UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

Aplicación de aprendizaje automático para evaluar lenguaje de videojuegos en Twitch

Autor:
D. Miguel Carralero Lanchares
Tutor:
Dña. Noemí Merayo Álvarez

TÍTULO: **Análisis de la respuesta emocional en redes sociales usando Técnicas de Aprendizaje Automático**
AUTOR: **D. Miguel Carralero Lanchares**
TUTOR: **Dña. Noemí Merayo Álvarez**
DEPARTAMENTO: **Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

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

FECHA:
CALIFICACIÓN:

Resumen de TFG

En el trabajo se aborda el análisis de la respuesta emocional en chats de transmisiones de videojuegos en Twitch. El crecimiento exponencial de esta plataforma ha generado una necesidad crucial de comprender las emociones que experimentan los usuarios al interactuar en los chats de las transmisiones. En este sentido, y dada la dificultad de analizar manualmente la gran cantidad de mensajes generados en tiempo real, se aplican técnicas de Inteligencia Artificial y Procesamiento del Lenguaje Natural (PLN) para clasificar la polaridad (positiva, negativa, indeterminada) y emociones (aprobación, desaprobación, decepción, interés, enfado, etc.) expresadas en sus mensajes/comentarios.

El objetivo principal de este estudio es mejorar los resultados de estudios previos sobre el análisis de sentimientos en transmisiones de videojuegos en Twitch. Para lograrlo, se amplía el corpus de datos, incluyendo una mayor diversidad de géneros de videojuegos, streamers y comunidades, así como diferentes terminologías y jergas propias de este ámbito. Se utiliza un modelo de lenguaje basado en Representación de Codificador Bidireccional de Transformadores (BERT), adaptado al español, para entrenar y optimizar dicho modelo de clasificación. Se realiza un preprocesado de los datos para eliminar información redundante y normalizarlos. Posteriormente, se evaluó el rendimiento del modelo utilizando diferentes métricas de precisión y se compararon los resultados con los de estudios previos para determinar la efectividad de las mejoras implementadas.

Tras el entrenamiento y optimización, se han desarrollado herramientas pioneras para aplicar el modelo, incluyendo una función para realizar predicciones sobre nuevos mensajes y una aplicación con interfaz gráfica que permite cargar archivos, obtener predicciones y visualizar los resultados mediante gráficos.

Palabras clave

Análisis de la respuesta emocional, Twitch, videojuegos, Procesamiento del Lenguaje Natural, Inteligencia Artificial, Deep Learning, Modelos basados en Representación de Codificador Bidireccional de Transformadores, Emociones, Polaridad

Abstract

The work carried out addresses the analysis of emotional responses in Twitch video game stream chats. The exponential growth of this platform has created a crucial need to understand the emotions experienced by users as they interact in stream chats. In this regard and given the difficulty of manually analysing the large number of messages generated in real time, Artificial Intelligence and Natural Language Processing (NLP) techniques are applied to classify the polarity (positive, negative, undetermined) and emotions (approval, disapproval, disappointment, interest, anger, etc.) expressed in the messages/comments.

This study's main objective is to improve the results of previous studies on sentiment analysis in Twitch video game streams. To achieve this, the data corpus is expanded, including a greater diversity of video game genres, streamers and communities, as well as different terminology and jargon specific to this field. A language model based on Bidirectional Encoder Representations from Transformers (BERT), adapted to Spanish, is used to train and optimize this classification model. Data preprocessing is performed to remove redundant information and normalize it. Subsequently, the model's performance was evaluated using different accuracy metrics, and the results were compared with those of previous studies to determine the effectiveness of the implemented improvements.

After training and optimization, pioneering tools have been developed to apply the model, including a function to make predictions on new messages and an application with a graphical interface that allows loading files, obtaining predictions, and visualizing the results through graphs.

Keywords

Emotional response analysis, Twitch, Videogames, Natural Language Processing, Artificial Intelligence, Deep Learning, Bidirectional Encoder Representations from Transformers Models, Emotions, Polarity

Agradecimientos

Primero me gustaría agradecer a mis padres por todo el apoyo y por la educación que me dieron desde pequeño. Siempre respaldándome en todos los ámbitos de mi formación, por lo cual siempre estaré agradecido. Mamá y Papá, este grado es tan mío como vuestro.

Me gustaría agradecer también a Noemí, que me presentó un trabajo tan desafiante, que siempre apreció todas y cada una de mis aportaciones y que desde el principio me ha ayudado para que el proyecto saliese adelante.

Agradecérselo también a mis amigos de toda la vida (Abel, Marco, Adrián y Mir) y a mis amigos de la carrera (Jorge, Maloti y Alpvi). Quienes me han apoyado y acompañado en las largas sesiones de estudio como en los cafés de por medio.

Y a mi pareja, Irene. Simplemente gracias. Me ayudaste a sacar fuerzas en el último año de carrera cuando ni siquiera yo creía que las tenía, me apoyaste los días en los que creía que todo me sobrepasaba y me has animado a poder sacar la mejor versión de mi tanto en este trabajo como en mi vida. Muero de ganas de terminar esta fase, para poder empezar la siguiente contigo.

Índice

Índice de figuras

Índice de tablas

Índice de ecuaciones

1

Introducción

1.1 Motivación

El presente Trabajo de Fin de Grado (TFG) se adentra en el análisis de la respuesta emocional en el contexto de las plataformas de streaming de videojuegos, un campo de creciente relevancia debido al auge de la industria del videojuego y su impacto cultural y económico a nivel global. Este trabajo se motiva por el potencial de la Inteligencia Artificial (IA) y el Procesamiento del Lenguaje Natural (PLN) para abordar los desafíos inherentes al análisis masivo de datos en tiempo real generados en estas plataformas. En concreto, este estudio se centra en la plataforma Twitch en el ámbito de los videojuegos, que ha experimentado un crecimiento exponencial en los últimos años, convirtiéndose en un espacio de interacción social en tiempo real donde millones de espectadores interactúan con sus creadores de contenido favoritos a través de chats en vivo. La vasta cantidad de usuarios y la inmensa cantidad de transmisiones de videojuegos en vivo hacen que el análisis manual del contenido de los mensajes en los chats sea una tarea impracticable.

En este contexto, la IA y el PLN emergen como herramientas cruciales para abordar este desafío, permitiendo extraer información valiosa de los textos, como la identificación de emociones, la detección de usuarios problemáticos o el análisis de la polaridad de los mensajes. Este estudio se distingue por su enfoque en el análisis de sentimientos en el contexto específico de las transmisiones en directo de juegos en Twitch. La creación de un corpus especializado y el entrenamiento de modelos de aprendizaje profundo basados en la arquitectura de Representaciones de Codificador Bidireccional de Transformadores (BERT, *Bidirectional Encoder Representations from Transformers*) permiten abordar las particularidades del lenguaje utilizado en este entorno, incluyendo el uso de jerga, emoticonos y estrategias para evadir filtros de lenguaje ofensivo o inapropiado.

Otro factor que motivó este trabajo fue el potencial de estas herramientas para los creadores de contenido o profesionales de las redes sociales. Un estudio de estas características puede resultar de gran utilidad para ellos, puesto que les permite analizar hasta cierto punto el nivel de *engagement*[1], el comportamiento emocional y el entendimiento de sus directos por parte de la audiencia. Esta información puede ser clave para saber qué funciona y qué no, y así tomar decisiones informadas para mejorar la calidad de su contenido, aumentar la interacción con su comunidad y gestionar el lenguaje y los comportamientos tóxicos o de odio que puedan surgir en estas plataformas tan masivas.

1.2 Objetivos

1.2.1 Objetivo principal

El objetivo principal de este trabajo es utilizar técnicas de Inteligencia Artificial para analizar la respuesta emocional en los mensajes del chat de Twitch, centrándose en la clasificación de la polaridad (positiva, negativa e indeterminada) y de un conjunto de emociones muy presentes en esta plataforma y en el ámbito de los videojuegos (aprobación, desaprobación, decepción, interés, enfado e indeterminado).

Este trabajo se basa en un estudio previo [1] que también abordó el análisis de la respuesta emocional en mensajes de Twitch. Sin embargo, se busca mejorar los resultados obtenidos en dicho estudio mediante la ampliación del corpus de datos y la optimización de los hiperparámetros de un modelo de clasificación denominado RoBERTuito, una variante de un modelo de BERT adaptada al español y al contexto de redes sociales.

El objetivo final es optimizar un modelo de arquitectura BERT capaz de clasificar con alta precisión la polaridad y las emociones expresadas en los mensajes del chat de Twitch, contribuyendo así al entendimiento de la dinámica de las comunidades online y al desarrollo de herramientas de análisis de sentimientos más efectivas en el contexto de los videojuegos y las retransmisiones en directo.

[1] Engagement: Interacciones significativas entre una marca y su audiencia que generan interés, participación y lealtad. Obtenido de: https://blog.hubspot.com/service/customer-engagement-guide

1.2.2 Objetivos específicos

Para lograr el objetivo principal de este TFG, se establecen los siguientes objetivos específicos:

1. Ampliación el corpus de datos de Twitch, incluyendo una mayor diversidad de géneros de videojuegos, streamers con diferentes niveles de competitividad y comunidades de seguidores de distintos tamaños. Se buscará equilibrar la representación de emociones y polaridades, prestando especial atención a las emociones negativas y aumentar la diversidad lingüística del *dataset* incorporando jerga y expresiones propias del ámbito *gamer*.

2. Recopilación de mensajes *Leet Speak* para enriquecer el corpus y mejorar la capacidad del modelo para detectar emociones negativas expresadas de forma encubierta.

3. Búsqueda y aplicación de parámetros óptimos del modelo RoBERTuito (un modelo de clasificación basado en BERT) para maximizar su precisión en la clasificación de polaridad y emociones, utilizando el nuevo corpus ampliado y balanceado.

4. Evaluación el rendimiento del modelo optimizado en la clasificación de polaridad y emociones utilizando métricas como precisión, exhaustividad, puntuación F1 y precisión global. Comparación de los resultados obtenidos con los del estudio previo para determinar si se ha logrado una mejora significativa.

5. Aplicar los modelos entrenados para realizar predicciones en tiempo real sobre nuevos mensajes de Twitch, con el objetivo de desarrollar una herramienta práctica y visual para el análisis de sentimientos en esta plataforma.

1.3 Metodología del trabajo

La metodología de trabajo seguida para el desarrollo de los objetivos se ha estructurado en cuatro fases principales:

1.3.1 Fase de documentación

En esta primera fase se ha realizado una revisión exhaustiva de la literatura disponible sobre el análisis de sentimientos y emociones en el contexto de los videojuegos y plataformas de *streaming*, así como de las técnicas de PLN y aprendizaje automático más relevantes. Se ha hecho hincapié en el estudio de los modelos de lenguaje basados en *Transformers*, como BERT y sus variantes, incluyendo RoBERTuito, el modelo utilizado en este trabajo. Esta fase ha permitido fundamentar el problema, comprender la importancia del análisis de emociones en redes sociales y seleccionar las técnicas y enfoques más adecuados para el presente trabajo, así como identificar áreas de mejora tanto en el modelo como en el corpus de datos.

1.3.2 Fase de análisis

Esta fase se centró en comprensión de los datos de Twitch, evaluando el impacto de diferentes estrategias en la composición del corpus y en el rendimiento del modelo. Se examinaron los chats, transmisiones y mensajes individuales, buscando mejorar la representación de polaridades y emociones. Se analizaron los resultados del modelo, incluyendo métricas de rendimiento y matrices de confusión, y se compararon con trabajos previos para validar la eficacia de las estrategias implementadas. Para la realización de esta fase se ha partido de los siguientes trabajos:

- TFG de Javier Estévez Asensio [1].
- TFG de Pablo González Gómez [2].

1.3.3 Fase de pruebas

Fase en la cual se ha entrenado el modelo BERT con los datos preprocesados y los hiperparámetros optimizados, y se evaluó su rendimiento en la clasificación de polaridad y emociones. Se compararon los resultados obtenidos con los del estudio previo para determinar si se ha logrado una mejora significativa.

1.3.4 Fase de escritura del informe

Fase en la cual se ha redactado el presente informe, en el que se detallan los objetivos, la metodología, los resultados y las conclusiones del trabajo. Se ha puesto especial énfasis en la claridad y rigurosidad de la exposición, así como en la justificación de las decisiones tomadas a lo largo del proceso.

1.4 Estructura de la memoria

La memoria del proyecto se organiza en los siguientes capítulos:

Capítulo 1: Se introduce el problema a abordar, los objetivos que se busca conseguir y se explica la metodología que se va a utilizar para la resolución del problema.

Capítulo 2: Se proporciona un estado del arte sobre la inteligencia artificial, el aprendizaje automático, el aprendizaje profundo y el procesamiento del lenguaje natural, con especial atención a la arquitectura *Transformer* y los modelos BERT. Se contextualiza el problema del análisis de sentimientos en el ámbito de los videojuegos y Twitch.

Capítulo 3: Se describen las principales herramientas software utilizadas en la realización del estudio y análisis.

Capítulo 4: Se detalla la metodología empleada para la ampliación del corpus de datos de Twitch, incluyendo los criterios de selección de transmisiones y *streamers*, así como el proceso de descarga, procesamiento y estructuración de los datos. Se analizan los resultados obtenidos en términos de distribución de polaridades y emociones, y se discuten las implicaciones de estos hallazgos.

Capítulo 5: Se describe el proceso de análisis de la respuesta emocional utilizando el modelo RoBERTuito. Se detallan las etapas de preprocesamiento, entrenamiento del modelo, elección de hiperparámetros y análisis de resultados. Se evalúa el rendimiento del modelo de clasificación de polaridades y emociones, y se comparan los resultados con los estudios previos.

Capítulo 6: Se presentan dos aplicaciones prácticas del modelo desarrollado, una función *predict* para realizar predicciones sobre nuevos datos y una aplicación con interfaz gráfica que permite cargar archivos de mensajes, obtener predicciones y visualizar los resultados mediante gráficos.

Capítulo 7: Se presentan las conclusiones finales del estudio, destacando los logros alcanzados y las limitaciones encontradas. Se proponen líneas futuras de investigación para mejorar y ampliar el alcance del trabajo realizado.

Capítulo 8: Se indican brevemente los requisitos mínimos de hardware y software necesarios para ejecutar las herramientas mencionadas en el estudio.

2

Estado del arte

2.1 Introducción

En este capítulo se describe por un lado un breve resumen de los modelos de aprendizaje automático en los que se centrará el TFG y por otro lado un estado del arte de la plataforma virtual Twitch y su potencial, especialmente en el ámbito de los videojuegos.

2.2 Marco teórico del aprendizaje profundo en PLN

La Inteligencia Artificial (IA), un campo en constante evolución, ha experimentado un crecimiento exponencial en las últimas décadas impulsada por avances significativos en áreas como el Aprendizaje Automático (*Machine Learning*, ML) y el Aprendizaje Profundo (Deep Learning, DL). La IA, en su esencia, busca dotar a los sistemas de la capacidad de imitar la inteligencia humana, permitiéndoles realizar tareas que tradicionalmente requerían de la cognición humana, como el aprendizaje, el razonamiento, la resolución de problemas e incluso la creatividad.

Dentro de este amplio campo, el Machine Learning (ML) [4] emerge como un subcampo fundamental. El ML se centra en el desarrollo de algoritmos que permiten a las máquinas aprender de los datos y mejorar su rendimiento en tareas específicas sin necesidad de ser programadas explícitamente para cada situación. Esto se logra mediante el entrenamiento de modelos con grandes cantidades de datos, lo que les permite identificar patrones y generalizar a partir de ellos.

El DL, a su vez, es una rama del ML que utiliza redes neuronales artificiales con múltiples capas para modelar y resolver problemas complejos [4]. Estas redes, inspiradas en el funcionamiento del cerebro humano, están compuestas por unidades interconectadas llamadas neuronas artificiales que procesan la información de manera distribuida y paralela. La profundidad de estas redes, es decir, el número de capas, les permite aprender

representaciones jerárquicas de los datos, lo que resulta crucial para tareas como el reconocimiento de imágenes, la traducción automática y, en nuestro caso, el análisis de sentimientos/emociones.

A modo de ilustración, la Figura 1 presenta un diagrama que detalla la jerarquía de la Inteligencia Artificial, mostrando cómo el ML y el DL se enmarcan como subcampos dentro de este campo más amplio.

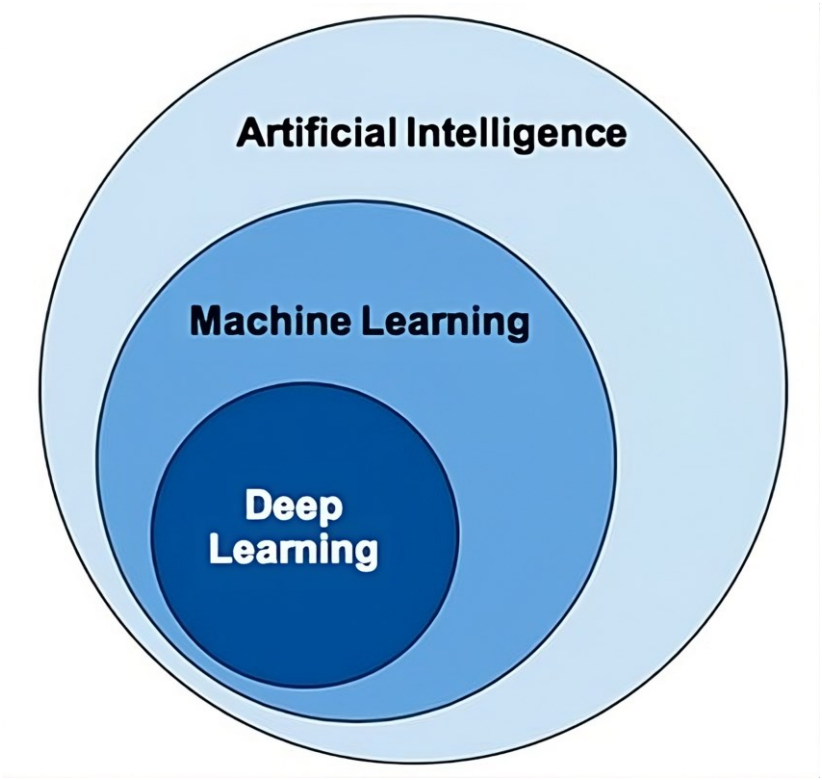


Figura 1: Jerarquía de la Inteligencia Artificial. Artificial Intelligence vs. Machine Learning vs. Deep Learning: What's the Difference? [5]

En el ámbito del procesamiento del lenguaje natural (PLN), las redes neuronales han demostrado ser especialmente efectivas. Al igual que en otras áreas, la arquitectura *Transformer*, introducida por primera vez en el artículo *Attention Is All You Need* [6] en 2017, ha revolucionado el campo. Los modelos *Transformer* se basan en un mecanismo de atención que permite a la red ponderar la importancia de diferentes partes de la entrada, lo que resulta en un mejor entendimiento del contexto y las relaciones entre palabras. Esto ha llevado a mejoras significativas en tareas como la traducción automática, el resumen de textos y la respuesta a preguntas.

Los modelos BERT, otro modelo desarrollado por Google en 2018 [7], ha sido clave en el avance del PLN. BERT son modelos preentrenados con grandes cantidades de texto, lo

que le permite capturar el significado de las palabras en su contexto de manera más precisa que los modelos anteriores. Esta capacidad lo hace ideal para tareas como el análisis de sentimientos, la identificación de entidades nombradas y la clasificación de textos, tareas fundamentales en el presente trabajo.

La convergencia de la IA, el ML y el DL, junto con la aparición de arquitecturas como *Transformer* y modelos como BERT, ha abierto un nuevo horizonte para el análisis del lenguaje natural. Estas tecnologías permiten abordar problemas complejos de comprensión y generación de lenguaje, con aplicaciones en áreas tan diversas como la medicina, el derecho, la educación, el marketing y, como en este caso, el análisis de la respuesta emocional en plataformas de *streaming* como Twitch. El presente trabajo se sitúa en la vanguardia de la investigación en PLN, aplicando estas tecnologías para comprender y analizar el lenguaje utilizado en los videojuegos y sus comunidades online.

2.3 Twitch y los videojuegos

La industria de los videojuegos ha experimentado un crecimiento exponencial en las últimas décadas, convirtiéndose en un fenómeno cultural y económico de alcance global, con un volumen de negocio que ronda los cien mil millones de euros y más de mil cien millones de jugadores en línea en todo el mundo [9]. Plataformas de streaming como Twitch han amplificado este impacto, creando comunidades virtuales masivas donde millones de espectadores interactúan con sus creadores de contenido favoritos a través de chats en vivo. En 2022, el número de minutos vistos en Twitch alcanzó los 1,35 billones, y la plataforma contó con un total de 7,59 millones de streamers activos a mediados de 2023 [9].

La vasta cantidad de usuarios de Twitch y la inmensa cantidad de transmisiones de videojuegos en vivo que ocurren simultáneamente en la plataforma (210.4 millones de horas transmitidas en el tercer trimestre de 2022) hacen que el análisis manual del contenido de los mensajes del chat sea una tarea impracticable. Incluso en una sola transmisión, miles de mensajes son enviados en tiempo real, lo que dificulta la identificación y comprensión de las emociones expresadas por los espectadores.

En este contexto, la IA y el PLN se presentan como herramientas cruciales para abordar este desafío. El PLN, al combinar la informática y la lingüística, permite extraer

información valiosa de los textos, como la identificación de emociones, la detección de *trolls*[2] [11] o el análisis de la polaridad de los mensajes [12].

Si bien existen estudios previos que han aplicado técnicas de PLN al análisis de contenido de videojuegos, pocos se han centrado en el análisis de la respuesta emocional en tiempo real en plataformas de streaming. La mayoría de estos trabajos se han enfocado en el análisis de reseñas de usuarios o resúmenes de juegos, utilizando léxicos y herramientas existentes en lugar de modelos de aprendizaje automático entrenados en corpus especializados [13, 14]. Otros estudios han explorado el uso de modelos de aprendizaje automático para identificar actitudes de *trolling* a partir de emoticonos, pero no han profundizado en el análisis de la respuesta emocional en los chats.

En este sentido, el presente trabajo se distingue por su enfoque en el análisis de sentimientos en el contexto específico de las transmisiones en directo de juegos en Twitch. La creación de un corpus especializado y el entrenamiento de modelos de aprendizaje profundo basados en BERT, como RoBERTuito, permiten abordar las particularidades del lenguaje utilizado en este entorno, incluyendo el uso de jerga [15, 16], emoticonos y estrategias de evasión como el *Leet Speak*[3].

La aplicación de técnicas de PLN al análisis de sentimientos en Twitch tiene un impacto significativo en un entorno que mueve a millones de personas y genera ingresos considerables. Esta tecnología permite a los streamers comprender mejor las opiniones y emociones de su audiencia, lo que les ayuda a tomar decisiones informadas sobre el contenido y las estrategias de sus transmisiones. Además, el análisis de sentimientos puede contribuir a la detección de comportamientos tóxicos y al fomento de una comunidad más segura e inclusiva.

[2] Trolls: Usuarios de internet que publican mensajes provocadores, irrelevantes o fuera de tema con el objetivo de generar controversia o molestar a otros usuarios. Obtenido de: https://ijnet.org/es/story/todo-lo-que-hay-que-saber-sobre-trolls-bots-y-botnets

[3] Leet Speak: Sistema de escritura que utiliza caracteres alfanuméricos y símbolos para reemplazar letras, utilizado principalmente en Internet para crear un lenguaje codificado o para evadir filtros de contenido. Obtenido de: https://en.wikipedia.org/wiki/Leet

3

Herramientas utilizadas

3.1 Introducción

Para llevar a cabo este trabajo ha sido necesario utilizar las herramientas y plataformas software que a continuación se describen.

3.2 Python

Python [17] es un lenguaje de programación de alto nivel, interpretado y de propósito general. Fue creado por Guido van Rossum en el año 1991 y posee una amplia variedad de bibliotecas y *frameworks*[4] especializados en desarrollo de Inteligencia Artificial y Ciencia de Datos. A continuación, se enumeran una serie de características que lo hacen una buena elección para este proyecto:

- Bibliotecas[5]: especializadas en desarrollo de Inteligencia Artificial, las cuales tienen una gran cantidad de algoritmos y herramientas para tareas como la clasificación.

[4] Framework: Conjunto de herramientas, guías y estructuras predefinidas que se utilizan para desarrollar y organizar software de manera eficiente. Obtenido de: https://www.inboundcycle.com/diccionario-marketing-online/framework

[5] Bibliotecas o Librerías: Colecciones de funciones, clases y métodos predefinidos que extienden la funcionalidad básica del lenguaje Obtenido de: https://iddigitalschool.com/bootcamps/que-son-las-librerias-de-python/

- Sintaxis sencilla y legible: esto es especialmente importante a la hora de desarrollar aprendizaje automático en el que se busca una implementación y análisis rápidos de los modelos y algoritmos.
- Fusión con otras tecnologías: se combina eficazmente con herramientas desarrolladas para otras tecnologías, como puede ser la biblioteca de visualización de datos Matplotlib.
- Flexibilidad y extensibilidad: permite unificar de manera sencilla el aprendizaje automático con otros aspectos de la IA, en el caso de este estudio, el procesado de lenguaje natural.

Debido a todas las ventajas mencionadas, Python es un lenguaje muy utilizado a la hora de trabajar para resolver problemas con Inteligencia Artificial y en particular con técnicas de aprendizaje automático, lo cual ha llevado a la conclusión de que este trabajo debía realizarse utilizando Python.

3.3 Librerías utilizadas

Para el desarrollo y evaluación del modelo BERT, utilizaremos una serie de librerías que nos permitirán trabajar con datos, entrenar y evaluar el modelo, así como visualizar los resultados obtenidos, en concreto:

- Pandas: Es una librería que ayuda a manejar y trabajar con datos de manera eficiente. Proporciona estructuras de datos especiales, como DataFrames, que nos permiten organizar y manipular nuestros datos de forma sencilla [18].
- Numpy: Es una librería esencial para el procesamiento numérico en Python. Proporciona herramientas para trabajar eficientemente con matrices y arreglos de datos multidimensionales [19].
- Matplotlib.pyplot: Es una librería de visualización en Python que permite crear gráficos y visualizaciones de datos de manera fácil y flexible [20].
- Seaborn: Es una librería de visualización de datos que ayuda a crear gráficos estadísticos atractivos y comprensibles. Es especialmente útil para explorar patrones y relaciones en nuestros datos [21].

- Sys: Es una librería que proporciona funciones y variables para interactuar con el intérprete de Python y su entorno, como la gestión de argumentos de línea de comandos o la salida estándar [22].
- Logging: Es una librería que permite registrar eventos y mensajes durante la ejecución de un programa, lo cual es útil para depurar errores y monitorear el comportamiento del software [23].
- Tensorflow: Es una potente biblioteca de código abierto desarrollada por Google para el aprendizaje automático y el aprendizaje profundo. Proporciona herramientas y recursos para construir y entrenar modelos complejos, como redes neuronales, para diversas aplicaciones [24].
- Gc: Es el recolector de basura de Python, que se encarga de liberar automáticamente la memoria que ya no está siendo utilizada por el programa [25].
- Torch: Es una librería que proporciona funciones y variables para interactuar con el intérprete de Python y su entorno, como la gestión de argumentos de línea de comandos o la salida estándar [26].
- Evaluate: Es una biblioteca de código abierto alojada en PyPI que facilita la evaluación de modelos de aprendizaje automático en diversas tareas, como la clasificación de texto, la traducción automática y la respuesta a preguntas. Proporciona métricas estándar y herramientas para comparar el rendimiento de diferentes modelos [27].
- Scikit-learn (o sklearn): se trata de una de las bibliotecas más populares y ampliamente utilizadas en Python para el aprendizaje automático. Proporciona una amplia gama de algoritmos de aprendizaje supervisado y no supervisado, así como herramientas para el preprocesamiento de datos, evaluación de modelos, selección de características y más [28].
- Hugging Face Transformers: es una librería dentro de la comunidad Hugging Face que permite experimentar con modelos basados en *transformers*[6] para diversas

[6] Transformers (Arquitectura de redes neuronales): Modelo de aprendizaje profundo basado en el mecanismo de atención, que permite procesar secuencias de datos (como texto) considerando todas las partes

tareas como procesamiento natural del lenguaje, visión computacional, audio y otras. Ofrece documentación para usar, entrenar y compartir modelos [29].

En resumen, estas librerías y funciones son herramientas que nos facilitan para trabajar con el modelo BERT propuesto en este trabajo, facilitando la carga de datos, el entrenamiento, la evaluación y la visualización de los resultados obtenidos.

3.4 Google Colab

Google Colaboratory (Colab) es un entorno de desarrollo en la nube creado por Google, que permite a los usuarios colaborar en *notebooks*[7] interactivos basados en Jupyter. Estos cuadernos se ejecutan en servidores de Google, ofreciendo recursos computacionales gratuitos para el análisis de datos y la ejecución de código. Colab proporciona una interfaz intuitiva con herramientas preinstaladas para realizar tareas de aprendizaje automático y análisis de datos de manera eficiente [30].

La principal ventaja de Colab es su accesibilidad, ya que no requiere configuración local y permite la colaboración en tiempo real. Además, proporciona acceso a recursos potentes, como GPUs, que aceleran el procesamiento de datos y el entrenamiento de modelos de aprendizaje automático.

3.5 Wandb

Wandb [31] se trata de una plataforma que permite monitorizar búsqueda de hiperparámetros, métricas del sistema y predicciones para comparar modelos en tiempo real. Es utilizada por grandes empresas dedicadas a investigación para la automatización

de la entrada simultáneamente, en lugar de hacerlo de forma secuencial. Obtenido de: https://la.blogs.nvidia.com/blog/que-es-un-modelo-transformer/

[7] Interfaz Notebook (o cuaderno computacional): Entorno virtual de cuaderno utilizado para la programación literaria. Combina la funcionalidad del software procesamiento de textos con ambos shell y kernel del lenguaje de programación de esa computadora portátil. Obtenido de: https://es.wikipedia.org/wiki/Interfaz_de_notebook

de procesos de entrenamiento y elaboración de reportes. Permite el uso de su API[8] de forma gratuita creando una cuenta, ya sea académica o personal.

3.6 Twitch

Twitch es una plataforma de streaming en vivo [32] que se ha consolidado como un espacio de interacción social en tiempo real, especialmente en el ámbito de los videojuegos y eventos en directo, aunque ha diversificado su contenido a áreas como la música o el arte. La plataforma facilita la comunicación bidireccional entre creadores de contenido y espectadores a través de chats en vivo. Estos chats, tras un análisis adecuado, pueden ser una fuente valiosa de información para medir el *engagement* de los usuarios, evaluar la respuesta emocional que generan los creadores de contenido y comprender las dinámicas de interacción social que se desarrollan en la plataforma. El estudio de estos datos puede proporcionar información relevante para investigadores y profesionales del marketing, entre otros, interesados en comprender el comportamiento de las audiencias en entornos digitales.

3.7 TwitchDownloader

TwitchDownloader [33] es una herramienta de software de código abierto que facilita la descarga de vídeos, clips y chats de la plataforma Twitch. Esta herramienta es especialmente útil para investigadores y analistas que desean recopilar datos de Twitch para su posterior análisis. TwitchDownloader ofrece una interfaz cómoda y sencilla con funciones para descargar vídeos completos (VODs o *Video on demand*), clips cortos, chats en diferentes formatos (JSON, HTML, texto plano), y actualizar o visualizar chats descargados con emojis y emotes[9].

[8] API (Interfaz de Programación de Aplicaciones): Conjunto de definiciones y protocolos que facilitan la creación e integración de software de aplicaciones, permitiendo que diferentes aplicaciones se comuniquen y compartan datos entre sí. Obtenido de: https://es.wikipedia.org/wiki/API

[9] Emote: Entrada de chat de un cliente basada en texto que indica que una acción está tomando lugar. A diferencia de los emoticones, no son arte de texto, sino que describen la acción usando palabras o imágenes (similares a los emojis). Obtenido de: https://es.wikipedia.org/wiki/Emote

3.8 Métricas de rendimiento

Con el fin de evaluar la eficacia del modelo de clasificación y compararlo con enfoques previos, se recurre a una serie de métricas objetivas ampliamente utilizadas en el campo.

Consideremos, por ejemplo, un clasificador de emociones diseñado para determinar la presencia o ausencia de enfado en un texto. Al realizar una predicción, existen cuatro posibles resultados, dependiendo de la relación entre el resultado obtenido y la salida esperada:

- Verdadero positivo (*True positive* o TP): El clasificador predice correctamente la presencia de enfado en un texto que efectivamente lo contiene.
- Falso positivo (*False positive* o FP): El clasificador predice erróneamente la presencia de enfado en un texto que no lo contiene.
- Falso negativo (*False negative* o FN): El clasificador predice erróneamente la ausencia de enfado en un texto que sí lo contiene.
- Verdadero negativo (*True negative* o TN): El clasificador predice correctamente la ausencia de enfado en un texto que no lo contiene.

Estos posibles resultados pueden ser representados en una matriz de confusión, como se ilustra en la Figura 2, donde las filas corresponden a las etiquetas verdaderas (presencia o ausencia de odio) y las columnas a las etiquetas predichas por el clasificador.

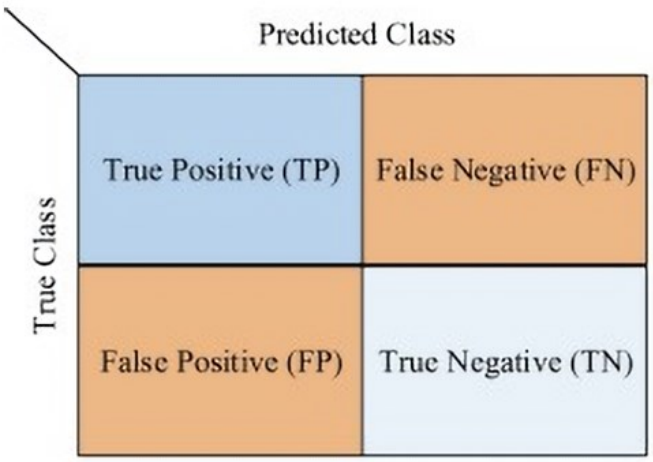


Figura 2: Matriz de confusión. *Artificial Intelligence-Based Brain-Computer Interface*, cap. 14, figura 7 - ScienceDirect [34]

Para evaluar el rendimiento del modelo de clasificación desarrollado en este estudio, se emplearon diversas métricas basadas en los resultados de la matriz de confusión. Estas métricas son:

- *Accuracy* (Exactitud): Mide la proporción de clasificaciones correctas sobre el total de predicciones (Ecuación 1).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Ecuación 1: Función Accuracy

- *Precision* (Precisión): Evalúa, para cada clase, la proporción de instancias correctamente clasificadas sobre el total de instancias predichas como pertenecientes a esa clase, indicando la capacidad del modelo para evitar falsos positivos (Ecuación 2).

$$Precision = \frac{TP}{TP + FP}$$

Ecuación 2: Función Precision

- *Recall* (Exhaustividad o sensibilidad): Calcula la proporción de instancias correctamente clasificadas sobre el total de instancias que realmente pertenecen a esa clase, reflejando la capacidad del modelo para evitar falsos negativos (Ecuación 3).

$$Recall = \frac{TP}{TP + FN}$$

Ecuación 3: Función Recall

- *F1-score* (Puntuación F1): Una medida sintetiza la precisión (*Precision*) y la exhaustividad (*Recall*), ofrece un equilibrio entre ambas y resulta particularmente útil para evaluar el rendimiento de modelos en escenarios donde las clases no están igualmente representadas (Ecuación 4).

$$F_1\text{-}score = 2\,\frac{precision \times recall}{precision + recall}$$

Ecuación 4: F1-score con β = 1.

4

Ampliación del corpus de videojuegos en la plataforma Twitch

4.1 Introducción

Este capítulo tiene como objetivo describir la metodología empleada para la ampliación de la base de datos de Twitch en entornos de videojuegos. Esta base de datos será utilizada para el análisis de la respuesta emocional generada ante diferente contenido de videojuegos transmitido en la plataforma Twitch, y empleada en una versión inicial en trabajos previos [1].

En la Tabla 1, se muestra la amplia variedad de géneros y experiencias de juego recogidas en el *dataset*[10] tras finalizar la ampliación.

Juego	Género Principal	Tipo de interacción de los jugadores
World of Warcraft	MMORPG	Juego basado en la cooperación con miembros del equipo
League of Legends	MOBA	Juego competitivo basado en equipos (cooperar con el equipo, competir con otros equipos)
Heroes of the Storm	MOBA	
Fornite	Battle royale shooter	
Apex Legends	Battle royale shooter	
Diablo 4	ARPG	Juego para un jugador
Rise of the Ronin	ARPG	
Red Dead Redeption 2	Action / Adventure	
Dark Souls II y III	ARPG	Juego para un jugador, también tiene cooperativo y competitivo mecánica en el juego en línea
Dragon Ball Sparking Zero	Fighting game	
Fornite	Battle Royale	

[10]Dataset o Conjunto de datos: Colección de datos habitualmente tabulada. Obtenido de: https://es.wikipedia.org/wiki/Conjunto_de_datos

Counter Strike: Global Offensive 2	First person shooter	Juego competitivo basado en equipos (cooperar con el equipo, competir con otros equipos)
Valorant	First person shooter	
Call of Duty: Warzone	Battle royale shooter	
Call of Duty: Modern Warfare II y III	First person shooter	Juego competitivo para un jugador
Minecraft	Sandbox / Open world survival	Juego para un jugador, también tiene cooperación
Lego Fornite	Open world survival	
War Thunder	Vehicular combat multiplayer	Juegos competitivos basado en equipos (cooperar con el equipo, competir con otros equipos)
Starfield	RPG	Juego para un jugador
Pokemon Community Game	RPG	
Assassin's Creed Origins	ARPG	
Elden Ring	ARPG	
Genshin Impact	ARPG	
Outlast	First-person survival horror	
Teamfight Tactics	Auto battler	
Suika Games	Puzzle	

Tabla 1: Género de videojuegos y tipo de interacción del jugador seleccionado para crear y ampliar la base de datos

Se describirá el proceso de selección de comentarios y mensajes, con un enfoque particular en el aumento de mensajes con polaridades y emociones desbalanceadas, así como en la incorporación de una mayor diversidad lingüística en la temática *gamer*.

Esta ampliación permitirá equilibrar la base de datos y obtener una representación más completa y precisa de las opiniones y emociones de los usuarios en las plataformas de *streaming*, lo que potencialmente conllevará a la obtención de mejores resultados en el análisis final de la respuesta emocional mediante técnicas de inteligencia artificial, en concreto técnicas de ML.

4.2 Descripción del corpus

Esta base de datos o corpus contiene como muestras los mensajes escritos por los espectadores en diferentes transmisiones en directo, todos estos relacionados con videojuegos. Para nuestro estudio esta base de datos o corpus se encuentra en un archivo con extensión .xlsx, el cual es un formato de hoja de cálculo utilizado por el programa informático Microsoft Excel. Dicho archivo está compuesto por tres columnas, cada una de ellas asociada a una etiqueta principal: texto, polaridad y emociones [35].

1. En la columna de “Texto” encontramos una variable independiente, que representa la característica de entrada. Estos son mensajes/comentarios en bruto de usuarios de

la plataforma Twitch reaccionando al directo, incluyendo ortografía, puntuación y formato original. Esta variable requerirá de un procesamiento adicional para poder analizarla, los detalles sobre este procesamiento se explicarán en los siguientes apartados.

2. En la columna de "Polaridad", encontramos la variable objetivo-referida al sentimiento general del texto/mensaje asignado, con tres valores:

    a. Positivo: esta etiqueta se utiliza para aquellos mensajes que expresan un sentimiento favorable o agrado hacia un tema o entidad específica.
    b. Negativo: la contraparte de la etiqueta anterior, que expresa sentimiento desfavorable o desagrado hacia un tema o entidad específica.
    c. Indeterminado: recoge los mensajes que no han podido ser debidamente etiquetados, ya sea por falta de un contexto o que no representan un sentimiento claro.

3. En la columna de "Emociones", se observa la segunda variable objetivo pendiente de la emoción general del comentario/texto. Se ha utilizado un conjunto de categorías de emociones predefinidas en base al modelo de Plutchnik [35]:

    a. Aprobación/Empatía/Confianza (aprobación): Esta etiqueta se aplica a los mensajes que expresan acuerdo y aprobación, generalmente dirigidas a lo que está sucediendo en la transmisión, al tema tratado del directo, o para expresar empatía comprensión, solidaridad, compromiso o confianza con el streamer, con respecto a lo que dice o hace.
    b. Desaprobación: La contraparte de la etiqueta anterior es la de "Desaprobación" que describe mensajes que muestran disgusto y desacuerdo con el hablante de forma moderada (en caso de tener un carácter más intenso, encontraremos esos mensajes bajo la etiqueta de "Enfado"). También incluido bajo esta categoría hay mensajes donde los usuarios del chat indican que el contenido o el tema les aburre, les resulta repetitivo o no les interesa por razones personales.
    c. Decepción/Tristeza (tristeza): Esta etiqueta hace referencia a expectativas que no se han cumplido, y se establece, por tanto, un contraste entre lo que se esperaba que sucediera y lo que ha sucedido. Esta categoría también incluye

sentimientos de tristeza que se expresan explícitamente, ya sea a través de palabras (tristeza, llanto, caída, etc.) o emotes/emoticonos.

d. Enfado/Ira (enfado): etiqueta referida cuando una emoción de carácter negativo (desaprobación, decepción, etc.) se expresa de forma intensa y categórica, o cuando en el chat estén presentes mensajes de resentimiento y enfado que contengan reproches, insultos, etc.

e. Interés/Aceptación/Hype (hype[11]): Una etiqueta que hace referencia a la expresión de ese sentimiento provocado por el deseo y las expectativas sobre un evento futuro.

f. Indeterminado: numerosos mensajes carecen del contexto suficiente para ser convenientemente etiquetados, ya que los mensajes de Twitch se producen en los chats de forma muy fragmentada, en una dinámica ágil y conversacional que discurre, en ocasiones, a gran velocidad, muy ligada a lo que sucede en pantalla. Además, en la comunicación digital hay que considerar el uso del humor, el sarcasmo y la hipérbole, que también pueden dificultar la categorización.

4.3 Metodología de selección de chats en Twitch

La selección adecuada de transmisiones, junto con sus respectivos chats, resulta crucial para la ampliación del corpus en este estudio. Estos chats consisten en registros textuales de los mensajes enviados por los espectadores a lo largo de las transmisiones en directo, recopilados de forma cronológica. De ellos se extrajeron las diferentes muestras que conformaron la base de datos inicial.

Partimos de que nuestro corpus anterior, utilizado en trabajos previos [1], contaba con una serie de criterios específicos en la selección de directos. Por coherencia con los estudios anteriores, mantendremos muchos de ellos:

- Canales en español. Se descartaron a los streamer bilingües, cuyos chats podrían alternar inglés y español. Aunque queda constancia de la alta proporción de términos

[11] Hype: Excitación o anticipación excesiva generada en torno a un producto, evento o persona, a menudo impulsada por estrategias de marketing y la difusión en redes sociales. Obtenido de: https://concepto.de/hype/

en inglés registrados en las transmisiones de Twitch, debido a que es una red muy utilizada por jóvenes y porque contienen un léxico especializado en videojuegos [36].

- Canales cuyo contenido principal sean los videojuegos.
- Canales con un límite de espectadores (1000). Este criterio estuvo presente en el corpus anterior, pues en los chats con un mayor número de participantes, la comunicación o interacción se vuelve casi imposible. Es decir, la elevada tasa de mensajes entrantes provoca el denominado "*scroll factor*" [37], tendiendo a convertirse en lo que se ha descrito como "una cascada ilegible de texto" [38].

  Sin embargo, en esta nueva ampliación del corpus abordaremos chats sin límite de espectadores, implementando métodos variados de selección de mensaje. Esto permitirá la extracción de mensajes con un fuerte componente emocional, nuevos tipos de muestras con lenguajes diversos y expresiones comunes en este tipo de transmisiones masivas, enriqueciendo el corpus y promoviendo una comprensión más profunda de las emociones y la comunicación.
- Canales en los que el streamer tiene cámara web. Se dio preferencia a estos *streams*[12], pues la participación suele ser mayor en los canales donde el streamer muestra su rostro a través de una cámara web integrada en la transmisión en vivo [39].

Si bien la base de datos previa incluye chats de diferentes transmisiones, seleccionados en función de esta serie de criterios específicos, no se logró un balance adecuado en cuanto a la distribución de emociones. En la Tabla 2, se muestra la distribución proporcional de mensajes por cada emoción en el corpus de partida [35]; haciendo así visible que uno de los objetivos de la ampliación es *incrementar el número de muestras/mensajes de las emociones que se encuentran descompensadas*.

[12] Streams (en el contexto de streaming): Transmisión continua de datos de audio o video a través de internet, permitiendo a los usuarios consumir contenido multimedia en tiempo real sin necesidad de descargarlo por completo. Obtenido de: https://es.wikipedia.org/wiki/Streaming

Clase de emoción	N.º de muestras	% de cada clase
Aprobación/Empatía/Confianza	711	32%
Desaprobación	482	21,7%
Decepción/Tristeza	271	12,2%
Enfado/Ira	168	7,6%
Interés/Aceptación/Hype	268	12,1%
Indeterminado	315	14,2%

Tabla 2: Distribución del corpus anterior en las diferentes categorías de emociones

Es importante destacar que no todos los chats contienen mensajes relevantes o interesantes para nuestro análisis. De hecho, algunos pueden carecer de opiniones o mensajes que puedan ser categorizados correctamente según las emociones objetivo. Por lo tanto, la afinación en la selección de chats y directos concretos es fundamental para encontrar muestras más representativas de las emociones descompensadas que buscamos analizar. Y es que, si elegimos directos al azar, sin buscar valores que nos afinen la búsqueda, no podremos encontrar muestras que equilibren el corpus de manera efectiva.

Es cierto que añadir muestras de directos o retransmisiones seleccionadas al azar, en un gran número, incrementa la probabilidad de balancear el corpus a medida que se incorporan más muestras. Sin embargo, si deseamos ser más eficientes en el proceso de ampliación, nuestra hipótesis plantea que debemos buscar chats concretos que cumplan determinados criterios específicos (criterios que definiremos en los siguientes puntos), en lugar de una selección totalmente aleatoria. De este modo, podremos optimizar el proceso de ampliación del corpus, obteniendo muestras más representativas de las emociones descompensadas de forma más precisa y sistemática.

Además, de cara a la ampliación del corpus, se añade otro objetivo complementario: *aumentar la diversidad lingüística "gamer"*. La incorporación de una mayor variedad de expresiones y léxico característicos de la comunidad de videojuegos nos permitirá obtener un corpus más representativo y fiel al lenguaje propio de este ámbito. De este modo, al cumplir con el objetivo principal de equilibrar la distribución emocional y lograr una mayor diversidad lingüística de los jugadores, el corpus ampliado reflejará de manera más precisa la riqueza expresiva y emocional presente en los chats de Twitch centrados en videojuegos.

En los siguientes apartados se detallará el procedimiento y metodología que se siguió para seleccionar los nuevos chats y comentarios en Twitch en el entorno de videojuegos.

4.3.1 Afinación del proceso de búsqueda

4.3.1.1 Búsqueda por videojuego

En el proceso de afinación y enriquecimiento del corpus lingüístico relacionado con los videojuegos, se contemplan dos enfoques principales de búsqueda de transmisiones en directo: búsqueda en nuevos géneros y lanzamientos y búsquedas por emociones. A continuación, se describirán cada una de ellas con más detenimiento.

Búsqueda de transmisiones centradas en nuevos géneros y lanzamientos

Este enfoque se basa en la premisa de que cada nuevo videojuego, expansión, contenido descargable (DLC o *DownLoadable Content*) o lanzamiento reciente incorpora una terminología especializada, expresiones idiomáticas y jergas propias de su género y comunidad de jugadores [15]. Al incluir transmisiones que abordan estos nuevos títulos, se logra enriquecer y diversificar el corpus lingüístico.

En la Tabla 3, se muestra la variedad de títulos y géneros empleados en el desarrollo y ampliación del corpus, incluyendo los nuevos géneros y lanzamientos desde 2020.

Títulos	CS: GO 2, Valorant, Call of Duty: Warzone, Call of Duty: Modern Warfare II y III, Minecraft, Lego Fornite, War Thunder, Starfield, Pokemon Community Game, Assassin's Creed Origins, Elden Ring, Genshin Impact, Outlast, Teamfight Tactics, Suika Games
Géneros	First person shooter, Sandbox, Open world survival, Vehicular combat multiplayer, RPG, First-person survival horror, Auto battler, Puzzle

Tabla 3: Títulos y géneros de videojuegos presentes en la base de datos

El objetivo principal es ampliar la diversidad lingüística del conjunto de datos, capturando el lenguaje y las expresiones características de cada nueva incorporación al mercado de los juegos. Esto no solo implica la adición de nuevas palabras y términos técnicos, sino también la incorporación de modismos, giros idiomáticos y formas de expresión propias de cada comunidad de jugadores. Algunos ejemplos de terminología *gamer* son:

- "**GG**": *Good Game* (Buen juego).
- "**Noob**": Jugador principiante.
- "**Camper**": Jugador que se esconde en un lugar para atacar a sus oponentes.

- "**Farmear**": Repetir la misma acción para obtener experiencia o dinero.
- "**P2W**": *Pay to Win* (Pagar para ganar).

Además, este enfoque permite mantener el dataset actualizado y relevante, ya que constantemente se están lanzando nuevos títulos, géneros emergentes y actualizaciones significativas en este mundo. Al incluir transmisiones relacionadas con estas novedades, se garantiza que el corpus refleje las tendencias y el lenguaje más reciente utilizado por los jugadores.

La Tabla 4 presenta una amplia selección de nuevas expresiones lingüísticas surgidas en el ámbito de los videojuegos, adaptadas a cada género y que reflejan la evolución del lenguaje en este entorno digital.

Género de videojuego	Términos
MOBA	*"new **meta**?"* *"**Bot feeder**"* *"este tio **gankea** nivel 2 y pretende ganar vaya manco"*
Shooter	*"**NT**"* *"Eso práctica **aim**"* *"se viene **lurkeadas** epicas de hitbox con el nuevo agente"*
ARPG	*"pilla el **loot**"* *"Demasiado **CC**"* *"la **build** de charco es monstruosa"*

Tabla 4: Ejemplo de terminología y expresiones idiomáticas presentes en el dataset

Búsqueda de transmisiones centradas en el carácter emocional

El segundo enfoque de búsqueda se centra en el carácter emocional de las transmisiones, con el objetivo de lograr una representación equilibrada de diferentes estados afectivos dentro del corpus lingüístico, ya que algunas emociones estaban inicialmente poco representadas. Este abarca una amplia gama de emociones, desde la euforia y el entusiasmo hasta emociones más intensas como el enfado, el rechazo y la decepción, todas ellas asociadas al mundo de las transmisiones de videojuegos. A continuación, se detallan los diferentes tipos de emociones y las características de las transmisiones que se buscan para cada uno:

A) Emoción: Hype y entusiasmo

Se buscan transmisiones que reflejen la euforia y el entusiasmo generados por las futuras novedades en el mercado de juegos, como *trailers*, presentaciones, ferias y eventos. Esto permite capturar expresiones y matices lingüísticos asociados a la emoción del "*hype*" y la anticipación.

B) Emoción: Enfado y rechazo

La búsqueda se centra en transmisiones que aborden videojuegos controversiales u "odiados" por diferentes motivos, como comunidades tóxicas, prácticas controvertidas (*pay-to-win*) o decepciones generalizadas. Esto enriquece el corpus con expresiones de enfado, rechazo y críticas negativas.

C) Emoción: Decepción y desaprobación

Se consideran *streamings* que expresen decepción y desaprobación hacia juegos que no cumplieron con las expectativas de los jugadores. Esto permite capturar matices lingüísticos asociados a estas emociones negativas.

4.3.1.2 Búsqueda por creador/streamer

La selección adecuada de *streamers* y creadores de contenido en Twitch es un aspecto fundamental para lograr un *dataset* lingüístico equilibrado y representativo del lenguaje utilizado en el ámbito de los videojuegos. El objetivo principal es capturar la diversidad lingüística y emocional presente en esta comunidad, considerando factores como el género, el nivel de competitividad, el tamaño de la audiencia y las características personales de los creadores de contenido. Seguiremos una serie de nuevos criterios para cumplir con los dos objetivos mencionados anteriormente, y que se describen a continuación con más detalle.

Criterio de representación equilibrada de género

Con el fin de reflejar la diversidad inherente a la comunidad de jugadores, se ha implementado un criterio de selección que busca una representación equilibrada de género entre los streamers incluidos en el corpus. Esta decisión se fundamenta en la premisa de que el género puede influir en el estilo de comunicación, la terminología empleada y las expresiones idiomáticas utilizadas durante las transmisiones. El proceso de selección ha implicado la identificación y evaluación de creadores de contenido de diferentes géneros, con el objetivo de lograr una representación equitativa en el conjunto de datos.

La Tabla 5 presenta la distribución por género de los/as creadores/oras de contenido en Twitch que conforman el conjunto de datos utilizado en este estudio, incluyendo tanto a aquellos/as cuyas transmisiones fueron analizadas en la presente investigación como a los/as que ya formaban parte del corpus preexistente. Los datos se obtuvieron a partir de los directos realizados por estos/as.

Streamers	**Mujeres**	biyin_, bollostream, IamCristinini, ipandarina, littleragergirl, luly, Marangi, Mayichi, mery_soldier, Nissaxter, odii, paolapinar, paracetamor, PauSenpaii, phoebina, Pikcal, SrtaPeliverde, Tigry86, Vivi_Streams, uxusan y Zeiling (22)
	Hombres	Anytimeshield, belvid, champepro, ElmiilloR, evangelion0, Exilom11, FolagorLives_Valorant, hawnktv, IlloJuan, kryp, KoffingSonriente, Mixwell, Nano__hots, Paxelol, RaevenCCA, raevencca, Rubius, scept_, SeVenJungle, tense198_v2, TheGrefg, Wolfangkillers, xixauxas, yyyugo_tv y zullhammer (25)

Tabla 5: Creadores de Twitch incluidos en el dataset

Las observaciones preliminares sugieren que existen matices lingüísticos y enfoques comunicativos distintos entre diferentes géneros, y por tanto planteamos la hipótesis de que se reflejará también en sus correspondientes chats, lo que enriquecerá la variedad de la base de datos.

Si bien la investigación sobre el impacto del género en la recepción del discurso en plataformas como YouTube o Twitch aún se encuentra en sus primeras etapas, la hipótesis planteada en este apartado sugiere que la representación equilibrada de género en el corpus puede contribuir al balanceo y aumento de la diversidad emocional de este.

Existen trabajos de investigación dedicados a evaluar la desigualdad del género y la inclusión en el mundo de los videojuegos [40, 41, 42, 43], así como la presencia de la sexualización femenina en ellos [44, 45, 46]. Sin embargo, este estudio se distingue por ser pionero en el estudio del lenguaje y sus características emocionales en el ámbito de los videojuegos desde la perspectiva de la creación de contenido y, como hemos explicado

anteriormente, con un enfoque centrado en la recepción o la vista del público. Recopilando lo que serían datos novedosos sobre este tipo de estudios/trabajos.

En un futuro, esto podría permitir comprender mejor las experiencias de las creadoras femeninas en el entorno digital, identificar las prácticas discriminatorias que enfrentan y desarrollar estrategias para promover un ambiente más inclusivo y respetuoso en el mundo del *streaming*.

Criterio de diversidad en los niveles de competitividad

Se ha contemplado la inclusión de streamers con diferentes niveles de competitividad, abarcando desde jugadores casuales hasta profesionales y de élite. Este criterio persigue la identificación de terminología específica y expresiones idiomáticas propias de cada nivel de habilidad y competitividad en los videojuegos. Se espera que los jugadores profesionales empleen un lenguaje más técnico y especializado, mientras que los jugadores casuales puedan utilizar un vocabulario más accesible y coloquial. Esta diversidad enriquece el corpus y lo hace más representativo de la amplia gama de habilidades y perfiles presentes en la comunidad *gamer*.

Criterio del tamaño de la comunidad de seguidores

Se han seleccionado streamers con comunidades de seguidores de diferentes tamaños, desde canales con audiencias masivas hasta aquellos con comunidades más pequeñas y acotadas. El objetivo principal es observar la variedad lingüística presente en los chats y la naturaleza de los mensajes según el volumen de participantes. En los canales con una audiencia numerosa, se espera encontrar una mayor variedad de registros lingüísticos, mientras que, en las comunidades más pequeñas, la interacción y el lenguaje utilizado pueden ser más íntimos y personalizados.

Criterio de inclusión de streamers con carácter agresivo, hostil y polarizador

Con el propósito de lograr un equilibrio emocional en el *dataset*, se han incluido streamers con personalidades y estilos de comunicación más agresivos, hostiles o polarizadores. Estos perfiles suelen generar reacciones emocionales intensas en la audiencia, lo que se refleja en el lenguaje utilizado en los chats.

La incorporación de estos permite capturar una amplia gama de expresiones y matices lingüísticos asociados a emociones como el enfado, la desaprobación, la frustración, la confrontación o la ira, enriqueciendo así la diversidad emocional del corpus.

4.3.2 Descarga de chats

Para la obtención de los datos, se empleó una herramienta de software libre denominada TwitchDownloader [33]. Este programa permite la descarga y extracción de los chats de las transmisiones seleccionadas en diversos formatos:

- JSON (Notación de Objetos JavaScript, del inglés *JavaScript Object Notation*): Conserva toda la información original del chat.
- Archivo HTML (Lenguaje de Marcas de Hipertexto, del inglés *HyperText Markup Language*): Facilita la visualización del chat en un navegador web.
- Archivo de texto plano: Ofrece una representación simple y legible del contenido del chat.

Adicionalmente, TwitchDownloader brinda la posibilidad de:

- Actualizar el contenido de un archivo de chat JSON previamente generado.
- Guardar el chat en un formato diferente al original.
- Visualizar el chat con diferentes emotes estáticos y animados utilizados en la plataforma Twitch.

Esta herramienta resultó útil para la extracción y para el futuro análisis de los datos de chat de las transmisiones.

4.3.3 Procesamiento y estructuración de los datos

En este apartado se describe el proceso de procesamiento y estructuración de mensajes de Twitch a partir de un archivo de texto plano (.txt) y convertirlo a uno de Excel (.xlsx). El objetivo es extraer los mensajes relevantes, adaptarlos a un formato en celdas y poder así seleccionar los mensajes eliminando la información innecesaria del chat. A continuación, se detallan los pasos a seguir:

1. *Extracción de mensajes:*
   1.1. Localización de mensajes. Los mensajes se encuentran en el archivo de texto plano, como se puede ver en la Figura 3, con el siguiente formato:

[Fecha y hora] usuario: mensaje

Ejemplo:

[2023-11-25 14:35:36 UTC] aguacaterosa: holaa pandaaa, verás el pase de micros hoy??

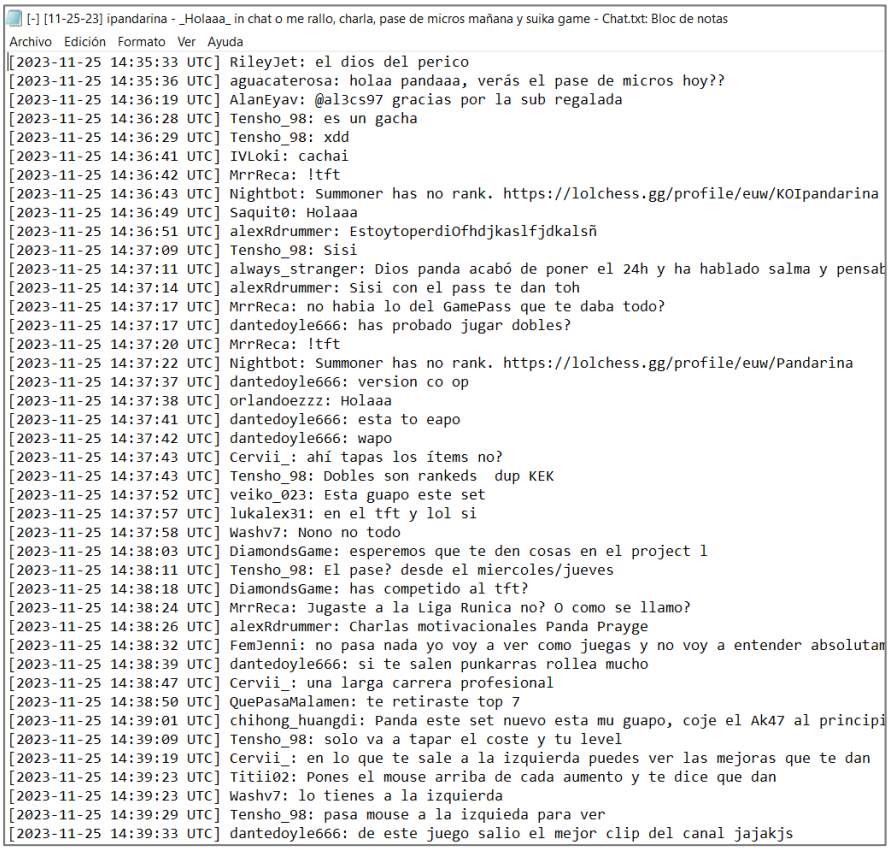
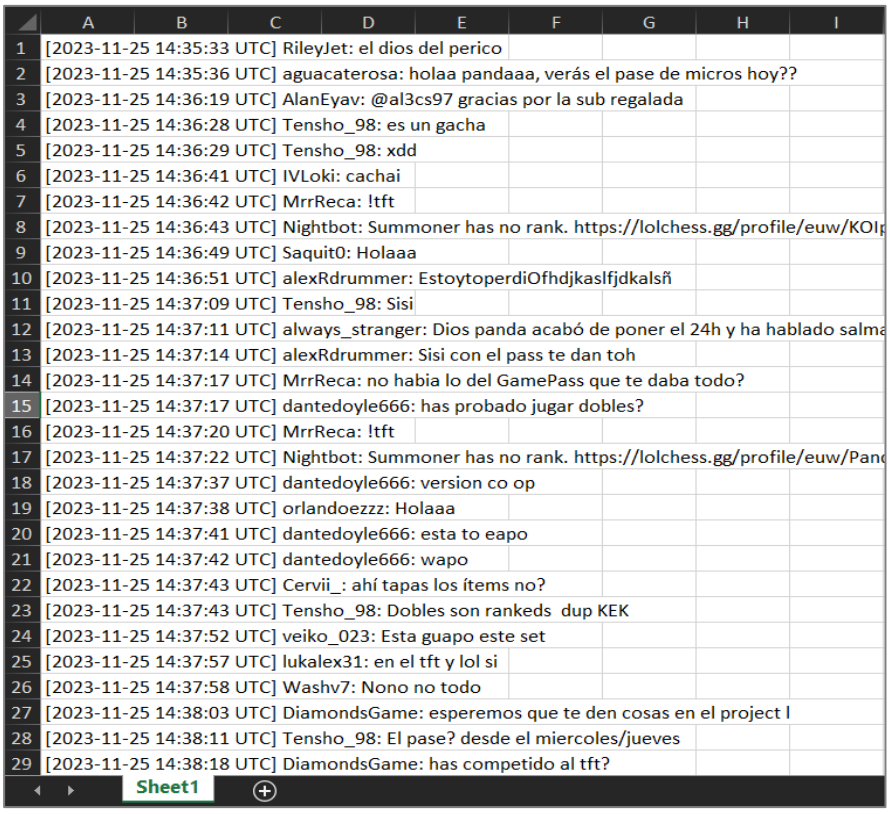

[-] [11-25-23] ipandarina - _Holaaa_ in chat o me rallo, charla, pase de micros mañana y suika game - Chat.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
[2023-11-25 14:35:33 UTC] RileyJet: el dios del perico
[2023-11-25 14:35:36 UTC] aguacaterosa: holaa pandaaa, verás el pase de micros hoy??
[2023-11-25 14:36:19 UTC] AlanEyav: @al3cs97 gracias por la sub regalada
[2023-11-25 14:36:28 UTC] Tensho_98: es un gacha
[2023-11-25 14:36:29 UTC] Tensho_98: xdd
[2023-11-25 14:36:41 UTC] IVLoki: cachai
[2023-11-25 14:36:42 UTC] MrrReca: !tft
[2023-11-25 14:36:43 UTC] Nightbot: Summoner has no rank. https://lolchess.gg/profile/euw/KOIpandarina
[2023-11-25 14:36:49 UTC] Saquit0: Holaaa
[2023-11-25 14:36:51 UTC] alexRdrummer: EstoytoperdiOfhdjkaslfjdkalsñ
[2023-11-25 14:37:09 UTC] Tensho_98: Sisi
[2023-11-25 14:37:11 UTC] always_stranger: Dios panda acabó de poner el 24h y ha hablado salma y pensab
[2023-11-25 14:37:14 UTC] alexRdrummer: Sisi con el pass te dan toh
[2023-11-25 14:37:17 UTC] MrrReca: no habia lo del GamePass que te daba todo?
[2023-11-25 14:37:17 UTC] dantedoyle666: has probado jugar dobles?
[2023-11-25 14:37:20 UTC] MrrReca: !tft
[2023-11-25 14:37:22 UTC] Nightbot: Summoner has no rank. https://lolchess.gg/profile/euw/Pandarina
[2023-11-25 14:37:37 UTC] dantedoyle666: version co op
[2023-11-25 14:37:38 UTC] orlandoezzz: Holaaa
[2023-11-25 14:37:41 UTC] dantedoyle666: esta to eapo
[2023-11-25 14:37:42 UTC] dantedoyle666: wapo
[2023-11-25 14:37:43 UTC] Cervii_: ahí tapas los ítems no?
[2023-11-25 14:37:43 UTC] Tensho_98: Dobles son rankeds dup KEK
[2023-11-25 14:37:52 UTC] veiko_023: Esta guapo este set
[2023-11-25 14:37:57 UTC] lukalex31: en el tft y lol si
[2023-11-25 14:37:58 UTC] Washv7: Nono no todo
[2023-11-25 14:38:03 UTC] DiamondsGame: esperemos que te den cosas en el project l
[2023-11-25 14:38:11 UTC] Tensho_98: El pase? desde el miercoles/jueves
[2023-11-25 14:38:18 UTC] DiamondsGame: has competido al tft?
[2023-11-25 14:38:24 UTC] MrrReca: Jugaste a la Liga Runica no? O como se llamo?
[2023-11-25 14:38:26 UTC] alexRdrummer: Charlas motivacionales Panda Prayge
[2023-11-25 14:38:32 UTC] FemJenni: no pasa nada yo voy a ver como juegas y no voy a entender absolutam
[2023-11-25 14:38:39 UTC] dantedoyle666: si te salen punkarras rollea mucho
[2023-11-25 14:38:47 UTC] Cervii_: una larga carrera profesional
[2023-11-25 14:38:50 UTC] QuePasaMalamen: te retiraste top 7
[2023-11-25 14:39:01 UTC] chihong_huangdi: Panda este set nuevo esta mu guapo, coje el Ak47 al principi
[2023-11-25 14:39:09 UTC] Tensho_98: solo va a tapar el coste y tu level
[2023-11-25 14:39:19 UTC] Cervii_: en lo que te sale a la izquierda puedes ver las mejoras que te dan
[2023-11-25 14:39:23 UTC] Titii02: Pones el mouse arriba de cada aumento y te dice que dan
[2023-11-25 14:39:23 UTC] Washv7: lo tienes a la izquierda
[2023-11-25 14:39:29 UTC] Tensho_98: pasa mouse a la izquieda para ver
[2023-11-25 14:39:33 UTC] dantedoyle666: de este juego salio el mejor clip del canal jajakjs

Figura 3: Chat de un directo en archivo .txt

1.2. Extracción del contenido. En este paso se extrae el contenido completo del fichero:

- Seleccionar todo el contenido del archivo de texto plano (Ctrl + E).
- Copiar el contenido seleccionado.
- Abrir un archivo de Microsoft Excel (.xlsx).
- Pegar el contenido en la primera celda.
- Cada línea de mensaje se pegará en una fila separada, como se puede visualizar en la Figura 4.



	A
1	[2023-11-25 14:35:33 UTC] RileyJet: el dios del perico
2	[2023-11-25 14:35:36 UTC] aguacaterosa: holaa pandaaa, verás el pase de micros hoy??
3	[2023-11-25 14:36:19 UTC] AlanEyav: @al3cs97 gracias por la sub regalada
4	[2023-11-25 14:36:28 UTC] Tensho_98: es un gacha
5	[2023-11-25 14:36:29 UTC] Tensho_98: xdd
6	[2023-11-25 14:36:41 UTC] IVLoki: cachai
7	[2023-11-25 14:36:42 UTC] MrrReca: !tft
8	[2023-11-25 14:36:43 UTC] Nightbot: Summoner has no rank. https://lolchess.gg/profile/euw/KOI
9	[2023-11-25 14:36:49 UTC] Saquit0: Holaaa
10	[2023-11-25 14:36:51 UTC] alexRdrummer: EstoytoperdiOfhdjkaslfjdkalsñ
11	[2023-11-25 14:37:09 UTC] Tensho_98: Sisi
12	[2023-11-25 14:37:11 UTC] always_stranger: Dios panda acabó de poner el 24h y ha hablado salma
13	[2023-11-25 14:37:14 UTC] alexRdrummer: Sisi con el pass te dan toh
14	[2023-11-25 14:37:17 UTC] MrrReca: no habia lo del GamePass que te daba todo?
15	[2023-11-25 14:37:17 UTC] dantedoyle666: has probado jugar dobles?
16	[2023-11-25 14:37:20 UTC] MrrReca: !tft
17	[2023-11-25 14:37:22 UTC] Nightbot: Summoner has no rank. https://lolchess.gg/profile/euw/Pan
18	[2023-11-25 14:37:37 UTC] dantedoyle666: version co op
19	[2023-11-25 14:37:38 UTC] orlandoezzz: Holaaa
20	[2023-11-25 14:37:41 UTC] dantedoyle666: esta to eapo
21	[2023-11-25 14:37:42 UTC] dantedoyle666: wapo
22	[2023-11-25 14:37:43 UTC] Cervii_: ahí tapas los ítems no?
23	[2023-11-25 14:37:43 UTC] Tensho_98: Dobles son rankeds dup KEK
24	[2023-11-25 14:37:52 UTC] veiko_023: Esta guapo este set
25	[2023-11-25 14:37:57 UTC] lukalex31: en el tft y lol si
26	[2023-11-25 14:37:58 UTC] Washv7: Nono no todo
27	[2023-11-25 14:38:03 UTC] DiamondsGame: esperemos que te den cosas en el project l
28	[2023-11-25 14:38:11 UTC] Tensho_98: El pase? desde el miercoles/jueves
29	[2023-11-25 14:38:18 UTC] DiamondsGame: has competido al tft?

Sheet1

Figura 4: Chat de un directo en archivo .xlsx

2. *Estructuración de mensajes en Excel:*

2.1. División en columnas:

- Seleccionar todas las filas que contienen los mensajes.
- Acceder a la opción "Datos" en la barra de herramientas.
- Seleccionar "Texto en columnas".
- Elegir la opción "Delimitados".

- En "Separadores", elegir ":".
- Hacer clic en "Aceptar".

En las Figuras 5, 6 y 7 podemos visualizar y seguir los pasos seguidos en el asistente para convertir texto en columnas del programa Microsoft Excel.

2.2. Resultado: Los mensajes se dividirán en columnas separadas, quedando solo el mensaje en una columna.

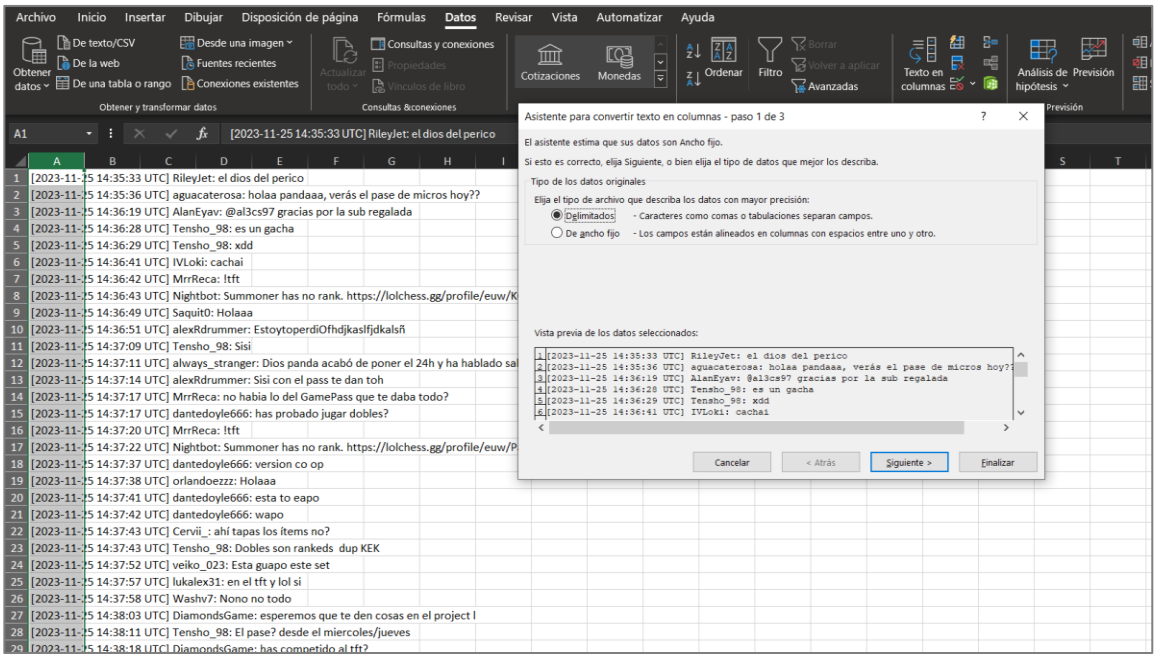


Figura 5: Asistente para convertir texto en columnas – Paso 1 de 3

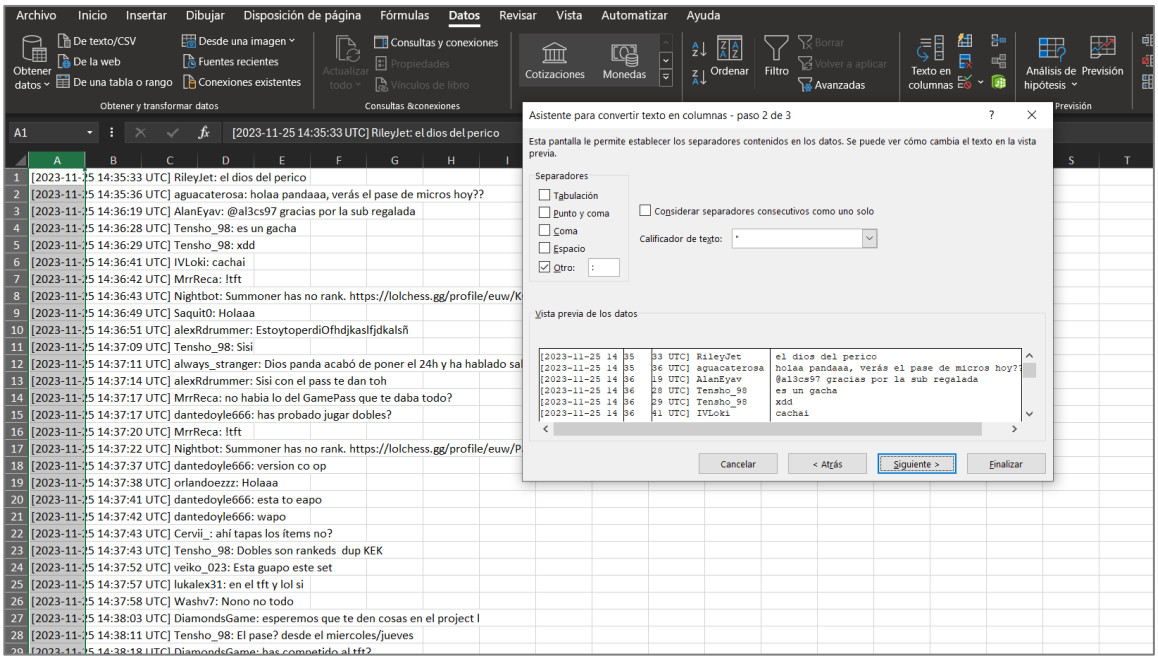


Figura 6: Asistente para convertir texto en columnas – Paso 2 de 3

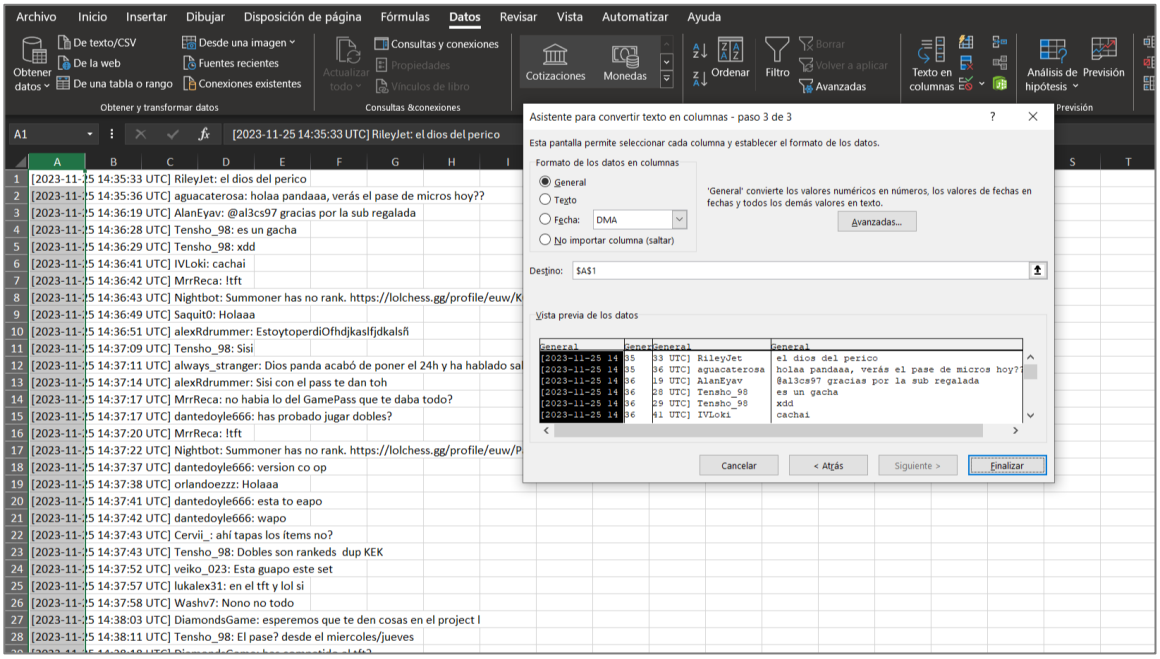


Figura 7: Asistente para convertir texto en columnas – Paso 3 de 3

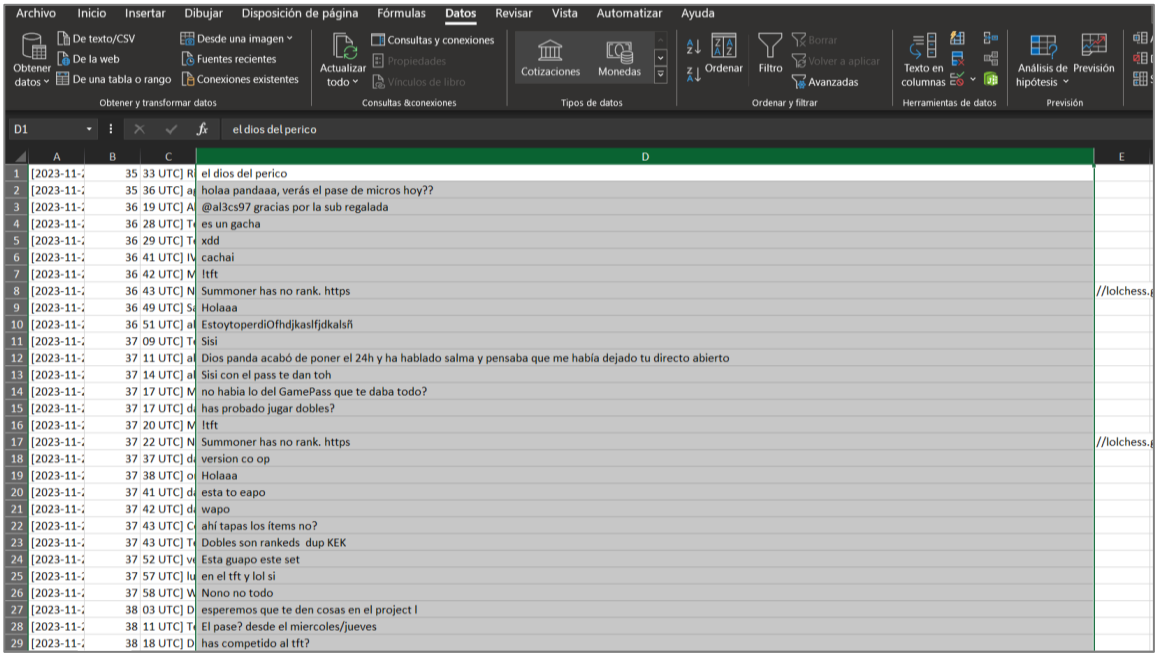


Figura 8: Texto del chat en columnas en un archivo .xlsx

Con los mensajes estructurados en columnas, como se muestra en la Figura 8, se puede proceder al análisis y selección de los mensajes relevantes.

Como consideraciones adicionales, cabe destacar la flexibilidad del proceso descrito, adaptable a diversos formatos de archivos de texto y herramientas de procesamiento de datos. Sin embargo, para optimizar el análisis y la selección de mensajes, es fundamental definir criterios claros y específicos antes de iniciar el proceso. Estos criterios deben estar

alineados con los objetivos del estudio y permitir una selección precisa y eficiente de los mensajes relevantes.

Adicionalmente, una futura automatización del proceso mediante herramientas o scripts puede ser una alternativa viable para ahorrar tiempo y mejorar la eficiencia, especialmente cuando se trabaja con grandes volúmenes de datos.

4.3.4 Selección de mensajes Leet Speak

El uso del *Leet Speak* o *leet* (337 5p34k o 1337 en la escritura *leet*) en la escritura en el chat de Twitch, especialmente en retransmisiones de videojuegos, presenta un desafío y una oportunidad para el análisis de sentimientos en el ámbito del PLN aplicado al español. Este lenguaje, caracterizado por la sustitución de letras por caracteres alfanuméricos, permite a los usuarios evadir filtros de lenguaje [49] y expresar de forma encubierta significados o emociones, lo que dificulta su detección y clasificación.

En el contexto de las transmisiones en directo, el uso de este lenguaje cobra especial relevancia, ya que permite a los espectadores expresar mensajes con emociones negativas hacia los creadores de contenido sin ser detectados por los sistemas de moderación. Estas emociones, como el enfado, la decepción y la desaprobación, son especialmente relevantes debido a que se encuentran subrepresentadas en nuestro conjunto de datos. Incorporar mensajes *leet* que expresen estas emociones podría contribuir a equilibrar el corpus y mejorar la capacidad del modelo para reconocerlas.

Para recopilar estos mensajes, se utilizaron bibliotecas web especializadas [49, 50] y se realizó una búsqueda exhaustiva en diferentes chats, empleando técnicas descritas próximamente en el Apartado 4.4.1.

A continuación, se presentan algunos ejemplos de mensajes *leet* encontrados, junto con su clasificación emocional:

- **M3n7!r050!** (¡Mentiroso!)
  Polaridad: Negativa / Clasificación: Enfado/Ira
- **N0 p/3d0 cr33r 3570** (No puedo creer esto)
  Polaridad: Negativa / Clasificación: Desaprobación
- **57up1d 817ch** (*Stupid bitch* o estúpida puta)
  Polaridad: Negativa / Clasificación: Enfado/Ira

Estos ejemplos ilustran cómo el *Leet Speak* puede ser utilizado para expresar emociones negativas de manera encubierta en el chat de Twitch. El análisis de estos mensajes, junto con el resto del corpus, permitirá desarrollar modelos de aprendizaje automático más precisos y robustos para la detección y clasificación de emociones en el lenguaje de los videojuegos en español.

4.4 Metodología de análisis de chats y comentarios

En este apartado se describe la metodología empleada para el análisis de mensajes en las retransmisiones en directo. Se detallan los procesos de búsqueda, análisis y las limitaciones del estudio.

4.4.1 Proceso de búsqueda y selección de comentarios

La búsqueda de mensajes relevantes para el conjunto de datos se realiza mediante métodos de lectura humana, adaptándose a las características de cada chat. Así, en chats grandes (más de 50.000 comentarios) se aplicó la siguiente metodología:

- Búsqueda específica de términos: Se emplean herramientas de búsqueda (como la 'Lupa' de Microsoft Excel en los ficheros .xlsx o la opción de 'Buscar' por palabras en ficheros .txt) para identificar palabras clave relacionadas con el lenguaje propio de los videojuegos, buscando variedad lingüística.
- Filtrado por emociones: Se filtran mensajes por palabras que indiquen sentimientos o insultos, permitiendo encontrar mensajes con diferentes emociones.
- Lectura diagonal: Se realiza una lectura rápida para identificar chats de interés.

Por otro lado, en chats pequeños (entre 300 y 5.000 comentarios) se siguió la siguiente metodología:

- Lectura minuciosa: Se leen todos los mensajes del chat para obtener una mayor precisión.

La selección de estos mensajes es crucial para construir un corpus de datos de calidad para el análisis de emociones en chats de *streams* de videojuegos. Los criterios de selección permiten filtrar mensajes irrelevantes (*spam*, comentarios personales) y enfocarse en aquellos que expresan opiniones, ideas o emociones relevantes al videojuego. Esto

garantiza un corpus más preciso y confiable para el entrenamiento de modelos de análisis de emociones. En concreto, seguiremos los siguientes criterios de selección:

- Relevancia: Expresan opiniones o ideas relevantes sobre el videojuego.
- Lenguaje propio: Utilizan lenguaje propio del videojuego (español o anglicismos).
- Emociones: Incluyen emociones de enfado/odio, tanto comunes como específicas del videojuego.
- Uso común: Expresan frases o ideas de uso común en la plataforma relacionadas con los videojuegos.
- Exclusión de mensajes irrelevantes: Se excluyen mensajes como spam, comentarios personales o aquellos con polaridad/sentimiento difícil de determinar (sarcasmo, ironía).

Este proceso de búsqueda y selección manual permite obtener un corpus de datos rico y diverso, que refleja el lenguaje utilizado en los chats de directos de videojuegos.

4.4.2 Proceso de análisis

El análisis del *dataset* de comentarios de Twitch se realizó en dos etapas:

1ª Etapa. ***<u>Categorización manual de mensajes:</u>***

Esta etapa la lleva a cabo el alumno y consiste en la categorización de los mensajes/comentarios relevantes por polaridad y emoción. Para ello, el alumno se basa en un decálogo para el etiquetado proporcionado por el equipo de investigación, el cual sirvió como guía fundamental para la categorización precisa y consistente de los mensajes. A mayores de este decálogo, el alumno también se basó en sus conocimientos previos en materia lingüística de videojuegos, en el apoyo que puede encontrar en redes sociales y en búsquedas en internet. De esta manera, identifica y categoriza los mensajes que son de interés para el estudio. Para garantizar un análisis objetivo e independiente del contexto, se realiza una reordenación alfabética de los mensajes. Esta estrategia elimina la influencia del contexto [47] o la agrupación de mensajes con significados similares, permitiendo una evaluación individualizada y objetiva de cada mensaje. De esta manera, se asegura que la categorización se base únicamente en el contenido intrínseco de cada mensaje, evitando cualquier clase de sesgo.

2ª Etapa. <u>Revisión y corrección por parte de expertos</u>

En esta segunda etapa, investigadores de la Universidad de Valladolid y la Universidad Autónoma de Madrid revisan y corrigen la categorización realizada por el alumno. Esta revisión por parte de expertos garantiza la calidad y fiabilidad de las etiquetas asignadas a cada mensaje, asegurando así la precisión del análisis posterior. Se utilizó una revisión por pares. Esta comprobación consistió en que, en un primer paso, se asignó a dos expertos independientes la tarea de etiquetar el corpus por separado. Posteriormente, después de la fase inicial de etiquetado, un tercer experto revisó los resultados para resolver cualquier discrepancia. En caso de desacuerdo entre los tres expertos, estos debatían y deliberaban hasta llegar a un consenso. Si no se podía alcanzar un consenso, el mensaje se descartaba. Este enfoque riguroso era fundamental para prevenir posibles errores o sesgos en el corpus etiquetado final.

La combinación de estas dos etapas permite obtener una categorización rigurosa y precisa de los mensajes del *dataset*, lo que es fundamental para el posterior entrenamiento y evaluación de modelos de aprendizaje automático para el análisis de emociones en chats de transmisiones de videojuegos.

4.4.3 Limitaciones

Durante la realización de la ampliación del corpus se encontraron las siguientes limitaciones en el estudio realizado:

- Tamaño del corpus: El tamaño del corpus es limitado, lo que impide realizar un análisis exhaustivo.
- Sesgos en la selección manual: La selección manual de mensajes puede estar sujeta a sesgos.
- Complejidad del análisis de sentimientos: No siempre es posible identificar con precisión la polaridad y el sentimiento de un mensaje, ya que a veces se mezclan varios o no están claros.
- Falta de recursos: La falta de recursos económicos y humanos limita la cantidad de datos que se pueden recopilar y analizar.

A pesar de estas limitaciones, la metodología empleada permite obtener una valiosa comprensión del lenguaje utilizado en los chats.

4.5 Análisis de los resultados

A continuación, se presenta un resumen del análisis de resultados obtenido del aumento y ampliación del corpus. En cuanto a la polaridad se puede observar que:

- Se ha duplicado el tamaño del conjunto de datos, lo que es positivo para la robustez de los modelos.
- Se ha aumentado significativamente la proporción de muestras negativas, pasando del 43,7% al 49,7%. Esto debería mejorar la capacidad de los modelos para identificar mensajes negativos, que eran un punto débil en trabajos anteriores.
- A pesar del desequilibrio en las polaridades, se espera que el enriquecimiento lingüístico compense este efecto.

En las Tablas 6 y 7, se muestran las comparativas de la distribución porcentual y la del número de muestras correspondientes a las diferentes polaridades del corpus, respectivamente. A modo de ilustrarnos los efectos de la ampliación del trabajo, en comparación con el conjunto de datos del estudio previo [1].

Clase de emoción	% de cada clase (anterior)	% de cada clase (actual)
Positivo	45,1%	36,7%
Negativo	43,7%	49,7%
Indeterminado	11,2%	13,6%

Tabla 6: Distribución y comparación del corpus en las diferentes categorías de polaridad

Polaridades	N.º de muestras en corpus (anterior):	N.º de muestras en corpus (actual):	N.º de muestras añadidas:
Positivo	1000	1686	686
Negativo	968	2279	1311
Indeterminado	247	624	377
Total:	2215	4589	2374

Tabla 7: Comparativa de las muestras para polaridad

Respecto a las emociones se pueden observar las siguientes características en el análisis de resultados:

- Al igual que con la polaridad, se ha duplicado el número de muestras, lo que permite un mejor entrenamiento de los modelos.

- Destaca el aumento considerable de muestras de emociones como "Enfado/Ira" y "Desaprobación", que eran problemáticas en trabajos previos. Esto debería mejorar la detección de estas emociones.
- También se observa un aumento de las muestras de "Aprobación", lo que se asocia a la inclusión de nuevos géneros de videojuegos. Esto enriquece el corpus y permite una mejor representación de la diversidad de emociones en el ámbito del *gaming*.
- A pesar de no haber logrado el equilibrio deseado, se espera que tanto el aumento de las emociones con menor número de muestras y el enriquecimiento lingüístico compensen este efecto.

En las Tablas 8 y 9, se muestran las comparativas de la distribución porcentual y la del número de muestras correspondientes a las diferentes categorías emocionales del corpus, respectivamente. De forma similar a las anteriores tablas, a modo de ilustrarnos los efectos de la ampliación del trabajo.

Clase de emoción	% de cada clase (anterior)	% de cada clase (actual)
Aprobación/Empatía/Confianza	32%	27%
Desaprobación	21,7%	21,1%
Decepción/Tristeza	12,2%	10,2%
Interés/Aceptación/Hype	12,1%	11%
Enfado/Ira	7,6%	16,3%
Indeterminado	14,2%	14,4%

Tabla 8: Distribución y comparación del corpus en las diferentes categorías de emoción

Emociones	N.º de muestras en corpus (anterior):	N.º de muestras en corpus (actual):	N.º de muestras añadidas:
Aprobación/Empatía/ Confianza	711	1239	528
Desaprobación	482	969	487
Decepción/Tristeza	271	469	198
Interés/Anticipación/ Hype	268	504	236
Enfado/Ira	168	746	578
Indeterminado	315	662	347
Total:	2215	4589	2374

Tabla 9: Comparativa de las muestras para emociones

4.6 Conclusiones

Las estrategias de diversificación del corpus han demostrado ser exitosas, incrementando notablemente su representatividad y riqueza lingüística. Esto se refleja en el aumento significativo del tamaño del conjunto de datos, la distribución más equilibrada

de las clases de polaridad y emoción, y la inclusión de una mayor variedad de géneros de entretenimiento digital y creadores de contenido. Es fundamental continuar monitoreando la distribución de las clases y emociones para garantizar un equilibrio adecuado y prevenir sesgos en los modelos. Este monitoreo permitirá realizar ajustes en las estrategias de recolección de datos de ser necesario.

Los resultados obtenidos son altamente prometedores y sientan las bases para el desarrollo de modelos más robustos y precisos para la detección de emociones en el lenguaje de este ámbito. La calidad y representatividad mejoradas del corpus deberían traducirse en un mejor rendimiento de estos modelos en el futuro.

Cabe destacar la adopción de un enfoque centrado en las redes sociales para la selección de títulos de entretenimiento digital y creadores de contenido, utilizando principalmente Reddit[13] y X[14] (anteriormente conocido como Twitter) [47]. Esta estrategia permite aprovechar el poder de la interacción social para identificar títulos y creadores que se ajusten a los criterios establecidos, enriqueciendo aún más el corpus.

En definitiva, la presente investigación ha demostrado la efectividad de las estrategias implementadas para diversificar el corpus lingüístico relacionado con el entretenimiento digital. Tanto en la efectividad de la metodología de selección de muestras como la afinación de búsqueda de las transmisiones. Los resultados obtenidos son alentadores y abren camino para el desarrollo de modelos más avanzados para el análisis de emociones en este ámbito.

[13] Reddit: Plataforma social de noticias y entretenimiento donde los usuarios pueden compartir enlaces, contenido multimedia y discutir temas diversos en comunidades temáticas llamadas "*subreddits*". Obtenido de: https://www.reddit.com/about/

[14] X (red social): Plataforma de microblogging anteriormente conocida como Twitter, que permite a los usuarios publicar mensajes cortos llamados "tweets" e interactuar con ellos a través de respuestas, retuits y me gusta. Obtenido de: https://es.wikipedia.org/wiki/X_Corp.

5

Análisis de la respuesta emocional con BERT: Polaridades y Emociones

5.1 Introducción

En este capítulo de la memoria se abordará el análisis de la respuesta emocional presente en los mensajes del chat de Twitch. Inicialmente, se realizará un análisis descriptivo del conjunto de datos para comprender su estructura y características finales. Posteriormente, se empleará el modelo RoBERTuito, una variante de BERT adaptada al español, para llevar a cabo el análisis. Para ello, se realizará un preprocesamiento de los datos y se seleccionarán los hiperparámetros óptimos utilizando la plataforma Wandb. Finalmente, se procederá al entrenamiento del modelo y al análisis de los resultados obtenidos, con el objetivo de evaluar su rendimiento y capacidad para detectar y clasificar de manera precisa las polaridades y emociones expresadas en cada mensaje del chat de Twitch.

5.2 Análisis descriptivo del dataset

Para esta tarea utilizaremos el conjunto de datos ampliado, descrito y analizado en secciones anteriores. Este surge a partir del estudio de un grupo de investigadores de la Universidad de Valladolid y la Autónoma de Madrid, cuyo principal objetivo es realizar el análisis de sentimientos en el ámbito de los videojuegos.

Los datos han sido extraídos de forma continuada desde el enero de 2022 y está formado por 4589 instancias con tres campos: el texto del mensaje a analizar, su polaridad y la emoción mayoritaria que transmite este comentario.

La Figura 9 muestra la distribución de las instancias en función de su polaridad: 1686 mensajes “Positivo”, 2279 mensajes “Negativo” y 624 mensajes “Indeterminado”. Y en la

Figura 10 se muestra la distribución de las instancias en función de su emoción: 1239 mensajes de "Aprobación/Empatía/Confianza", 969 mensajes de "Desaprobación", 469 mensajes de "Decepción/Tristeza", 504 mensajes de "Interés/Anticipación/Hype", 746 mensajes de "Enfado/Ira" y 662 mensajes "Indeterminado".

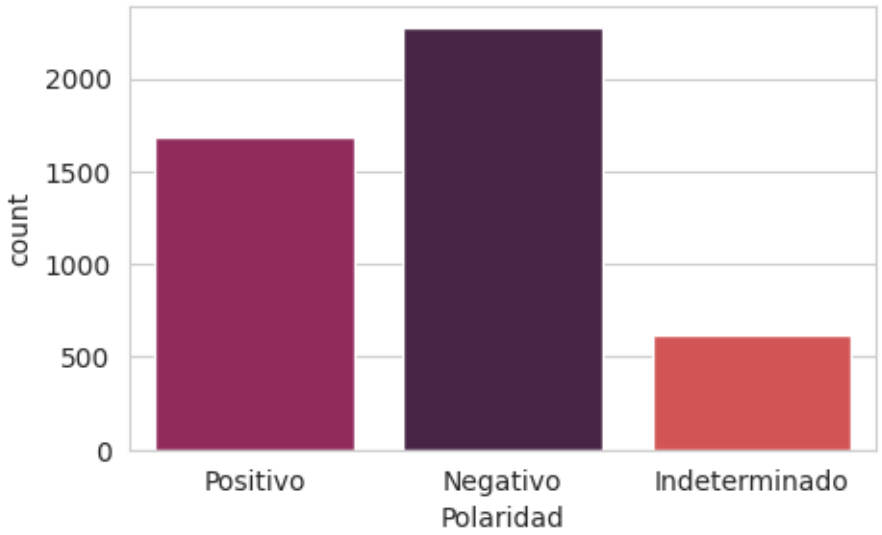


Figura 9: Distribución de polaridades en el conjunto de datos

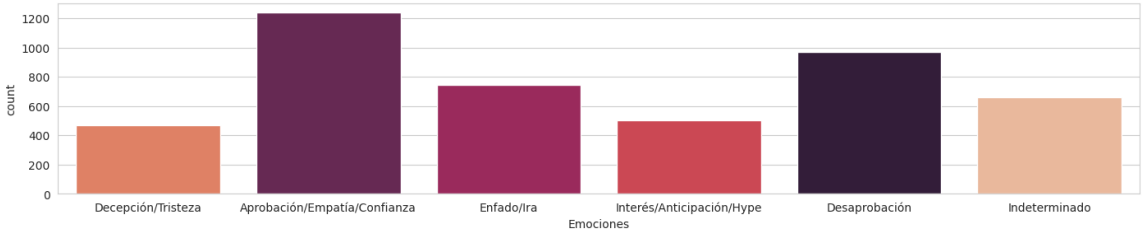


Figura 10: Distribución de emociones en el conjunto de datos

5.3 Modelo BERT utilizado: RoBERTuito

Los modelos BERT han transformado el panorama del procesamiento del lenguaje natural. Desarrollado por *Google AI Research*, este modelo de aprendizaje profundo (*Deep Learning*) emplea una arquitectura de *transformers* para aprender representaciones contextuales del texto, permitiéndole comprender mejor el contexto de las palabras dentro de una oración o párrafo. En este sentido, BERT ha demostrado ser efectivo en una amplia gama de aplicaciones de PLN, incluyendo el análisis de sentimientos (SA) [51]. BERT puede capturar la esencia emocional de los datos textuales y predecir con precisión el sentimiento y la emoción detrás de las palabras, lo cual es fundamental en áreas como el monitoreo y predicción de las respuestas emocionales en las redes sociales.

En este trabajo, hemos utilizado el modelo RoBERTuito, un modelo de lenguaje preentrenado para texto generado por usuarios en español, entrenado con más de 500 millones de *tweets*. Los experimentos realizados en un *benchmark*, conjunto de referencia o grupo de datos, que consta de tareas relacionadas con el texto generado por el usuario revelaron que este modelo exhibió un rendimiento superior en comparación con otros modelos de lenguaje preentrenados en el idioma español [51].

RoBERTuito se trata de un modelo creado en 2022 a partir de la arquitectura base del modelo RoBERTa, con un tamaño de dimensiones ocultas de 768 [52]. Su entrenamiento ha sido realizado por un grupo de trabajo bajo el nombre de pysentimiento [53].

5.4 Preprocesamiento de datos

Tras finalizar la ampliación del *dataset* y antes de entrenar un modelo, un paso crucial es el preprocesamiento de los textos, en este caso comentarios/mensajes de los espectadores en las transmisiones en directo de la plataforma Twitch. Con el objetivo de garantizar una comprensión homogénea por parte de los modelos de aprendizaje automático. Esto es debido que, a nivel de redacción de mensajes en redes sociales, suele ser más relajada y por tanto más propensa a variabilidad o a faltas de ortografía.

Para el preprocesado de los datos, se ha aprovechado el preprocesamiento integrado en el modelo RoBERTuito. En este proceso, se han tenido en cuenta los siguientes aspectos:

- Normalización de mayúsculas y minúsculas: Se convierten todas las letras a un formato consistente u homogéneo, ya sea mayúsculas o minúsculas, para evitar discrepancias.
- Supresión de acentos: Se remueven los acentos/tildes de las palabras, con el fin de reducir la variabilidad en su representación y facilitar el procesamiento del modelo.
- Supresión de signos de puntuación: Se remueven los signos de puntuación, como puntos, comas o signos de interrogación, con el fin de evitar que interfieran en la interpretación de las palabras.
- Estandarización de la risa: Se trata de convertir las expresiones de risa o emoticonos en una forma más estandarizada, de modo que la red pueda interpretarlas de manera consistente.

- Entrenamiento en español: El modelo RoBERTuito ha sido entrenado específicamente con datos en español, lo que le permite comprender y procesar mejor el lenguaje utilizado en el contexto de este estudio.
- Limitación de repeticiones de caracteres: Se limita la repetición de caracteres a un máximo de tres seguidos, como en el caso de "jajaja" que se convierte en "jaja". Esto evita que palabras alargadas artificialmente afecten el rendimiento del modelo.
- Tokenización de nombres de usuario: Los nombres de usuario se convierten en un *token*[15] especial, como "@usuario", para evitar que el modelo intente interpretarlos como palabras con significado.
- Reemplazo de hashtags por un token especial: Los *hashtags*[16] se sustituyen por un *token* especial, como "#hashtag", para preservar su función de categorización sin añadir ruido al modelo.
- Reemplazo de emojis por su representación textual: Los emojis se reemplazan por su descripción textual correspondiente (por ejemplo, "😂" se convierte en "cara riendo con lágrimas de alegría"), lo que permite al modelo interpretar su significado emocional.

Estos ajustes permiten adaptar los datos del modelo de manera más precisa a las particularidades del lenguaje utilizado en el chat de Twitch, optimizando así su capacidad para analizar y clasificar las emociones expresadas en los mensajes. En la Figura 11 se muestra la definición de la función y documentación incluida para los modelos del grupo pysentimiento para el preprocesado en RoBERTuito [55].

[15] Token: Referencia (un identificador) que regresa a los datos sensibles a través de un sistema de tokenización. Obtenido de: https://es.wikipedia.org/wiki/Token_(inform%C3%A1tica)

[16] Hashtag: Etiqueta de metadatos precedida de un carácter especial con el fin de que tanto el sistema como el usuario la identifiquen de forma rápida. Obtenido de: https://es.wikipedia.org/wiki/Hashtag

```
def preprocess_tweet(
        text, lang="es", user_token=None, url_token=None, preprocess_hashtags=True, hashtag_token=None, char_replace=True,
        demoji=True, shorten=3, normalize_laughter=True, emoji_wrapper="emoji", preprocess_handles=True):
    """
    Basic preprocessing

    Arguments:
    ---------

    text: str
        Text to preprocess

    lang: str (default 'es')
        Language used in the preprocessing. This is used for the demoji functionality and laughter preprocessing

    user_token: str (default "[USER]")
        Token used to replace user handles

    url_token: str (default "[URL]")
        Token used to replace urls

    preprocess_hashtags: boolean (default True)
        If true, applies preprocessing to hashtag, trying to split camel cases

    hashtag_token: str (default None)
        If preprocess_hashtags is True, adds hashtag_token before the preprocessed content of the hashtag

    shorten: int (default: 3)
        If not none, all occurrences of shorten or more characters are cut to this number

    char_replace: bool (default: True)
        If true, replaces or removes special characters to equivalent ones.

    demoji: boolean (default True)
        If true, converts emoji to text representations using `emoji` library, and wraps this with "emoji" tokens

    normalize_laughter: boolean (default True)
        Normalizes laughters. Uses different regular expressions depending on the lang argument.

    preprocess_handles: boolean (default True)
        If true, replaces user handles with user_token
    """

    user_token = user_token or default_args[lang]["user_token"]
    url_token = url_token or default_args[lang]["url_token"]
    hashtag_token = hashtag_token or default_args[lang]["hashtag_token"]
```

Figura 11: Documentación de la función de preprocesado utilizada en los modelos creados por el grupo de trabajo pysentimiento

Tras finalizar el proceso de preprocesamiento, se lleva a cabo la tokenización, la cual consiste en segmentar cada palabra individual de cada mensaje en unidades básicas denominadas "*tokens*".

La tokenización y los *tokens* constituyen elementos fundamentales en el procesamiento del lenguaje natural debido a que facilitan el análisis y la comprensión del significado del lenguaje por parte de los computadores. En las próximas secciones se analizará en detalle el proceso a nivel de programación.

5.5 Entrenamiento del modelo

El entrenamiento de modelos constituye la etapa fundamental en la creación o especialización de un modelo para un dominio específico. En este proceso, la clase *Trainer* desempeña un papel determinante, ya que es la encargada de recibir el modelo y los datos debidamente preparados para llevar a cabo el entrenamiento.

Cabe destacar que, en el caso de trabajar con datos etiquetados, es esencial que los campos que contienen el texto y la clase de respuesta se denominen "*text*" y "*label*" respectivamente, debido a consideraciones de diseño en la implementación del modelo.

5.5.1 Proceso de Tokenizado

Para entrenar un modelo sobre un nuevo conjunto de datos, es necesario preparar los datos para que sean compatibles con el modelo. Este proceso implica emular las funciones de la capa codificadora de la arquitectura *Transformer*s [5], que transforma el texto en una representación numérica mediante la tokenización. La clase *Tokenizer* [56] facilita esta tarea al proporcionar un vocabulario y funciones para convertir el texto en secuencias de *tokens*, añadir tokens especiales y aplicar técnicas de relleno o truncamiento. Así, se garantiza la compatibilidad de los datos con el modelo y se aprovecha el mecanismo de atención de la arquitectura. La librería *Hugging Face* proporciona un tokenizador específico para cada modelo compatible.

Para facilitar el mapeo de los datos, se divide el conjunto en entrenamiento y prueba (*test*), y se transforma en un *dataset* propio de la librería. Al aplicar la función de tokenizado proporcionada en la documentación, dependiente del *tokenizer*, se obtienen los siguientes elementos:

- *Text*: El texto original de los datos.
- *Label*: La etiqueta correspondiente al texto.
- *Input ids*: Representación numérica del texto, donde cada palabra se identifica con un número único del vocabulario del modelo.
- *Token type ids*: Utilizado en tareas de respuesta a preguntas o modelado de lenguaje, añade símbolos especiales para adaptar la entrada a la tarea. En este caso, no se utiliza y se representa como una lista de ceros [57].

- *Attention masks*: Indica al modelo qué tokens deben ser considerados en el mecanismo de atención al modelo BERT [5].

Este proceso de tokenización y preparación de datos es esencial para que el modelo pueda procesar y comprender la información contenida en el conjunto de datos, y así llevar a cabo el entrenamiento de manera efectiva.

5.5.2 Proceso de Entrenamiento

Una vez que los datos han sido adecuadamente procesados, es el momento de emplear la clase *Trainer* para iniciar el entrenamiento del modelo. Previamente, se establecen los parámetros de entrenamiento mediante la clase *TrainingArguments*, junto con otros parámetros relevantes. En este trabajo, además de los parámetros detallados en el apartado siguiente, se definen los siguientes:

- *output_dir*: Directorio donde se almacenarán los archivos relacionados con el entrenamiento. Es obligatorio especificarlo.

- *evaluation_strategy*: Indica en qué momento se evaluará el modelo con el conjunto de prueba. Puede realizarse al finalizar cada época (*epoch*) o cada lote, y se puede especificar el número de épocas o lotes antes de la evaluación. En este trabajo, se evalúa después de completar una sola época, ya que todos los datos de entrenamiento habrán sido procesados por el modelo.

- *fp16*: Habilita el uso de precisión de punto flotante de 16 bits para optimizar el uso de memoria y acelerar el entrenamiento. Es necesario para utilizar tamaños de *batch* de hasta 128 y mejorar la velocidad de entrenamiento. Aunque existen otras opciones, como *tf32*, las limitaciones de arquitectura de las GPUs utilizadas impidieron su prueba en este caso.

- *load_best_model_at_end y metric_for_best_model*: Se utilizan junto con un *callback* en la llamada a *Trainer* para implementar una parada temprana del entrenamiento. La métrica utilizada para determinar el mejor modelo es la precisión (*accuracy*).

La configuración de estos parámetros permite ajustar el entrenamiento del modelo a las necesidades específicas del problema y optimizar su rendimiento.

La declaración de un objeto *Trainer* implica especificar el modelo a entrenar, los parámetros de entrenamiento, los conjuntos de datos de entrenamiento y prueba (*test*), las métricas de evaluación, el *data collator* y *callbacks* opcionales. Las métricas de evaluación se declaran en una función que devuelve un diccionario, mientras que el *data collator*, encargado de crear los lotes (*batch*) de datos para el entrenamiento, se declara indicando el *tokenizer*.

En este trabajo, se utiliza un *callback earlyStop* para detener el entrenamiento si la última evaluación muestra un rendimiento inferior al anterior en la métrica de precisión (*accuracy*). Una vez configurado todo, se ejecuta la función *train* de la clase *Trainer* para iniciar el entrenamiento, como se muestra en la Figura 12.

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	No log	0.571952	0.837413	0.848639	0.837413	0.832794
2	No log	0.651366	0.844406	0.849343	0.844406	0.839322
3	0.535800	0.868442	0.832168	0.840024	0.832168	0.831833

Figura 12: Ejemplo de entrenamiento

Para obtener un desglose detallado de las métricas por clase, se ejecuta la función *evaluate* sobre el conjunto de prueba después del entrenamiento, lo que permite obtener las predicciones del modelo. A continuación, se genera un informe de clasificación con las salidas esperadas, como se ilustra en la Figura 13.

Al comparar ambas figuras, se observa que los resultados del desglose de métricas por clases coinciden con la segunda evaluación del modelo, debido a la parada temprana (*earlyStop*) y a los dos últimos parámetros utilizados en la clase *TrainingArguments*.

	precision	recall	f1-score	support
0	0.9030	0.9313	0.9169	160
1	0.7727	0.8947	0.8293	57
2	0.6757	0.8333	0.7463	30
3	0.8547	0.8547	0.8547	117
4	0.8383	0.8485	0.8434	165
5	0.9000	0.4186	0.5714	43
accuracy			0.8444	572
macro avg	0.8241	0.7969	0.7937	572
weighted avg	0.8493	0.8444	0.8393	572

Figura 13: Desglose de métricas del entrenamiento por clases

En todos los entrenamientos, se ha dividido el conjunto de datos utilizando la función *train_test_split* con estratificación en la clase de respuesta, garantizando así la

representación de todos los posibles valores de salida en ambos subconjuntos. Se han utilizado las proporciones sugeridas por la función, asignando un 66% para el conjunto de entrenamiento y un 33% el de la prueba.

5.6 Elección de hiperparámetros del modelo

En el ámbito del aprendizaje automático, la elección adecuada de los hiperparámetros resulta crucial para optimizar el rendimiento de un modelo. Entre los diversos métodos disponibles, la búsqueda de hiperparámetros utilizando la librería Hugging Face [29] y el *backend*[17] dedicado Wandb [31], se ha convertido en una herramienta popular por su facilidad de uso, eficiencia y capacidad de análisis.

La plataforma Wandb facilita este proceso a través de una interfaz intuitiva y herramientas de análisis avanzadas, permitiendo obtener configuraciones de hiperparámetros que maximizan el rendimiento del modelo. Por estos motivos, utilizaremos esta herramienta para este proceso.

Los parámetros de entrenamiento sobre los que realizaremos la búsqueda son:

- Tasa de aprendizaje (*Learning rate*): un multiplicador empleado en el algoritmo de propagación hacia atrás (retropropagación) que agiliza y acelera la convergencia del modelo. Sus valores variarían entre 10^{-6} y 10^{-4} siguiendo una distribución uniforme.
- Tamaño de lote (*Batch size*): indica el número de muestras propagadas a través del modelo antes de reajustar sus pesos. Las búsquedas sobre este hiperparámetro del entrenamiento y evaluación se realizan por separado. Sus valores se suelen tratar de potencias de 2. Y en nuestro caso de estudio variarán entre 2 y 128, siendo este el máximo por limitaciones de la memoria VRAM de las GPUs disponibles.
- Periodo/Épocas (*Epoch*): este parámetro marca cuando se han introducido todas las instancias del conjunto de entrenamiento al modelo. Sus valores variarán entre 5 y 15 siguiendo una distribución uniforme de números enteros.

[17] Backend: Parte no visible de una aplicación o sitio web que se ejecuta en el servidor y maneja la lógica, el procesamiento de datos y la comunicación con la base de datos. Obtenido de: https://www.arimetrics.com/glosario-digital/backend

En la más adelante en las Figuras 14 y 15, se muestran los fragmentos del código en Python para la realización de búsqueda de hiperparámetros con la plataforma Wandb.

```
Tunning

[ ] import wandb

    wandb.login()

[ ] id2label = {0: "NEG", 1: "NEU", 2: "POS"}
    label2id = {"NEG": 0, "NEU": 1, "POS": 2}

    model = AutoModelForSequenceClassification.from_pretrained('pysentimiento/robertuito-sentiment-analysis', num_labels=3, id2label=id2label, label2id=label2id, ignore_mismatched_sizes=True)
    tokenizer = AutoTokenizer.from_pretrained('pysentimiento/robertuito-sentiment-analysis')
    data_collator = DataCollatorWithPadding(tokenizer)


    dataset = load_corpus_Twitch(drop=0, n_labels=3, task="sentiment")
    train, test = train_test_split(dataset, stratify=dataset["label"])
    train.to_csv("corpus_train.csv", index=False)
    test.to_csv("corpus_test.csv", index=False)
    train_test = load_dataset("csv", data_files={"train": "./corpus_train.csv", "test": "./corpus_test.csv"})
    tokenized_datasets = train_test.map(tokenize_function, batched=True)
```

Figura 14: Fragmentos de código para la conexión con Wandb, el preprocesamiento, carga de datos, preentrenamiento y búsqueda de hiperparámetros en Python.

```
def wandb_hp_space(trial):
    return {
        "method": "random",
        "metric": {"name": "accuracy", "goal": "maximize"},
        "parameters": {
            "learning_rate": {"distribution": "uniform", "min": 1e-6, "max": 1e-4},
            "per_device_train_batch_size": {"values": [16, 32, 64, 128]},
            "per_device_eval_batch_size": {"values": [16, 32, 64, 128]},
            "num_train_epochs" : {"distribution": "int_uniform", "min": 5, "max": 15}
        },
    }

def model_init(trial):
    return AutoModelForSequenceClassification.from_pretrained(
        'pysentimiento/robertuito-sentiment-analysis',
        config=config
    )


training_args = TrainingArguments(report_to="wandb", output_dir="wandb-3labels-Twitch", overwrite_output_dir=True,
      evaluation_strategy="epoch", per_device_train_batch_size = 128, per_device_eval_batch_size=128, num_train_epochs = 10,
                                fp16=True, save_strategy="epoch", load_best_model_at_end = True, metric_for_best_model="accuracy") #necesario para earlyStop

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["test"],
    compute_metrics=compute_metrics,
    tokenizer=tokenizer,
    model_init=model_init,
    data_collator=data_collator,
    callbacks=[earlyStop]
)

best_trial = trainer.hyperparameter_search(
    direction="maximize",
    backend="wandb",
    hp_space=wandb_hp_space,
    n_trials=35 #originalmente 30
)


wandb.finish()
```

Figura 15: Fragmentos de código para la conexión con Wandb, selección de parámetros/valores y dirección de optimización junto con el número de entrenamientos.

La Figura 15 presenta los fragmentos de código más relevantes para la elección de hiperparámetros. A continuación, se detalla su funcionamiento: en primer lugar, es necesario iniciar sesión en la plataforma Wandb y utilizar la clave de API única proporcionada a cada usuario. Una vez establecida la conexión con el *backend* desde Python, se define el espacio de hiperparámetros, especificando los parámetros a optimizar y sus posibles valores. Posteriormente, se escribe el código correspondiente a un entrenamiento genérico del modelo que se desea optimizar. Finalmente, se indica el

número de entrenamientos a realizar y la dirección en la que se busca optimizar la métrica de rendimiento seleccionada.

En este estudio, se optó por realizar dos ejecuciones con 25 y 35 entrenamientos, respectivamente, para evaluar si un mayor número de iteraciones influye en la optimización de la precisión (*accuracy*) del modelo. El análisis comparativo detallado de estos resultados se presentará más adelante.

Si bien los resultados de cada entrenamiento pueden visualizarse en el cuaderno, dada la cantidad de iteraciones, resulta más práctico acceder a la plataforma web de Wandb. Allí encontraremos los proyectos con sus configuraciones, actualizadas automáticamente al finalizar cada entrenamiento con las métricas obtenidas. La configuración para almacenar los resultados en diferentes proyectos se establece en el código fuente. En caso contrario, todos los resultados se almacenarán en un proyecto llamado *uncategorized*. A continuación, se ilustrará mediante imágenes cómo navegar por la web de Wandb para observar y visualizar las ejecuciones de las búsquedas de hiperparámetros:

1. Para comenzar, accedemos a la plataforma https://wandb.ai e iniciamos sesión utilizando una cuenta personal o académica, la cual nos proporcionará una clave API personal que nos permitirá establecer la conexión con la plataforma a través del cuaderno de Python.
2. Una vez dentro de la plataforma, en la pestaña *Home*, encontraremos a la izquierda los diferentes proyectos donde se almacenan las simulaciones y ejecuciones realizadas; como muestra la Figura 16. En el proyecto denominado "*huggingface"* se guardarán los resultados de los entrenamientos del modelo, los cuales podrán ser visualizados de manera más detallada desde el cuaderno de Python que desde la propia plataforma Wandb. Por otro lado, en el proyecto denominado "hiperparámetros" encontraremos los resultados de las elecciones de hiperparámetros, ya que hemos configurado previamente el sistema para que se almacenen en este lugar específico.

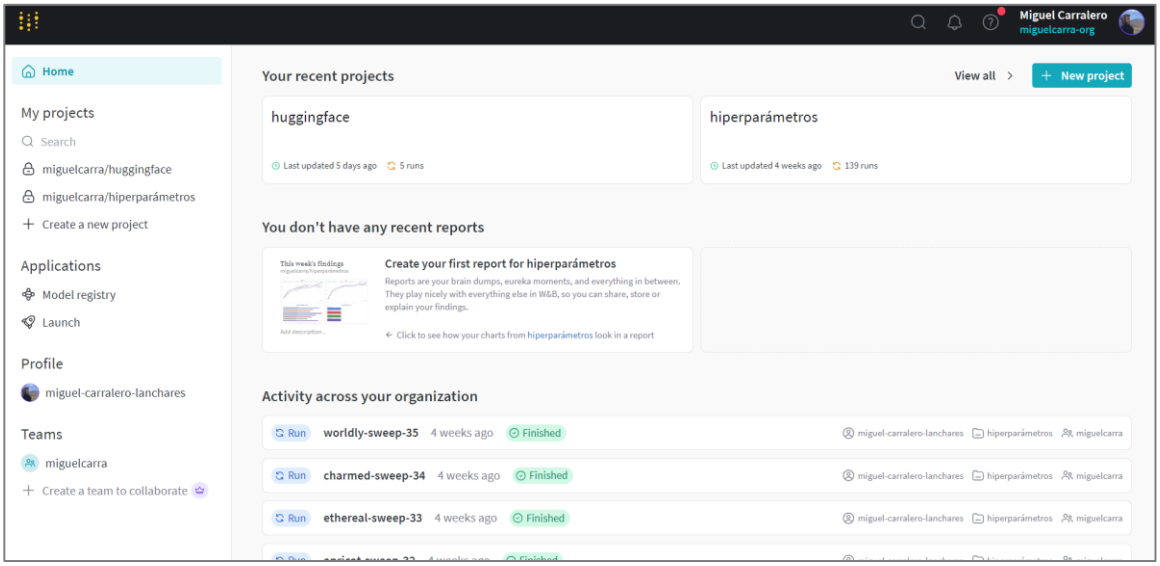


Figura 16: Repositorio *Home* de Wandb

3. A continuación, accedemos al espacio de trabajo del proyecto "hiperparámetros" y nos dirigimos a la pestaña *Sweeps* para localizar nuestra simulación. En la Figura 17 se muestra la vista del repositorio.

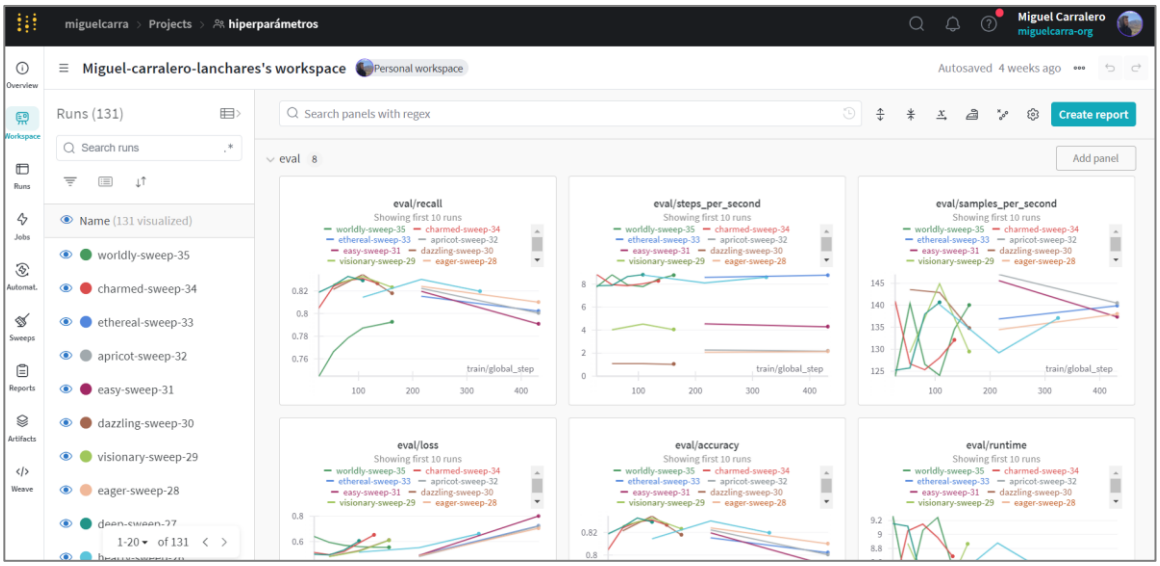


Figura 17: Proyecto “hiperparámetros” en Wandb

4. En la pestaña *Sweep*, como se visualiza en Figura 18, se mostrarán todas las simulaciones realizadas durante nuestro estudio. En este caso, nos centraremos en la simulación denominada "polaridad35", correspondiente a la ejecución de búsqueda de hiperparámetros para análisis de polaridad con 35 entrenamientos. Seleccionamos esta simulación para acceder a su espacio de trabajo específico.

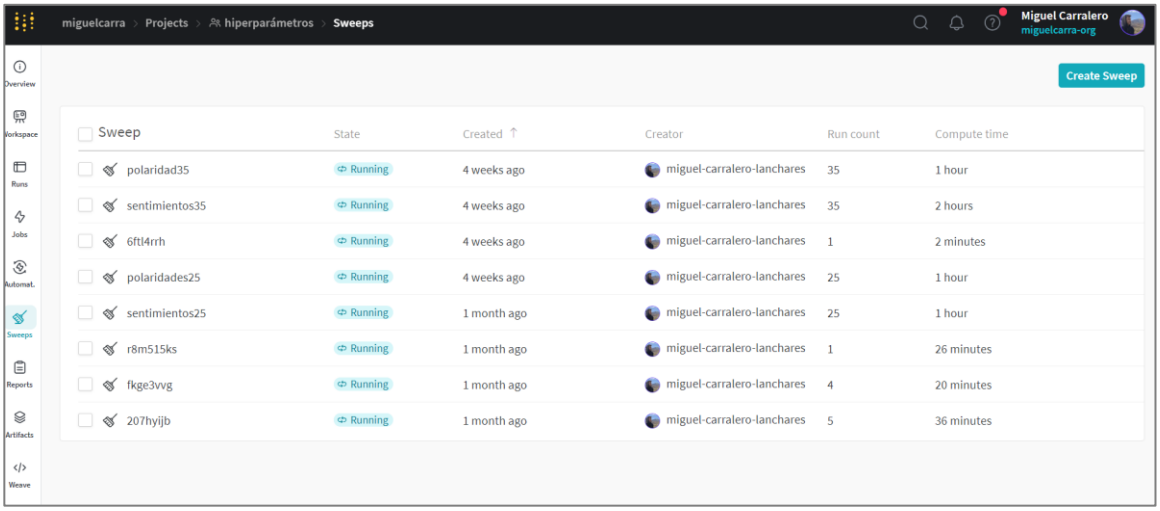


Figura 18: Pestaña *Sweeps* del proyecto "hiperparámetros" en Wandb

5. Una vez dentro del espacio de trabajo de la simulación "polaridad35", podremos visualizar un gráfico que nos permitirá obtener los resultados de la búsqueda de hiperparámetros, como muestra la Figura 19. Para ello, simplemente debemos modificar el último valor del gráfico, cambiando "*eval/loss*" por "*eval/accuracy*". Este cambio lo podemos visualizar en la Figura 20. De esta manera, obtendremos una representación visual de los resultados en términos de precisión en lugar de pérdida.

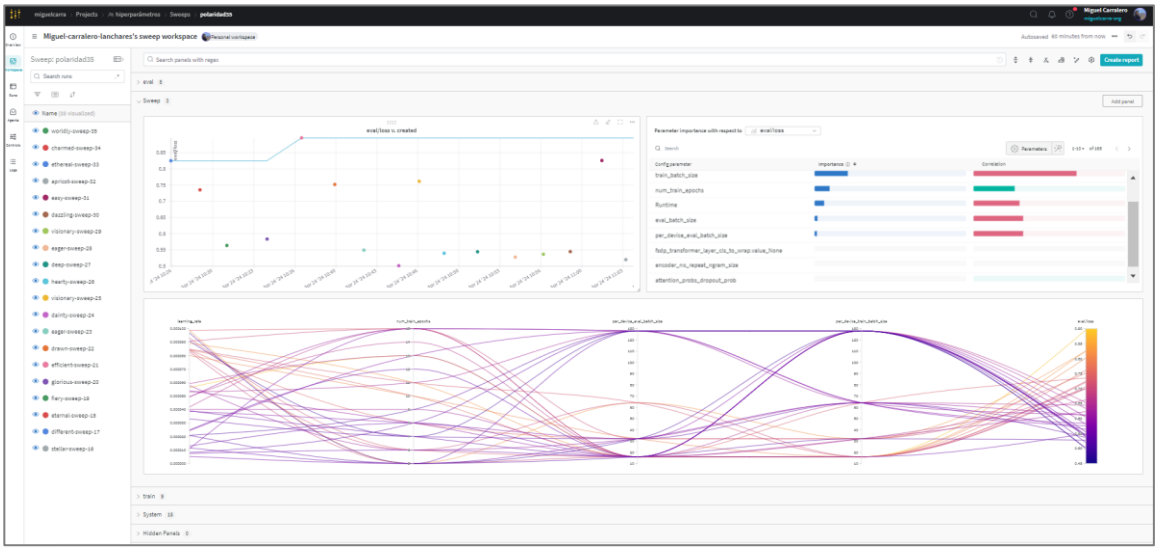


Figura 19: Búsqueda de hiperparámetros por polaridad para 35 ejecuciones en Wandb

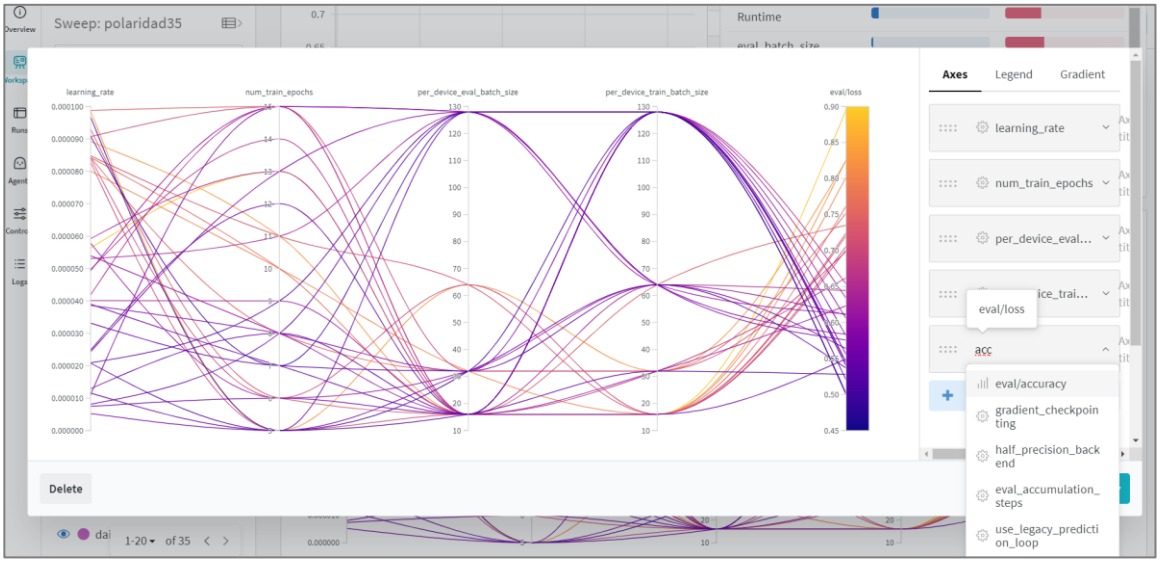


Figura 20: Modificación del gráfico para la visualización de los resultados de la búsqueda de hiperparámetros en Wandb

También desde el repositorio de "polaridad35" podremos acceder a una pestaña a la izquierda llamada "*Runs*" y visualizar las 35 ejecuciones de la búsqueda de hiperparámetros. Disponible para visualizar en la Figura 21.

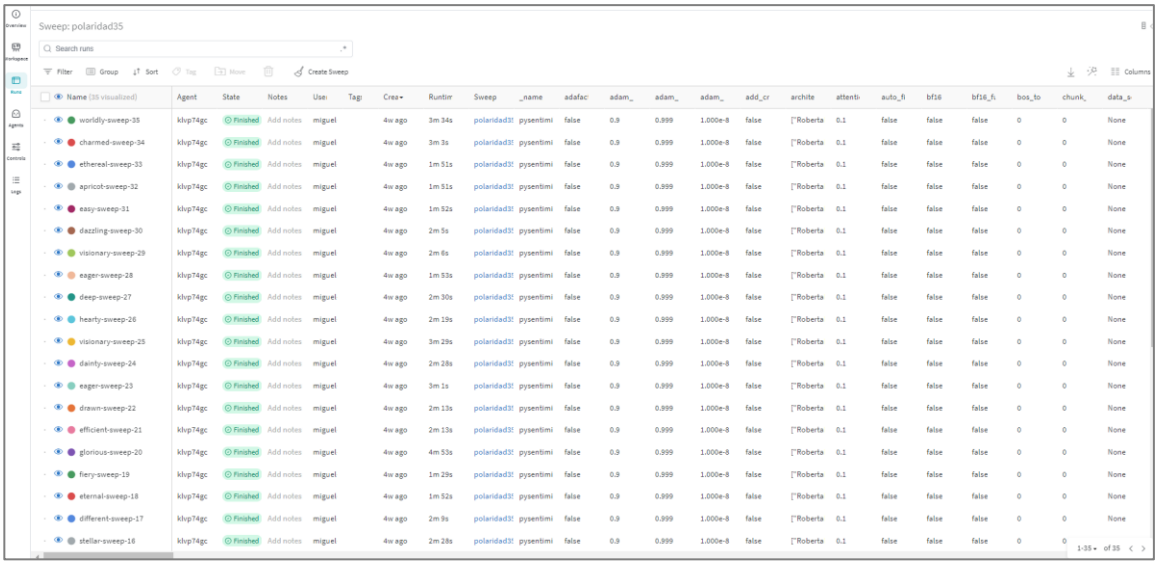


Figura 21: Pestaña *Runs* de la simulación de 35 ejecuciones en Wandb

5.6.1 Comparativa de resultados

Las Figuras 25, 26, 27 y 28 presentadas ilustran los resultados de las búsquedas de hiperparámetros realizadas para los modelos de polaridad (con 25 y 35 ejecuciones) y emociones (con 25 y 35 ejecuciones). Cada línea en los gráficos representa una configuración de parámetros utilizada en el entrenamiento, mostrando la variación de los valores a lo largo del proceso. El eje de abscisas superior indica los parámetros y su valor

en cada entrenamiento, mientras que el último valor representa la métrica de rendimiento del modelo, visualizada mediante un gradiente de colores. Las líneas de color anaranjado señalan configuraciones con una precisión (*accuracy*) cercana a 81%-83.5% (0.81-0.835) en el caso de polaridad y de 69%-70% (0.69-0.7) en el caso de emociones, sugiriendo que podrían ser óptimas.

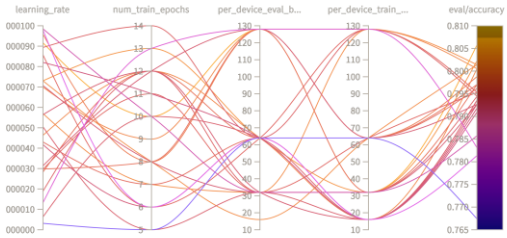


Figura 22: Búsqueda de hiperparámetros para Polaridad (25 ejecuciones) con Wandb

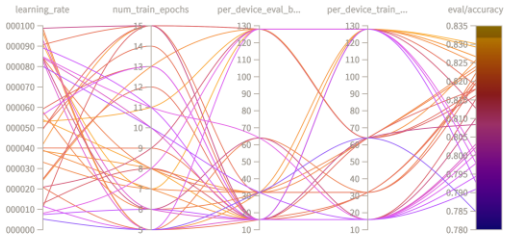


Figura 23: Búsqueda de hiperparámetros para Polaridad (35 ejecuciones) con Wandb

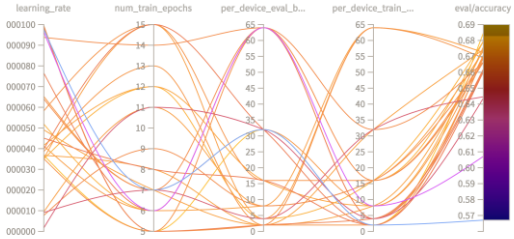


Figura 24: Búsqueda de hiperparámetros para Emociones (25 ejecuciones) con Wandb

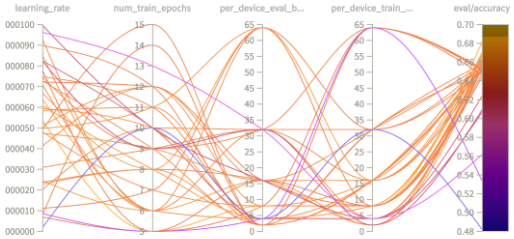


Figura 25: Búsqueda de hiperparámetros para Emociones (35 ejecuciones) con Wandb

A partir de estos resultados, seleccionaremos los hiperparámetros que mejor se ajusten a nuestras necesidades, pero es importante destacar que, según lo observado, no hay una diferencia significativa entre los resultados obtenidos con 25 y 35 ejecuciones. En cuanto a los valores óptimos, ambas gráficas alcanzan valores máximos de *accuracy* similares, lo que indica que el rendimiento máximo del modelo no se ve afectado significativamente por la diferencia en el número de ejecuciones. En cuanto a las ejecuciones intermedias en ambas gráficas muestran una distribución similar de los valores de precisión, agrupándose en torno a valores similares con una dispersión baja. Esto sugiere que el proceso de búsqueda de hiperparámetros ha explorado regiones similares del espacio de hiperparámetros en ambos casos, convergiendo hacia configuraciones con un rendimiento similar, independientemente del número de ejecuciones. Sin embargo, aunque las

diferencias en *accuracy* son mínimas, se ha optado por seleccionar la configuración con 35 ejecuciones para el modelo de polaridad y la configuración con 25 ejecuciones para el modelo de emociones, ya que, en ambos casos, estas configuraciones presentaron un ligero margen de mejora en la métrica de evaluación.

Como anotaciones finales, cabe mencionar que el tiempo de ejecución (*Run Time*) de las búsquedas varió en función del número de ejecuciones. Las elecciones con 25 simulaciones teniendo una duración media de 45 minutos, mientras que aquellas con 35 se extendieron a una hora.

En la Tabla 10 se presentan los valores de los hiperparámetros que serán utilizados en los entrenamientos correspondientes.

Hiperparámetros	Polaridad	Emociones
Learning rate	0.0000909	0.00003685
Eval Batch size	128	32
Train Batch size	128	2
Epoch	7	5
Accuracy	0,8301	0,6847
Acc. (%)	83,01%	68,47%

Tabla 10: Valores de los hiperparámetros para los entrenamientos

La plataforma Wandb facilita la selección individual de cada entrenamiento para examinar los valores de sus parámetros, así como la aplicación de filtros para excluir algunos. Además, ofrece métricas sobre la importancia y correlación de cada parámetro con la métrica a optimizar, lo que simplifica la identificación de la combinación de hiperparámetros que produce los mejores resultados. Estos valores óptimos se utilizarán para entrenar el modelo final, con el cual se obtendrán los resultados definitivos.

5.6.2 Entornos de computación en Google Colab

En el desarrollo de este trabajo, se optó por el entorno de computación en la nube de Google Colab [30] para llevar a cabo los experimentos y análisis necesarios. Esta elección fue motivada por su eficacia en trabajos previos y la ventaja de ofrecer recursos computacionales de forma gratuita. Colab proporciona acceso a una GPU NVIDIA Tesla T4, 12.69 GB de memoria RAM y 107.72 GB de almacenamiento, lo cual resulta suficiente para muchas tareas de aprendizaje automático.

Sin embargo, durante la fase de búsqueda de hiperparámetros utilizando la plataforma Wandb, se encontraron ciertas limitaciones en la versión gratuita. En particular, la capacidad de disco restringida limitó el alcance de las simulaciones a menos de 20 ejecuciones, lo que dificultó la exploración exhaustiva del espacio de hiperparámetros y la identificación de la configuración óptima. Además, la disponibilidad intermitente de GPUs para usuarios sin plan de pago supuso un obstáculo adicional en el flujo de trabajo.

Para solventar estas limitaciones y asegurar una búsqueda de parámetros más completa y eficiente, se decidió invertir en una suscripción de un mes al plan Colab Pro [58]. Este plan ofrece una serie de ventajas significativas, como hasta un total de 100 unidades informáticas, mayores tiempos de ejecución, mayor capacidad de RAM y almacenamiento, y acceso prioritario a GPUs y CPUs más potentes. En concreto, se utilizaron entornos virtuales con 200 GB de disco, lo que permitió realizar búsquedas de hasta 35 ejecuciones, cada una con una duración aproximada de una hora y un consumo de 1.84 unidades informáticas. Esta inversión no solo agilizó el proceso de búsqueda de parámetros, sino que también resultó en un ahorro considerable de unidades informáticas, permitiendo que la suscripción fuera aprovechada para otros proyectos del departamento.

La elección de Colab Pro demostró ser una decisión acertada, ya que facilitó la realización de experimentos más ambiciosos y la obtención de resultados más sólidos en un plazo de tiempo más corto. Aunque implicó un coste adicional, la inversión se vio compensada por la mejora en la eficiencia y la calidad del trabajo realizado.

5.7 Análisis de resultados para la polaridad con RoBERTuito

Con el objetivo de obtener resultados robustos y confiables, se llevó a cabo un proceso de validación cruzada mediante *10-Fold* con estratificación en las clases de respuesta. Este enfoque permitió entrenar el modelo en diferentes divisiones del conjunto de datos y calcular la media aritmética de las métricas de rendimiento obtenidas para cada clase de respuesta, así como las medias aritméticas (macro) y ponderada por el tamaño de las clases (*weighted*), y la precisión (*accuracy*) global del modelo.

Se llevaron a cabo las pruebas con las siguientes configuraciones de parámetros, determinadas a través de una búsqueda de hiperparámetros utilizando la plataforma Wandb [31] con 35 ejecuciones:

- RoBERTuito: learning_rate = 9,09 × 10^{-5}, train_batch_size = 128, eval_batch size = 128, num_train_epochs= 7

5.7.1 Resultados

La Tabla 11 muestra los resultados obtenidos en la clasificación de polaridad (Positivo, Negativo e Indeterminado) utilizando el modelo descrito, tanto en el estudio actual como en el trabajo publicado por Merayo et al. [3], que representa una optimización del estudio previo realizado por Javier Estévez [1]. Los resultados se presentan en términos de precisión (Prec), *recall* (Recall) y *F1-score* (F1) para cada clase, siendo estas las métricas presentadas en la Sección 3.8, así como para el promedio macro (*macro-avg*) y ponderado (*weighted-avg*). Además, se muestra la precisión global del modelo (Acc). Los valores de las métricas en el trabajo actual se presentan en formato de porcentaje, lo que facilita la interpretación de los resultados. El número de instancias de cada clase (Supp.) también se incluye para proporcionar contexto sobre la distribución de los datos. En resumen, la tabla proporciona una visión detallada del rendimiento del modelo RoBERTuito en la tarea de clasificación de polaridad, permitiendo evaluar su capacidad para distinguir entre las diferentes clases y su precisión general.

Polaridad	**Métrica**	**RoBERTuito (%) (estudio anterior)**	**RoBERTuito (%) (estudio actual)**	**Supp. (estudio actual)**
Negativo	Prec. Recall F1	75% 85% 80%	83,3% 87,5% 85,3%	227,9
Positivo	Prec. Recall F1	82,3% 85,6% 84%	80,4% 81,7% 80,9%	62,3
Indeterminado	Prec. Recall F1	52,5% 18,4% 27,1%	68,7% 52,2% 59,1%	62,3
macro-avg	Prec. Recall F1	- - -	77,4% 73,8% 75,1%	458,9
weighted-avg	Prec. Recall F1	- - -	80% 81% 80%	458,9
Global	Acc.	78%	81%	

Tabla 11: Resultados obtenidos en clasificación de Polaridad

Adicionalmente, para facilitar la comparación de la métrica de precisión (Prec.) y la precisión global de los modelos (Acc.) entre el estudio previo y el actual, y así observar los cambios derivados de las modificaciones realizadas en este trabajo, se presenta la Tabla 12. Esta permite evaluar la capacidad de precisión del modelo en la clasificación de mensajes por polaridad, contrastando los resultados obtenidos en ambos estudios.

Polaridad	**Prec. (%) (estudio anterior)**	**Prec. (%) (estudio actual)**
Negativo	75%	83,3%
Positivo	82,3%	80,4%
Indeterminado	52,5%	68,7%
Acc.	78%	81%

Tabla 12: Comparativa de la métrica de Precisión (Prec.) por Polaridad en el estudio anterior [3] y actual

El análisis de los resultados obtenidos revela que el modelo RoBERTuito presenta un desempeño global satisfactorio en la tarea de clasificación de polaridad, alcanzando un 81% de precisión (A*ccuracy*). Este valor representa una ligera mejora en comparación con los trabajos anteriores de referencia, el trabajo fin de grado de Javier [1] y el artículo publicado posteriormente [3], en el que se realizó una optimización a partir de los resultados del TFG, obteniendo una precisión de un 78%. No obstante, un examen más detallado por clase evidencia variaciones significativas en su rendimiento.

En la clase "Negativo", el modelo demuestra una notable capacidad para identificar textos con esta polaridad, logrando una precisión (Prec) del 83.3%, un *recall* del 87.5% y un *F1-score* de 85.3%. Estos resultados, que superan a los del modelo anterior (Prec: 75%, Recall: 85%, F1: 80%), indican una alta competencia en la detección de expresiones negativas y sugieren que el balanceo de clases en el corpus, enfocado en aumentar la representación de ejemplos negativos, ha tenido un impacto positivo.

En contraste, la clase "Positivo" muestra un desempeño ligeramente inferior, aunque aún satisfactorio, con una precisión del 80.4%, un *recall* del 81.7% y un *F1-score* de 80.9%. A pesar de mantener un buen rendimiento, estos valores son ligeramente inferiores a los del modelo anterior (Prec: 82,3%, Recall: 85,6%, F1: 84%), lo que podría atribuirse al enfoque en mejorar la clase "Negativo" durante el balanceo del corpus.

La clase "Indeterminado" emerge como el principal desafío para el modelo, presentando una precisión del 68.7%, un *recall* del 52.2% y un *F1-score* de 59.1%. Aunque estos

valores superan bastante a los del modelo anterior (Prec: 52,5%, Recall: 18,4%, F1: 27%), sugieren que la identificación de textos que no expresan una polaridad clara sigue siendo una tarea compleja, posiblemente debido a la ambigüedad inherente a este tipo de expresiones y a la menor cantidad de ejemplos en el corpus balanceado.

En síntesis, el modelo se posiciona como un medio eficaz en la clasificación de polaridad, particularmente en la identificación de textos negativos, superando el rendimiento de trabajos previos [1, 3]. Esta mejora se atribuye a la ampliación del corpus, especialmente al balanceo de mensajes de polaridad negativa, lo que ha permitido mejorar los resultados de esta clase y mantener altos los valores de precisión en la clase "Positivo". Además, el aumento en el número de mensajes de la clase "Indeterminado" en el corpus también ha contribuido a mejorar sus métricas, aunque de forma indirecta.

Sin embargo, el análisis revela la necesidad de optimizar su rendimiento en esta última clase. Para abordar este desafío, se podrían explorar diversas estrategias, como la ampliación del conjunto de datos de entrenamiento con ejemplos de esta clase, el ajuste de los hiperparámetros, la implementación de modificaciones a nivel del modelo que permitan un tratamiento más efectivo de los datos o la aplicación de técnicas de aumento de datos.

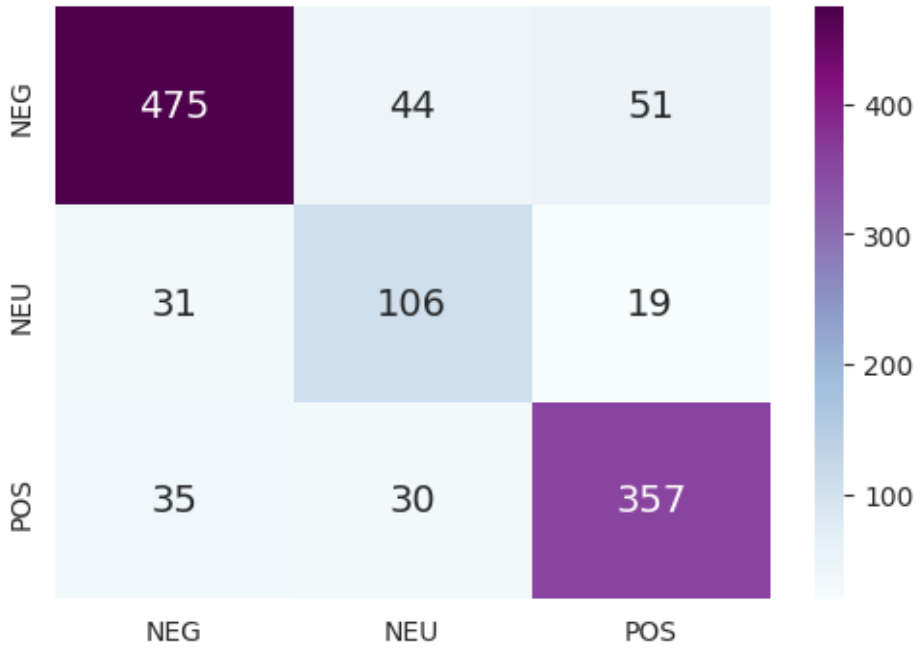


Figura 26: Matriz de confusión para una clasificación de Polaridad

El análisis de la matriz de confusión (Figura 26) respalda las observaciones previas sobre el rendimiento del modelo, es decir, las Tablas 11 y 12. La matriz revela una alta precisión

en la clasificación de textos Negativos (NEG), evidenciado por los 475 casos correctamente identificados. Sin embargo, se observa una cierta confusión con la clase Indeterminado/Neutra (NEU), con 44 casos erróneamente clasificados como negativos.

En la clase Indeterminado/Neutra (NEU), el modelo muestra un desempeño moderado, con 106 aciertos y una distribución relativamente equilibrada de errores entre las clases negativas y positivas. Esto sugiere que el modelo tiene dificultades para distinguir entre textos neutral y aquellos con polaridad, lo que coincide con el menor rendimiento observado en las métricas de la clase. Esta dificultad en la clasificación de textos indeterminados puede deberse a la naturaleza ambigua de esta clase. En muchos casos, es difícil determinar si un texto es verdaderamente neutral, ya que a menudo una opinión puede estar implícita o varias connotaciones pueden mezclarse al mismo tiempo. Además, la definición misma de "neutralidad" puede ser subjetiva y variar según el contexto.

Finalmente, la clase Positiva (POS) muestra un buen rendimiento, con 357 aciertos y una baja tasa de falsos positivos. Sin embargo, se evidencia cierta confusión con la clase negativa, con 35 casos erróneamente clasificados.

En resumen, la matriz de confusión confirma la habilidad del modelo para identificar textos negativos y positivos, pero revela dificultades en la distinción de textos neutrales, lo que respalda la necesidad de explorar estrategias de mejora enfocadas en esta clase, como se mencionó anteriormente.

5.8 Análisis de resultados para emociones con RoBERTuito

Se llevaron a cabo las pruebas utilizando las siguientes configuraciones de parámetros óptimos, determinadas mediante una búsqueda exhaustiva de hiperparámetros empleando la plataforma Wandb [31] con 25 ejecuciones:

- RoBERTuito: learning_rate = 3,685 × 10^{-5}, train_batch_size = 2, eval_batch size = 32, num_train_epochs= 5

5.8.1 Resultados

La Tabla 13 presenta los resultados en la clasificación de emociones. En este caso, las métricas se muestran para cada una de las clases de emociones identificadas, descritas en la Sección 3.8, así como para los promedios macro y ponderado. Además, se incluye la precisión global del modelo (Acc) y el número de instancias de cada clase (Supp.).

Emoción	Métrica	RoBERTuito (estudio anterior)	RoBERTuito (%) (estudio actual)	Supp. (estudio actual)
Aprobación/Empatía /Confianza	Prec. Recall F1	76% 76,5% 75,7%	71% 77,1% 74%	123,9
Desaprobación	Prec. Recall F1	59,7% 65,3% 62%	57,1% 65,3% 60,6%	96,9
Decepción/Tristeza	Prec. Recall F1	85% 93% 89%	80% 73% 76,1%	46,9
Interés/Anticipación /Hype	Prec. Recall F1	74,7% 67,5% 70%	71,7% 63,7% 67,3%	50,4
Enfado/Ira	Prec. Recall F1	59% 56,5% 56%	76,4% 71,3% 73,6%	74,6
Indeterminado	Prec. Recall F1	52,3% 39,7% 44%	64,1% 51,2% 56,6%	66,2
macro-avg	Prec. Recall F1	- - -	70% 67% 68%	458,9
weighted-avg	Prec. Recall F1	- - -	68,9% 68,1% 68%	458,9
Global	Acc.	68%	68.1%	

Tabla 13: Resultados obtenidos en clasificación de Emociones

Para facilitar la comparación de la métrica de precisión (Prec.) entre el estudio previo y el actual, y así observar los cambios derivados de las modificaciones realizadas en este trabajo, se presenta la Tabla 14. Esta permite evaluar la capacidad de precisión del modelo en la clasificación de mensajes por emociones, contrastando los resultados obtenidos en ambos estudios.

Utilizando como referencia la Figura 27 correspondiente a la matriz de confusión para una clasificación de emociones y las tablas con los resultados de las métricas, podremos analizar y estudiar las capacidades de del modelo entrenado.

Polaridad	Prec. (%) (estudio anterior)	Prec. (%) (estudio actual)
Aprobación/Empatía/Confianza	76%	71%
Desaprobación	59,7%	57,1%
Decepción/Tristeza	85%	80%
Interés/Anticipación/Hype	74,7%	71,7%
Enfado/Ira	59%	76,4%
Indeterminado	52,3%	64,1%
Acc.	68%	68,1%

Tabla 14: Comparativa de la métrica de Precisión (Prec.) por Emociones en el estudio anterior [3] y actual

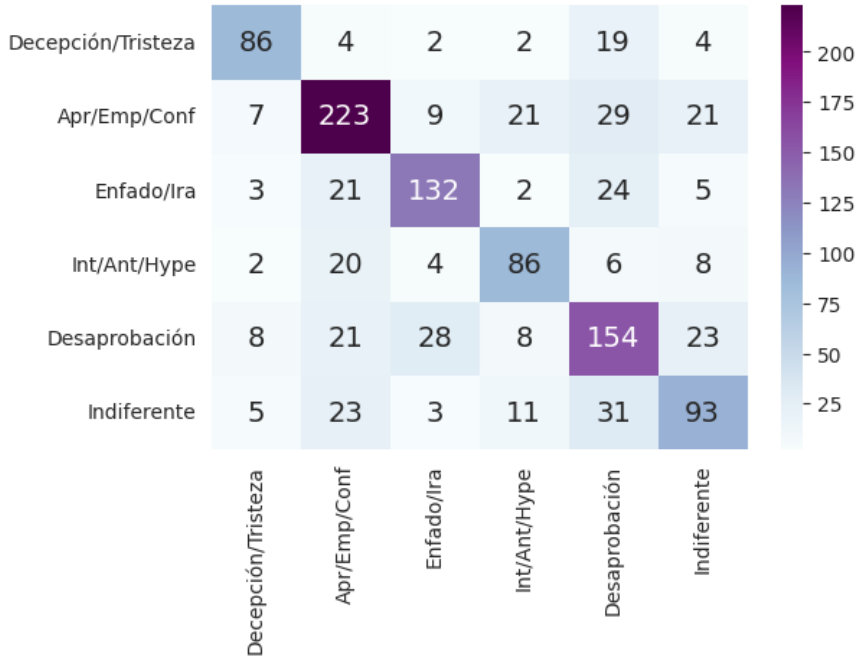


Figura 27: Matriz de confusión para una clasificación de Emociones

Este, tras el entrenamiento, exhibe un desempeño destacable en la identificación precisa de ciertas emociones, mientras que enfrenta desafíos con otras. La emoción con mayor precisión es "Decepción/Tristeza" con un valor de 80%, lo cual es notable considerando que tiene un número relativamente bajo de instancias (469, como se detalla en la Sección 5.2 sobre el corpus de datos). Esta alta precisión, ligeramente inferior a la de trabajos anteriores (85%), sugiere que los datos de entrenamiento para esta emoción eran de alta calidad y distinguibles, permitiendo al modelo aprender patrones claros para identificarla correctamente. La matriz de confusión respalda el valor de la alta precisión observada en la clasificación de la emoción, mostrando la menor cantidad de falsos positivos (31 errores) en comparación con otras categorías emocionales. No obstante, se evidencia que el mayor número de falsos positivos (19) se produce con la emoción "Desaprobación", lo que podría

explicar la disminución en la precisión de esta categoría respecto al estudio anterior. Este fenómeno de confusión entre ambas emociones resulta plausible, considerando que la "Desaprobación" fue una de las emociones objetivo del balanceo del corpus, incrementando así la cantidad de datos disponibles para el entrenamiento y, por ende, la posibilidad de que el modelo encuentre similitudes en la naturaleza y expresión lingüística de ambas categorías emocionales.

Por otro lado, tanto en el estudio actual como en el trabajo publicado por Noemí Merayo et al. [3], que representa una optimización del estudio previo realizado por Javier Estévez [1], la emoción "Desaprobación" presenta la menor precisión (57,1%) a pesar de contar con un mayor número de instancias (969) que en el estudio anterior. Esto indica que el modelo tiene dificultades para diferenciarla de otras emociones, posiblemente debido a la complejidad y ambigüedad inherente a esta emoción, o a la falta de claridad en los datos de entrenamiento. Si bien no se observa una mejora en comparación con el modelo anterior, que obtuvo una precisión de 59,7% para esta misma clase, los resultados actuales aún reflejan la necesidad de refinar la capacidad del modelo para identificar correctamente la "Desaprobación". Además, es posible que el modelo la confunda con otras emociones negativas, lo que contribuye a esta disminución de la precisión, ya que por muchas veces cuando hay desaprobación puede ir acompañada también de emociones como la tristeza, la decepción o incluso el enfado. La matriz de confusión respalda esta observación, mostrando una cantidad significativa de falsos positivos (88) para "Desaprobación".

Cabe destacar que las emociones "Aprobación/Empatía/Confianza" (71%) e "Interés/Anticipación/Hype" (71.7%) mantienen un porcentaje de acierto en la clasificación similar a estudios anteriores con este modelo (74,7%). Esto sugiere que, de forma indirecta, se han mantenido los buenos resultados en la identificación de estas emociones, evidenciando la capacidad del modelo para capturar y aprender patrones sólidos de manera consistente, como se refleja en la baja cantidad de falsos positivos en la matriz de confusión. No obstante, se observa un descenso en la precisión de estas categorías respecto al estudio previo [3], posiblemente debido al balanceo del corpus y a la mayor presencia de falsos positivos con emociones como "Desaprobación" (29) e "Interés/Anticipación/Hype" (21) en el caso de "Aprobación/Empatía/Confianza" y con “Aprobación/Empatía/Confianza" (20) en el caso de "Interés/Anticipación/Hype". Esto es esperable, ya que la "Desaprobación" puede presentar desafíos relacionados con el

sarcasmo y la ironía, mientras que el "Hype" es una emoción extremadamente positiva que puede ser difícil de distinguir de otras emociones positivas.

Además, se observa una mejora notable en la precisión para "Enfado/Ira" (76,4%) e "Indeterminado" (64,1%) en comparación con trabajos previos, siendo 59% y 52,3% respectivamente. Gracias al balanceo de mensajes en el corpus realizado en este trabajo, la mejora en el rendimiento en la clasificación de estas emociones es más que evidente, lo cual se refleja en una disminución de falsos positivos en la matriz de confusión [1].

En resumen, el modelo actual muestra un rendimiento notable en la identificación de emociones, especialmente en aquellas que fueron objeto de balanceo en el corpus de entrenamiento. Se obtuvieron mejoras significativas en la precisión de "Enfado/Ira" e "Indeterminado" en comparación con trabajos previos [1, 3]. Sin embargo, "Desaprobación" sigue siendo la emoción con menor desempeño, lo que sugiere que el balanceo de clases no fue suficiente para superar completamente los desafíos inherentes a esta categoría. Por otro lado, las emociones "Aprobación/Empatía/Confianza", “Decepción/Tristeza” e "Interés/Anticipación/Hype" mantuvieron un buen rendimiento, similar al de estudios anteriores, lo que indica que el modelo ha capturado patrones sólidos para identificar estas emociones de manera consistente. Estos resultados subrayan la importancia de contar con datos de entrenamiento de alta calidad y representativos, así como de abordar la complejidad y ambigüedad inherente a ciertas emociones, como la "Desaprobación", para lograr un desempeño óptimo en la clasificación de emociones.

Es importante señalar que la precisión global del modelo (68%) se mantiene similar a la de trabajos anteriores, a pesar de las mejoras observadas en algunas emociones. Esto plantea interrogantes sobre si el rendimiento del modelo está limitado por la cantidad de datos de entrenamiento disponibles o si es necesario un rediseño del modelo, tal vez prescindiendo de RoBERTuito y utilizando uno con capas adicionales específicas para la tarea de clasificación. Estas cuestiones se analizarán en detalle en las conclusiones finales y se abordarán en futuras líneas de investigación.

6

Aplicaciones del modelo: Predicción y presentación de los resultados

6.1 Introducción

Tras el entrenamiento del modelo RoBERTuito, se abren dos vías principales para realizar predicciones sobre nuevos datos, que se describirán en este capítulo de la memoria.

La primera consiste en utilizar la misma clase *Trainer* empleada durante el entrenamiento, junto con su función *evaluate*. Esta opción, aunque sencilla, puede resultar menos flexible para ciertas aplicaciones.

La segunda vía, más versátil, implica guardar los archivos de configuración del modelo (*config.json*) y del tokenizador (*tokenizer.json*) generados durante el entrenamiento. Estos archivos pueden ser utilizados posteriormente para cargar el modelo y el tokenizador, ya sea en conjunto con la clase *pipeline* de Hugging Face o directamente para realizar predicciones. En función de esta segunda aproximación, se exploraron dos formas de realizar predicciones:

1. *Predicción con función Predict:* Esta herramienta, desarrollada en el presente trabajo, aprovecha los archivos de configuración guardados *(config.json* y *tokenizer.json*) para cargar el modelo RoBERTuito y el tokenizador correspondiente. A través de la clase *pipeline* de Hugging Face, se simplifica el proceso de inferencia, permitiendo realizar predicciones de polaridad y emociones de manera rápida y eficiente sobre nuevos datos.

2. *Predicción con interfaz gráfica:* En colaboración con otra compañera de la carrera, se ha adaptado una aplicación con interfaz gráfica desarrollada en su TFG a nuestro modelo de estudio. Esta aplicación utiliza los archivos de configuración para cargar el modelo y el *tokenizer*, y realizar predicciones sobre nuevos datos de entrada a partir de un archivo con mensajes, devolviendo otro archivo con las predicciones correspondientes. Además, ofrece la posibilidad de visualizar la clasificación de las predicciones mediante gráficos, lo que facilita la interpretación y análisis de los resultados.

6.2 Desarrollo de una función de predicción

Se ha desarrollado una función *predict*, siguiendo el enfoque de cargar el modelo y el tokenizador a partir de los archivos de configuración guardados. Esta función permite realizar predicciones de polaridad o emociones sobre un texto o una lista de textos.

Como se ilustra en la Figura 28, la función toma como argumentos la tarea a realizar (*task*, ya sea "polaridad" o "emociones") y los datos a clasificar (*data*). Dependiendo de la tarea seleccionada, se define el número de salidas del modelo y la configuración para interpretar dichas salidas, que el modelo proporciona en formato numérico. Posteriormente, se carga el modelo y el *tokenizer* desde los archivos de configuración almacenados en la nube, estos fueron creados siguiendo los detalles descritos en el Apartado 5.5. La Figura 29 muestra las líneas de código empleadas para guardar estos archivos, correspondientes a los modelos entrenados para cada tarea, en este caso y a modo de ejemplo la tarea de clasificación por emociones.

Para optimizar el proceso de predicción, se ajusta el nivel de registro de mensajes (*log*) para evitar la generación de avisos irrelevantes. Finalmente, se utiliza la clase *pipeline* para realizar las predicciones sobre los datos de entrada. El parámetro top_k=1 indica que solo se desea obtener la respuesta con mayor probabilidad para cada texto. La función devuelve únicamente las etiquetas predichas para cada texto, almacenadas en el campo *label*, sin incluir las probabilidades asociadas.

```
## Prediction Function
#task = "polaridad", "emociones"
#data = string data or list of strings to predict from

def predict(task, data):
  if(task=="emociones"):
    num_labels=6
    id2label= {0:"Decepción/Tristeza", 1: "Aprobación/Empatía/Confianza", 2: "Enfado/Ira",
               3: "Interés/Anticipación/Hype", 4: "Desaprobación", 5:"Indeterminado"}
    label2id= {"Decepción/Tristeza" :0,"Aprobación/Empatía/Confianza" :1,"Enfado/Ira" :2,
               "Interés/Anticipación/Hype" :3,"Desaprobación" :4, "Indeterminado" :5}
  else:
    num_labels=3
    id2label= {0: "NEG", 1: "NEU", 2: "POS"}
    label2id= {"NEG" :0,"NEU" :1,"POS" :2}

  transformers.logging.set_verbosity_error()

  model_path = "./drive/MyDrive/Colab Notebooks/TFG/modelostwitch/model_robertuito_"+ task
  tok_path = "./drive/MyDrive/Colab Notebooks/TFG/modelostwitch/tok_robertuito_"+ task

  model = AutoModelForSequenceClassification.from_pretrained(model_path, num_labels=num_labels, label2id = label2id, id2label = id2label, ignore_mismatched_sizes=True)
  tokenizer = AutoTokenizer.from_pretrained(tok_path)

  pipe = TextClassificationPipeline(model=model,tokenizer=tokenizer,top_k=1,device=0)
  transformers.logging.set_verbosity_warning()
  return([i[0]["label"] for i in pipe(data)])
```

Figura 28: Función de predicción para la clasificación de textos

```
model.save_pretrained("/content/drive/MyDrive/Colab Notebooks/TFG/modelostwitch/model_robertuito_emociones")
tokenizer.save_pretrained("/content/drive/MyDrive/Colab Notebooks/TFG/modelostwitch/tok_robertuito_emociones")
```

Figura 29: Sentencias de código utilizada para guardar el modelo y tokenizador destinados a la clasificación por Emociones

La Tabla 15 muestra ejemplos de predicciones realizadas por el modelo RoBERTuito para la clasificación de polaridad y emociones en un conjunto de textos de prueba, utilizando las sentencias de las Figuras 30 y 31.

```
#Sentimmientos
predict(task="polaridad",
        data=["te quiero","te odio","me das pena",
              "no tengo una opinion clara al respecto",
              "siento profundamente tu reciente pérdida",
              "soy vegano porque siento que debo hacer algo por el bienestar animal",
              "hasta la polla de este juego",
              "que puto coñazo",
              "conazo de juego",
              "p_t4s",
              "sin miedo al exito",
              "me paso tu chat por el papo",
              "te como el papo tio"])
```

Figura 30: Sentencia de código utilizada para obtener los resultados en términos de Polaridad de la Tabla 15

```
#Emociones
predict(task="emociones",
        data=["te quiero","te odio","me das pena",
              "no tengo una opinion clara al respecto",
              "siento profundamente tu reciente pérdida",
              "soy vegano porque siento que debo hacer algo por el bienestar animal",
              "hasta la polla de este juego",
              "que puto coñazo",
              "conazo de juego",
              "p_t4s",
              "sin miedo al exito",
              "me paso tu chat por el papo",
              "te como el papo tio"])
```

Figura 31: Sentencia de código utilizada para obtener los resultados en términos de Emociones de la Tabla 15

Texto	Predicción de polaridad	Predicción de emoción
te quiero	Positivo	Aprobación/Empatía/Confianza
te odio	Negativo	Enfado/Ira
me das pena	Negativo	Decepción/Tristeza
no tengo una opinion clara al respecto	Negativo	Desaprobación
siento profundamente tu reciente pérdida	Negativo	Decepción/Tristeza
soy vegano porque siento que debo hacer algo por el bienestar animal	Positivo	Aprobación/Empatía/Confianza
hasta la polla de este juego	Negativo	Enfado/Ira
que puto coñazo	Negativo	Enfado/Ira
conazo de juego	Negativo	Desaprobación
p_t4s	Negativo	Enfado/Ira
sin miedo al éxito	Positivo	Aprobación/Empatía/Confianza
me paso tu chat por el papo	Negativo	Enfado/Ira
te como el papo tio	Positivo	Aprobación/Empatía/Confianza

Tabla 15: Ejemplo de predicción de Polaridad y Emoción obtenidas con las ejecuciones de las Figuras 34 y 35

En la Tabla 15 se observa que la clasificación de los comentarios "me paso tu chat por el papo" como “Enfado/Ira” y "te como el papo tío" como “Aprobación/Empatía/Confianza” demuestra la capacidad del modelo para discernir la carga emocional de expresiones coloquiales de Twitch, incluso sin un análisis exhaustivo del contexto. Este resultado es especialmente relevante en el caso de emociones como el “Enfado/Ira”, cuya identificación precisa es crucial para comprender la dinámica de la comunidad y la percepción de los espectadores hacia los *streamers*.

6.3 Integración del modelo en una interfaz gráfica

En el marco de este trabajo, se estableció una colaboración con otra compañera de carrera, autora de un TFG centrado en la modelización de algoritmos BERT para evaluar la respuesta emocional en Instagram en el ámbito de la salud mental. Su aplicación, diseñada para analizar archivos .csv o .xlsx con mensajes y generar predicciones por polaridad o emoción, se convirtió en el punto de partida para una expansión de funcionalidades y aplicaciones para nuestro caso de uso concreto, esto es, para nuestro modelo BERT en el ámbito de Twitch y videojuegos.

Esta colaboración se enfocó en dotar a la herramienta de una mayor versatilidad en el análisis de la respuesta emocional, extendiendo su aplicabilidad más allá de la red social inicial a otras plataformas relevantes como Twitch. Para lograr este objetivo, se integraron los modelos de clasificación de polaridad y emociones desarrollados en este trabajo, permitiendo a los usuarios seleccionar el modelo más adecuado según la plataforma de origen de los datos. De esta forma, el software se convierte en una solución adaptable y personalizable para el análisis de sentimientos en diversas redes sociales, como X, Instagram, Pinterest, Twitch, entre otras.

La herramienta resultante, cuya interfaz se muestra en la Figura 32, ofrece una experiencia de usuario intuitiva y eficiente, permitiendo desde introducir comentarios para obtener predicciones individuales a cargar datos en formato tabulares (es decir en formato de hoja de cálculo) y obtener predicciones de polaridad y emociones de manera rápida y sencilla en el mismo formato. Como se ilustra en la Figura 33, la aplicación devuelve las predicciones en un archivo con dos columnas: "*Comments*" y "*Prediction*", donde esta última contiene la emoción o polaridad predicha para cada mensaje de entrada.

A nivel de código, la aplicación, desarrollada en Python utilizando Anaconda [59] y Jupyter Notebook [60], carga el modelo y el tokenizador desde un directorio específico, procesa los datos de entrada y genera las predicciones correspondientes, que se guardan en un archivo de salida. Las sentencias de código de la Figura 34 ilustran este proceso de carga.

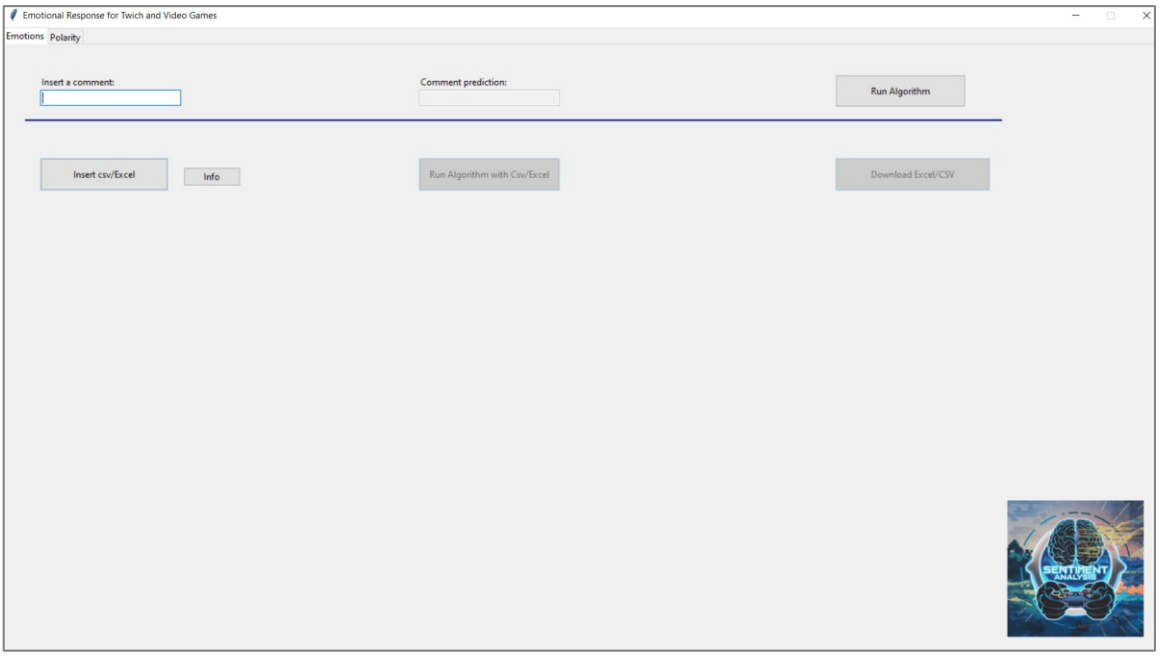


Figura 32: Interfaz gráfica de la aplicación

	A	B
1	**Comments**	**Prediction**
2	nivel 80? lo tenia bien guardado el vicio genshin	Neutral
3	HUH	Neutral
4	Que enferma	Anger/Rage
5	ENFERMA DE MRDA	Anger/Rage
6	??	Neutral
7	HUH	Neutral
8	BOOBEST BOOBEST	Approval/Empathy
9	ENFERMA ALERT ENFERMA ALERT ENFERMA ALERT ENFERMA A	Anger/Rage
10	ES MENOOOR	Disapproval
11	ENFERMITA	Anger/Rage
12	queeeee??	Neutral
13	ENFERMA ALERT ENFERMA ALERT ENFERMA ALERT ENFERMA A	Anger/Rage
14	Que dices nissa, los jugadores de genshin no hacemos eso	Disapproval
15	?? ?? ??	Neutral

Figura 33: Ejemplo de archivo de salida con predicciones por Polaridad

Si bien el código completo no puede ser divulgado por cuestiones de propiedad intelectual, se presentan algunas implementaciones propias en la Figura 35, que ilustra cómo la herramienta permite seleccionar el modelo y ajustar los parámetros de configuración para adaptarse a las particularidades de cada plataforma y conjunto de datos, como el lenguaje específico de los videojuegos y las retransmisiones en directo. Para mejorar la accesibilidad y comprensión de los resultados, dándole más horizontes y potencialidad a la interfaz, se han traducido las clases de polaridad y emoción al inglés. En el caso de

"Emociones", se ha simplificado las categorías con 3 emociones como es el caso de "*Approval/Empathy/Confidence*" a "*Approval/Empathy*", facilitando así la visualización y análisis de los datos.

```
# Cargar el modelo pre-entrenado y el tokenizador
model_path = r".\modelo_entrenado_twitch\model_robertuito_emociones"
tokenizer_path = r".\modelo_entrenado_twitch\tok_robertuito_emociones"

model = AutoModelForSequenceClassification.from_pretrained(model_path)
tokenizer = AutoTokenizer.from_pretrained(tokenizer_path)
```

Figura 34: Fragmentos de código para el proceso de carga de modelos y tokenizadores

```
# Mapear las etiquetas predichas a sus correspondientes etiquetas de texto
id2label = {0: "Disappointment/Sadness", 1: "Approval/Empathy", 2: "Anger/Rage",
            3: "Interest/Hype", 4: "Disapproval", 5: "Neutral"}

predicted_labels_text = [id2label[label_id] for label_id in predicted_labels_list]
```

Figura 35: Fragmentos de código para el proceso de ajuste de parámetros de configuración para la clasificación para la clasificación emocional en el modelo de Twitch

Además de las predicciones, la aplicación genera gráficos que resumen y visualizan los resultados de manera clara y concisa. Estos gráficos, como se muestra en la Figura 36, pueden mostrar la distribución global de las clases mediante un gráfico de barras, así como la evolución a lo largo de la transmisión. Esta funcionalidad de visualización resulta especialmente útil para investigadores y profesionales que deseen comprender y comunicar los patrones de clasificación presentes en los datos.

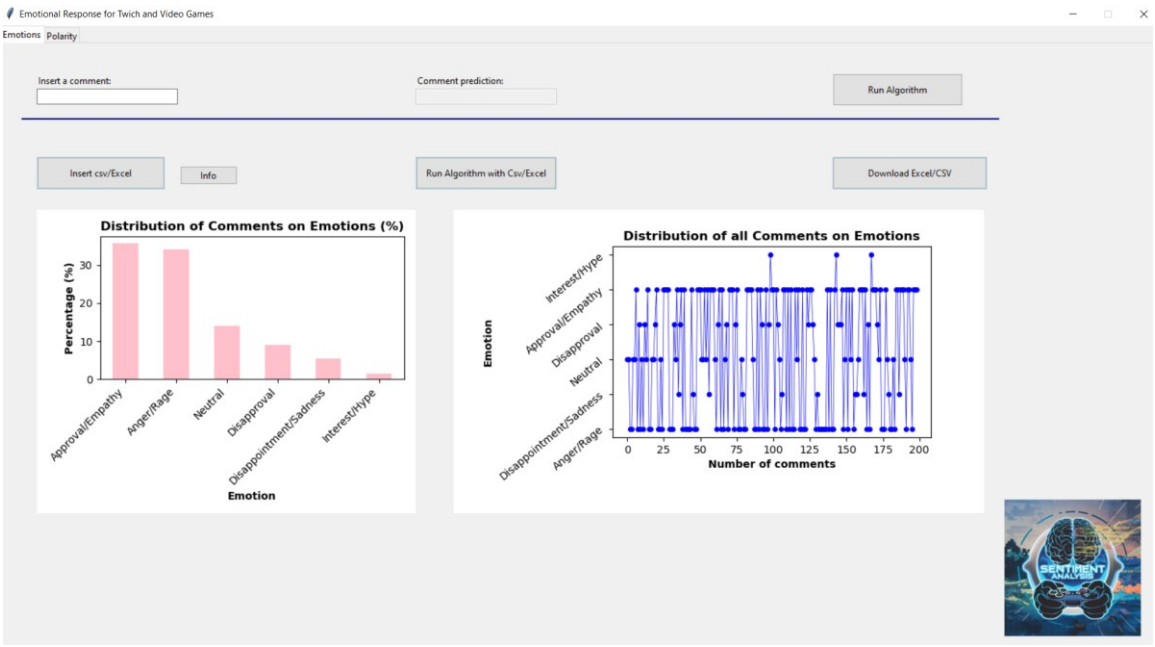


Figura 36: Interfaz gráfica de la aplicación y gráficos generados con las predicciones

A modo de ejemplo y para evaluar el rendimiento del software, se realizó una prueba de predicción en un chat de Twitch con 200 mensajes. El chat seleccionado pertenece a la *streamer* Nissaxter, conocida por su carácter polémico y agresivo, mientras jugaba a *Genshin Impact*, un juego que también genera opiniones encontradas debido a sus mecánicas de monetización. El conjunto de datos recoge mensajes de la mitad de la transmisión, evitando así la polarización positiva que suele darse al inicio y al final de los directos (debido a que se suelen tratar de saludos y despedidas de los espectadores a la streamer).

Los resultados de esta prueba, detallados en las Figuras 42, 43, 44 y 45, demuestran la eficacia de la herramienta para analizar la respuesta emocional en Twitch, capturando tanto la distribución general de las clases como su evolución temporal. En el caso analizado, se observa una predominancia de la polaridad negativa, en particular de la emoción "Enfado/Ira", lo que se alinea con el contexto del chat, caracterizado por la reputación polémica de la streamer y la controversia en torno al juego.

Sin embargo, la presencia significativa de la emoción "Aprobación/Empatía/Confianza" plantea una aparente contradicción. Es posible que estas expresiones positivas sean genuinas, pero también podrían estar siendo utilizadas de manera irónica o sarcástica, lo cual sería coherente con el tono general del chat. Este fenómeno resalta la importancia de considerar el contexto y las particularidades de cada comunidad al analizar el sentimiento

en línea, ya que la interpretación de las emociones expresadas puede variar según la dinámica y las normas sociales establecidas en cada plataforma o comunidad.

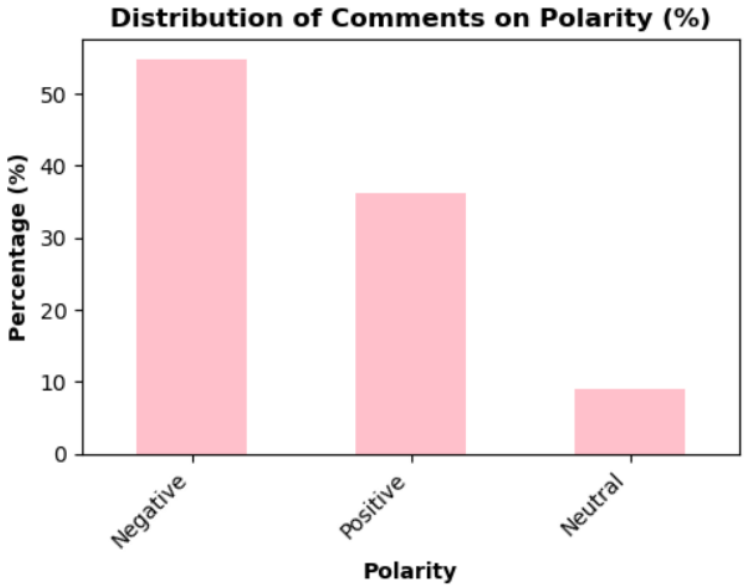


Figura 37: Distribución de comentarios por Polaridad (200 mensajes)

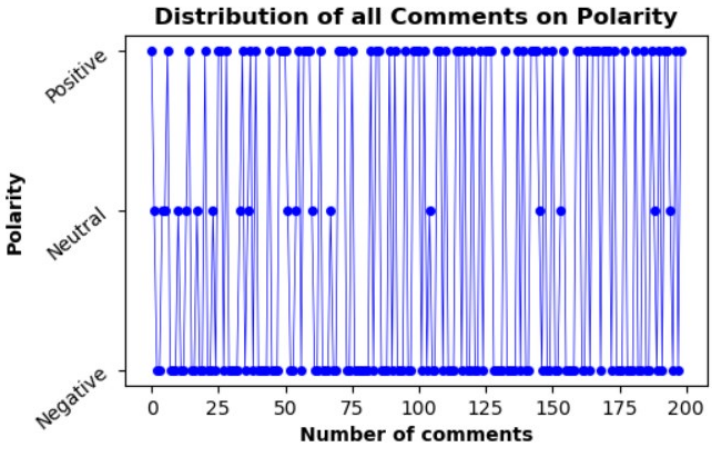


Figura 38: Distribución de comentarios a lo largo
de la transmisión por Polaridad (200 mensajes)

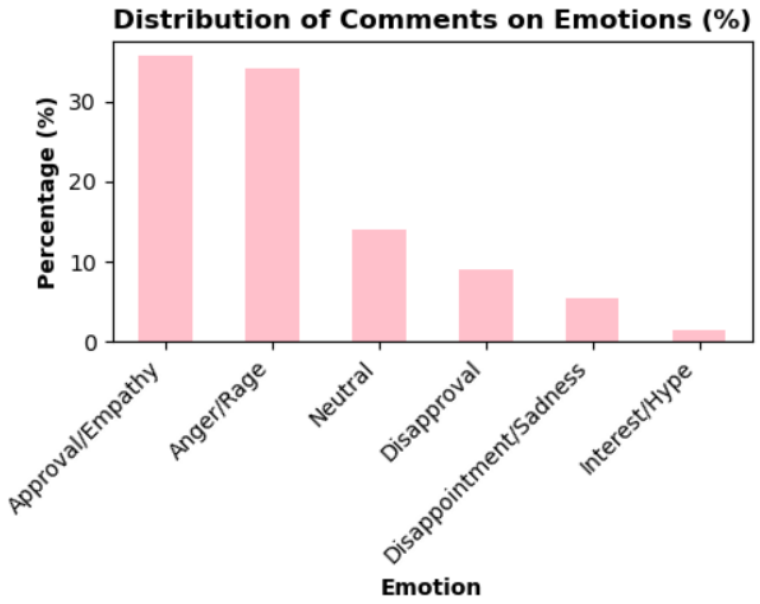


Figura 39: Distribución de comentarios por Emociones (200 mensajes)

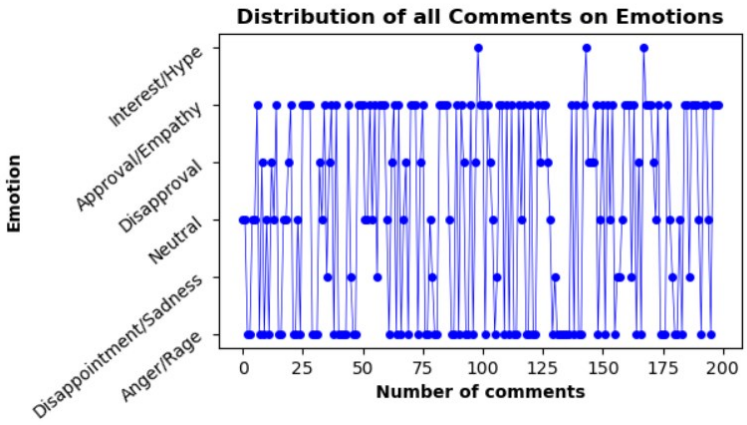


Figura 40: Distribución de comentarios a lo largo
de la transmisión por Emociones (200 mensajes)

Esta colaboración interdisciplinaria ha culminado en el desarrollo de una herramienta versátil y robusta para el análisis de sentimientos en redes sociales, con un amplio espectro de aplicaciones potenciales en diversos ámbitos, desde la investigación académica hasta el marketing y la gestión de comunidades online. La capacidad de adaptación del programa a diferentes plataformas y conjuntos de datos, así como su interfaz intuitiva y sus funciones de visualización, la convierten en un recurso valioso para cualquier persona interesada en comprender y analizar las emociones expresadas en las redes sociales.

En particular, los modelos entrenados a partir del conjunto de datos ampliado y enriquecido con lenguaje específico de videojuegos y *Leet Speak*, representan una contribución significativa al campo del procesamiento del lenguaje natural en español, especialmente

en el dominio de los videojuegos. Estos modelos, junto con el software desarrollado, pueden ser de gran utilidad para otros grupos de investigación en PLN, proporcionando un punto de partida sólido para futuros trabajos en el análisis de sentimientos en este contexto. Además, la facilidad para importar y adaptar estos modelos a otras herramientas de análisis emocional amplía aún más su potencial de aplicación y transferencia a otros dominios y plataformas.

7

Conclusiones y líneas futuras

7.1 Conclusiones

El presente Trabajo de Fin de Grado ha logrado ampliar y enriquecer un corpus lingüístico previamente creado relacionado con el análisis de sentimientos en transmisiones de videojuegos en Twitch. La metodología empleada, basada en la selección estratégica de transmisiones y streamers, ha demostrado ser efectiva para diversificar el conjunto de datos, incorporando una mayor variedad de géneros de videojuegos, niveles de competitividad y comunidades de seguidores. Además, la inclusión de mensajes en *Leet Speak* ha enriquecido el corpus con expresiones emocionales encubiertas, mejorando la capacidad del modelo para detectar emociones negativas.

Los resultados obtenidos en la clasificación de polaridad y emociones utilizando el modelo RoBERTuito son alentadores. Se ha logrado una mejora significativa en la detección de emociones específicas, como "Enfado/Ira" e "Indeterminado", gracias al balanceo de emociones en el *dataset* y a la incorporación de nuevos datos. Aunque la emoción "Desaprobación" sigue siendo un desafío, el modelo ha demostrado un rendimiento sólido en la identificación de polaridad general y otras emociones clave, mejorando en casi todos los casos los resultados obtenidos en trabajos previos [1, 3], y manteniendo una precisión similar en aquellos casos en los que no se ha superado.

La adaptación de una interfaz gráfica existente ha demostrado ser una herramienta valiosa para el análisis de sentimientos en Twitch y otras plataformas de redes sociales. Su capacidad para cargar modelos entrenados, realizar predicciones y visualizar resultados mediante gráficos, facilita la interpretación y el análisis de la respuesta emocional en diferentes contextos. La adaptabilidad del software a la plataforma Twitch y a conjuntos de datos específicos de videojuegos, así como su interfaz intuitiva, lo convierten en un

recurso útil para creadores de contenido, profesionales e investigadores interesados en comprender las dinámicas emocionales en la comunidad de jugadores en línea.

En resumen, este TFG ha sentado las bases para futuras investigaciones en el análisis de sentimientos en el ámbito de los videojuegos y las redes sociales. Los resultados obtenidos y la herramienta desarrollada contribuyen al avance del conocimiento en este campo y ofrecen nuevas oportunidades para comprender y gestionar la respuesta emocional en línea.

7.2 Líneas futuras

Aunque los resultados obtenidos, especialmente en la clasificación de emociones, son prometedores, existen áreas de mejora que pueden ser exploradas en futuras investigaciones. Algunas de las líneas futuras a seguir serían las siguientes:

- Ampliación y diversificación continua del corpus: A pesar de los esfuerzos realizados en este trabajo, el corpus de datos sigue siendo limitado en comparación con la vasta cantidad de mensajes generados en Twitch. Se propone continuar ampliando el corpus, incorporando transmisiones de una mayor variedad de juegos, streamers y comunidades, así como explorando otras fuentes de datos, como redes sociales y foros de discusión. Es importante mantener un corpus balanceado en polaridad y emociones con este aumento.
- Automatización de la selección y etiquetado de mensajes: Actualmente, la selección y el etiquetado de mensajes para el corpus de entrenamiento se realizan de forma manual, lo que consume mucho tiempo y recursos. Se propone investigar métodos para automatizar este proceso, utilizando técnicas de aprendizaje activo, *clustering* o modelos de lenguaje preentrenados para identificar y etiquetar mensajes relevantes de manera más eficiente. Esto permitiría acelerar el proceso de ampliación del corpus y reducir el tiempo de trabajo dedicado a esta tarea.
- Profundización en el análisis de emociones complejas: El modelo actual se centra en la clasificación de polaridad y emociones básicas, pero las emociones humanas son mucho más complejas y matizadas. Se propone investigar la detección de emociones más específicas, como el sarcasmo, la ironía, el humor y la frustración, que son comunes en el lenguaje de los videojuegos. Esto podría requerir el

desarrollo de nuevos modelos y la adaptación de técnicas de PLN para capturar estos matices emocionales.

- Búsqueda más exhaustiva de hiperparámetros: Si bien se ha realizado una búsqueda inicial de hiperparámetros utilizando la plataforma Wandb, se podrían emplear otras técnicas de optimización, como la optimización bayesiana, para encontrar una combinación óptima de hiperparámetros que mejore aún más el rendimiento del modelo.
- Mejora en el preprocesamiento de los datos: Aunque se ha utilizado el preprocesamiento integrado en el modelo RoBERTuito, se podrían explorar otras técnicas de preprocesamiento, como la eliminación de *stopwords* o la corrección ortográfica, para adaptar mejor los datos a las particularidades del lenguaje utilizado en Twitch. Esto podría contribuir a reducir el ruido en los datos y mejorar la calidad del modelo.
- Estudio de modelos de clasificación alternativos: En este trabajo se ha empleado el modelo RoBERTuito para la clasificación de emociones y polaridad. Sin embargo, se propone explorar el potencial de otros modelos BERT, como BERT-base, y evaluar si la inclusión de capas de clasificación específicas para la tarea mejora el rendimiento en comparación con RoBERTuito, que no ha sido específicamente adaptado para este contexto. Se podrían evaluar modelos de clasificación alternativos, como los basados en redes neuronales recurrentes (RNN) o modelos de atención, para determinar si ofrecen ventajas en términos de precisión o eficiencia en la clasificación de emociones y polaridad en mensajes de Twitch.
- Desarrollo de herramientas gráficas de análisis más sofisticadas: La interfaz gráfica empleada en este trabajo, ampliada con el modelo de esta investigación, constituye un primer paso hacia la creación de herramientas prácticas para el análisis de sentimientos en Twitch. Se propone desarrollar una interfaz gráfica más avanzada y potente que permita el análisis en tiempo real de grandes volúmenes de datos, la detección de patrones de comportamiento emocional a lo largo del tiempo, y la identificación de usuarios o comunidades tóxicas, incluyendo así diversas funcionalidades. Estas herramientas podrían ser de gran utilidad para streamers, moderadores de comunidades y empresas interesadas en comprender y gestionar la respuesta emocional en línea.

Con este trabajo se han fortalecido las bases existentes en trabajos previos y se han añadido nuevos puntos clave para favorecer el progreso de la investigación en el campo del análisis de sentimientos en transmisiones de videojuegos en Twitch. Se ha demostrado que el enriquecimiento, la ampliación y el balanceo del corpus de datos tienen un impacto significativo en el rendimiento del modelo, independientemente de los cambios realizados en este. Las líneas futuras de investigación se centrarán en seguir desarrollando modelos más sofisticados, en explorar nuevas técnicas de preprocesamiento y continuar con la ampliación del corpus para mejorar aún más la precisión en la detección de emociones complejas y ambiguas, prestando especial atención al enriquecimiento lingüístico y balanceo del dataset.

8

Anexo

8.1 Requisitos de computación

En el desarrollo de este trabajo, se empleó Google Colaboratory (Colab) como entorno de computación en la nube para llevar a cabo los experimentos y análisis. Colab proporcionó acceso a recursos computacionales gratuitos, incluyendo una GPU NVIDIA Tesla T4, 12.69 GB de memoria RAM y 107.72 GB de almacenamiento. Estos recursos resultaron ser suficientes para la mayoría de las tareas de aprendizaje automático, como el entrenamiento y evaluación del modelo RoBERTuito en la clasificación de polaridad y emociones. No obstante, como se detalla en la Sección 5.6.2, la versión gratuita presentó limitaciones en cuanto a la capacidad de disco y la disponibilidad de GPUs, especialmente durante la búsqueda de hiperparámetros utilizando la plataforma Wandb. Para solventar estas restricciones y garantizar una búsqueda de hiperparámetros más exhaustiva, se optó por una suscripción a Colab Pro (con 200 GB de almacenamiento). Cabe destacar que, si bien la suscripción ofrece la ventaja de ejecutar procesos en segundo plano, esta funcionalidad no siempre resulta eficiente debido a las limitaciones en las unidades informáticas disponibles, especialmente en tareas de larga duración y alta demanda computacional.

En este contexto, la estabilidad y el éxito de las ejecuciones dependieron en gran medida de una conexión a Internet estable y fiable. En nuestro caso, se utilizó una conexión con una velocidad de descarga de 285 Mb/s, una velocidad de subida de 105 Mb/s y una latencia de 8 ms. Esta conexión a Internet resultó crucial para garantizar la continuidad de los procesos, especialmente durante la búsqueda de hiperparámetros en la plataforma Wandb, que requería una comunicación constante y sin interrupciones con los servidores. Se observó que conexiones a Internet menos robustas, como aquellas con velocidades de subida y bajada inferiores a 30 Mb/s y una latencia superior a 10 ms, podían ocasionar

problemas de estabilidad y fallos en las ejecuciones, particularmente en la fase de búsqueda de hiperparámetros.

En conclusión, este estudio demuestra la viabilidad de llevar a cabo investigaciones en el ámbito del aprendizaje automático y el procesamiento del lenguaje natural utilizando recursos computacionales virtuales básicos, como los proporcionados por Google Colab en su versión gratuita. La suscripción a Colab Pro, aunque beneficiosa en ciertas situaciones, no es estrictamente necesaria para la mayoría de las tareas. Sin embargo, contar con una conexión a Internet de alta velocidad y baja latencia es fundamental para garantizar la estabilidad y el éxito de las ejecuciones, especialmente en tareas que requieren una comunicación constante con servidores externos, como la búsqueda de hiperparámetros.

9

Bibliografía

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

[2] P. G. Gómez, "Aplicación de técnicas de aprendizaje automático para la clasificación y predicción en diferentes casos de uso", 2023.

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

[4] "¿Qué es machine learning?", sap.com. Disponible en: https://www.sap.com/spain/products/artificial-intelligence/what-is-machine-learning.html.

[5] M. Sussmann, "What are the differences between artificial intelligence, machine learning, deep learning and generative AI?", sumologic.com, 23-abr-2024. Disponible en: https://www.sumologic.com/blog/machine-learning-deep-learning/.

[6] A. Vaswani et al., "Attention is all you need", arxiv.org, 2017. Disponible en: http://arxiv.org/abs/1706.03762.

[7] J. Devlin, M.-W. Chang, K. Lee, y K. Toutanova, "BERT: Pre-training of deep bidirectional Transformers for language understanding", arxiv.org, 2018. Disponible en: http://arxiv.org/abs/1810.04805.

[8] T. Wolf et al., "Transformers: State-of-the-art natural language processing", en Proceedings of the 2020 Conference on Empirical Methods in Natural Language

Processing: System Demonstrations, pp. 38–45, 2020. Disponible en: https://aclanthology.org/2020.emnlp-demos.6/.

[9] J. Clement, "Online gaming - Statistics & Facts", statista.com, 2023. Disponible en: https://www.statista.com/topics/1551/online-gaming/.

[10] J. A. Velez, M. R. Gotlieb, G. Graybeal, A. Abitbol, y J. A. Villarreal, "Live streams and revenue streams: Twitch as a hybrid gaming culture", en Video Games, New York, NY : Routledge, 2018. | Series: Electronic media research series: Routledge, pp. 193–207, 2018. Disponible en: https://www.semanticscholar.org/paper/43951816685d4f61dfa06f53e6890f4cc3ff2bb4.

[11] F. Barbieri, L. Espinosa Anke, M. Ballesteros, J. Soler, y H. Saggion, "Towards the understanding of gaming audiences by modeling twitch emotes", en Proceedings of the 3rd Workshop on Noisy User-generated Text, pp. 11–20, 2017. Disponible en: https://aclanthology.org/W17-4402/.

[12] J. Deng, F. Cuadrado, G. Tyson, y S. Uhlig, "Behind the game: Exploring the twitch streaming platform", en 2015 International Workshop on Network and Systems Support for Games (NetGames), 2015. Disponible en: https://www.semanticscholar.org/paper/Behind-the-game%3A-Exploring-the-twitch-streaming-Deng-Cuadrado/eb2bf642f844d9d548d1c48a7a71a2c3abc92e6c.

[13] Y. Agca, "Analyzing Video Game Content and Sentiment A Study on Categories, Emotional Responses, and Success Factors", en International Research in Social, Human and Administrative Sciences XIV, pages 67–79, 2023. Disponible en: https://www.researchgate.net/publication/371172928_Analyzing_Video_Game_Content_and_Sentiment_A_Study_on_Categories_Emotional_Responses_and_Success_Factors.

[14] T. Guzsvinecz y J. Szűcs, "Length and sentiment analysis of reviews about top-level video game genres on the Steam platform", Computers in Human Behavior, 2023. Disponible en: https://www.researchgate.net/publication/373882591_Length_and_sentiment_analysis_of_reviews_about_top-level_video_game_genres_on_the_steam_platform.

[15] J. Orta Casado y B. Peña-Acuña, "Video games lexicon included in Spanish language: a multiple case study", linguodidactica, vol. 1, pp. 15–35, 2022. Disponible en: https://www.uhu.es/publicaciones/ojs/index.php/linguodidactica/article/view/7428.

[16] J. Olejniczak, "A linguistic study of language variety used on twitch.Tv: Descriptive and corpus-based approaches", 2015. Disponible en: https://www.researchgate.net/publication/317168136_A_Linguistic_Study_of_Language_Variety_Used_on_TwitchTv_Descriptive_and_Corpus-Based_Approaches.

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

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

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

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

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

[22] Sys. Disponible en: https://docs.python.org/3/library/sys.html.

[23] Logging. Disponible en: https://docs.python.org/3/library/logging.html.

[24] Tensorflow. Disponible en: https://www.tensorflow.org/.

[25] Gc. Disponible en: https://docs.python.org/3/library/gc.html.

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

[27] Evaluate. Disponible en: https://pypi.org/project/evaluate/.

[28] Scikit-learn (o sklearn). Disponible en: https://scikit-learn.org/stable/.

[29] Hugging Face Transformers. Disponible en: https://huggingface.co/docs/transformers/index.

[30] Google colab. Disponible en: https://colab.research.google.com/.

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

[32] Twitch. Disponible en: https://www.twitch.tv/.

[33] L. Pardo, Twitch Downloader repository. 2023. Disponible en: https://github.com/lay295/TwitchDownloader.

[34] F. Demir, "14- Deep autoencoder-based automated brain tumor detection from MRI data", 2022. Disponible en: https://www.sciencedirect.com/science/article/abs/pii/B9780323911979000138.

[35] R. Plutchik, "Chapter 1 - A general psychoevolutionary theory of emotion", Theories of Emotion, pp. 3-33, 1980. Disponible en: https://www.sciencedirect.com/science/article/abs/pii/B9780125587013500077?via%3Dihub.

[36] F. Yus, "La comunicación en la era digital", Pragmática, pp. 16-17, 2020. Disponible en: https://personal.ua.es/francisco.yus/site/Akal.pdf.

[37] W. A. Hamilton, O. Garretson, y A. Kerne, "Streaming on twitch: Fostering participatory communities of play within live mixed media", en Proceedings of the

SIGCHI Conference on Human Factors in Computing Systems, 2014. Disponible en: https://doi.org/10.1145/2556288.2557048.

[38] H. Jodén y J. Strandell, "Building viewer engagement through interaction rituals on Twitch.tv", Inf. Commun. Soc., vol. 25, núm. 13, pp. 1969–1986, 2022. Disponible en: https://doi.org/10.1080/1369118x.2021.1913211.

[39] J. Kneer, Y. Zhang, B. G. Żerebecki, y T. Wulf, "Same gaming: An exploration of relationships between gender traits, sexual orientation, motivations, and enjoyment of playing video games", Simul. Gaming, vol. 53, núm. 5, pp. 423–445, 2022. Disponible en: https://doi.org/10.1177/10468781221113030.

[40] T. L. Dietz, "An examination of violence and gender role portrayals in video games: Implications for gender socialization and aggressive behavior", Sex Roles, vol. 38, núm. 5/6, pp. 425–442, 1998. Disponible en: https://doi.org/10.1023/a:1018709905920.

[41] R. Pan, "Video Games and Gender Equality: How Has Video Gaming Become a Mens Privilege?", Communications in Humanities Research, 2023. Disponible en: https://chr.ewapublishing.org/article/5170a57a9eb64c5183fcbf49ec176ca5.

[42] K. Perry, "Damsels and darlings: decoding gender equality in video game communities", Fem. Media Stud., vol. 22, núm. 5, pp. 1102–1119, 2022. Disponible en: https://doi.org/10.1080/14680777.2021.1883085.

[43] E. Downs y S. L. Smith, "Keeping abreast of hypersexuality: A video game character content analysis", Sex Roles, vol. 62, núm. 11–12, pp. 721–733, 2010. Disponible en: https://doi.org/10.1007/s11199-009-9637-1.

[44] T. Lynch, J. E. Tompkins, I. I. van Driel, y N. Fritz, "Sexy, strong, and secondary: A content analysis of female characters in video games across 31 years: Female game characters across 31 years", J. Commun., vol. 66, núm. 4, pp. 564–584, 2016. Disponible en: https://doi.org/10.1111/jcom.12237.

[45] J. Kohlburn, H. Cho, y H. Moore, "Players' perceptions of sexuality and gender-inclusive video games a pragmatic content analysis of steam reviews", Converg. Int. J. Res. New Media Technol., vol. 29, núm. 2, pp. 379–399, 2023. Disponible en: https://doi.org/10.1177/13548565221137481.

[46] D. Recktenwald, "Toward a transcription and analysis of live streaming on Twitch", J. Pragmat., vol. 115, pp. 68–81, 2017. Disponible en: https://www.researchgate.net/publication/314030873_Toward_a_transcription_and_analysis_of_live_streaming_on_Twitch.

[47] B. Bankov, "The Impact of Social Media on Video Game Communities and the Gaming Industry", 2019. Disponible en: https://www.researchgate.net/publication/337144821_The_Impact_of_Social_Media_on_Video_Game_Communities_and_the_Gaming_Industry.

[48] E.A. Andreeva, "A modern english internet slang.leetspeak". Disponible en: https://elib.sfu-kras.ru/bitstream/handle/2311/18331/s19_001.pdf?sequence=1.

[49] "Leet Speak Cheat Sheet", gamehouse.com. Disponible en: https://www.gamehouse.com/blog/leet-speak-cheat-sheet/.

[50] "An Explanation of l33t Speak", h2g2.com. Disponible en: https://h2g2.com/edited_entry/A787917.

[51] J. M. Pérez, D. A. Furman, L. Alonso Alemany, y F. M. Luque, "RoBERTuito: a pre-trained language model for social media text in Spanish", en Proceedings of the Thirteenth Language Resources and Evaluation Conference, 2022, pp. 7235–7243. Disponible en: https://aclanthology.org/2022.lrec-1.785/.

[52] J. M. Pérez, D. A. Furman, L. A. Alemany, y F. Luque, "RoBERTuito: a pre-trained language model for social media text in Spanish", arxiv.org, 2021. Disponible en: https://arxiv.org/abs/2111.09453.

[53] J. M. Pérez et al., "pysentimiento: A Python Toolkit for Opinion Mining and Social NLP tasks", arxiv.org, 2021. Disponible en: https://arxiv.org/abs/2106.09462.

[54] J. Herrero Llanos, "Análisis de sentimientos en Twitter mediante técnicas de Deep Learning", Universidad de Valladolid, 2022. Disponible en: https://uvadoc.uva.es/handle/10324/57316.

[55] pysentimiento: A Python multilingual toolkit for Sentiment Analysis and Social NLP tasks. Disponible en: https://github.com/pysentimiento/pysentimiento.

[56] "Tokenizer", Huggingface.co. Disponible en: https://huggingface.co/docs/transformers/main_classes/tokenizer.

[57] Token type ids, "Glossary", Huggingface.co. Disponible en: https://huggingface.co/docs/transformers/glossary#tokentype-ids.

[58] Servicios de Suscripción Google Colab. Disponible en: https://colab.research.google.com/signup.

[59] Anaconda. Disponible en: https://www.anaconda.com/.

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