que consideraremos con más detalle en el Apartado 4.5. El análisis descriptivo general también ha servido como entrada para ajustar el diseño experimental del proyecto, modificando el preprocesamiento, la arquitectura del modelo y otras consideraciones similares, para que coincidan con las características reales de los datos [33].

4.3 Modelo SimpleTransformers utilizado: DistilBERT

Para abordar las tareas de clasificación de polaridad y emociones, se ha utilizado un modelo basado en DistilBERT, implementado a través de la librería SimpleTransformers. Esta combinación ha permitido sacar el máximo partido de los modelos basados en la arquitectura Transformer, al mismo tiempo que facilitó el trabajo al reducir la complejidad técnica y el costo computacional durante el entrenamiento [5].

DistilBERT es, en esencia, una versión simplificada de BERT, pensada para ser más rápida y ligera sin perder demasiado rendimiento en comparación con el modelo original.

Esta ventaja viene de una técnica llamada distilación del conocimiento (*knowledge distillation*) [4][13]. El concepto es sencillo: en lugar de entrenarse directamente con las etiquetas del conjunto de datos, un modelo más pequeño (el estudiante) aprende a imitar a otro modelo más grande y preciso (el profesor). Así, el estudiante recibe no solo la categoría correcta, sino también la información interna que el profesor tiene sobre la confianza en cada clase, lo que le permite alcanzar un gran rendimiento con menos recursos. Gracias a este proceso, el modelo resultante (DistilBERT) mantiene gran parte del conocimiento del modelo original (BERT), pero con una reducción significativa en el número de parámetros, lo que se traduce en menor consumo de memoria y tiempos de entrenamiento más cortos [4].

En este trabajo no se ha llevado a cabo el proceso de destilación como tal, sino que se ha utilizado un modelo DistilBERT ya preentrenado, específicamente el modelo: "*francisco-perez-sorrosal/dccuchile-distilbert-base-spanish-uncased-finetuned-with-spanish-tweets-clf-cleaned-ds"* [7].

Este modelo ha sido extraído del repositorio de modelos de Hugging Face. Fue publicado por el investigador Francisco Pérez-Sorrosal y está basado en la versión distilada del distilBERT en español desarrollada por la Universidad Católica de Chile ("*dccuchile/distilbert-base-spanish-uncased"*), la cual fue ajustada (*fine-tuned*) para tareas de clasificación de textos en español, concretamente sobre tweets etiquetados por polaridad, lo que lo hace especialmente adecuado para este proyecto, dado que el corpus empleado contiene comentarios breves, espontáneos y emocionalmente cargados, procedentes de redes sociales como Instagram.

Esta elección de modelo preajustado me ha permitido aprovechar su capacidad para capturar matices emocionales y expresivos propios del lenguaje informal en español, reduciendo el tiempo de entrenamiento y mejorando la generalización en tareas de análisis de sentimientos.

Para facilitar su implementación y ajuste, se ha utilizado la librería SimpleTransformers, una interfaz de alto nivel construida sobre transformers (Hugging Face) y PyTorch. Esta herramienta facilita la configuración y entrenamiento de modelos como DistilBERT con tan solo unas líneas de código, encargándose de gran parte del trabajo, como de la tokenización y la predicción. Además, simplifica la preparación de los

datos y el seguimiento de las métricas, lo que ha permitido centrar el enfoque del proyecto en lo realmente importante: el diseño experimental y el análisis de resultados [5].

En este proyecto, el modelo se ha entrenado para realizar clasificación de texto en dos tareas diferentes: detectar la polaridad (con tres clases) e identificar emociones (con seis clases), utilizando en ambos casos los mismos procedimientos de preprocesamiento y los mismos hiperparámetros base. La configuración final y las decisiones tomadas en cuanto al entrenamiento se detallan en los apartados siguientes.

4.4 Preprocesamiento de datos

Antes de comenzar el entrenamiento, se realiza un preprocesamiento para asegurar que tanto los comentarios textuales como las etiquetas estén en el formato que el modelo DistilBERT pueda utilizar.

El proceso se inicia con la importación de las librerías necesarias y la carga del conjunto de datos desde Google Drive, como se muestra en el fragmento de código de la Figura 4.

```
# Import necessary libraries
import pandas as pd  # For data manipulation and analysis
import numpy as np   # For numerical operations

# Mount Google Drive to access files stored in the user's Drive
from google.colab import drive
drive.mount('/content/drive')

# Load the CSV file containing the mental health corpus
dataset = pd.read_csv("/content/drive/MyDrive/TFGSOFÍA/CorpusSaludMentalCompleto.csv")

# Display the count of each unique value in the 'Emotions' column
# This helps understand the class distribution in the dataset
dataset['Emotions'].value_counts()
```

Figura 4: Fragmento de código encargado de la preparación del conjunto de datos y la importación de las librerías necesarias.

A continuación, las etiquetas de emoción fueron transformadas de texto a valores numéricos utilizando la clase *ClassLabel* de la librería datasets, lo que facilita su uso posterior en el modelo (ver Figura 5). Se sigue el mismo proceso para las etiquetas de polaridad, normalizándolas en sus tres clases: positivo, negativo e indeterminado.

```python
# Import ClassLabel from the datasets library to manage categorical labels
from datasets import ClassLabel

# Define the ClassLabel with six emotion categories
c21 = ClassLabel(
    num_classes=6,
    names=[
        'Amor/Admiración',                    # Love/Admiration
        'Gratitud',                           # Gratitude
        'Tristeza/Pena',                      # Sadness/Grief
        'Enfado/Desprecio/Burla',             # Anger/Contempt/Mockery
        'Comprensión/Empatía/Identificación', # Understanding/Empathy/Identification
        'Indeterminado'                       # Undetermined
    ]
)

# Convert emotion labels from text to integer class IDs
label_int = [c21.str2int(label) for label in dataset['Emotions']]
```

Figura 5: Fragmento de código que carga y codifica las etiquetas de emociones en formato numérico.

Posteriormente, se dividió el dataset en tres subconjuntos: entrenamiento, validación y prueba. Se aplicó una división estratificada a cada uno de ellos para asegurar que las clases se asignen equitativamente (Figura 6).

```python
from sklearn.model_selection import train_test_split

# Select the relevant columns: text data and emotion labels
X = dataset['Text']
y = dataset['Emotions']

# Split the dataset: 70% training, 15% validation, 15% test
X_train, X_temp, y_train, y_temp = train_test_split(
    X, label_int, stratify=label_int, test_size=0.3, random_state=42
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, stratify=y_temp, test_size=0.5, random_state=42
)
```

Figura 6: Fragmento de código para la división estratificada del conjunto de datos.

Esta división se procesa en 2 pasos, utilizando la función *train_test_split* de *scikit-learn* con estratificación, para mantener la frecuencia de las clases similar a la del conjunto original [12]. Primero, el 70% de los datos se dividió para entrenamiento, el 30% restante se dividió equitativamente en validación y prueba (15% cada uno). Esto es justificable ya que:

- Permitimos que el modelo aprenda los patrones de las diferentes clases emocionales o de polaridad con el 70% del corpus durante el entrenamiento.

- Se retiene el 15% para validación, lo que permite evaluar el modelo mientras se entrena. Este conjunto se emplea para verificar el rendimiento en los datos no vistos y se utiliza para aplicar técnicas como la detención temprana para evitar el sobreajuste [34].
- El 15% restante es el conjunto de prueba, el cual se reserva exclusivamente para evaluar qué tan bien funciona el modelo al enfrentarse a datos que no ha visto antes. De esta manera, la evaluación es más justa y realmente muestra la capacidad de generalización del modelo.

Esta división equilibrada y estratificada de los datos es importante para entrenar y evaluar en conjuntos de datos representativos, algo que es especialmente crítico en tareas sensibles como la clasificación emocional.

Una vez finalizada la división, los datos fueron organizaron en *DataFrames* con el formato requerido por la librería SimpleTransformers, incluyendo las columnas *text* (comentario) y *labels* (categoría correspondiente). Además, se aplicó una función de limpieza para eliminar caracteres especiales, símbolos y otros elementos no alfabéticos que pudieran introducir ruido en el entrenamiento. El código correspondiente puede verse en la Figura 7.

```
# Create DataFrames in the format required by SimpleTransformers
train_df = pd.DataFrame({'text': X_train, 'labels': y_train})
val_df = pd.DataFrame({'text': X_val, 'labels': y_val})   # Validation set
test_df = pd.DataFrame({'text': X_test, 'labels': y_test})  # Test set

# Function to clean text: removes emojis and special characters
def clean_text(text):
    if pd.isna(text) or not isinstance(text, str):
        return ''  # Handle NaN or non-string values safely
    return re.sub(r'[^\w\s]', '', text)  # Remove non-word characters

import re
import pandas as pd

# Clean the text in all three datasets
train_df['text'] = train_df['text'].apply(clean_text)
val_df['text'] = val_df['text'].apply(clean_text)
test_df['text'] = test_df['text'].apply(clean_text)
```

Figura 7: Código de limpieza de texto aplicado a los comentarios.

Esta serie de transformaciones ha preparado el corpus y lo ha dejado listo para ser incorporado al modelo. Finalmente, los datos están formateados, cumpliendo completamente con el formato de los datos mencionados anteriormente en

SimpleTransformers, es decir, permitiendo el entrenamiento, validación y posterior prueba del modelo en las etapas posteriores del proyecto.

4.5 Balanceo de clases

En particular, el entrenamiento de modelos para tareas de clasificación puede verse afectado por distribuciones de clases no uniformes, como suele ser el caso en problemas de múltiples clases. Se han utilizado una variedad de técnicas de equilibrio para hacer que el modelo sea menos inclinado hacia la clase mayoritaria, con el fin de refinar el rendimiento del modelo y detectar las clases pertenecientes a la minoría.

4.5.1 Desbalanceo en el dataset: distribución de clases y problemas asociados

Durante el análisis inicial del conjunto de datos (Apartado 4.2), se detectó un desbalance significativo en la distribución de clases, tanto en polaridad como en emociones. Algunas clases, como *Amor/Admiración* o la polaridad *Positiva*, presentan un número elevado de muestras, mientras que otras, como *Enfado/Desprecio/Burla* o "*Tristeza/Pena*", aparecen con bastante menor frecuencia.

Esta situación es bastante común en tareas de clasificación de emociones, y presenta un reto importante: el modelo tiende a aprender con mayor facilidad los patrones de las clases dominantes, mientras que tiene más dificultades para identificar correctamente a las clases menos frecuentes. Esto provoca que el rendimiento general pueda verse afectado y que las métricas para las clases minoritarias, como el *recall* o el *F1-score*, sean mucho más bajas [6].

La presencia de este fenómeno no solo se ha observado en este trabajo, sino también en estudios recientes como el de Chaves-Villota [11]. En su investigación, se mostró que las emociones minoritarias como *miedo* eran particularmente difíciles de clasificar debido a su escasa representación en su conjunto de datos. Por esta razón, es clave aplicar estrategias específicas para reducir el desbalance antes de entrenar el modelo.

4.5.2 Técnicas de balanceo aplicadas

En este trabajo, para enfrentarnos a este problema, hemos utilizado una técnica de ponderación de clases (*class weighting*) durante el entrenamiento.

Con el objetivo de compensar el desbalance de clases identificado en el conjunto de datos, se ha aplicado la técnica de ponderación de clases (*class weighting*). Esta estrategia permite ajustar la importancia que tiene cada clase en la función de pérdida, es decir, se asigna a las clases minoritarias un peso mayor, de modo que los errores cometidos al clasificarlas afectan más al aprendizaje que los errores en clases mayoritarias.

En este trabajo se han evaluado dos técnicas distintas de ponderación de clases: la primera se basa en ponderación por intervalos según frecuencia y la segunda en una ponderación inversamente proporcional a la frecuencia.

4.5.2.1 Balanceo 1: Ponderación por intervalos según frecuencia

En este primer enfoque se emplea una asignación discreta de pesos en función de la frecuencia porcentual de cada clase, utilizando un sistema de rangos predefinido [11]. Según esta técnica, las clases con menos del 5 % del total reciben un peso de 2; entre 5 % y 20 %, un peso de 1.5; entre 20 % y 35 %, un peso de 1; y aquellas con más del 35 %, un peso de 0.5 (Ecuación 1). Esta regla busca penalizar más los errores en clases poco representadas y reducir el sesgo hacia las clases mayoritarias.

$$wi = \begin{cases} 2, & \% < 5 \\ 1.5, & 5 < \% < 20 \\ 1, & 20 < \% < 35 \\ 0.5, & \% > 35 \end{cases}$$

Ecuación 1: Ponderación de clases con la primera técnica.

Aplicando esta regla a la distribución de clases del corpus, se obtuvo la siguiente tabla (Tabla 1).

Emoción	*Frecuencia*	*Porcentaje (%)*	*Peso w_i*
Amor/Admiración	*1267*	*28.98%*	*1.0*
Comprensión/Empatía/Identificación	*1148*	*26.26%*	*1.0*
Enfado/Desprecio/Burla	*830*	*18.98%*	*1.5*

Gratitud	*548*	*12.53%*	*1.5*
Tristeza/Pena	*293*	*6.7%*	*1.5*
Indeterminado	*286*	*6.54%*	*1.5*

Tabla 1: Resultados tras aplicar la ponderación de clases con la primera técnica.

El vector de pesos resultante en el orden definido por la codificación del dataset *(c2l = ClassLabel(num_classes=6, names=['Amor/Admiración', 'Gratitud', 'Tristeza/Pena', 'Enfado/Desprecio/Burla', 'Comprensión/Empatía/Identificación', 'Indeterminado']),* es: weight = [1, 1.5, 1.5, 1.5, 1, 1.5].

Este vector indica que las clases mayoritarias (*Amor/Admiración* y *Comprensión/Empatía/Identificación*) tienen suficiente representación, mientras que el resto requiere un refuerzo adicional durante el entrenamiento.

4.5.2.2 Balanceo 2: Ponderación inversamente proporcional a la frecuencia.

El segundo enfoque consiste en calcular los pesos de cada clase utilizando una fórmula basada en la frecuencia absoluta, por lo que se trata de una estrategia más precisa. La fórmula empleada se muestra en la Ecuación 2, donde wi es el peso asignado a la clase i, n es el número total de muestras del dataset (en nuestro caso 4372), n_c es el número de clases (6 para emociones) y n_i es el número de muestras pertenecientes a la clase i.

$$wi = \frac{n}{n_c\, n_i}$$

Ecuación 2: Ponderación de clases con la segunda técnica.

Aplicando esta fórmula se obtienen los siguientes resultado de la Tabla 2:

Emoción	*Frecuencia (ni)*	*Peso w_i (Técnica 2)*
Amor/Admiración	*1267*	*0.575*
Comprensión/Empatía/Identificación	*1148*	*0.635*

Enfado/Desprecio/Burla	*830*	*0.878*
Gratitud	*548*	*1.329*
Tristeza/Pena	*293*	*2.487*
Indeterminado	*286*	*2.545*

Tabla 2: Resultados tras aplicar la ponderación de clases con la segunda técnica.

El vector resultante ordenado es: weight = [0.575, 1.329, 2.487, 0.878, 0.635, 2.545]. Esta técnica permite un ajuste más fino del aprendizaje, reflejando con mayor precisión el desequilibrio real del corpus.

En el estudio original [11], se observó que este tipo de ponderación mejora significativamente la *recall* y la *F1-score* de clases minoritarias como *anger*, *fear*, *joy* o *sadness*. En este trabajo se ha confirmado esa tendencia, especialmente en clases como *Tristeza/Pena* y *Enfado/Desprecio/Burla*, al aplicar ambos esquemas de ponderación, tal y como se explicará en los siguientes apartados.

Estas ponderaciones se aplicaron fuera del diccionario de configuración del modelo, a través del parámetro *weight* pasado directamente a la función *train_model()*, lo que permite modificar el cálculo de la pérdida según los pesos definidos.

4.5.2.3 Ponderación de clases para polaridad

Para la tarea de clasificación de polaridad, donde las clases son: *Positiva*, *Negativa* e *Indeterminada*, también se aplicaron las dos técnicas de ponderación descritas previamente.

En primer lugar, se analiza la ponderación por intervalos según frecuencia. Aplicando los rangos establecidos en la Ecuación 1, se obtuvieron los siguientes resultados de la Tabla 3.

Polaridad	*Frecuencia*	*Porcentaje (%)*	*Peso* w_i
Positiva	*2751*	*64.44 %*	*0.5*

Negativa	*1281*	*30 %*	*1.0*
Indeterminada	*340*	*8 %*	*1.5*

Tabla 3: Resultados tras aplicar la ponderación de clases con la primera técnica para polaridad.

Este esquema refleja claramente la descompensación en el número de ejemplos por clase. La clase *Positiva*, al superar el 35 % del total, recibe el menor peso (0.5), mientras que *Indeterminada*, con tan solo un 8 %, recibe un refuerzo importante (1.5). Este enfoque tiene la ventaja de ser sencillo de aplicar y permite compensar, al menos parcialmente, el sesgo hacia las clases más frecuentes.

Para una asignación más precisa, se aplicó la segunda técnica basada en la frecuencia absoluta. Según la fórmula recogida en la Ecuación 2 donde n=4372 (número total de muestras), n_c=3 (número de clases) y n_i es el número de ejemplos en cada clase, se obtuvieron los resultados de la Tabla 4.

Polaridad	*Frecuencia (ni)*	*Peso w_i (Técnica 2)*
Positiva	*2751*	*0.53*
Negativa	*1281*	*1.14*
Indeterminada	*340*	*4.28*

Tabla 4: Resultados tras aplicar la ponderación de clases con la segunda técnica para polaridad.

Este segundo método proporciona un ajuste más fino, especialmente en el caso de la clase *Indeterminada*, que recibe un peso notablemente alto (4.28), muy superior al de las clases *Positiva* y *Negativa*. Esto busca corregir el fuerte desbalance del conjunto de datos, en el que más del 60 % de los ejemplos pertenecen a la clase *Positiva*.

Ambas técnicas tienen el objetivo común de mitigar el sesgo hacia las clases mayoritarias. Sin embargo, los pesos resultantes y su impacto en el modelo son distintos: el balanceo 1 suaviza el desbalance de forma progresiva, pero puede resultar insuficiente cuando el grado de desproporción es muy alto, como ocurre con *Indeterminada*. Por otro lado, el balanceo 2 asigna pesos más extremos a las clases menos frecuentes para obligar al modelo a prestarles más atención. Aunque esta estrategia puede ayudar a detectar mejor

a las clases minoritarias, también conlleva el riesgo de crear inestabilidad, haciendo que el modelo reduzca su rendimiento en las clases mayoritarias.

Los efectos de estos diferentes enfoques de ponderación y su impacto en el rendimiento del modelo se analizan en detalle en el Apartado 4.8.2, donde se compara cómo afectan a las diferentes métricas y a la capacidad de identificar correctamente cada clase.

4.5.3 Otras técnicas alternativas no aplicadas

Aunque en este proyecto se ha optado por la ponderación de clases (*class weighting*) como método principal para abordar el desbalance de clases, existen otras técnicas complementarias que también podrían aplicarse en futuros trabajos [36]. Entre las más utilizadas se encuentran:

- Oversampling: replicar (multiplicar) datos siguiendo la distribución de la clase minoritaria. Aunque podría lograr un mejor rendimiento para clases con pocos ejemplos, el peligro de sobreajuste es alto (overfitting).

- Undersampling: trata de reducir las muestras de las clases mayoritarias. Es útil para equilibrar el conjunto de datos, pero puede causar la pérdida de conocimiento útil y disminuir la capacidad de generalización del modelo.

- Generación de datos sintéticos: el uso de herramientas como SMOTE (*Synthetic Minority Over-sampling Technique*) o técnicas más recientes basadas en modelos generativos o LLMs (*Large Language Models*) permiten obtener ejemplos artificiales a partir de un conjunto de datos original. Estas técnicas son especialmente útiles cuando las clases minoritarias son extremadamente escasas, pero dependen de la calidad y consistencia de los ejemplos generados.

Estas técnicas no han sido aplicadas en el presente trabajo, ya que la ponderación de clases ha sido suficiente para mejorar los resultados en las clases menos representadas sin alterar la distribución original del dataset ni introducir datos sintéticos. No obstante, su aplicación futura podría complementar la estrategia actual y aportar mejoras adicionales en contextos con mayor desbalance o menor tamaño de muestra.

4.6 Tokenización con DistilBERT

En el ámbito del PLN, uno de los pasos fundamentales antes de entrenar cualquier modelo es tokenizar los textos de entrada. Este proceso consiste en convertir los mensajes en secuencias numéricas que el modelo pueda interpretar. Para ello, cada oración es dividida en fragmentos denominados *tokens*, que pueden ser palabras, subpalabras o incluso caracteres, dependiendo del tipo de tokenizador utilizado [28].

En modelos basados en la arquitectura Transformer, como BERT o DistilBERT, esta tokenización debe seguir una estructura específica, incluyendo:

- La división del texto en *tokens* según un vocabulario predefinido.
- La conversión de cada *token* en un índice numérico correspondiente a su posición en ese vocabulario.
- La adición de tokens especiales, como [*CLS*] al inicio y [*SEP*] al final de cada secuencia, necesarios para tareas de clasificación.
- El relleno (*padding*) de las secuencias más cortas y la truncación de aquellas que superen la longitud máxima admitida por el modelo (*max_seq_length*), para asegurar que todas tengan el mismo tamaño.

Sin embargo, en este trabajo no ha sido necesario realizar este preprocesamiento de forma manual. Al utilizar la librería SimpleTransformers, el proceso de tokenización se realiza de forma completamente automatizada. Internamente, SimpleTransformers emplea el tokenizador oficial de DistilBERT, disponible en la biblioteca transformers de Hugging Face [4][5]. Este tokenizador viene adaptado al modelo preentrenado utilizado, lo que garantiza que los textos se procesan correctamente siguiendo el formato que el modelo espera.

Gracias a esta automatización, el pipeline de entrenamiento se simplifica considerablemente, lo que permite asignar mayor atención al diseño experimental, la selección de hiperparámetros y el análisis de resultados, sin preocuparse por los detalles técnicos del preprocesamiento textual.

4.7 Elección de hiperparámetros del modelo

El ajuste de hiperparámetros es una parte esencial del entrenamiento en modelos de aprendizaje profundo, ya que determina aspectos fundamentales como la tasa de

aprendizaje, el tamaño del lote o el número de épocas de entrenamiento. Dado que encontrar la combinación óptima de valores puede requerir muchas pruebas, se ha utilizado la plataforma Wandb para realizar una búsqueda automatizada y eficiente [9].

4.7.1 Configuración de hiperparámetros óptimos

La búsqueda de hiperparámetros se llevó a cabo mediante un sweep aleatorio (*random search*), en el que se probaron múltiples configuraciones de forma automática. Se ha demostrado que esta técnica es más efectiva que métodos más costosos como *grid search* [37].

Como se muestra en la Figura 8, se ha definido un espacio de búsqueda sobre tres hiperparámetros fundamentales que influyen directamente en el rendimiento y la estabilidad del modelo:

- Tasa de aprendizaje (*learning_rate*): determina qué tan rápido el modelo actualiza sus pesos durante el entrenamiento. Con valores bajos obtenemos un aprendizaje más lento pero estable, mientras que valores altos permiten una convergencia más rápida con riesgo de inestabilidad. En este caso, se permitió que la tasa variara entre $1x10^{-6}$ y $1x10^{-4}$ siguiendo una distribución uniforme continua. Este rango es común en tareas de fine-tuning con modelos Transformer [38].
- Tamaño de lote (*train_batch_size*): indica el número de muestras a procesar antes de actualizar los parámetros internos del modelo. Los lotes pequeños producen más ruido, pero generan mejor generalización; los lotes grandes aceleran el entrenamiento, pero son más costosos en términos de memoria. Se establecieron valores de potencia de dos, discretos, de 2 a 64, adaptados a los recursos computacionales disponibles [38].
- Número de épocas (*num_train_epochs*): representa cuántas veces el modelo recorre el conjunto completo de datos de entrenamiento. Si se asignan pocas épocas puede llevar a un entrenamiento incompleto, mientras que un número excesivo puede provocar sobreajuste (*overfitting*). En este proyecto, se fijó un rango entre 5 y 15 épocas, seleccionando valores enteros aleatorios dentro de este intervalo en cada ejecución del sweep.

Este diseño permite evaluar el impacto de distintas configuraciones sobre el rendimiento del modelo sin necesidad de realizar ajustes manuales, y facilita la exploración del espacio de búsqueda de manera eficiente gracias al sistema automatizado de ejecución y seguimiento proporcionado por Wandb [9].

Además, para evaluar objetivamente cada combinación de hiperparámetros, se definió como métrica principal de optimización el *accuracy* (precisión), que mide la proporción de predicciones correctas sobre el total de muestras evaluadas. Esta métrica se utilizó como criterio de selección del mejor modelo dentro del sweep, indicando qué configuración generalizaba mejor en el conjunto de validación.

Junto a ella, se registró también de forma complementaria el coeficiente de correlación de Matthews (*mcc*), una métrica más equilibrada en contextos con clases desbalanceadas, ya que tiene en cuenta verdaderos y falsos positivos y negativos [31]. Este coeficiente devuelve un valor entre –1 y +1, donde 1 indica una clasificación perfecta, 0 equivale a un clasificador al azar y –1 refleja un comportamiento totalmente inverso al deseado. Se considera que un *mcc* por encima de 0,7 es indicativo de una capacidad de discriminación muy alta, incluso en el caso de clases con escasa representación; valores entre 0,3 y 0,7 indican un rendimiento moderado que deja margen de mejora, y si su valor se encuentra por debajo de 0,3 significa que el modelo ha superado el rendimiento por mera casualidad o está muy desequilibrado hacia la clase mayoritaria. En este trabajo, el *mcc* se empleó como métrica secundaria de evaluación y también como criterio para el mecanismo de *early stopping*, deteniendo el entrenamiento cuando no se detectaban mejoras consistentes [34].

Gracias al seguimiento automático de esta métrica junto con otras en la plataforma Wandb, fue posible evaluar y comparar de manera precisa el comportamiento del modelo en diferentes ejecuciones, facilitando así la selección de los hiperparámetros óptimos.

```
# Configuration for W&B hyperparameter sweep
sweep_config = {
    "method": "random",  # Random search over the parameter space
    "metric": {"name": "accuracy", "goal": "maximize"},  # Optimize for accuracy
    "parameters": {
        "learning_rate": {"distribution": "uniform", "min": 1e-6, "max": 1e-4},  # Range for learning rate
        "train_batch_size": {"values": [2, 4, 8, 16, 32, 64]},  # Different batch sizes to try
        "num_train_epochs": {"distribution": "int_uniform", "min": 5, "max": 15}  # Range for number of epochs
    }
}

# Initialize the sweep in W&B under the "EMOCIONES" project
sweep_id = wandb.sweep(sweep_config, project="EMOCIONES")
```

Figura 8: Configuración e inicialización del sweep aleatorio.

A continuación, se diseñó la función de entrenamiento personalizada que sería ejecutada automáticamente por Wandb en cada iteración del sweep. Esta función define el proceso completo: desde la inicialización del modelo hasta el registro de resultados finales.

Tal como se observa en la Figura 9, se inicia una nueva ejecución de W&B con *wandb.init()* y se cargan los hiperparámetros asignados aleatoriamente para esa ejecución a través de *wandb.config*. Estos parámetros se inyectan directamente en la configuración del modelo *(learning_rate, train_batch_size y num_train_epochs*), lo que permite ajustar automáticamente cada experimento sin intervención manual. La configuración interna del modelo incluye:

- Evaluación continua en el conjunto de validación (*evaluate_during_training=True*).
- Activación del *early stopping*, deteniendo el entrenamiento si no hay mejoras tras tres evaluaciones consecutivas del mcc.
- Establecimiento de *accuracy* como métrica principal para seleccionar el mejor modelo (*metric_for_best_model*) y guiar el proceso de optimización, mientras que mcc se utiliza como métrica complementaria para monitorizar la calidad en contextos desbalanceados.
- Desactivación del guardado de checkpoints intermedios para reducir consumo de espacio.
- Vinculación del experimento con el proyecto “EMOCIONES” en la plataforma Wandb, para almacenar los resultados automáticamente.

Tras cada entrenamiento, el modelo se evalúa sobre el conjunto de validación, y se registran las dos métricas clave: *accuracy* y *mcc*. Estos resultados quedan asociados a cada ejecución del sweep, lo que permite compararlas posteriormente.

Finalmente, el sweep se ejecuta 30 veces utilizando *wandb.agent*, lo que permite explorar distintas combinaciones de hiperparámetros de forma automatizada y reproducible.

```python
# Define the training function that will be called by W&B agent during the sweep
def train(train_df=train_df, val_df=val_df):
    with wandb.init():  # Start a new W&B run
        config = wandb.config  # Load sweep-configured hyperparameters

        # Initialize a DistilBERT classification model with 6 output labels
        model = ClassificationModel(
            "distilbert",  # Model type
            model_name,    # Pretrained model name
            num_labels=6,  # Number of emotion classes
            args={
                # Hyperparameters from sweep configuration
                "learning_rate": config.learning_rate,
                "train_batch_size": config.train_batch_size,
                "num_train_epochs": config.num_train_epochs,
                # Output settings
                "overwrite_output_dir": True,
                "output_dir": f"outputs/{wandb.run.name}",  # Directory for all model files
                "best_model_dir": f"outputs/{wandb.run.name}/best_model",  # Save best model separately
                # Evaluation settings
                "reprocess_input_data": True,
                "evaluate_during_training": True,
                "use_early_stopping": True,
                "early_stopping_patience": 3,  # Stop if no improvement after 3 evaluations
                "early_stopping_metric": "mcc",
                "early_stopping_metric_minimize": False,  # We want to maximize MCC
                "metric_for_best_model": "mcc",
                "evaluation_metric": "mcc",
                # Avoid saving unnecessary checkpoints
                "save_model": False,
                "save_model_every_epoch": False,
                "save_eval_checkpoints": False,
                "save_steps": -1,
                # Disable multiprocessing to prevent compatibility issues in Colab
                "use_multiprocessing": False,
                "use_multiprocessing_for_evaluation": False,
                # Link with the W&B project
                "wandb_project": "EMOCIONES",
            },
            ignore_mismatched_sizes=True  # Allow model to adapt if output layer size differs
        )

        # Train the model and evaluate on validation data during training
        model.train_model(train_df, eval_df=val_df, compute_metrics=compute_metrics)

        # Final evaluation on validation set
        preds, _ = model.predict(val_df["text"].tolist())
        acc = accuracy_score(val_df["labels"], preds)
        mcc = matthews_corrcoef(val_df["labels"], preds)

        # Log final metrics to W&B
        wandb.log({"accuracy": acc, "mcc": mcc})

# Launch the hyperparameter sweep: run the training function 30 times with different configs
wandb.agent(sweep_id, train, count=30)
```

Figura 9: Código para lanzar el sweep que indica los hiperparámetros óptimos.

4.7.2 Visualización y selección de los parámetros óptimos en W&B

Aunque los resultados de cada entrenamiento pueden consultarse directamente desde el cuaderno de Python, realizar un análisis comparativo completo desde ahí resulta poco práctico cuando se han llevado a cabo múltiples ejecuciones. Por ello, se ha utilizado la interfaz web de Weights & Biases (Wandb) para visualizar, comparar y seleccionar la mejor combinación de hiperparámetros obtenida durante el sweep [9].

Al finalizar cada entrenamiento, Wandb almacena automáticamente las métricas obtenidas (como accuracy y mcc), junto con los hiperparámetros utilizados, dentro del

proyecto correspondiente. Para acceder, basta con iniciar sesión en la plataforma a través de https://wandb.ai con una cuenta personal o académica. Una vez dentro, desde la pestaña Home, se muestra el listado de proyectos creados por el usuario, como se ilustra en la Figura 10.

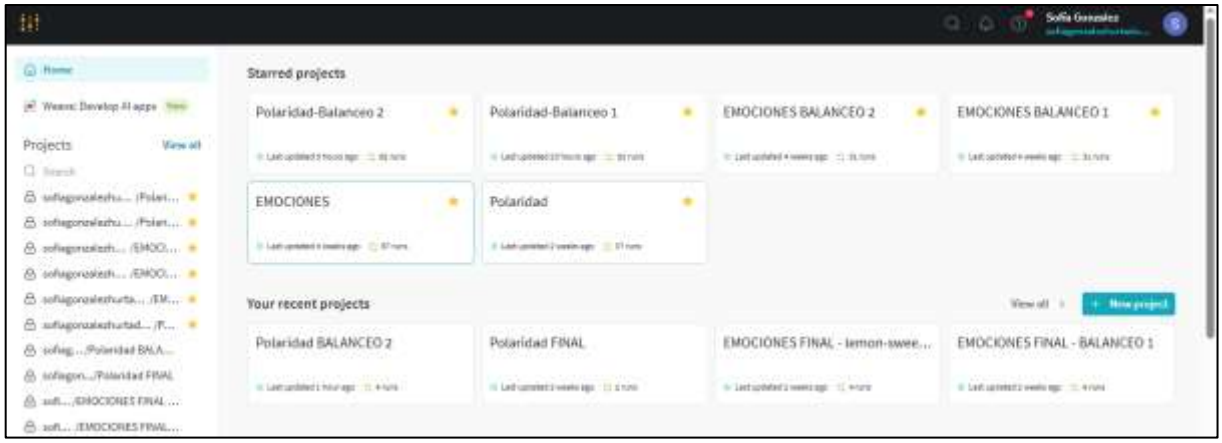

Figura 10: Pestaña Home de Wandb.

En este caso particular, se han creado seis proyectos distintos, correspondientes a las dos tareas (emociones y polaridad) y las tres configuraciones empleadas en cada una de ellas, es decir:

- Sin balanceo de clases.
- Con balanceo mediante la primera técnica (por intervalos).
- Con balanceo mediante la segunda técnica (proporcional inversa).

Cada proyecto agrupa las ejecuciones relacionadas con su configuración específica. Dentro de cada uno, se accede a la pestaña Sweeps, donde se almacenan todas las simulaciones ejecutadas automáticamente por Wandb. Por ejemplo, en el proyecto denominado “EMOCIONES”, se localizó la simulación identificada como “bs3zsg3l”, correspondiente a la búsqueda de hiperparámetros para la clasificación de emociones. Esta simulación constó de un total de 30 ejecuciones, cada una con una combinación distinta de hiperparámetros generada aleatoriamente (ver Figura 11).

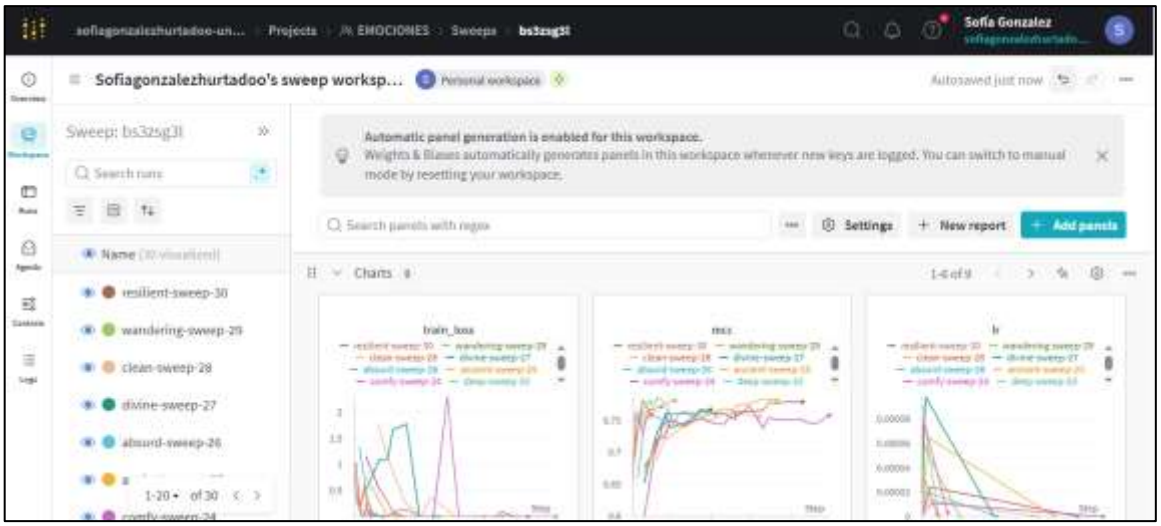

Figura 11: Pestaña de la simulación “bs3zsg3l”, dentro del proyecto de “EMOCIONES”.

Una vez dentro del espacio de trabajo de la simulación, Wandb proporciona gráficos interactivos que muestran la evolución de las métricas de evaluación a lo largo de las distintas ejecuciones (ver Figura 12).

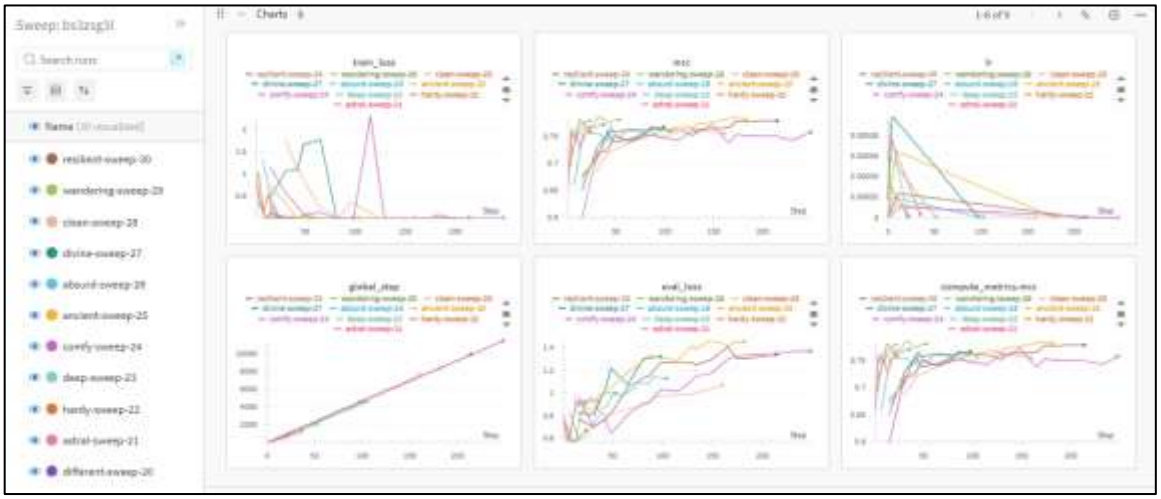

Figura 12: Gráficas de la simulación “bs3zsg3l”, dentro del proyecto de “EMOCIONES”.

Una de las herramientas más útiles que proporciona la plataforma W&B para el análisis de los resultados de un sweep es la visualización de hiperparámetros mediante gráficas de coordenadas paralelas, como se muestra en la Figura 13. Esta gráfica representa en paralelo los tres hiperparámetros ajustados en el sweep: *learning_rate* (tasa de aprendizaje), *num_train_epochs* (número de épocas), y *train_batch_size* (tamaño de lote), y los relaciona visualmente con la métrica de rendimiento objetivo: *accuracy* (precisión).

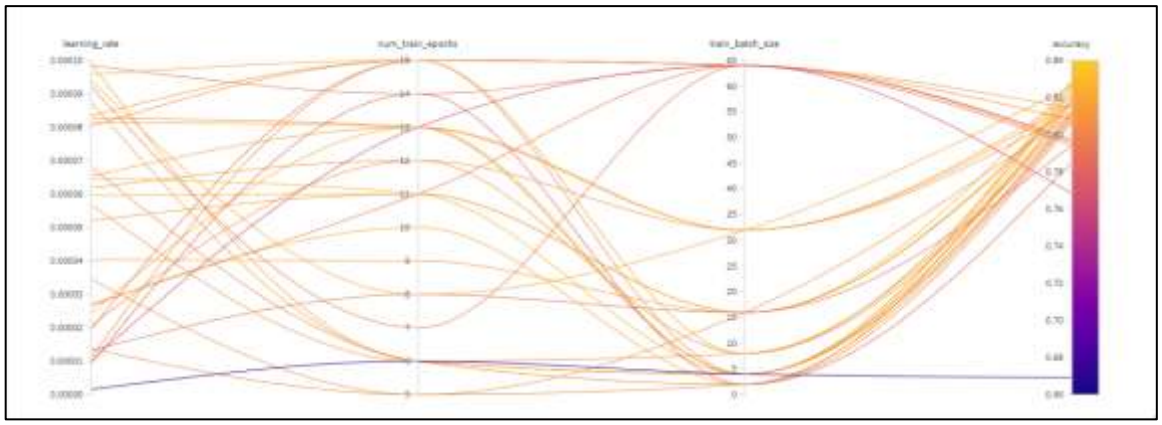

Figura 13: Gráfica de hiperparámetros para el proyecto "EMOCIONES".

Cada línea de la gráfica representa una ejecución específica del modelo, es decir, una combinación única de hiperparámetros. El color las líneas indica el valor de *accuracy* alcanzado por esa ejecución, siguiendo la escala de la derecha: los colores más cálidos (naranja-amarillo) indican una mayor precisión, mientras que los más fríos (azul-morado) indican un rendimiento inferior.

Esta visualización permite detectar patrones de rendimiento y entender mejor cómo afectan ciertos rangos de hiperparámetros al resultado final. Por ejemplo, en esta simulación se observa que las configuraciones con tamaños de lote pequeños o intermedios (entre 8 y 32), junto con tasas de aprendizaje entre 0.00007 y 0.000008, tienden a alcanzar los valores más altos de precisión.

En base a esta información, se seleccionó la combinación de hiperparámetros que maximiza la precisión dentro del sweep, criterio que se ha utilizado en este proyecto como referencia principal para elegir el mejor modelo.

4.7.3 Ejecuciones de sweep realizadas y resumen de configuraciones óptimas

Para encontrar los hiperparámetros más adecuados en cada escenario, se han llevado a cabo seis barridos aleatorios de hiperparámetros (sweeps).

En la Figura 14 se muestra la mejor combinación obtenida para la tarea de clasificación de emociones sin aplicar balanceo de clases. La ejecución destacada en naranja alcanzó la mayor precisión (accuracy), con un valor de 0.8293. Los hiperparámetros seleccionados fueron:

- Tasa de aprendizaje (*learning_rate*): 0.00006476
- Número de épocas (*num_train_epochs*): 11
- Tamaño de lote (*train_batch_size*): 4

Esta configuración, identificada automáticamente por Wandb como la de mayor rendimiento en el conjunto de validación, fue utilizada para entrenar el modelo final en la tarea de emociones sin balanceo. Este ejemplo demuestra la utilidad de herramientas como Wandb para guiar el ajuste de modelos, facilitando la selección de la configuración óptima de manera clara, informada y justificada.

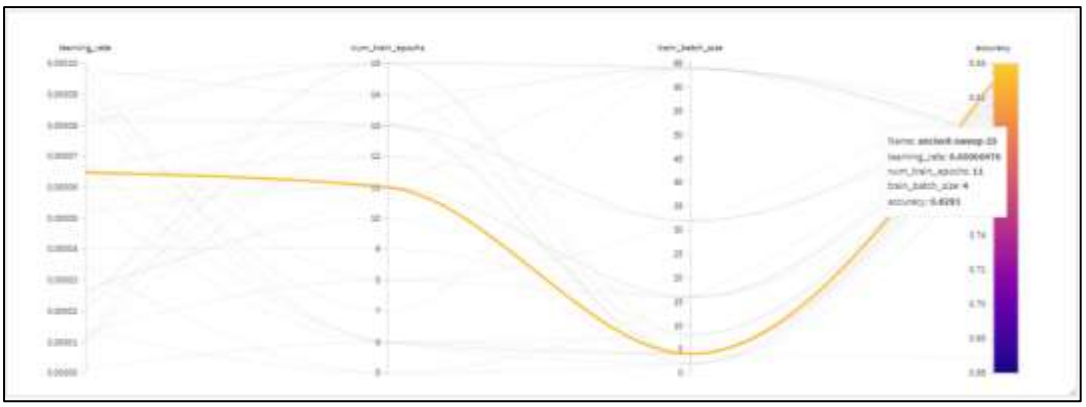

Figura 14: Hiperparámetros óptimos para el proyecto "EMOCIONES".

Tras analizar en el apartado anterior los parámetros óptimos para el proyecto de emociones sin balanceo, vamos a observar el caso de la tarea de clasificación de emociones con la primera técnica de balanceo (ponderación por intervalos), para la que se ejecutó un sweep aleatorio con múltiples combinaciones de hiperparámetros. La Figura 15 muestra el análisis visual de los resultados, donde cada línea representa una ejecución distinta y su color indica el valor de accuracy.

A la derecha del sweep, se destaca la mejor ejecución obtenida, correspondiente al experimento identificado como *"distinctive-sweep-6"*, que alcanzó una precisión de 0.8338. Los valores óptimos de hiperparámetros para esta configuración fueron: tasa de aprendizaje (*learning_rate*): 0.00006799, número de épocas (*num_train_epochs*): 7 y amaño de lote (*train_batch_size*): 8.

Esta configuración fue seleccionada como la más eficaz para el entrenamiento definitivo del modelo en este escenario.

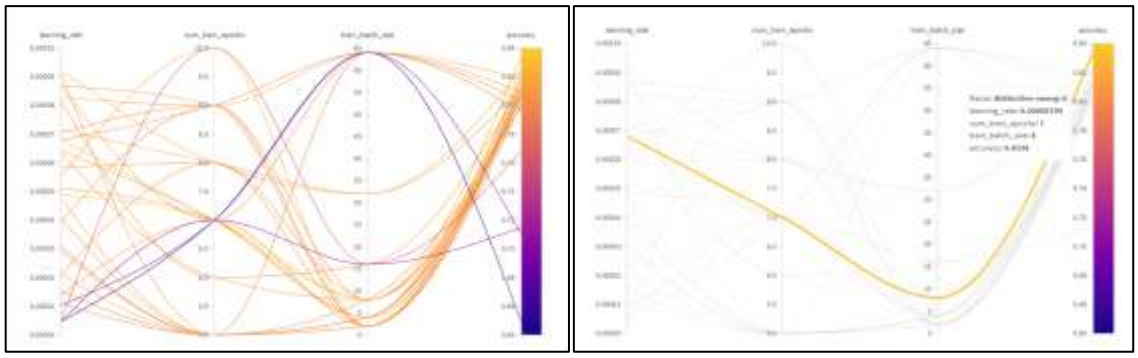

Figura 15: Hiperparámetros óptimos para emociones con balanceo 1.

En el caso de la tarea de clasificación de emociones con la segunda técnica de balanceo (ponderación proporcional inversa), se observa que la mejor ejecución obtenida, correspondiente al experimento identificado como *"lemon-sweep-11",* alcanzó una precisión de 0.8293, cuyos valores óptimos de hiperparámetros en este caso fueron: tasa de aprendizaje (*learning_rate*): 0.00007278, número de épocas (*num_train_epochs*): 6, tamaño de lote (*train_batch_size*): 16 (ver Figura 16).

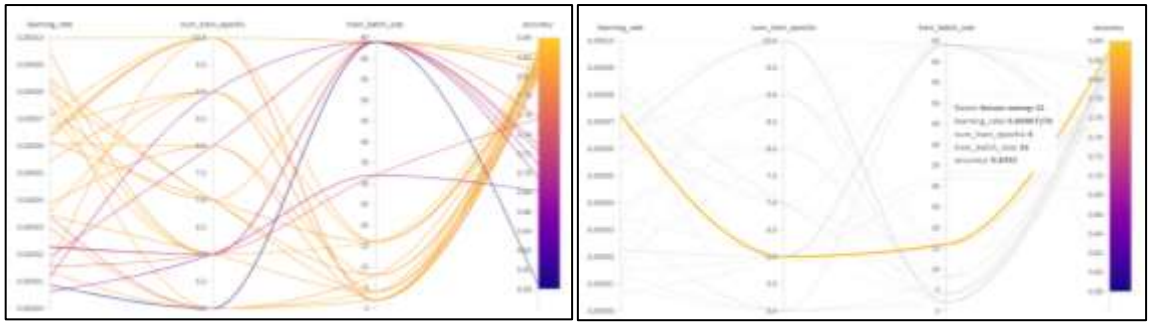

Figura 16: Hiperparámetros óptimos para emociones con balanceo 2.

En cuanto a la tarea de clasificación de polaridad, también se llevaron a cabo tres barridos independientes, correspondientes a los tres escenarios experimentales: sin balanceo, con ponderación por intervalos y con ponderación proporcional inversa.

Para el caso sin balanceo, el sweep permitió identificar como mejor ejecución aquella etiquetada como *"balmy-sweep-15",* con una precisión final de 0.875. La configuración óptima en este caso fue: tasa de aprendizaje (*learning_rate*): 0.00007583, número de épocas (*num_train_epochs*): 13 y tamaño de lote *(train_batch_size*): 8. Esta combinación mostró un rendimiento excelente, y fue seleccionada para entrenar el modelo final en este escenario (ver Figura 17).

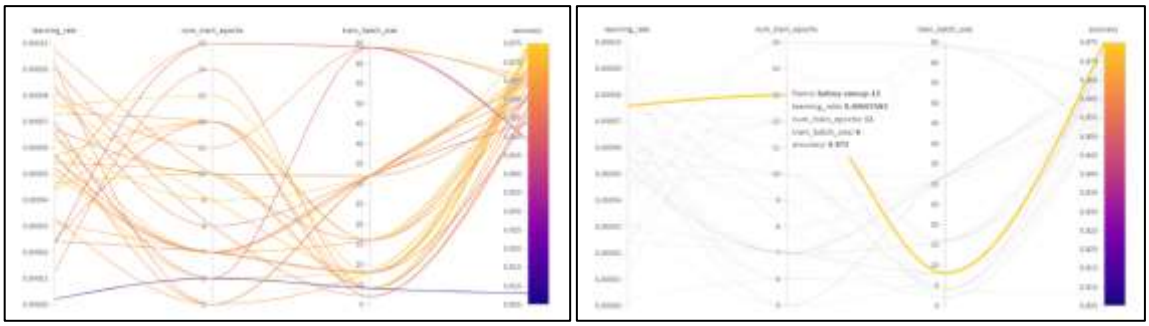

Figura 17: Hiperparámetros óptimos para polaridad sin balanceo.

En el escenario con ponderación por intervalos, la mejor ejecución correspondió al sweep "*olive-sweep-10"*, que alcanzó una *accuracy* de 0.8796, ligeramente superior a la configuración sin pesos. Esta ejecución se caracterizó por: tasa de aprendizaje: 0.00005103, número de épocas: 14 y tamaño de lote: 2. Este resultado sugiere que, aunque con un *batch size* muy pequeño, el modelo es capaz de beneficiarse de la ponderación por intervalos para mejorar su rendimiento (ver Figura 18).

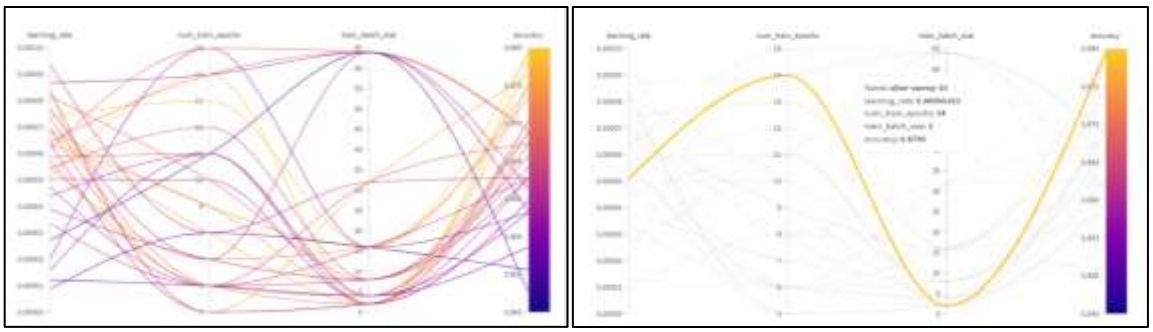

Figura 18: Hiperparámetros óptimos para polaridad con balanceo 1.

Por último, en el caso de ponderación proporcional inversa, el sweep identificó la ejecución "*cerulean-sweep-10*" como la más precisa, alcanzando una accuracy de 0.8765. La configuración asociada fue: tasa de aprendizaje: 0.00001619, número de épocas: 15 y tamaño de lote: 4 (ver Figura 19).

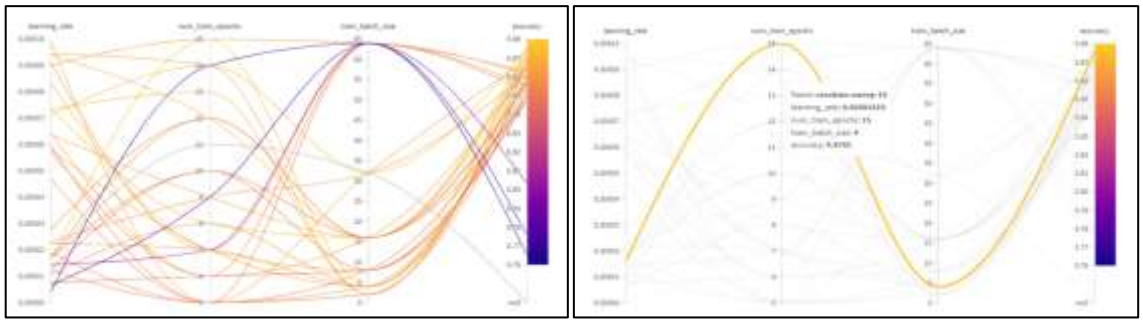

Figura 19: Hiperparámetros óptimos para polaridad con balanceo 2.

A continuación, se resume la configuración óptima identificada para cada uno de los seis escenarios evaluados, según los resultados de la Tabla 5.

Tarea	***Técnica***	***learning_rate***	***num_train_epochs***	***train_batch_size***	***accuracy***
Emociones	***Sin balanceo***	*0.00006476*	*11*	*4*	*0.8293*
Emociones	***Balanceo 1***	*0.00006799*	*7*	*8*	*0.8338*
Emociones	***Balanceo 2***	*0.00007278*	*6*	*16*	*0.8293*
Polaridad	***Sin balanceo***	*0.00007583*	*13*	*8*	*0.875*
Polaridad	***Balanceo 1***	*0.00005103*	*14*	*2*	*0.8796*
Polaridad	***Balanceo 2***	*0.00001619*	*15*	*4*	*0.8765*

Tabla 5: Resumen de los hiperparámetros óptimos.

La Tabla 5 muestra que, para clasificación de emociones, los modelos alcanzaron su mejor rendimiento con menos épocas de entrenamiento (6–7) en comparación con los de polaridad, que necesitaron entre 13 y 15 épocas. Por otro lado, las tasas de aprendizaje (*learning rate*) se mantuvieron similares en todos los escenarios, manteniendo en un rango cercano a 10^{-5}, mientras que el tamaño de lote varió en cada caso para adaptarse a las necesidades específicas de cada técnica de balanceo.

4.8 Entrenamiento y evaluación del modelo

Una vez seleccionada la mejor combinación de hiperparámetros para cada modelo (véase Apartado 4.7), se procedió a entrenar dichos modelos finales.

En el primer bloque de código (Figura 20) se define el diccionario *best_args*, que contiene los hiperparámetros y ajustes específicos del entrenamiento. En el caso del modelo de emociones sin balanceo, se establecen los siguientes valores:

- *num_train_epochs*: 11
- *learning_rate*: 0.00006476
- *train_batch_size*: 4

Además, se activan funcionalidades clave para optimizar el proceso de entrenamiento:

- Se habilitó la evaluación continua *(evaluate_during_training = True)* junto a la detección de parada anticipada *(early_stopping = True)* con una paciencia de 3 épocas para interrumpir el proceso si no hay mejoras.
- El coeficiente de Correlación de Matthews se empleó tanto para decidir cuándo detener el entrenamiento como para seleccionar el mejor modelo alcanzado.
- Para ahorrar espacio, se desactivó el guardado de *checkpoints* intermedios, conservando únicamente la versión final del modelo (*save_model = True).*
- Se desactivó el multiprocesamiento para garantizar la compatibilidad con el entorno de ejecución de Google Colab.

Además de los hiperparámetros, se implementó una función llamada *compute_metrics* (Figura 21), que calcula las métricas relevantes al finalizar cada época de entrenamiento. Esta función recibe como entrada las predicciones del modelo (*preds*) y las etiquetas reales (*labels*). Si las predicciones tienen forma vectorial, primero se convierten en etiquetas predichas seleccionando la clase con la puntuación más alta mediante *argmax*.

Una vez convertidas las predicciones en etiquetas, se calcula la precisión (*accuracy_score*) comparando las etiquetas reales con las predichas. Además, se calcula también el coeficiente de correlación de Matthews (mcc), mediante la función *matthews_corrcoef*.

```python
from simpletransformers.classification import ClassificationModel
from sklearn.metrics import accuracy_score, matthews_corrcoef
import wandb
# Start a new run in Weights & Biases (W&B) for experiment tracking
wandb.init(project="EMOCIONES FINAL")

# Define the best hyperparameters obtained from the sweep
best_args = {
    "reprocess_input_data": True,           # Preprocess input data before training
    "overwrite_output_dir": True,           # Overwrite previous outputs if they exist
    "fp16": False,                          # Disable mixed-precision training (for compatibility)
    "num_train_epochs": 11,                 # Number of training epochs
    "train_batch_size": 4,                  # Small batch size (helps generalization, suitable for low-memory)
    "learning_rate": 6.476e-5,              # Learning rate selected from the sweep
    "output_dir": "modelo_final/",          # Path to save the final model
    # Enable evaluation and early stopping during training
    "evaluate_during_training": True,
    "use_early_stopping": True,
    "early_stopping_patience": 3,           # Stop training after 3 evaluations without improvement
    "early_stopping_metric": "mcc",         # Use MCC (Matthews Correlation Coefficient) for early stopping
    "early_stopping_metric_minimize": False, # We want to maximize MCC
    "metric_for_best_model": "mcc",         # Save the best model based on this metric
    # Save the final model but avoid unnecessary checkpoints
    "save_model": True,
    "save_eval_checkpoints": False,
    "save_steps": -1,                       # Don't save checkpoints every N steps
    # W&B integration
    "wandb_project": "EMOCIONES FINAL",
    # Avoid multiprocessing to ensure compatibility in certain environments (e.g. Google Colab)
    "use_multiprocessing": False,
    "use_multiprocessing_for_evaluation": False,
}
```

Figura 20: Código que define los mejores parámetros para emociones sin class weighting.

Ambas métricas (*accuracy y mcc*) se devuelven en un diccionario y se utilizan automáticamente durante el entrenamiento, tanto para la monitorización del rendimiento como para aplicar el mecanismo de parada anticipada (*early stopping*) y seleccionar el mejor modelo.

```python
from sklearn.metrics import accuracy_score, matthews_corrcoef

# Custom evaluation metrics used during training and validation
def compute_metrics(preds, labels):
    if len(preds.shape) > 1:
        preds = preds.argmax(axis=1)  # Convert logits to predicted class labels
    acc = accuracy_score(labels, preds)  # Accuracy
    mcc = matthews_corrcoef(labels, preds)  # Matthews Correlation Coefficient
    return {"accuracy": acc, "mcc": mcc}
```

Figura 21: Función llamada compute_metrics.

Una vez definidos los parámetros y las métricas, se instancia el modelo mediante la clase *ClassificationModel* y se lanza el entrenamiento mediante el método *train_model*, el cuál entrena el modelo sobre el conjunto *train_df*, lo evalúa periódicamente sobre *val_df*, y calcula las métricas definidas mediante la función *compute_metrics* (ver Figura 22).

```
# Initialize the classification model with the selected pretrained checkpoint
model = ClassificationModel(
    "distilbert",  # Model type
    "francisco-perez-sorrosal/dccuchile-distilbert-base-spanish-uncased-finetuned-with-spanish-tweets-clf-cleaned-ds",
    num_labels=6,  # Number of emotion classes
    args=best_args,
    use_cuda=False,  # Use GPU if available
    ignore_mismatched_sizes=True         # Allow adaptation if classifier layer sizes differ
)

# Train the model using the training and validation sets
model.train_model(train_df, eval_df=val_df, compute_metrics=compute_metrics)
```

Figura 22: Código que inicializa y entrena el modelo.

Como se explica en el Apartado 4.3, se seleccionó como base el modelo preentrenado *francisco-perez-sorrosal/dccuchile-distilbert-base-spanish-uncased-finetuned-with-spanish-tweets-clf-cleaned-ds*, disponible en la plataforma Hugging Face.

Una vez entrenado el modelo con la configuración óptima de hiperparámetros, se procedió a su evaluación final utilizando el conjunto de *test*, el cual no fue utilizado durante la fase de entrenamiento ni validación. Esta evaluación permite obtener una estimación clara y realista del rendimiento general del modelo y su capacidad de generalización.

Para llevarla a cabo, se implementó la función *EvaluateModel* (Figura 23), que realiza las siguientes acciones:

1) Predicción sobre el conjunto de test: utiliza el método *predict()* para obtener las predicciones *(y_pred*) sobre los comentarios del conjunto *test_df*.
2) Cálculo de la matriz de confusión: genera una matriz de confusión normalizada, comparando las etiquetas verdaderas (*y_true*) con las predichas (ver Figura 24). Esta matriz se representa gráficamente mediante un mapa de calor (*heatmap*), donde se pueden observar los aciertos y errores por clase.
3) Cálculo de métricas de evaluación:
    - Accuracy: es el porcentaje total de aciertos del modelo.
    - F1-score: es la media armónica entre precisión y recall, calculada con promedio macro para darle la misma importancia a todas las clases.
    - Precisión: es el porcentaje de predicciones correctas sobre el total de predicciones positivas.
    - Recall: Capacidad del modelo para detectar todos los ejemplos de cada clase presente en el conjunto de test.
4) Informe de clasificación (*classification_report*): imprime un desglose detallado de métricas por clase (precisión, recall y F1-score), lo cual permite

evaluar qué clases están mejor o peor clasificadas por el modelo (ver Figura 25).

```python
# Evaluation function
def EvaluateModel(yTrue, yPredict):
    # Use the correct order of label indices
    labels = list(range(len(class_names)))

    # Compute confusion matrices
    cm = confusion_matrix(yTrue, yPredict, labels=labels, normalize='true')
    cm_ = confusion_matrix(yTrue, yPredict, labels=labels)

    # Plot confusion matrix with class names
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm_, annot=True, fmt='d', cmap='Blues',
                xticklabels=class_names,
                yticklabels=class_names)
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.title('Confusion Matrix')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

    # Print evaluation metrics
    print("Accuracy:", accuracy_score(yTrue, yPredict))
    print("F1-score:", f1_score(yTrue, yPredict, average="macro"))
    print("Precision:", precision_score(yTrue, yPredict, average="macro"))
    print("Recall:", recall_score(yTrue, yPredict, average="macro"))

    # Print detailed classification report
    print("\nClassification Report:\n", classification_report(
        yTrue, yPredict, target_names=class_names))

# Get predictions from the model on the test set
y_pred, raw_outputs = model.predict(test_df["text"].tolist())

# Evaluate the model using true labels and predictions
EvaluateModel(test_df["labels"], y_pred)
```

Figura 23: Función EvaluateModel.

Este proceso de evaluación proporciona una visión completa del rendimiento del modelo más allá de la precisión global, especialmente relevante en tareas con clases desbalanceadas o sensibles como la detección de emociones.

4.8.1 Análisis de los resultados para emociones

En este apartado se comparan los resultados obtenidos por los tres modelos entrenados para la tarea de clasificación de emociones: sin balanceo, con balanceo 1 (ponderación por intervalos) y con balanceo 2 (ponderación proporcional inversa).

Comenzamos por el análisis del modelo sin aplicar ninguna técnica de balanceo, cuyos resultados están recogidos en las Figuras 24 y 25.

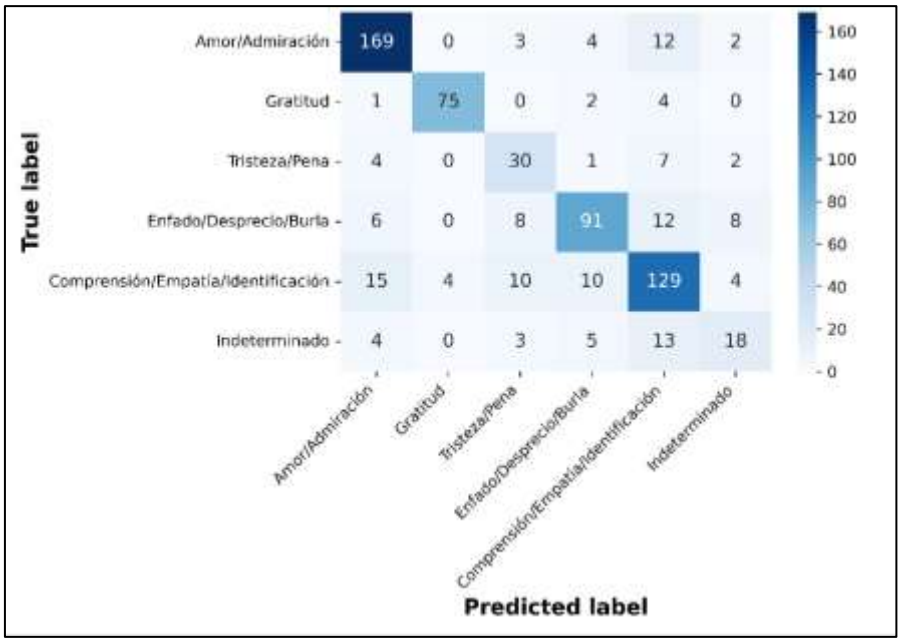

Figura 24: Matriz de confusión para emociones sin balanceo.

La matriz de confusión ofrece una visión clara de los aciertos y errores del modelo. En este caso se observa como las clases *"Amor/Admiración*" y "*Gratitud*" tienen predicciones muy concentradas en la diagonal, lo que denota alta fiabilidad del modelo en estos casos. Por otro lado, la categoría *Indeterminado* muestra una dispersión notable, confundida con frecuencia con *Comprensión/Empatía* y *Amor/Admiración*, lo que indica que al modelo le cuesta distinguirla como una categoría propia. También se aprecian confusiones entre *Tristeza* y *Enfado*, algo esperable dado que ambas pueden compartir matices emocionales similares negativos o de malestar.

```
Accuracy:     0.7804878048780488
F1-score:     0.7307183137245327
Precision:    0.7362839889766867
Recall:       0.7304217772554936

Classification Report:

Label                                Precision    Recall    F1-score   Support
-------------------------------------------------------------------------------
0 - Amor/Admiración                       0.85      0.89        0.87       190
1 - Gratitud                              0.95      0.91        0.93        82
2 - Tristeza/Pena                         0.56      0.68        0.61        44
3 - Enfado/Desprecio/Burla                0.81      0.73        0.76       125
4 - Comprensión/Empatía/ID                0.73      0.75        0.74       172
5 - Indeterminado                         0.53      0.42        0.47        43
-------------------------------------------------------------------------------
accuracy                                                        0.78       656
macro avg                                 0.74      0.73        0.73       656
weighted avg                              0.78      0.78        0.78       656
```

Figura 25: Métricas de evaluación e informe de clasificación.

Los resultados de este primer modelo, mostrados en la Figura 25, reflejan un buen rendimiento general, con una *accuracy* global del 78 %, lo que significa que el modelo

acierta en 78 de cada 100 predicciones sin importar la clase. Además, mantiene un equilibrio razonable entre precisión y exhaustividad en la mayoría de las categorías. La precisión media (macro) es del 73.6 %.

La diferencia entre *accuracy* y precision se explica por lo que mide cada una: la *accuracy* refleja todos los aciertos del modelo en general, mientras que la precision evalúa qué tan fiables son esas predicciones para cada clase. Por ejemplo, la clase "*Gratitud*" presenta una precisión muy alta (0.95) y un F1-score excelente de 0.93, mostrando gran estabilidad en sus métricas. También destacan con buenos resultados las clases "*Amor/Admiración*" y "*Comprensión/Empatía*", ambas con F1-scores superiores a 0.74. en este sentido, la clase "*Enfado/Desprecio/Burla*", también presenta buenos resultados, alcanzando niveles de precisión del 0.81 y con valores de Recall y F1-Score por encima del 0.73.

Sin embargo, las clases con menor presencia en el entrenamiento sufren una caída en estas métricas. Por ejemplo, "*Tristeza/Pena*" solo aparece en un pequeño porcentaje del corpus, y aunque el modelo identifica bien la mayoría de los casos reales (recall = 0,68), su precision baja a 0,56 porque con frecuencia etiqueta como tristeza comentarios que pertenecen a otras categorías. De manera similar, "*Indeterminado*", que cuenta con muy pocas muestras y además es semánticamente difuso, presenta un F1-score de 0,47 y una precision del 0,53, reflejando la escasez de ejemplos concretos que ayuden al modelo a aprender sus patrones distintivos.

El modelo entrenado utilizando la técnica de balanceo 1 presenta buenos resultados (Figuras 26 y 27), considerando que las clases fueron clasificadas uniformemente con una precisión general del 78.65%.

Con el uso de dicha técnica, que otorga un peso de 1.5 a cada clase minoritaria (es decir, todas excepto "*Amor/Admiración*" y "*Comprensión/ Empatía/ Identificación*"), se logra una mejora general, especialmente en "*Tristeza/Pena*" (F1 de 0.65 en comparación con 0.61 sin balanceo) y un ligero incremento en " *Enfado* ".

Al aplicar esta técnica, que asigna un peso de 1.5 a todas las clases menos representadas (es decir, todas excepto "*Amor/Admiración*" y "*Comprensión/ Empatía/ Identificación*"), se observa una mejora general, especialmente en "*Tristeza/Pena*" (F1 de

0.65 frente a 0.61 sin balanceo), así como un ligero incremento en "*Enfado*". No obstante, la clase "*Indeterminado*" sufre una disminución en el F1-score, probablemente porque el modelo está sesgado por el aumento de peso asignado y comienza a especializarse demasiado sin tener suficientes muestras representativas, y, por lo tanto, se equivoca más a menudo. La ganancia global es modesta, pero el aprendizaje queda algo más repartido sin dañar a las categorías dominantes.

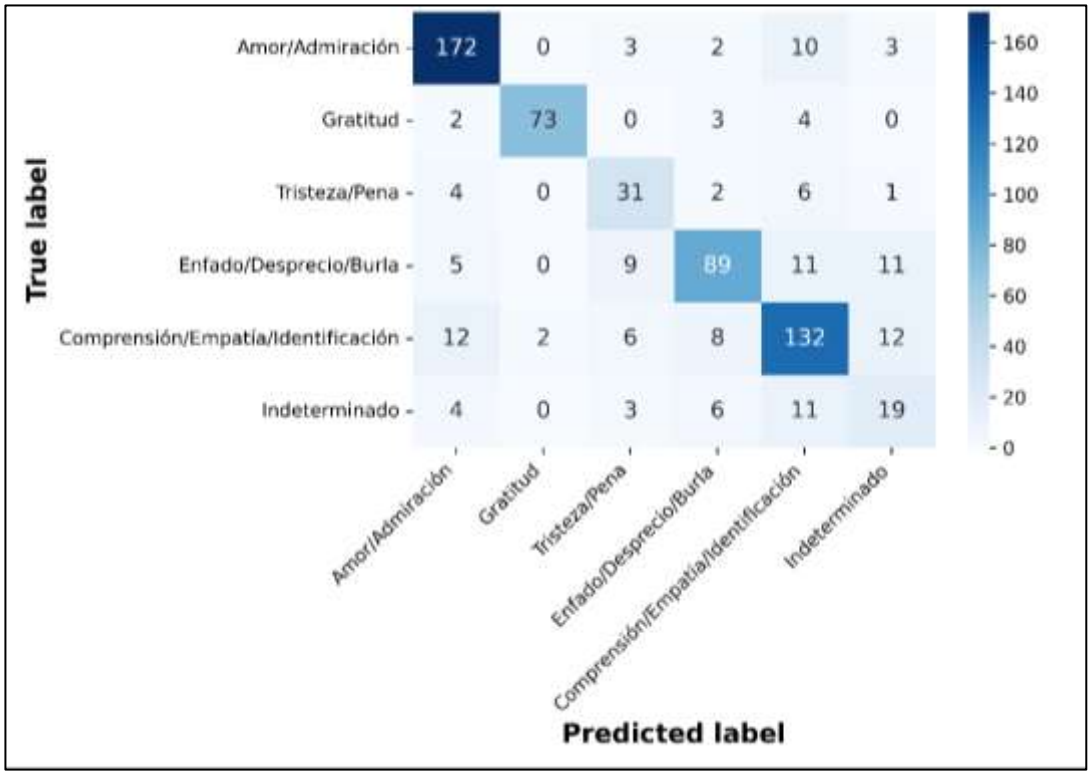

Figura 26: Matriz de confusión para emociones con balanceo 1.

```
Accuracy:     0.7865853658536586
F1-score:     0.7345845476874909
Precision:    0.7357606440890553
Recall:       0.7368924734101019

Classification Report:

Label                                  Precision    Recall    F1-score   Support
-------------------------------------------------------------------------------
0 - Amor/Admiración                         0.86      0.91        0.88       190
1 - Gratitud                                0.97      0.89        0.93        82
2 - Tristeza/Pena                           0.60      0.70        0.65        44
3 - Enfado/Desprecio/Burla                  0.81      0.71        0.76       125
4 - Comprensión/Empatía/Identificación      0.76      0.77        0.76       172
5 - Indeterminado                           0.41      0.44        0.43        43
-------------------------------------------------------------------------------
accuracy                                                          0.79       656
macro avg                                   0.74      0.74        0.73       656
weighted avg                                0.79      0.79        0.79       656
```

Figura 27: Métricas de evaluación e informe de clasificación para emociones con balanceo 1.

En cuanto al modelo entrenado con balanceo 2 (Figuras 28 y 29), este asigna pesos continuos que reflejan con mayor precisión el grado real de desbalance entre clases. Como resultado, se obtiene el mejor rendimiento global de los tres modelos evaluados, con una *accuracy* del 79.73 % y un F1-score macro de 0.7469.

La mayor ganancia se produce en la clase "*Tristeza/Pena*", que, al recibir uno de los pesos más altos (2.487), alcanza un F1-score de 0.71, muy superior al obtenido sin balanceo (0.61) o con balanceo 1 (0.65). También destaca "*Enfado/Desprecio/Burla*", cuyo peso intermedio (0.878) basta para que el F1-score pase de 0.76 a 0.78, lo que sugiere que la ganancia no depende solo de un peso elevado, sino de la claridad de los patrones lingüísticos.

En la clase "*Gratitud*", con un peso de 1.329 (inferior al del balanceo 1, 1.5), se observa un ligero descenso del F1-score de 0.93 a 0.91. Esta pérdida, aunque leve, indica que una penalización menor reduce marginalmente la capacidad del modelo para distinguir esta emoción con la misma precisión.

La clase "*Indeterminado*", a pesar de contar con el peso más alto (2.545), apenas mejora: su F1 pasa de 0.43 (balanceo 1) a 0.44 y sigue por debajo del modelo sin balanceo (0.47). Esta limitación se debe probablemente a la ambigüedad intrínseca de la etiqueta y a su solapamiento con otras emociones; un peso excesivo puede incluso propagar la confusión en lugar de corregirla.

De esta forma, la ponderación proporcional inversa consigue potenciar de manera clara las categorías más infrarrepresentadas y mantiene casi intacta la fiabilidad en las emociones mayoritarias. Aun así, el caso de "*Indeterminado*" demuestra que elevar mucho el peso de una clase ambigua no basta por sí solo: si los mensajes carecen de rasgos

distintivos, el modelo seguirá dudando entre etiquetas próximas, por muy alta que sea la penalización asignada a sus errores.

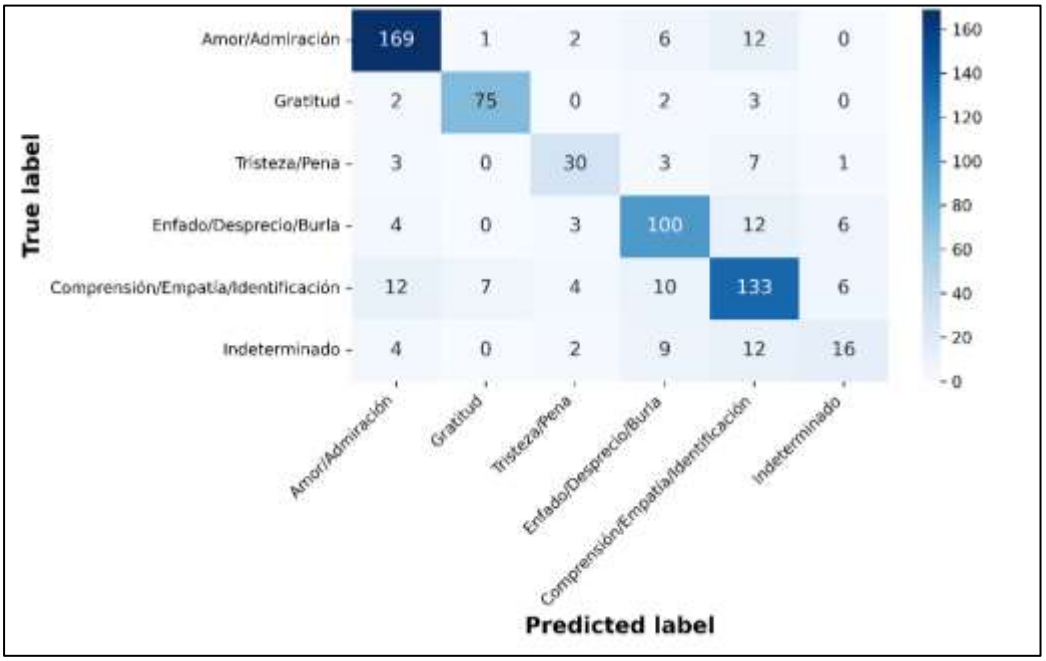

Figura 28:Matriz de confusión para emociones con balanceo 2.

```
Accuracy:     0.7972560975609756
F1-score:     0.7469624205224695
Precision:    0.7617379104102322
Recall:       0.7385458082632456

Classification Report:

Label                                  Precision    Recall    F1-score   Support
-------------------------------------------------------------------------------
0 - Amor/Admiración                         0.87      0.89        0.88       190
1 - Gratitud                                0.90      0.91        0.91        82
2 - Tristeza/Pena                           0.73      0.68        0.71        44
3 - Enfado/Desprecio/Burla                  0.77      0.80        0.78       125
4 - Comprensión/Empatía/Identificación      0.74      0.77        0.76       172
5 - Indeterminado                           0.55      0.37        0.44        43
-------------------------------------------------------------------------------
accuracy                                                          0.80       656
macro avg                                   0.76      0.74        0.75       656
weighted avg                                0.79      0.80        0.79       656
```

Figura 29: Métricas de evaluación e informe de clasificación para emociones con balanceo 2.

A continuación, se comparan los resultados obtenidos por los tres modelos entrenados para la tarea de clasificación de emociones, representados en las Figuras 25, 26 y 27. Los resultados generales se resumen en la Tabla 6, donde se aprecia que el modelo con balanceo 2 obtiene el mejor desempeño global.

Técnica	*Accuracy*	*F1-score*	*Precision*	*Recall*	*Mcc*
Sin balanceo	*0.7804*	*0.7307*	*0.7363*	*0.7304*	*0.7207*
Balanceo 1	*0.7866*	*0.7345*	*0.7357*	*0.7369*	*0.7291*
Balanceo 2	*0.7973*	*0.7469*	*0.7617*	*0.7385*	*0.7411*

Tabla 6: Resultados modelos de análisis de emociones.

Además de la precisión tradicional, el recall y el F1-score, el rendimiento global de los tres modelos se ha evaluado mediante el coeficiente de correlación de Matthews (*mcc*). Las conclusiones de esta sección se refuerzan con los resultados: el modelo sin balanceo obtuvo un *mcc* de 0.7207, y al añadir el Balanceo 1, el valor aumentó un poco (0.7291). El Balanceo 2, por otro lado, obtuvo el *mcc* más alto 0.7411, lo que indica que permite un aprendizaje más equilibrado y estable entre las clases.

También el análisis por clase permite observar con mayor detalle cómo afecta cada técnica al comportamiento del modelo. La Tabla 7 muestra que los mayores saltos de F1 se concentran en las clases infrarrepresentadas. Sin aplicar balanceo, el modelo se enfoca en las categorías más frecuentes y apenas capta las menos representadas. Al introducir Balanceo 1, las clases intermedias ganan algo de terreno gracias a una penalización moderada de sus errores, pero las mejoras son limitadas. En cambio, Balanceo 2, al asignar pesos inversamente proporcionales a la frecuencia real de cada etiqueta, consigue un refuerzo mucho más eficaz de las emociones escasas, por ejemplo “Tristeza/Pena”, y al mismo tiempo, mantiene intacta la precisión en las clases con abundantes ejemplos.

Clase	*Métrica*	*Sin balanceo*	*Balanceo 1*	*Balanceo 2*
Amor / Admiración	*Precision* *Recall* *F1-score*	*0.85* *0.89* *0.87*	*0.86* *0.91* *0.88*	***0.87*** ***0.89*** ***0.88***
Gratitud	*Precision* *Recall* *F1-score*	*0.95* *0.91* *0.93*	*0.97* *0.89* *0.93*	***0.90*** ***0.91*** ***0.91***
Tristeza / Pena	*Precision* *Recall* 	*0.56* *0.68* *0.61*	*0.60* *0.70* *0.65*	***0.73*** ***0.68*** ***0.71***

	F1-score			
Enfado / Desprecio / Burla	*Precision* *Recall* *F1-score*	*0.81* *0.73* *0.76*	*0.81* *0.71* *0.76*	***0.77*** ***0.80*** ***0.78***
Comprensión / Empatía / Identificación	*Precision* *Recall* *F1-score*	*0.73* *0.75* *0.74*	*0.76* *0.77* *0.76*	***0.74*** ***0.77*** ***0.76***
Indeterminado	*Precision* *Recall* *F1-score*	*0.53* *o.42* *0.47*	*0.41* *0.44* *0.43*	***0.55*** ***0.37*** ***0.44***

Tabla 7: Resultados por clase para cada modelo de análisis de emociones.

En general, la estrategia de Balanceo 2 emerge como la estrategia de mejor rendimiento para el análisis de emociones, al aumentar efectivamente la precisión para las clases menos representadas sin perjudicar a las mayoritarias. No obstante, este método no ha sido suficiente en el caso de clases ambiguas como "*Indeterminado*", y en trabajos futuros se recomienda aplicarlo mezclado con otras técnicas que refuercen la capacidad del modelo para distinguir correctamente este tipo de categoría.

4.8.2 Análisis de los resultados para polaridad

Esta sección presenta y compara los resultados obtenidos por los tres modelos entrenados para la tarea de clasificación de polaridad.

Comenzando por el análisis del modelo sin aplicar técnicas de balanceo (ver Figuras 30 y 31), se observa un rendimiento general elevado, alcanzando una *accuracy* del 84,14 %. Sin embargo, al analizar las métricas macro, los resultados son más moderados: un F1-score de 0,70, una precision de 0,74 y un *recall* de 0,67. Esto indica que, aunque en términos globales el modelo funciona bien, su rendimiento varía al evaluar cada clase por separado. Este desequilibrio se puede explicar por la distribución de las clases en el corpus, en la que la clase "*Positiva*" está sobrerrepresentada respecto a las demás.

Esto se especifica en el informe de clasificación. La clase "*Positiva*" obtiene la mejor puntuación con 0.90 en F1, la clase "*Negativa*" tiene 0.80. Sin embargo, la clase "*Indeterminada*" es la que más sufre del desequilibrio donde el F1-score es el más bajo del conjunto (0.39) con una precisión de 0.58 y un recall de solo 0.29.

La matriz de confusión confirma esta tendencia: mientras que las clases "*Positiva*" y "*Negativa*" son reconocidas con gran acierto, la categoría "*Indeterminado*" se confunde con frecuencia con las otras dos. Esto refleja la dificultad del modelo para identificarla correctamente, algo que puede deberse tanto a su escasa representación en el corpus como a su propia ambigüedad, ya que muchos comentarios sin polaridad clara suelen compartir rasgos similares a los de opiniones positivas o negativas.

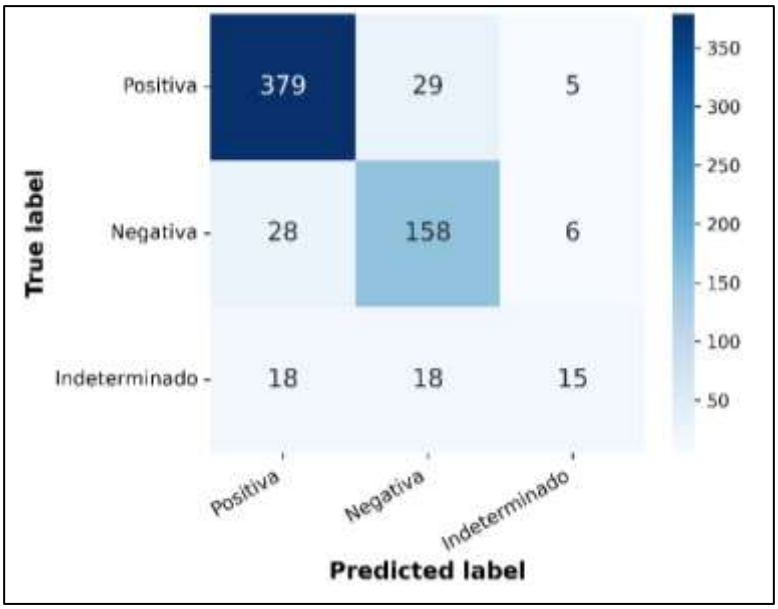

Figura 30: Matriz de confusión para polaridad sin balanceo.

```
Accuracy:     0.8414634146341463
F1-score:     0.6967049230384629
Precision:    0.7464731633741678
Recall:       0.6782366195055596

Classification Report:

Label                                  Precision    Recall    F1-score   Support
-------------------------------------------------------------------------------
Positiva                                    0.89      0.92        0.90       413
Negativa                                    0.77      0.82        0.80       192
Indeterminado                               0.58      0.29        0.39        51
-------------------------------------------------------------------------------
accuracy                                                          0.84       656
macro avg                                   0.75      0.68        0.70       656
weighted avg                                0.83      0.84        0.83       656
```

Figura 31: Métricas de evaluación e informe de clasificación para polaridad sin balanceo de clases.

Al introducir la técnica de balanceo 1 (Figuras 32 y 33), que asigna pesos según intervalos de frecuencia (0.5 para la clase mayoritaria "*Positiva*", 1.0 para "*Negativa*", y

1.5 para "*Indeterminada*"), se observa una leve mejora en el rendimiento general. La *accuracy* aumenta a 85,06 %, y el F1-score macro sube a 0,7333.

La clase que más se beneficia es "*Indeterminada*". Gracias al peso más alto (1,5), el modelo mejora notablemente su detección, alcanzando un *recall* de 0,41 (frente al 0,29 inicial) y elevando su F1-score de 0,39 a 0,48. Esto indica que, al darle mayor relevancia en la fase de entrenamiento, el modelo aprende a identificarla mejor, aunque sigue enfrentando dificultades debido a la ambigüedad natural de esta categoría.

También en la clase "*Negativa"*, con un peso intermedio (1.0), se aprecia también una mejora, especialmente en precisión (de 0.77 a 0.83), reflejando que un peso equilibrado beneficia su clasificación. En cambio, "*Positiva*", pese a tener un peso más bajo (0.5), mantiene su excelente rendimiento (F1 = 0.91), confirmando que esta categoría sigue destacando aunque reciba una menor penalización.

La matriz de confusión respalda estos resultados, destacando una clara disminución en los errores para la clase "*Indeterminada*" y menos confusiones con las otras dos clases.

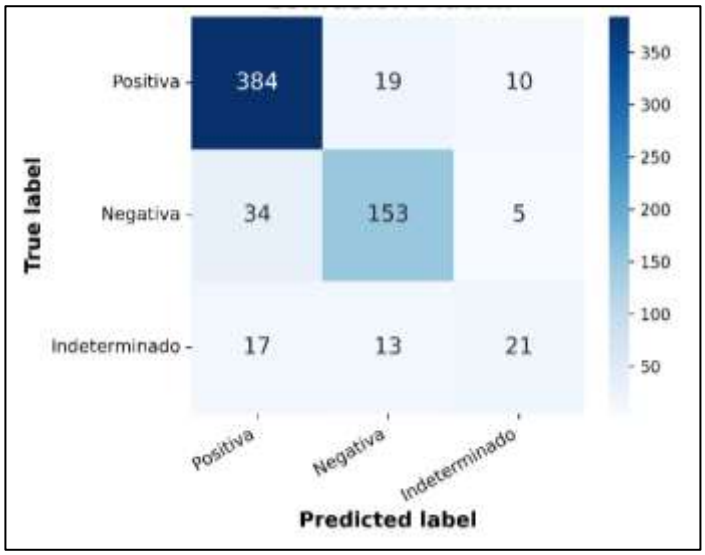

Figura 32: Matriz de confusión para polaridad con balanceo 1.

```
Accuracy:     0.850609756097561
F1-score:     0.7333633618604342
Precision:    0.7643729936833386
Recall:       0.7128072627356027

Classification Report:

Label                                  Precision    Recall    F1-score   Support
-------------------------------------------------------------------------------
Positiva                                    0.88      0.93        0.91       413
Negativa                                    0.83      0.80        0.81       192
Indeterminado                               0.58      0.41        0.48        51
-------------------------------------------------------------------------------
accuracy                                                          0.85       656
macro avg                                   0.76      0.71        0.73       656
weighted avg                                0.84      0.85        0.85       656
```

Figura 33: Métricas de evaluación e informe de clasificación para polaridad con balanceo 1.

Por último, al aplicar la ponderación inversamente proporcional a la frecuencia real de las clases (Figuras 34 y 35), con pesos continuos (0.53 para "*Positiva*", 1.14 para "*Negativa*", y 4.28 para "*Indeterminada*"), Balanceo 2, los resultados globales no experimentan mejora, incluso muestran un leve deterioro en comparación con balanceo 1. La *accuracy* baja a 83.99 %, con un F1-score macro también inferior (0.6849). Esto se debe a que la tarea de polaridad con el uso de Balanceo 2, asigna un peso excesivo a la clase "*Indeterminada*", y al tratarse de una categoría ambigua y poco definida, forzar su aprendizaje desestabiliza la función de pérdida y reduce la capacidad del modelo para optimizar correctamente el conjunto global.

La clase "*Positiva*" se mantiene robusta (F1 = 0.90) al recibir un peso muy similar al Balanceo 1. La clase "*Negativa*", con un peso ligeramente superior al anterior modelo, incrementa levemente su rendimiento, alcanzando un F1 de 0.82.

Sin embargo, la clase "*Indeterminada*" muestra una disminución en todas las métricas, sobre todo en el F1-score que cae a 0,33. A pesar de asignarle un peso tan alto, el modelo no logra mejorar su detección e, incluso, aumenta la confusión con las clases mayoritarias. Esto sugiere que darle un peso excesivo a una categoría tan ambigua y escasamente representada acaba provocando que el modelo se sobreajuste a patrones limitados, perdiendo capacidad de generalización e incrementando los errores.

La matriz de confusión confirma este fenómeno, ya que indica cómo aumentan los errores en la clasificación de "*Indeterminada*", que ahora son confundidos más frecuentemente con las clases predominantes.

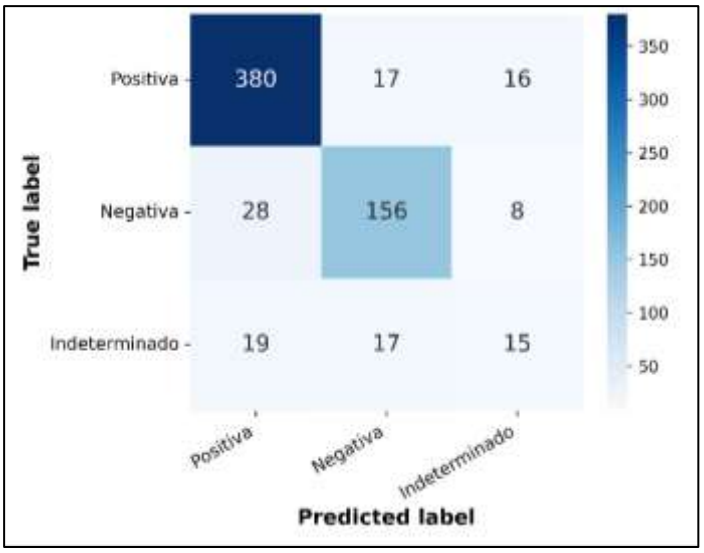

Figura 34: Matriz de confusión para polaridad con balanceo 2.

```
Accuracy:     0.8399390243902439
F1-score:     0.6849497215989363
Precision:    0.6985325861943635
Recall:       0.6755714997863552

Classification Report:

Label                                  Precision    Recall    F1-score   Support
-------------------------------------------------------------------------------
Positiva                                    0.89      0.92        0.90       413
Negativa                                    0.82      0.81        0.82       192
Indeterminado                               0.38      0.29        0.33        51
-------------------------------------------------------------------------------
accuracy                                                          0.84       656
macro avg                                   0.70      0.68        0.68       656
weighted avg                                0.83      0.84        0.83       656
```

Figura 35: Resultados para polaridad con balanceo 2.

A continuación, se comparan los tres modelos de polaridad mediante una tabla que resume sus principales métricas de rendimiento (Tabla 8).

Técnica	***Accuracy***	***F1-score***	***Precision***	***Recall***	***Mcc***
Sin balanceo	*0.8414*	*0.6967*	*0.7464*	*0.6782*	*0.6832*
Balanceo 1	*0.8506*	*0.7333*	*0.7643*	*0.7128*	*0.7003*
Balanceo 2	*0.8399*	*0.6849*	*0.6985*	*0.6755*	*0.6999*

Tabla 8: Resultados generales para los modelos de análisis de polaridad.

En cuanto a la evolución del coeficiente de correlación de Matthews (*mcc*), los resultados reflejan la misma tendencia observada en las métricas de rendimiento global. El modelo sin balanceo obtiene un *mcc* de 0.6832, que ya indica un desempeño sólido. Al

aplicar Balanceo 1, el *mcc* mejora ligeramente hasta alcanzar 0.7003, reflejando que esta técnica contribuye a un mejor equilibrio en la clasificación, especialmente en la clase minoritaria. Por el contrario, con Balanceo 2 el *mcc* cae ligeramente a 0.6999, lo que confirma que, pese a una ponderación más agresiva, el modelo no logra generalizar mejor.

Como se puede ver en la comparativa completa mostrada en la Tabla 9 para cada clase, en el caso de la tarea de polaridad, el método Balanceo 1 es el mejor método de ponderación. El modelo logra aumentar significativamente la detección de la clase minoritaria "*Indeterminada*" a cambio de un costo moderado en las clases más frecuentes. Mientras que, el Balanceo 2 se convierte en lo opuesto al dar un peso demasiado alto a esta clase, resultando en una pérdida de precisión, y muestra la importancia de seleccionar cuidadosamente los pesos que se les asignan no solo en función de su ocurrencia acumulada sino también en función del grado de su ambigüedad y también de su claridad semántica. Como tal, cualquier método de equilibrio debe tener en cuenta tanto la distribución como la naturaleza del problema en estudio.

Clase	***Métrica***	***Sin balanceo***	***Balanceo 1***	***Balanceo 2***
Positiva	*Precision* *Recall* *F1-score*	*0.89* *0.92* *0.90*	***0.88*** ***0.93*** ***0.91***	*0.89* *0.92* *0.90*
Negativa	*Precision* *Recall* *F1-score*	*0.77* *0.82* *0.80*	***0.83*** ***0.80*** ***0.81***	*0.82* *0.81* *0.82*
Indeterminado	*Precision* *Recall* *F1-score*	*0.58* *0.29* *0.39*	***0.58*** ***0.41*** ***0.48***	*0.38* *0.29* *0.33*

Tabla 9: Resultados por clase para cada modelo de análisis de polaridad.

4.9 Comparativa con otros modelos BERT

Para cerrar el análisis de resultados de este capítulo, en esta sección se presenta una comparación directa entre DistilBERT (utilizado con la configuración de Balanceo 2 para emociones y Balanceo 1 para polaridad, ya que son las más eficientes) y el modelo RoBERTuito, analizado por Lucía Diez Platero en su trabajo de fin de grado [39].

RoBERTuito es una adaptación al español del modelo RoBERTa, que es una variante optimizada del modelo BERT basado en la arquitectura Transformer. Está específicamente entrenado en textos de redes en español para tareas de procesamiento de lenguaje natural, por lo que suele ofrecer resultados robustos en clasificación de sentimientos y análisis emocional, particularmente en dominios informales debido a su ajuste fino a este tipo de escenarios.

En la comparación de resultados en la tarea de clasificación de emociones (Tabla 10), se aprecia que RoBERTuito obtiene valores más altos en casi todas las métricas, tanto a nivel de clase como en los promedios globales. Además, la accuracy de RoBERTuito alcanza un 85,58 %, por encima del 80 % de DistilBERT.

Emociones	***Métrica***	***DistilBERT (Balanceo 2)***	***RoBERTuito***
Amor/Admiración	*Precision* *Recall* *F1-score*	*87%* *89%* *88%*	*90.92%* *92.58%* *91.71%*
Gratitud	*Precision* *Recall* *F1-score*	*90%* *91%* *91%*	*91.88%* *93.79%* *92.77%*
Tristeza/Pena	*Precision* *Recall* *F1-score*	*73%* *68%* *71%*	*72.74%* *75.42%* *73.59%*
Enfado/Desprecio/Burla	*Precision* *Recall* *F1-score*	*77%* *80%* *78%*	*87.63%* *87.34%* *87.40%*
Comprensión/Empatía/Identificación	*Precision* *Recall* *F1-score*	*74%* *77%* *76%*	*83.59%* *83.53%* *83.39%*
Indeterminado/Neutral	*Precision* *Recall* *F1-score*	*55%* *37%* *44%*	*65%* *52.43%* *57.40%*
macro	*Precision* *Recall* *F1-score*	*76%* *74%* *75%*	*81.96%* *80.85%* *81.05%*
weighted	*Precision* *Recall* *F1-score*	*79%* *80%* *79%*	*85.59%* *85.58%* *85.39%*
Global	*Accuracy*	*80%*	*85.58%*

Tabla 10: Comparativa de resultados entre DistilBERT (Balanceo 2) y RoBERTuito para emociones.

En cuanto a polaridad (Tabla 11), RoBERTuito mantiene esa diferencia favorable en cada clase, la cual se refleja en los promedios macro y en la accuracy global de 89,09% frente a 85% de DistilBERT.

Es importante mencionar, sin embargo, que RoBERTuito y DistilBERT no se evaluaron exactamente sobre el mismo conjunto de prueba, por lo que los resultados podrían estar parcialmente influidos por ligeras diferencias en los datos de evaluación.

Polaridad	***Métrica***	***DistilBERT (Balanceo 1)***	***RoBERTuito***
Positivo	*Precision* *Recall* *F1-score*	*88%* *93%* *91%*	*93.82%* *93.34%* *93.57%*
Negativo	*Precision* *Recall* *F1-score*	*83%* *80%* *81%*	*85.88%* *89.46%* *87.52%*
Indeterminado	*Precision* *Recall* *F1-score*	*58%* *41%* *48%*	*63.14%* *53.24%* *56.74%*
macro	*Precision* *Recall* *F1-score*	*76%* *71%* *73%*	*80.95%* *78.68%* *79.19%*
weighted	*Precision* *Recall* *F1-score*	*84%* *85%* *85%*	*89.11%* *89.09%* *88.91%*
Global	*Accuracy*	*85%*	*89.09%*

Tabla 11: Comparativa de resultados entre DistilBERT (Balanceo 1) y RoBERTuito para polaridad.

Aunque RoBERTuito ofrece mejores resultados en términos de rendimiento, DistilBERT sigue siendo una alternativa muy interesante en muchos escenarios prácticos. En primer lugar, su reducido tamaño (menos de la mitad del tamaño de un BERT regular), supone una gran ventaja cuando los recursos son limitados. Por ejemplo, en Google Colab o entornos con GPUs modestas, podemos usar menos memoria y entrenar más rápido con DistilBERT. Además, el hecho de que esté integrada con la librería SimpleTransformers significa que todo el proceso desde la tokenización hasta la configuración de los hiperparámetros y la obtención de métricas de rendimiento se puede realizar fácilmente con unas pocas líneas de código, lo que significa que investigadores o desarrolladores sin experiencia profunda en ingeniería de modelos puedan comenzar más fácilmente.

Por lo tanto, si el objetivo final es maximizar la precisión absoluta, y hay suficientes recursos para entrenar/desplegar modelos más grandes, RoBERTuito lograría un alto rendimiento en ambas tareas (emociones y polaridad). Pero si simplemente desea obtener un resultado competitivo más rápidamente (o más barato o simple) en un lugar donde no puede dedicar mucho hardware o tiempo al problema, DistilBERT es una gran opción para obtener resultados de alta calidad a un costo y complejidad mucho menores.

4.10 Coste del proyecto

Para el desarrollo de este trabajo se han utilizado múltiples recursos computacionales y bibliotecas especializadas en procesamiento de lenguaje natural (NLP) y aprendizaje automático. Como se explica en el Capítulo 3, el entorno principal de ejecución ha sido Google Colab, una plataforma que permite ejecutar código Python en la nube con acceso gratuito o de pago a recursos avanzados [21].

Debido al elevado número de experimentos realizados, especialmente durante los barridos de hiperparámetros con Wandb, fue necesario contratar Google Colab Pro en tres ocasiones (Figura 36), lo que supuso una inversión total de aproximadamente 34 € (11.19 € cada mes). Este plan ofrece ventajas como acceso prioritario a GPUs más potentes, 100 unidades informáticas al mes, mayor tiempo de ejecución y almacenamiento temporal, lo que permitió ejecutar más de 180 pruebas automáticas. Cada sweep constaba de 30 pruebas, con una duración media de 1 hora y un coste de entre 45-60 unidades informáticas [21].

Figura 36: Google Colab Pro.

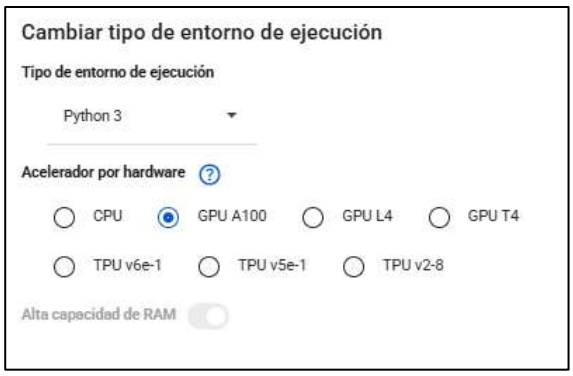

Figura 37: Entornos de ejecución disponibles.

Durante los sweeps, se configuró el entorno de ejecución para aprovechar una GPU A100 (Figura 37), lo que permitió reducir de manera significativa el tiempo de entrenamiento de los modelos más exigentes. Por otro lado, para evaluar los modelos finales, también se realizaron pruebas en CPU en la versión gratuita de Colab. Gracias a esta estrategia combinada, se logró un equilibrio perfecto entre velocidad, eficiencia y ahorro de costes a lo largo de toda la experimentación.

En resumen, la combinación de recursos cloud avanzados (Colab Pro + GPU A100) junto con un ecosistema Python bien integrado ha sido fundamental para el éxito y viabilidad de este proyecto.

5

Conclusiones y Líneas futuras

5.1 Conclusiones

Este Trabajo de Fin de Grado presenta el desarrollo de un sistema de clasificación automática de emociones y polaridad en textos relacionados con la salud mental en redes sociales, tomando como base un modelo preentrenado en español de DistilBERT. El modelo fue implementado y adaptado mediante la biblioteca SimpleTransformers, lo que permitió entrenarlo y ajustarlo a nuestro corpus etiquetado con seis emociones y tres tipos de polaridad.

Los resultados han sido consistentes tanto en términos de precisión general como en el análisis por clase para ambos tipos de clasificación. En la tarea de clasificación emocional, se ha demostrado que la técnica de balanceo por ponderación inversa permite mejorar el rendimiento en clases difíciles como "*Tristeza/Pena*" o "*Enfado*", sin perjudicar la precisión en clases mayoritarias como "*Gratitud*" o "*Amor/Admiración*". Para la polaridad, aunque el balanceo 2 no ha mostrado una mejora significativa, el balanceo 1 ha permitido una mejor detección de la clase "*Indeterminada*", históricamente más difícil por su ambigüedad.

Además, se realizó una búsqueda de hiperparámetros a través de barridos en Wandb, para encontrar la mejor combinación para cada configuración experimental. Esta estrategia, basada en el análisis por clase y en la evaluación con métricas macro, permitió valorar de manera justa el rendimiento del modelo en un corpus marcado por un fuerte desbalance de clases.

En conclusión, este trabajo demuestra que el uso de modelos preentrenados adaptados al español junto al empleo de técnicas de balanceo puede ser altamente eficaz para tareas de clasificación emocional en textos de redes sociales, con posibles aplicaciones en otros campos como la psicología, la moderación automática o el análisis social.

5.2 Líneas futuras

A partir del trabajo realizado, se sugieren las siguientes líneas de mejora e investigación de cara al futuro:

- Ampliación y enriquecimiento del corpus: aunque el modelo ha ofrecido resultados prometedores, incorporar más textos en clases poco representadas, así como incluir datos de nuevas plataformas, podría mejorar su capacidad de generalización. Por otro lado, se plantea como línea futura la posibilidad de incorporar nuevas categorías de emoción, con el objetivo de ampliar el rango de matices emocionales que el sistema sea capaz de analizar y clasificar.
- Refinamiento de las técnicas de balanceo: utilizar otros métodos de balanceo como el sobremuestreo (SMOTE) o la introducción de ejemplos artificiales podrían ayudar a mejorar o superar los resultados obtenidos con las técnicas de ponderación actuales.
- Automatización del etiquetado: utilizar modelos ya entrenados para generar etiquetas en nuevos datos no anotados facilitaría futuras expansiones del corpus y permitiría construir sistemas en tiempo real.
- Evaluación en contexto real: será interesante evaluar la efectividad de los modelos en escenarios del mundo real (por ejemplo, moderación de contenido o monitoreo comunitario), para evaluar su valor práctico más allá de las condiciones ideales presentes en nuestros experimentos.

Estas líneas abren la puerta a seguir explorando la intersección entre inteligencia artificial y salud mental desde una perspectiva técnica, pero con claras implicaciones sociales.

6

Bibliografía

[1] S. Kemp, “Digital 2024: Global Overview Report,” DataReportal, 31-ene-2024. [Online]. Disponible en: https://datareportal.com/reports/digital-2024-global-overview-report

[2] E. Cambria y B. White, “Jumping NLP curves: A review of natural language processing research,” IEEE Computational Intelligence Magazine, vol. 9, no. 2, pp. 48–57, 2014.

[3] A. Vaswani et al., “Attention is all you need,” Advances in Neural Information Processing Systems, vol. 30, 2017.

[4] V. Sanh, L. Debut, J. Chaumond y T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” arXiv preprint arXiv:1910.01108, 2019.

[5] T. Rajapakse, “SimpleTransformers,” [Online]. Disponible en: https://github.com/ThilinaRajapakse/simpletransformers. [Accedido: 24-may-2025].

[6] H. He y E. A. Garcia, “Learning from imbalanced data,” IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263–1284, 2009.

[7] F. Pérez-Sorrosal, “dccuchile-distilbert-base-spanish-uncased-finetuned-with-spanish-tweets-clf-cleaned-ds,” Hugging Face. Disponible en: https://huggingface.co/francisco-perez-sorrosal/dccuchile-distilbert-base-spanish-uncased-finetuned-with-spanish-tweets-clf-cleaned-ds

[8] R. C. Cabral, S. C. Han, J. Poon y G. Nenadic, “MM-EMOG: Multi-Label Emotion Graph Representation for Mental Health Classification on Social Media,” Robotics, vol. 13, no. 3, 2024.

[9] Weights & Biases. Disponible en: https://wandb.ai/site

[10] D. M. Powers, “Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation,” Journal of Machine Learning Technologies, vol. 2, no. 1, pp. 37–63, 2011.

[11] Chaves-Villota, et al., “EmoSPeech 2024@IberLEF: Multimodal Speech-text Emotion Recognition in Spanish,” in CEUR Workshop Proceedings, vol. 3756, 2024. Disponible en: https://ceur-ws.org/Vol-3756/EmoSPeech2024_paper8.pdf

[12] Scikit-learn. Disponible en: https://scikit-learn.org/stable/

[13] J. Devlin, M. Chang, K. Lee y K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv preprint arXiv:1810.04805, 2019.

[14] T. B. Brown et al., “Language models are few-shot learners,” Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020.

[15] U.S. Department of Health and Human Services, “Social Media and Youth Mental Health,” Surgeon General’s Advisory, mayo 2023. [Online]. Disponible en: https://www.hhs.gov/surgeongeneral/reports-and-publications/youth-mental-health/social-media/index.htmlNew York Post+6HHS.gov+6Wikipedia+6

[16] Pew Research Center, “Teens, Social Media and Mental Health,” 22-abr-2025. [Online]. Disponible en: https://www.pewresearch.org/internet/2025/04/22/teens-social-media-and-mental-health/Pew Research Center+1Pew Research Center+1

[17] D. Sridhar, “Children are speaking to strangers online – and grooming is on the rise. This is how to protect them,” The Guardian, 21-may-2025. [Online]. Disponible en: https://www.theguardian.com/commentisfree/2025/may/21/children-strangers-online-grooming-on-rise-protect-themtheguardian.com

[18] Y. Yuan et al., “Mental Health Coping Stories on Social Media: A Causal-Inference Study of Papageno Effect,” arXiv preprint arXiv:2302.09885, 2023. [Online]. Disponible en: https://arxiv.org/abs/2302.09885

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

[20] Aurora, “¿Qué son las librerías de Python?,” ID Digital School, 18-jul-2023. [Online]. Disponible en: https://iddigitalschool.com/bootcamps/que-son-las-librerias-de-python/

[21] Google, “Colaboratory,” [Online]. Disponible en: https://colab.research.google.com/

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

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

[24] Hugging Face Datasets. Disponible en: https://huggingface.co/docs/datasets/

[25] Torch. Disponible en: https://pytorch.org/

[26] Matplotlib. Disponible en: https://matplotlib.org/

[27] Seaborn. Disponible en: https://seaborn.pydata.org/

[28] C. D. Manning, P. Raghavan, y H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[29] T. Fawcett, “An introduction to ROC analysis,” Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.

[30] J. Han, M. Kamber, y J. Pei, Data Mining: Concepts and Techniques, 3rd ed., Morgan Kaufmann, 2011.

[31] B. W. Matthews, “Comparison of the predicted and observed secondary structure of T4 phage lysozyme,” Biochimica et Biophysica Acta (BBA), vol. 405, no. 2, pp. 442–451, 1975.

[32] R. Plutchik, “A general psychoevolutionary theory of emotion,” en Theories of Emotion, Academic Press, 1980, pp. 3–33.

[33] S. Gatt y E. Krahmer, “A survey of the state of the art in natural language generation: Core tasks, applications and evaluation,” Disponible en: https://arxiv.org/abs/2006.01413

[34] L. Prechelt, “Early Stopping — But When?,” in Neural Networks: Tricks of the Trade, Springer, 1998, pp. 55–69.

[35] G. Batista, R. C. Prati y M. C. Monard, “A study of the behavior of several methods for balancing machine learning training data,” ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 20–29, 2004.

[36] N. V. Chawla, K. W. Bowyer, L. O. Hall, y W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.

[37] J. Bergstra y Y. Bengio, “Random search for hyper-parameter optimization,” Journal of Machine Learning Research, vol. 13, pp. 281–305, 2012.

[38] T. Wolf et al., “Transformers: State-of-the-Art Natural Language Processing,” Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, 2020. [Online]. Disponible en: https://aclanthology.org/2020.emnlp-demos.6

[39] Diez Platero, L. (2024). Análisis emocional sobre contenido de salud mental en Instagram mediante modelos BERT (Trabajo Fin de Grado). E.T.S.I. Telecomunicación, Universidad de Valladolid.