Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática
(Mención en Computación)

Análisis emocional en redes sociales basados en modelos de aprendizaje automático transformers BERT

Autor:
D. Javier Estévez Asensio
Tutores:
D. Jesús Mª Vegas Hernández
Dña. Noemí Merayo Álvarez

Resumen

En la actualidad resulta de lo más habitual la comunicación por medios telemáticos y en especial por redes sociales. Este nuevo medio ha traído cambios en la forma de expresarse de las personas, por lo que su análisis resulta de gran interés.

A su vez se está viviendo una época de grandes avances en modelos de deep learning para su uso en tareas de procesamiento natural del lenguaje, gracias en gran medida a los modelos transformers, los cuales permiten un análisis más profundo del contexto del mensaje.

En este trabajo se va a explorar la librería transformers desarrollada por la comunidad de inteligencia artificial Hugging Face, analizando modelos utilizados para el análisis de la polaridad y emociones con distintos conjuntos de datos, logrando precisiones incluso superiores al 95 % en el análisis de polaridad.

Los resultados obtenidos se compararán con otros modelos utilizados con el mismo fin para analizar sus ventajas y diferencias.

Abstract

Currently, communication through telematic means, especially through social networks, has become extremely common. This new medium has changed the way people express themselves, making its analysis of great interest.

Simultaneously, we are experiencing a period of great advances in deep learning models for natural language processing tasks, largely due to transformer models. These models allow for a deeper analysis of the message context.

This study aims to explore the transformers library developed by Hugging Face artificial intelligence community. We will analyze models used for polarity and emotion analysis using various datasets, achieving accuracy values surpassing 95 % in polarity analysis.

The obtained results will be compared with other models used for the same purpose to assess their advantages and differences.

Agradecimientos

Agradezco a mi familia, por brindarme una educación completa y apoyarme desde el primer día hasta el último.

A mis tutores, por permitirme realizar este TFG junto a ellos y ayudarme con el mismo.

Y por último, me gustaría agradecer a la promoción 2018/2023 de InDat por unos años maravillosos junto a ellos.

Tabla de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

La necesidad de comunicarnos de forma online está muy presente en la vida cotidiana. Esta necesidad se vió reforzada durante el periodo de cuarentena a la que se vió obligada la población debido a la pandemia provocada por el virus *COVID-19* [1].

La comunicación a través de redes sociales tiene serias limitaciones dadas por una longitud de mensajes limitada en ocasiones, la falta de comunicación no verbal y la pérdida del tono utilizado en conversaciones cara a cara. Estos problemas se han intentado suplir aportando a la comunicación por redes matices propios, pero aún así es habitual la malinterpretación del mensaje transmitido.

Es por ello que resulta de gran interés encontrar un clasificador capaz de aprender de estas características propias de las redes e interpretar con gran precisión los sentimientos que se pretenden transmitir al escribir un mensaje.

La inteligencia artificial ha realizado grandes avances dentro del procesamiento natural del lenguaje (NLP por sus siglas en inglés) por lo que resulta un campo de estudio ideal para el trabajo propuesto. En concreto se utilizará un tipo de modelo basado en transformers [2], los cuales han supuesto un gran avance en los años recientes en aplicaciones NLP, dando lugar a aplicaciones aclamadas a nivel mundial por su sofisticación. [3]

1.2. Objetivos

Los objetivos marcados en este trabajo fin de grado son el análisis y aplicación de modelos basados en transformers [2], concretamente los modelos BERT [4] desarrollados por un grupo de investigadores del grupo Google AI para la clasificación de sentimientos y emociones con diferentes conjuntos de datos de dos de las redes sociales más utilizadas a nivel mundidal como son Twitter [5] y Twitch [6].

Se utilizarán también los resultados de este trabajo para comparar la precisión de estos clasificadores con aquellos propuestos mediante el uso de diversas técnicas de inteligencia artificial en trabajos anteriores.

1.3. Metodología

Se ha dividido el proyecto en diferentes fases según los objetivos propuestos de cada una de ellas, que se comentan a continuación.

1.3.1. Documentación

Durante esta fase se ha realizado un proceso de aprendizaje sobre el problema abordado, incluyendo conceptos sobre inteligencia artificial ya vistos durante los estudios de grado y otros nunca vistos anteriormente como la arquitectura transformers y el modelo BERT [4].

También se incluye en esta fase la lectura y ejecución del trabajo fin de grado realizado por Jesús Herrero Llanos [7] el año pasado, de temática similar, ya que este trabajo se puede considerar una continuación de su trabajo para compararlo con nuevas técnicas de inteligencia artificial.

Por último se realizan tareas de documentación sobre la librería Hugging Face Transformers [8], ya que será utilizada para la ejecución de las pruebas de este trabajo.

1.3.2. Análisis

En esta fase se realiza una búsqueda de aquellos modelos más interesantes publicados en la comunidad Hugging Face para la tareas de análisis de sentimientos y emociones en español y se analizan sus diferencias.

1.3.3. Pruebas

A lo largo de esta fase se realizan las diferentes pruebas que permitan la comparación de los distintos modelos entre sí, así como con otros modelos de machine learning desarrollados para los mismos objetivos.

1.3.4. Redacción de la memoria

La escritura del informe de este trabajo, incluyendo su planificación, contenidos y conclusiones finales se realiza durante esta fase.

1.4. Estructura de la memoria

Los contenidos de esta memoria se organizan en el siguiente orden:

- Introducción, en el que se proporciona información acerca de la motivación y los objetivos del trabajo propuesto en este proyecto.
- Planificación, en el que se aborda la asignación de tiempo a las distintas tareas, estimación de presupuesto y prevención de riesgos.
- Estado del arte, donde se pretende dar una visión general de las tecnologías y ramas del conocimiento relacionadas con el TFG.
- Herramientas y metodología, donde se exponen la información teórica acerca de las métricas de rendimiento utilizadas para evaluar los modelos y las herramientas utilizadas para la ejecución del proyecto.

- Análisis de polaridad, capítulo donde se habla de los contenidos más prácticos del trabajo, incluyendo el análisis descriptivo del conjuntos de datos y los procesos realizados para ejecutar las pruebas con sus correspondientes resultados.
- Análisis de emociones, con contenidos similares a los mencionados en el capítulo anterior para la tarea de clasificación de emociones.
- Conclusiones, donde se comentan las conclusiones del trabajo realizado, así como dificultades encontradas y líneas de trabajo futuras.
- Anexos, apartado en el que se presenta información adicional (que ya se verá si es necesario)

Capítulo 2

Planificación

2.1. Planificación

Con el fin de asegurar un correcto desarrollo del proyecto se ha propuesto una metodología de trabajo ya mencionada en el capítulo anterior, la cual se detalla en este capítulo.

Para una correcta planificación es necesario realizar un diagrama de secuencia que incluya un desglose completo de las tareas que conformarán el proyecto, incluyendo sus fechas propuestas de inicio y duración y si hay restricciones entre ellas.

La fecha de inicio de este trabajo fin de grado se supone el 16 de enero, fecha en la que se realiza la reunión con los tutores que da inicio al proyecto, y se estima su final el 15 de junio, con tiempo suficiente para realizar el depósito del trabajo.

Se ha utilizado la herramienta GanttProject [9] para la realización del diagrama de Gantt para este trabajo:

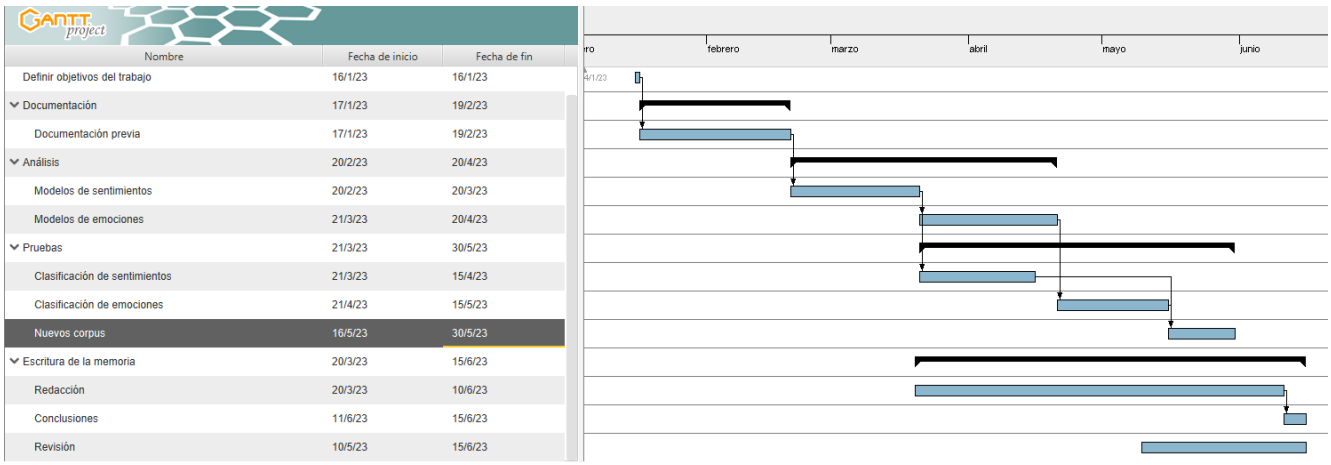

Figura 2.1: Diagrama de Gantt del proyecto

Como se ve en la figura 2.1 existen tareas con relaciones fin-inicio como es el caso de las pruebas. Esto se debe a que aunque al usar Google Colab se podía disponer de recursos suficientes para ejecutar pruebas para análisis de sentimientos y emociones simultáneamente, existe una limitación de sesiones activas para ejecutar código.

La tarea de escritura de la memoria se trata de una tarea hamaca ya que puede realizarse durante el trabajo en las distintas fases a medida que se recopile información suficiente para ello.

2.2. Presupuesto

Según la planificación del apartado anterior el proyecto tiene una duración estimada de 5 meses, a lo largo de los cuales pueden surgir costes que deben ser previstos dentro del presupuesto del proyecto.

El principal coste del proyecto es el de contratar a un trabajador que cuente con características de un ingeniero de datos, un perfil de trabajador que cuenta con grandes conocimientos en el área de la Inteligencia Artificial.

De acuerdo a la sección 6 de la guía docente de un Trabajo de Fin de Grado [10] se puede suponer una duración de proyecto de 300 horas. Mediante una consulta en webs especializadas tenemos que un trabajador con el perfil demandado percibe un salario promedio de 15,06€/h [11], lo que hace un gasto total de 4.518,00€ de salario al trabajador.

Como gastos materiales, se debe contar con un equipo con unas características no muy exigentes ya que todas las pruebas se realizarán en servidores en la nube, por lo que no se utilizan los recursos de nuestro propio ordenador. Suponemos el coste de un ordenador con estas características en 500€.

A el gasto anterior se puede agregar el coste de electricidad necesario para su funcionamiento. Se ha estimado que durante las 300 horas se pueda realizar un consumo en torno a 300W. Debido a los acontecimientos recientes este recurso se ha visto afectado por grandes fluctuaciones de precio, llegando a tener diferencias notables de precio en distintos momentos del día. Tomando el precio promedio para consumidores adscritos a la tarifa regulada PVPC de electricidad en España [12] en el mes de Febrero, de 0,206025€ kw/h, se tiene un coste estimado de 18,5423€ en electricidad.

Por último, los servidores mencionados tienen una serie de limitaciones en cuanto a recursos disponibles. En el caso de la plataforma a utilizar [13] estas limitaciones se pueden evitar mediante la compra de unidades de cómputo o suscripción mensual de un plan de trabajo mejorado, incluyendo mejores GPU con disponibilidad asegurada, mayores tiempos de ejecución, más disco y memoria principal.

La mensualidad de una suscripción al servicio Pro tiene un coste de 11.99€/mes. En caso de ser necesaria se pueden estimar un mínimo de 2 meses de uso de acuerdo al diagrama de Gantt 2.1 usado en la planificación del proyecto, ya que son las tareas del apartado de pruebas las más dependientes de la potencia de cómputo dentro del proyecto.

Como presupuesto total del proyecto se estima un gasto de 5.036,54€ fijos y 23,98€ posibles en concepto de potencia de cómputo adicional.

2.3. Riesgos

A lo largo de todo proyecto existe la posibilidad de sufrir contratiempos que afecten a la planificación realizada para parte o la totalidad del proyecto, suponiendo retrasos no previstos y

perjudicando a la correcta ejecución del proyecto.

Se ha realizado en la tabla 2.1 un de análisis de posibles riesgos, indicando la probabilidad de ocurrencia y el impacto que supondrían en el proyecto de forma cualitativa con 3 posibles valores: bajo, medio y alto.

En la última columa se cuenta con posibles soluciones o métodos para evitar el riesgo. Alguna de ellas ha sido ya mencionada en el apartado de costes, suponiendo el coste variable por falta de potencia computacional.

Referencia	Riesgo	Probabilidad	Impacto	Gestión
R1	Estimación insuficiente de tiempos	Media	Medio	Realizar una correcta planificación del proyecto
R2	Falta de conocimientos	Alta	Alto	Realizar una correcta documentación previa
R3	Recursos hardware insuficientes	Media	Medio	1. Asumir un contratiempo en la etapa de clasificación 2. Pago de suscripción en el servicio informático contratado
R4	Dedicación a otros proyectos	Media	Medio	Realizar una correcta planificación del proyecto
R5	Enfermedad del trabajador	Baja	Alto	Asumir el contratiempo
R6	Ampliación de los contenidos del proyecto	Media	Medio	Realizar una correcta planificación del proyecto
R7	Fallo de los servicios informáticos	Media	Alto	Asumir el contratiempo
R8	Selección del trabajador para procesos electorales	Baja	Alto	Asumir el contratiempo

Tabla 2.1: Análisis de riesgos del proyecto

En la mayoría de supuestos se procura gestionar el riesgo evitándolo mediante una correcta planificación o ejecución de tareas predecesoras.

En el caso de los riesgos R5 y R8 es difícil encontrar una solución para evitar el riesgo en primer lugar.

Con respecto al riesgo R7 cabría la posibilidad de mantener una copia local de los documentos para su ejecución en un equipo local, pero dadas las especificaciones del equipo informático planificado y las limitaciones en cuanto a hardware que supone el problema a tratar no se trata de una solución viable.

Capítulo 3

Estado del arte

3.1. Inteligencia artificial

La inteligencia artificial o AI por sus siglas en inglés es un campo de la ciencia de datos que combina la informática, la estadística y la ciencia cognitiva para elaborar sistemas capaces de resolver problemas complejos buscando emular los procesos cognitivos presentes en los seres humanos [14].

Para ello la inteligencia artificial generalmente parte de un conjunto de datos con los que se genera un aprendizaje previo, y debe ser capaz de aprender de la experiencia para adaptarse a nuevas situaciones.

Los objetivos de la inteligencia artificial son diversos, entre los que se encuentran tareas más sencillas de clasificación hasta reconocimiento de imágenes, detección de idiomas, procesado del lenguaje natural.

Existen dos enfoques de la inteligencia artificial según el alcance de sus objetivos: [15]

- La IA débil se especializa en un dominio de trabajo concreto, como el reconocimiento de imágenes.
- La IA fuerte busca un objetivo más ambicioso como el de igualar o mejorar las capacidades de la inteligencia humana de forma parcial o total.

El origen de la inteligencia artificial puede suponerse en los sistemas de clasificación basados en reglas, incapaces de aprender de la experiencia previa.

Desde entonces se realizan múltiples avances en el campo con hitos como la creación de la neurona de McCulloch Pits en 1943 en un intento de recrear el comportamiento de una neurona biológica; el perceptrón de Frank Rosenblatt en 1957, capaz de aprender de la experiencia previa; las redes neuronales recurrentes de Hopfield en 1982 o las redes LSTM de Hochreiter en 1997 [16].

3.2. Inteligencia artificial en la actualidad

Actualmente se está viviendo una auténtica revolución en el campo de la inteligencia artificial, proporcionando utilidades impresionantes como el asistente ChatGPT [17], herramientas para recrear la voz de una persona a partir de solo unas pocas palabras suyas [18], o crear imágenes a

partir de textos [19].

Esta explosión de proyectos se ha visto beneficiada por permitir el acceso al público general a sus productos, brindando la posibilidad de ganar popularidad a la vez que se obtiene una nueva fuente de entrenamiento para versiones posteriores del modelo.

Sin embargo su popularización también ha supuesto la aparición de nuevos métodos de fraude o la sofisticación de otros, como la suplantación de identidad mediante voz durante una llamada en tiempo real o el *deepfake* de celebridades o ejecutivos para la transmisión de *fake news* [20].

Por otro lado se encuentran denuncias como las de artistas que han visto como se ha utilizado su contenido sin consentimiento para el entrenamiento de IAs para generar arte [21]. Y es que como se ha comentado anteriormente estos modelos requieren de grandes conjuntos de datos para su entrenamiento antes de poder ser lanzados como producto.

La presentación pública de ChatGPT ha logrado posicionarlo como muchos como una herramienta de asistencia imprescindible en el día a día incluso en el ámbito laboral, y con un futuro tan prometedor se teme que nuevos modelos sean capaces de sustituir miles de puestos de trabajo realizados por seres humanos.

Es por ello que diversas organizaciones están estudiando la creación de nuevas normativas con el objetivo de regular las capacidades y limitaciones de estos modelos de inteligencia artificial, como es el caso del AI Act [22] propuesto por la Unión Europea.

Si se mira lo ocurrido en perspectiva, estamos viviendo el nacimiento de un nuevo hito en la historia al igual que como ocurrió con la popularización de Internet, por lo que la falta de legislación al respecto propicia estos malos usos. En un futuro cercano la IA servirá de ayuda en tareas sanitarias y por el bien social [23].

3.3. Machine learning

Una rama del conocimiento dentro del campo de la inteligencia artificial se trata del aprendizaje automático o machine learning [24], que busca desarrollar modelos de clasificación que permitan a las máquinas aprender y mejorar a partir de datos, generando patrones para la resolución de tareas para las que no han sido programados específicamente.

Existen distintas técnicas de entrenamiento dentro del machine learning, las cuales dependerán de los objetivos buscados:

- Aprendizaje supervisado, en el que el modelo se entrena con ejemplos con salidas conocidas, por lo que el entrenamiento se basa en la generación de respuestas correctas en relación a las entradas suministradas.

- Aprendizaje no supervisado, donde se proporcionan ejemplos de entrada y no de salida o datos no etiquetados. De esta forma se busca descubrir nuevos patrones o relaciones desapercibidas entre los datos.

3.4. Deep learning

En los últimos años se ha realizado una diferenciación dentro del aprendizaje automático, recibiendo el nombre de deep learning o aprendizaje profundo [24].

El deep learning busca la generación de conocimiento mediante la simulación del pensamiento humano a través de redes neuronales compuestas por múltiples capas.

Como se ilustra en la figura 3.1, una de las diferencias más importantes con respecto al aprendizaje automático es que el aprendizaje profundo es aprendizaje no supervisado. A diferencia del machine learning no requiere de una extracción de características de los datos, proceso elaborado previamente por un ser humano.

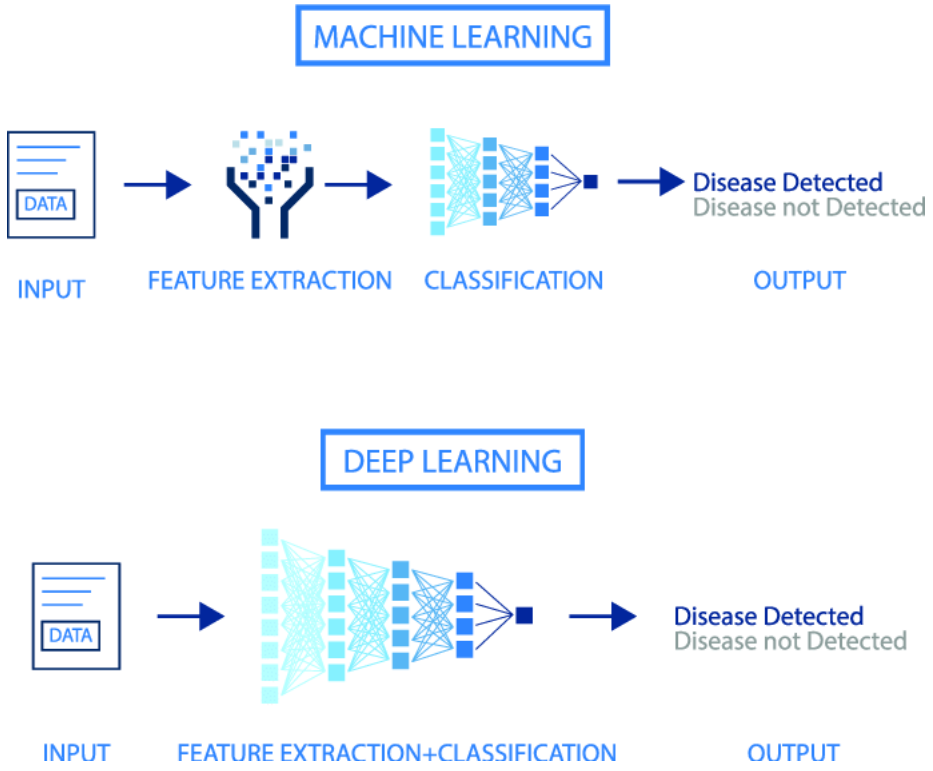

Figura 3.1: Machine learning vs Deep learning - Machine Learning and Deep Learning Approaches for Brain Disease Diagnosis: Principles and Recent Advances, página 3, figura 2. - ResearchGate [25].

El entrenamiento de estos modelos se realiza mediante la configuración de parámetros de entrenamiento básicos y la introducción de grandes cantidades de datos para que el sistema aprenda por sí mismo patrones a través de sus capas de procesamiento.

Al agregar múltiples capas los modelos de deep learning son capaces de aprender representaciones jerárquicas y complejas, haciéndolas adecuadas para el modelado de los problemas más complejos.

Se puede por lo tanto realizar una subdivisión de las diferentes áreas dentro de la inteligencia artificial mediante el diagrama de la figura 3.2:

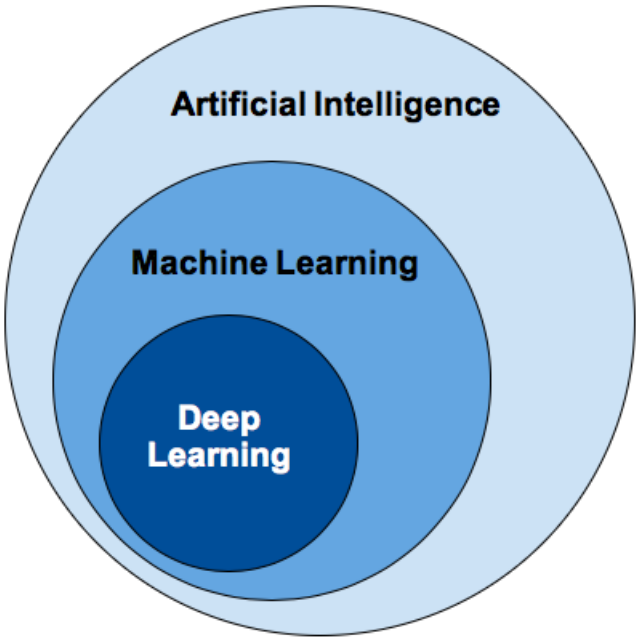

Figura 3.2: Jerarquía de la inteligencia artificial - Artificial Intelligence vs. Machine Learning vs. Deep Learning: What's the Difference? [26].

3.5. Redes neuronales

Como se ha comentado en el apartado anterior las redes neuronales surgen de aplicar un nuevo paradigma de trabajo a los algoritmos utilizados en machine learning. Mediante el uso de una red de capas de neuronas se crea información que se usa a su vez en las siguientes capas de entrenamiento, logrando elaborar reglas de clasificación mucho más complejas.

Una red neuronal conta principalmente de los siguientes elementos:

- Datos de entrada, los cuales se proporcionan a la red junto a sus respectivas salidas esperadas para poder realizar predicciones y medir su capacidad de predicción.
- Capas, las cuales reciben las salidas de la capa anterior y transmiten sus salidas para ser recibidas por la capa posterior. Destacan las capas primera y última por recibir los datos de entrada brutos y proporcionar las predicciones finales. El resto de capas se denominas capas ocultas y su número es variable.
- Función de activación, usada para transmitir la salida de una capa hacia la siguiente. Devuelve un valor acotado a partir de un valor de entrada.
- Optimizador, el cual se encarga de optimizar los pesos de los parámetros mediante el cálculo del gradiente de la función de coste para cada parámetro. Al buscar minimizar el error los pesos se modifican en la dirección contraria al gradiente. A este algoritmo se le denomina propagación hacia atrás.
- Función de pérdida, la cual se encarga de evaluar la desviación entre las predicciones de la red y las salidas esperadas. Valores menores corresponden a una mejor eficiencia de la red.

El entrenamiento de una red neuronal consiste en buscar valores de los pesos de las neuronas con el objetivo de lograr salidas iguales a las proporcionadas en el entrenamiento.

Para ello los pesos son modificados iterativamente mediante el algoritmo de propagación hacia atrás, generalmente multiplicando el vector del gradiente por un parámetro de entrenamiento llamado tasa de aprendizaje para agilizar la convergencia del modelo. En términos matemáticos se tiene:

$$W_{t+1} = W_t - f \times lr \tag{3.1}$$

Donde f es la función del optimizador y lr es la tasa de aprendizaje.

En la figura 3.3 se muestra el proceso que aplica el método del descenso del gradiente y su relación con el entrenamiento de un modelo.

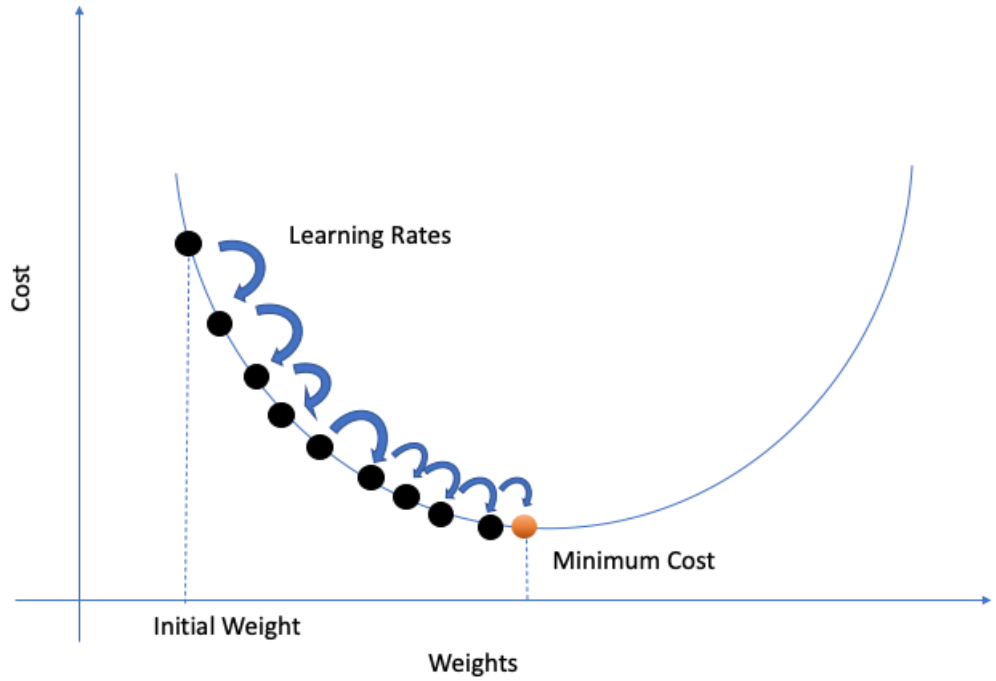

Figura 3.3: Método del descenso del gradiente - An Empirical Analysis of Generative Adversarial Network Training Times with Varying Batch Sizes, página 4, figura 5. - ResearchGate [27]

Para este trabajo se ha utilizado el optimizador Adam [28], una extensión del método del descenso del gradiente estocástico, por ser el que mejor resultados obtuvo en un TFG anterior con relación a este trabajo [7].

3.5.1. Transformers

Los transformers son una arquitectura de redes neuronales introducida por primera vez en el artículo *Attention Is All You Need* [2] en 2017 por Vaswani et al. suponiendo una revolución en el ámbito del procesamiento del lenguaje natural.

Una de las características más destacadas de esta arquitectura es su mecanismo de atención, el cual le permite utilizar la información de contenidos relevantes en el texto de entrada a la hora de generar la salida, sin importar lo lejos que se encuentren dichos contenidos entre sí. En algoritmos generativos de texto, al generar una palabra cada vez es capaz de utilizar sus propios resultados

como datos de entrada, dotando al resultado final de más coherencia.

En la figura 3.4 se puede ver como a partir del texto *"As aliens entered our planet"* se genera una continuación de la historia palabra por palabra, y las palabras a las que se presta atención para la generación de la siguiente.

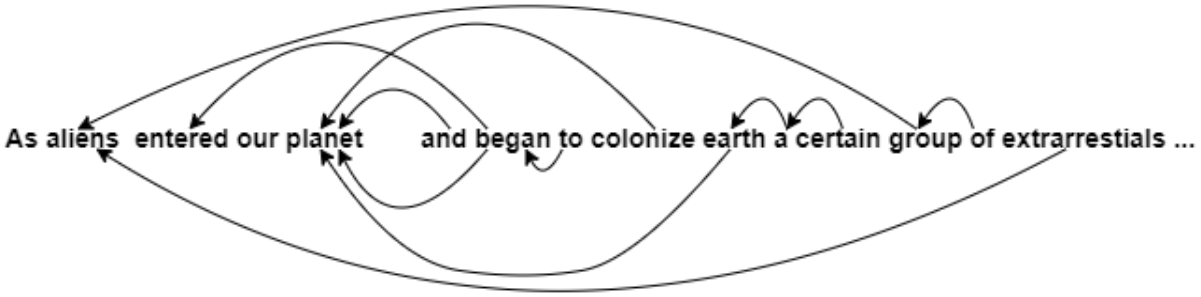

Figura 3.4: Mecanismo de atención de un texto generativo - Illustrated Guide to Transformers- Step by Step Explanation, - TowardsDataScience [8].

Las redes neuronales recurentes (RNN por sus siglas en inglés) también poseen capacidades de "atención" a inputs anteriores, pero el mecanismo de atención no sufre de pérdida de memoria, por lo que en el supuesto de la historia anterior, a medida que la historia se alargue, llegará un momento a partir del cual la ventana de inputs de los cuales una red RNN podría tomar referencias sería parcial, mientras que para los transformers seguiría siendo completa [8].

En la figura 3.5 se muestra la estructura habitual de un modelo transformers:

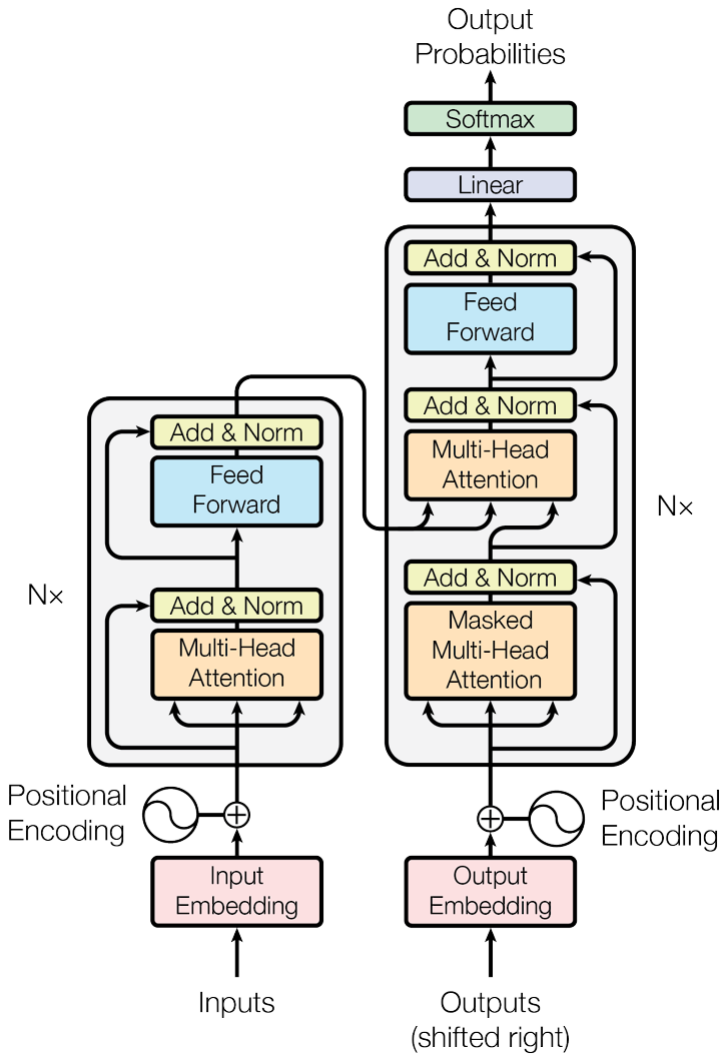

Figura 3.5: Estructura de un modelo transformers - Attention Is All You Need, página 3, figura 1 [2].

Podemos realizar un desglose de dicha estructura en los siguientes elementos:

- Input embedding, se encarga de crear un vector de representación para cada palabra de los datos de entrada para presentar los datos a la red en forma de números. Se pueden tomar estos vectores como una tabla de referencias. A este elemento también se le conocerá como tokenizador en el ámbito de este trabajo.

- La capa codificadora, compuesta por la agrupación del margen izquierdo, cuyo objetivo es crear una representación continua a partir de la secuencia de entradas de forma que contenga toda la información de atención aprendida relativa a la secuencia completa.
  Esta capa se puede repetir varias veces para obtener nuevas representaciones de atención para potenciar los resultados.

- La capa decodificadora, formada por la agrupación del margen derecho, parte de un token de inicio junto a ciertas palabras de la secuencia de entrada y las salidas de la capa codificadora para generar un conjunto de salidas.
  Estas pasan por una capa lineal que actua como un clasificador con un numero de salidas igual al número de palabras disponibles en el modelo. Estas salidas pasan por una capa

softmax final, obteniendo probabilidades para cada uno de los resultados. La salida con mayor probabilidad será la predicción del modelo.

Estos modelos han superando ampliamente enfoques anteriores en la aplicación de multitud de tareas, entre las que destacan la generación de texto, predicción de palabras ocultas, traducción, etiquetado de entidades y clasificación de sentimientos, tarea en la que se centra este trabajo.

A partir de este modelo han surgido extensiones como *Generative Pre-trained Transformer* o GPT, utilizado en el ya mencionado y popular asistente ChatGPT [17], T5 (Text-to-Text Transfer Transformer) [29] y BERT, del que hablaremos a continuación.

3.5.2. BERT

Las Representaciones de Codificador Bidireccional de Transformadores en su traducción al español, son una extensión de transformers creado por investigadores de Google AI en 2018 [4].

Su característica más destacada es la aplicación de transformers bidireccionalmente, en la que el mecanismo de atención se aplica de izquierda a derecha a mayores de hacerlo convencionalmente de izquierda a derecha, obteniendo un contexto en ambos sentidos de la frase de entrada.

Respecto al resto de características no es tan diferente de otros como GPT. Esto fue intencional, ya que consideraron OpenAI GPT como el modelo más comparable a BERT y esto permitía ofrecer ciertas comparativas entre ambos modelos. Mientras que GPT se entrenó con el dataset BooksCorpus [30] de 800 millones de palabras y utiliza la misma tasa de entrenamiento para cualquier tarea específica, BERT se entrenó juntando el anterior dataset y la Wikipedia al completo y utiliza diferentes tasas de entrenamiento para cada tarea.

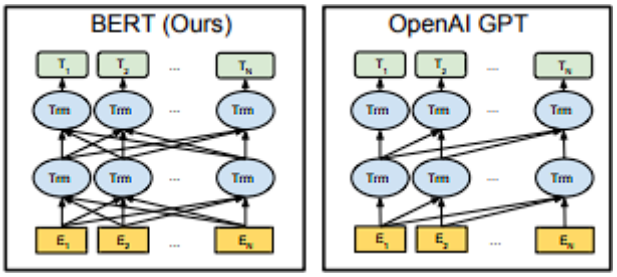

Figura 3.6: BERT vs OpenAI GPT - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, página 13, figura 3 [4].

Un gran punto a favor de BERT es la capacidad de adaptación a diferentes tareas de procesamiento natural del lenguaje. BERT se entrena con grandes cantidades de datos sin etiquetar para aprender representaciones de palabras y frases. Tras esto el modelo puede entrenarse con conjuntos de datos etiquetados para realizar un ajuste o fine-tuning hacia la tarea deseada.

En este trabajo usaremos modelos creados a partir de BERT o derivados publicados en la comunidad de inteligencia artificial Hugging Face [31].

Capítulo 4

Herramientas y metodología

4.1. Herramientas utilizadas

Para el desarrollo del proyecto se han utilizado las herramientas que se expondrán a continuación.

4.1.1. Python

Python [32] es un lenguaje de programación de alto nivel y fácil uso, lo que ha propiciado el desarrollo de multitud de librerías en materia relacionada con la ciencia de datos.

Es debido a los motivos mencionados anteriormente que Python es ha convertido en uno de los lenguajes de programación más utilizados para la investigación y creación de modelos de inteligencia artificial.

4.1.2. Wandb

Wandb [33] es una plataforma que permite monitorizar búsquedas de hiperparámetros, métricas del sistema y predicciones para comparar modelos en tiempo real.

Es utilizado por grandes empresas dedicadas a investigación para la automatización de procesos de entrenamiento y elaboración de reportes.

Permite el uso de su API de forma gratuita creando una cuenta.

4.1.3. Google Colab

Plataforma usada para ejecutar el código utilizado en este trabajo. Proporciona un entorno de ejecución dinámico en la nube con acceso gratuito limitado a GPUs con la que editar cuadernos Jupyter.

Existe la posibilidad de pagar una suscripción mensual con la que acceder a una mayor variedad de GPUs, CPUs mas rápidas y mayor capacidad de memoria RAM y disco.

4.1.4. Overleaf

Editor LaTex [34] online gratuito utilizado para escribir, editar y publicar artículos científicos. Overleaf [35] permite trabajo compartido y simultáneo de hasta 2 personas y un historial de cambios en la versión gratuita. Se ha utilizado para escribir esta memoria TFG.

4.1.5. Twitter

Twitter [5] es una red social de microblogging en la que los usuarios pueden compartir sus pensamientos mediante mensajes cortos llamados tweets en las feeds de las personas seguidoras de su perfil.

Twitter ha sido utilizado como medio de extracción de información y datos para el entrenamiento de modelos de procesado natural del lenguaje dada su gran popularidad entre la población y su fácil extracción mediante su API de libre uso.

Tras la reciente compra por parte de Elon Musk [36] la API dejará de ser gratuita al público general, pero no afecta al desempeño de este proyecto ya que se usarán dos conjuntos de datos extraídos con anterioridad.

Los elementos diferenciales de Twitter con respecto a otras webs son:

- Hashtag, representados por # junto a un mensaje que indica una tendencia. Permite el filtrado de tweets por temática.
- Retweet, redifusión del tweet de otro usuario a tus seguidores.
- Mención, nombre de otro usuario de la red precedido de @. Forma de referirse a otro usuario al escribir un tweet. También hace que dicho usuario reciba una notificación personal para enterarse de su contenido.

4.1.6. Twitch

Twitch [6] es la mayor plataforma de retransmisión en directo de contenido relacionado con videojuegos, deportes y entretenimiento entre otros perteneciente a Amazon [37].

Cuenta con chat en directo para permitir la interacción de los espectadores con los creadores de contenido y entre ellos, por lo que ha sido utilizado para extraer el contenido de los mensajes compartidos mediante su API para realizar un análisis de sentimientos y emociones en el ámbito de los videojuegos.

4.1.7. Hugging Face Transformers

Subdivisión dentro de la gran comunidad dedicada a la inteligencia artificial Hugging Face en la que se ofrece una librería completa para experimentar con diversos modelos relacionados basados en transformers.

Incluye documentación para el uso de modelos para predicción y entrenamiento de nuevos modelos y la posibilidad de subirlos a la web para su uso público.

Actualmente cuenta con modelos para tareas diversas agrupadas en cuatro conjuntos:

- Procesamiento natural del lenguaje: análisis de sentimientos, reconocimiento de entidades nombradas, traducción, respuesta a preguntas, predicción de palabras en un texto (modelado del lenguaje), resumen de texto, respuesta múltiple y generación de texto en el procesamiento natural del lenguaje.
- Visión computacional: clasificación de imágenes, detección de objetos y segmentación de imágenes.
- Audio: reconocimiento de la lengua hablada y clasificación de audio.
- Otras: respuesta a preguntas basadas en información presente en tablas, conversión de imagen a texto, extracción de información de documentos escaneados, clasificación de vídeo y respuesta a preguntas respecto a una imagen.

4.2. Métricas de rendimiento

Para evaluar la capacidad predictiva de los modelos y poder realizar una comparativa entre ellos se utilizan una serie de medidas objetivas utilizadas en clasificación.

Suponiendo un clasificador de emociones que indica si hay odio en un texto o no, existen cuatro posibles resultados al realizar una predicción en función de su resultado y la salida esperada:

- Verdadero positivo (TP) cuando el resultado es positivo y la salida esperada también.
- Falso positivo (FP) cuando el resultado es positivo y la salida esperada no.
- Falso negativo (FN), caso contrario al falso positivo.
- Verdadero negativo (TN), cuando el resultado es negativo y la salida esperada también.

Si se realizara una matriz de confusión de las posibles salidas de este clasificador las posibilidades quedarían situadas según la figura 4.1

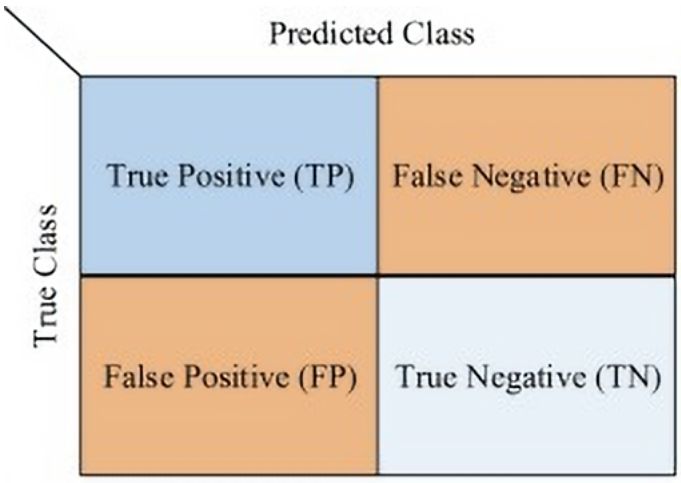

Figura 4.1: Matriz de confusión - Artificial Intelligence-Based Brain-Computer Interface, capítulo 14, figura 7. - ScienceDirect [38].

Las medidas de precisión utilizadas en este trabajo dependen de estos resultados y son las siguientes:

- Accuracy: mide la razón entre las clasificaciones correctas y los resultados totales.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.1}$$

- Precision: razón entre el número de instancias correctamente clasificadas y aquellas instancias clasificadas como positivas, sin importar su resultado real. Se calcula para cada posible clase de la salida.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

- Recall: razón entre el número de instancias correctamente clasificadas y el número de instancias realmente pertenecientes a la clase.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

- F_1-score: perteneciente a la familia de los F-score hace uso de precision y recall y viene dado por la fórmula:

$$F_\beta\text{-}score = (1 + \beta^2)\frac{precision \times recall}{(\beta \times precision) + recall} \tag{4.4}$$

  Esta fórmula depende de un parámetro β variable, que en nuestro caso es igual a 1, por lo que se tiene:

$$F_1\text{-}score = 2\,\frac{precision \times recall}{precision + recall} \tag{4.5}$$

Capítulo 5

Análisis de polaridad con modelos de BERT

5.1. Introducción

A lo largo del trabajo se han realizado dos tipos de análisis: de polaridad y de emociones.

En este capítulo nos centramos en la tarea de clasificación de sentimientos, al cual corresponde al problema de clasificación de textos.

La clasificación de textos puede realizarse a distintos niveles, entre los siguientes [39]:

- Nivel de documento: se analiza el sentimiento de un documento de texto en su totalidad.
- Nivel de oración: se divide el documento en oraciones y se extrae la polaridad de cada una de ellas por separado.
- Nivel de aspecto y entidad: donde una entidad está formada por distintos elementos o aspectos y sobre cada uno de ellos se expresa una opinión cuya polaridad puede ser distinta en cada caso. Es el caso más complejo de analizar, pero también el más específico.

En los conjuntos de datos analizados cada instancia solo puede tomar una polaridad, por lo que nuestro problema se trata de un problema de clasificación multiclase simple a nivel de documento, donde cada documento sería una instancia ya que no pertenecen a un conjunto coherente mayor.

Existen dos técnicas en el proceso de creación de una red neuronal. La creación mediante diccionarios predefinidos consiste en utilizar incrustaciones de palabras preentrenadas para ajustar los pesos de la red de forma general. La otra técnica es realizar el ajuste de pesos durante el preentrenamiento de la red utilizando datos de entrenamiento del propio corpus.

Esta última es la técnica utilizada en el trabajo, tanto para el problema de este capítulo como del siguiente, ya que así la red aprenderá de datos más específicos para la resolución de nuestro problema [40].

5.2. Análisis descriptivo de los dataset

Para esta tarea utilizamos tres conjuntos de datos con características distintas.

5.2.1. Twitter

Se trata del mismo conjunto de datos utilizado en el TFG de Jesús Herrero Llanos [7], obtenido por la concatenación de los conjuntos de datos utilizados por TASS (Taller de Análisis Semántico) a lo largo de varios años.

Al igual que en el desarrollo de su trabajo eliminaremos las 12.000 últimas instancias para poder ofrecer una comparativa más directa.

Consiste por lo tanto de 60.161 instancias de las que constan dos campos de datos:

- En el primero se encuentra el texto que en este caso se trata de un tweet.
- En el segundo se encuentra la clase respuesta, referida al sentimiento o polaridad del texto, con cuatro posibles salidas: positivo, negativo, neutro y sin sentimiento.

Por motivos de balanceo de clases y simplificación se han fusionado las clases neutro y sin sentimiento en una sola.

Las muestras del conjunto de datos siguen una distribución según la variable respuesta como se puede ver en la figura 5.1, con 22021 positivos, 15748 negativos y 22392 neutros.

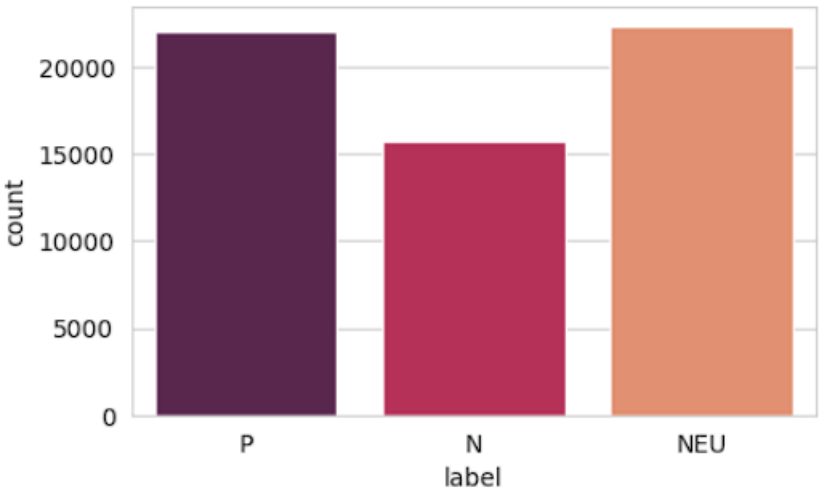

Figura 5.1: Conjunto de datos de Twitter para polaridad

5.2.2. Twitch

Este conjunto de datos surge a partir de una investigación de un grupo de investigadores de la Universidad de Valladolid y la Autónoma de Madrid en el que se busca realizar el análisis de polaridad en el ámbito de los videojuegos.

Los datos fueron extraídos en verano de 2022 y está formado por 2007 instancias con tres campos: el texto a analizar, su polaridad y la emoción mayoritaria que transmite el texto.

De estos tres campos nos interesan para el problema actual los dos primeros.
En la figura 5.2 vemos su división en 967 mensajes positivos, 822 negativos y 218 indeterminados.

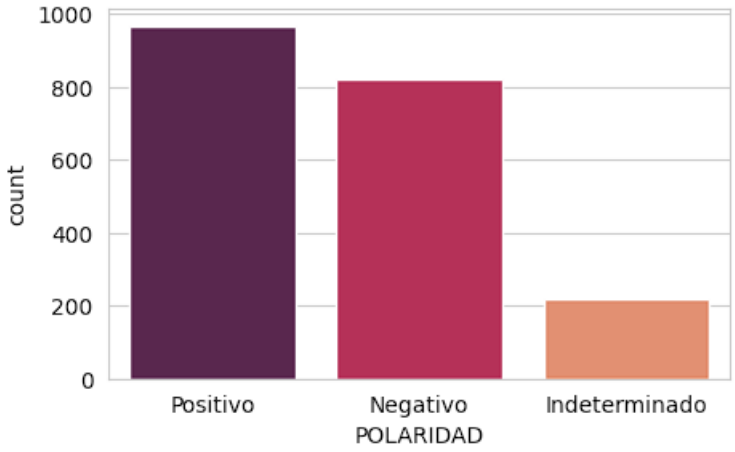

Figura 5.2: Conjunto de datos de Twitch para polaridad

5.2.3. Salud mental

Conjunto de datos extraído mediante la API de Twitter para el estudio del impacto de las redes sociales en la salud mental de las personas y estudiar la relación con personas influyentes dentro de ellas.

Contiene 2288 instancias con 7 campos: texto, emoticonos, polaridad, emociones, estigma, relación e influencer de los cuales utilizaremos el primero y tercero, cuyas descripciones coinciden con los de los anteriores conjuntos de datos.

Dos instancias dentro del conjunto fueron eliminadas por no haber sido etiquetadas.

La división por polaridad de los datos viene dada por 1526 textos positivos, 587 negativos y 173 indeterminados.

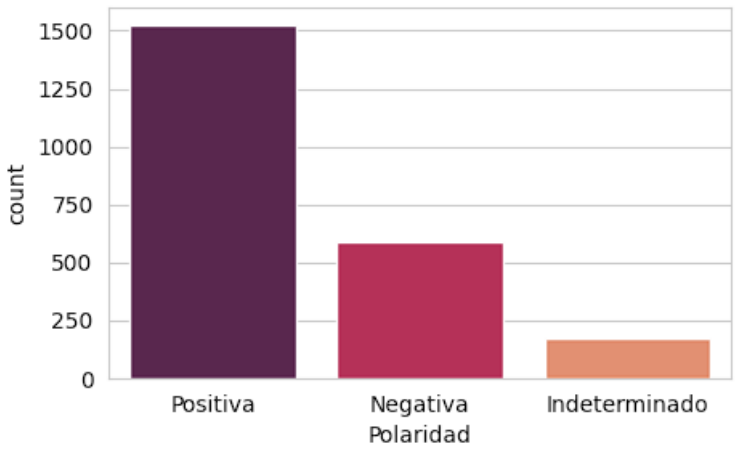

Figura 5.3: Conjunto de datos de salud mental para polaridad

5.3. Modelos utilizados

Los modelos utilizados para esta tarea han sido entrenados por un grupo de trabajo bajo el nombre de pysentimiento [41] y se compararán en la clasificación de sentimientos con dos salidas, *positiva* y *negativa*; y con tres salidas, añadiendo a las dos anteriores la clase *indeterminada*.

5.3.1. RoBERTa

El modelo RoBERTA [42] fue publicado por primera vez en 2019 y se trata de una aproximación a la optimización robusta del modelo BERT original [4], de ahí las nuevas siglas del modelo.

El enfoque de su desarrollo se basó en el análisis de las limitaciones de BERT y mejorar su desempeño en tareas de procesamiento del lenguaje natural.

Entre sus modificaciones está el uso de un número mayor de capas de codificación, aumentando tanto el número de parámetros como las dimensiones ocultas. Con este paso se busca obtener una representación aún más completa del lenguaje.

También se realizó un análisis en profundidad de los hiperparámetros óptimos para BERT, entre ellos el batch size, learning rate y los training steps.

Por último se utilizó una cantidad de datos de preentrenamiento muy superior al del modelo original para obtener una mejor comprensión del lenguaje. Junto a ello se le entrenó en una nueva tarea consistente en predecir si dos oraciones iban una detrás de la otra en un texto [43].

El modelo publicado en el repositorio Hugging Face se trata del modelo base fine-tuned para la clasificación de sentimientos de textos en español [44].

5.3.2. RoBERTuito

Se trata de un modelo creado en 2022 a partir de la arquitectura base del modelo RoBERTa, ccon un tamaño de dimensiones ocultas de 768 [45].

Para la elección de hiperparámetros se tuvo en cuenta las investigaciónes sobre hiperparámetros realizadas durante el desarrollo del modelo RoBERTa original [42] y BERTweet [46], modelo utilizado para la tarea de clasificación de tweets en inglés. Debido a las limitaciones de hardware parámetros como el batch size no pudo ser probado con el valor sugerido por RoBERTa, y tuvo que ser reducido de 8.000 a 4.000.

El modelo se entrenó durante tres semanas para la tarea de modelado del lenguaje (ocultar una palabra dentro de una frase y predecirla) con un corpus de cerca de 500 millones de tweets, donde la mayoría de ellos estaban escritos en español pero también había presencia de inglés y portugués.

Se entrenaron 3 variantes del modelo: una que mantiene las mayúsculas, uno que reduce todo a minúsculas y el último que reduce a minúsculas y elimina los acentos de las palabras.

En los resultados se comenta que no encontraron una mejora o empeoramiento significativo entre las dos últimas variantes, que sí lograron mejores resultados que el modelo que mantenía las mayúsculas.

El modelo subido al respositorio Hugging face [47] se trata de un modelo fine-tuned para la tarea de clasificación de polaridad utilizando el corpus TASS2000 [48] .

5.4. Preprocesamiento de datos

El preprocesamiento de los datos busca obtener una homogeneización de palabras, ya que la redacción de mensajes en la red suele ser más relajada y por lo tanto está sujeta a mayor variabilidad o faltas de ortografía.

Para el preprocesado se ha probado a realizar las mismas acciones propuestas por Jesús Herrero Llanos en su TFG dada la similitud del problema, que son:

- Normalización de mayúsculas y minúsculas: se reduce todas las palabras a minúsculas.
- Eliminación de tildes.
- Reducción de la repetición de caracteres.
- Normalización de las risas: a partir de más de dos repeticiones del monosílabo *ja* se considera la onomatopeya de la risa y se reduce a *jaja* para homogeneizar.
- Normalización de las jergas.
- Eliminado de menciones, hashtags, retweets y enlaces.
- Eliminación de signos de puntuación.

Tras realizar pruebas con y sin realizar preprocesado de los datos se vió que los resultados coincidían, dado que los modelos utilizados incluyen su propio preprocesado de los datos en caso de que el usuario no planteara la posibilidad de realizarlo.

Este preprocesado incluye la mayoría de los pasos anteriores, modificando algunos como la sustitución de hashtags por "hashtag", menciones por *@usuario* y enlaces por *"url"* o la sustitución de caracteres especiales o emojis por texto. En la figura 5.4 se muestra la definición de la función y documentación incluida para los modelos del grupo pysentimiento.

Como los resultados obtenidos eran iguales y el preprocesado incluído en los modelos abarca casi la totalidad de técnicas enumeradas anteriormente, se decidió introducir los textos sin modificar para utilizar el preprocesado de los modelos.

```
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
    """
```

Figura 5.4: Documentación de la función de preprocesado utilizada en los modelos creados por el grupo de trabajo pysentimiento [49].

5.5. Entrenamiento de un modelo

El entrenamiento de modelos es la fase clave para la creación o especialización de un modelo a un dominio.

Para ello se dispone de la clase *Trainer*, a la cual hay que suministrarla del modelo y unos datos correctamente preparados.

Para el entrenamiento con datos etiquetados es importante que los campos donde se encuentran el texto y la clase respuesta reciban los nombres *text* y *label* respectivamente por razones de diseño.

5.5.1. Tokenizado

Para entrenar un modelo sobre un nuevo conjunto de datos primero se deben preparar los datos para ser aceptados por el modelo.

Básicamente se trata de realizar las funciones de la capa codificadora tratada en el apartado 3.5.1

Para ello se utiliza la clase *Tokenizer* [50]. La librería Hugging Face tiene un tokenizer para cada modelo soportado.

Para facilitar las funciones de mapeado el conjunto de datos se divide en entrenamiento y test y se transforma a un dataset propio de la librería.

Al mapear los datos a una función de tokenizado proporcionada en la documentación, dependiente del tokenizer, se nos devuelven los siguientes elementos:

- Text: el mismo que en nuestros datos.
- Label: el mismo que en nuestros datos.
- Input_ids: contiene el texto tras identificar cada palabra igual del conjunto de datos con un número. (no se como explicarlo mejor)
- Token_type_ids: utilizado en tareas de respuesta a preguntas o modelado de lenguaje. Añade símbolos especiales como máscaras o separadores para adaptar la entrada a la tarea a realizar. En nuestro caso no se usa y por ello es una lista con todos los valores igual a cero [51].
- Attention_masks: indica al modelo sobre los tokens a los que se le debería prestar atención según el mecanismo de atención de los modelos BERT [2].

5.5.2. Entrenamiento

Una vez tenemos los datos convenientemente tratados es momento de utilizar la clase *Trainer*.

Previamente se declaran los parámetros de entrenamiento con la clase *TrainingArguments* junto con otros parámetros. Para este trabajo, además de los parámetros del apartado siguiente definimos:

- output_dir: directorio donde incluir archivos relativos al entrenamiento. Obligatorio declararlo.
- evaluation_strategy: indica bajo que hito se evalúa el modelo con el conjunto test.
  Se puede realizar al completar época o al completar batch y se puede especificar el número de épocas o batch antes de realizarlo.
  Para este trabajo se evalúa tras completar una sola época, ya que todos los datos de entrenamiento habrán pasado por el modelo.
- fp16: indica el uso de la precisión de punto flotante de 16 bits para mejorar el uso de memoria y la velocidad de entrenamiento.
  Necesario para poder utilizar tamaños de batch de hasta 128 y mejorar la velocidad de entrenamiento.
  Existen varias opciones, de las cuales tf32 parecía ser muy interesante, pero por limitaciones de arquitectura de las GPUs usadas no se pudo probar.
- load_best_model_at_end y metric_for_best_model: utilizadas en conjunto a un callback que se indica en la llamada a *Trainer* para realizar una parada temprana del modelo. La métrica utilizada para el mejor modelo es accuracy.

Para declarar un *Trainer* se indica el modelo, los parámetros de entrenamiento, los conjuntos de entrenamiento y test, las métricas de evaluación del modelo, el data collator y callbacks opcionales.

Las métricas de evaluación se declaran en una función para ser devueltas por un diccionario.

El data collator se declara indicando el tokenizer y se encarga de crear los batch de datos para entrenar el modelo.

Para este trabajo el callback utilizado es un earlyStop callback para detener el modelo en caso de que la última evaluación proporcione peores resultados que la anterior sobre una métrica indicada. En nuestro caso la métrica se trata de accuracy.

Una vez declarado todo se puede ejecutar la función train de la clase Trainer para realizar un entrenamiento como el de la figura 5.5

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 5.5: Ejemplo de entrenamiento

Para obtener un desglose de las métricas para las clases de la respuesta se deberá ejecutar tras el entrenamiento la función evaluate sobre el conjunto de test para obtener las predicciones del modelo. Tras ello se realiza un classification report con las salidas esperadas obteniendo un resultado como el de la figura 5.6.

	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 5.6: Desglose de las métricas de entrenamiento por clases

Si comparamos ambas imágenes podemos ver como los resultados del classification report coinciden con la segunda evaluación del modelo, debido a la parada temprana junto a los dos últimos parámetros utilizados en la clase TrainingArguments.

Para todos los entrenamientos se ha realizado la división del conjunto de datos utilizando la función train_test_split con estratificación en la clase respuesta para asegurar representación de todos los posibles valores de la salida en ambos subconjuntos de datos. Para las proporciones de los subconjuntos se han utilizado los parámetros sugeridos en la función, por lo que se utiliza un 66 % para el conjunto de entrenamiento y un 33 % para el conjunto de test.

5.6. Elección de hiperparámetros

Hugging Face permite la búsqueda de hyperparámetros a través de backends dedicados. Entre los soportados se encuentran optuna, sigopt, raytune and wandb [52].

Como ya hemos comentado en la sección 4.1 nosotros utilizaremos wandb para este proceso.

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

- Learning rate: la tasa de entrenamiento es un multiplicador utilizado en el algoritmo de propagación hacia atrás que agiliza la convergencia del modelo.
  Sus valores variarán entre 10^{-6} y 10^{-4} siguiendo una distribución uniforme.

- Batch size: indica el número de muestras que se propagan a través del modelo antes de reajustar sus pesos.
  Se realiza la búsqueda sobre el batch size del entrenamiento y la evaluación por separado.
  Sus valores suelen seleccionarse como potencias de 2. En nuestro caso variarán entre 2 y 128, siendo este el máximo por limitaciones de la memoria VRAM de las GPUs disponibles.

- Epoch: las épocas marcan cuando todas las instancias del conjunto de entrenamiento han sido introducidas al modelo.
  Sus valores variarán entre 5 y 15 siguiendo una distribución uniforme de números enteros.

En la figura 5.7 se incluyen fragmentos de código importantes para realizar la búsqueda de hiperparámetros con la plataforma wandb.

Primero se debe tener la sesión iniciada en su plataforma y utilizar la clave de API única para cada usuario.

Una vez conectados con el backend desde Python se define el espacio de hiperparámetros, donde se indican los parámetros y sus posibles valores.

A continuación escribimos el código de un entrenamiento genérico del modelo a optimizar.

Por último se especifica el número de entrenamientos a realizar y la dirección en la que queremos optimizar la métrica de rendimiento seleccionada. En nuestro caso utilizamos 30 entrenamientos buscando maximizar el accuracy del modelo.

```
config = {
          "wandb":{
            "project": "wandb-2labels-final",
            "api_key": "[redacted]"
        }
}


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}
        },
    }


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

Figura 5.7: Fragmentos de código para la conexión con wandb, selección de parámetros y sus valores y dirección de optimización junto al número de entrenamientos

Al ejecutar se realizarán tantos entrenamientos como hemos especificado. Su salida se verá en el cuaderno, pero al realizar muchos entrenamientos es más interesante dirigirse a la página web de Wandb.

Allí tendremos en nuestros proyectos las diferentes configuraciones, las cuales se actualizan al terminar cada entrenamiento con las métricas obtenidas.

En la figura 5.8 tenemos un ejemplo de búsqueda de hiperparámetros para uno de los modelos utilizados, donde cada línea del gráfico equivale a una configuración de parámetros usada para el entrenamiento. La línea oscila a lo largo de la gráfica indicando el valor utilizado en el entrenamiento para cada parámetro, en el eje de abscisas en la parte superior. El último valor de este eje indica la métrica o métricas del modelo y se apoya de un gradiente de colores para identificar las configuraciones según los resultados. Por lo tanto una línea de color anaranjado nos indica que ha obtenido valores cercanos de accuracy cercanos a 0.94, por lo que podría tratarse de una configuración óptima.

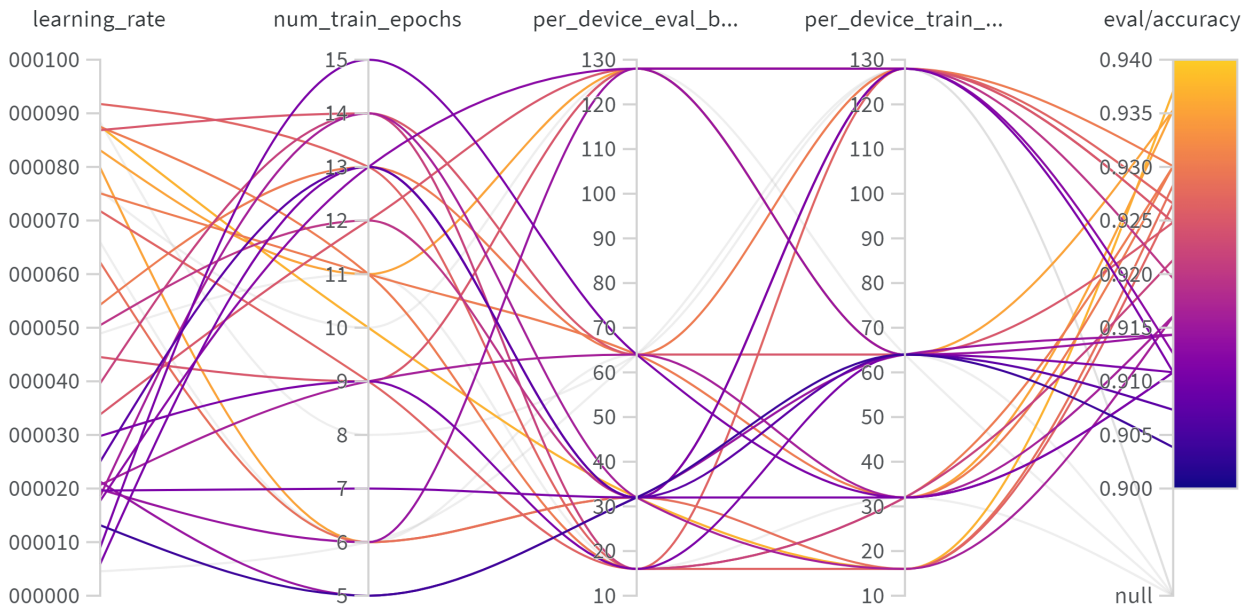

Figura 5.8: Búsqueda de hiperparámetros con wandb

El entorno wandb permite seleccionar cada entrenamiento por separado para ver los valores de sus parámetros o aplicar filtros para excluir algunos, incluso ofrece métricas en términos de importancia y correlación de cada parámetro con la métrica a optimizar.

De esta forma resulta más sencillo determinar cuál es la combinación de hiperparámetros con la que obtenemos mejores resultados en nuestro modelo. Estos valores serán los utilizados para entrenar el modelo con el que obtendremos los resultados.

5.7. Resultados

Para la obtención de resultados robustos realizamos un proceso de validación cruzada, mediante el cual se divide el conjunto de datos mediante un 10-Fold con estratificación en las clases de la respuesta.

Se entrena el modelo con las distintas divisiones y se realiza la media aritmética de las métricas de rendimiento obtenidas para cada una de las clases respuesta, las medias aritmética (macro) y ponderada por el tamaño de las clases (weigthed) y la precisión (accuracy) del modelo.

Para cada uno de los conjuntos de datos realizaremos las mismas pruebas de clasificación, primero solo con las clases positiva y negativa y posteriormente se incluirá la polaridad indeterminada o neutra al clasificador.

5.7.1. Resultados de polaridad para el dataset de Twitter

A continuación se muestran los resultados de polaridad obtenidos en la clasificación del conjunto de datos de Twitter.

5.7.1.1. Análisis de dos polaridades: positiva y negativa

Realizamos las pruebas con las siguientes configuraciones de parámetros, obtenidas mediante una búsqueda de hiperparámetros con wandb [33] de 30 repeticiones:

- RoBERTa: learning_rate = 6,542 $\times 10^{-5}$, train_batch_size = 64, eval_batch_size = 32, num_train_epochs= 14

- RoBERTuito: learning_rate = 8,169 $\times 10^{-5}$, train_batch_size = 64, eval_batch_size = 16, num_train_epochs= 6

En la tabla 5.1 encontramos los resultados, donde POS y NEG se refiere a las polaridades negativa y positiva respectivamente y P, R y F_1 se refiere a las métricas de precision, recall y F_1-score presentadas en la seccion 4.2. Al final encontramos la clase macro-avg que muestra la media aritmética de las métricas de las anteriores clases, y la métrica de accuracy global del modelo, denominada Acc.

Modelo	NEG			POS			macro-avg			
	P	R	F_1	P	R	F_1	P	R	F_1	Acc
RoBERTa	0.895	0.901	0.897	0.933	0.928	0.930	0.914	0.914	0.914	0.917
RoBERTuito	0.883	0.918	0.900	0.944	0.917	0.930	0.913	0.918	0.915	0.918

Tabla 5.1: Resultados obtenidos en clasificación de sentimientos con 2 salidas para el corpus de Twitter

La clase positiva obtiene mejores resultados que la negativa, lo que significa que el modelo es capaz de clasificar con mayor precisión aquellos textos con mensaje positivo correctamente. Esto ocurre para ambos modelos.

Utilizando las métricas globales de los clasificadores (clase macro-avg) y el accuracy global vemos que ambos clasificadores alcanzan una precisión bastante buena, por encima de 0.9 en todas las métricas. El clasificador RoBERTuito es mejor en todas ellas salvo en precision, aunque estas diferencias son muy ligeras de un modelo a otro.

Esto se ve también en la figura 5.9, donde ambos clasificadores obtienen resultados muy similares para la clasificación del conjunto test.

Recordemos que para las matrices de confusión en las filas se encuentran los valores reales de los textos y en las columnas los valores predichos por el clasificador, por lo que el valor de la esquina superior derecha de estas matrices muestra el número de textos positivos que fueron clasificados erróneamente como negativos por el clasificador.

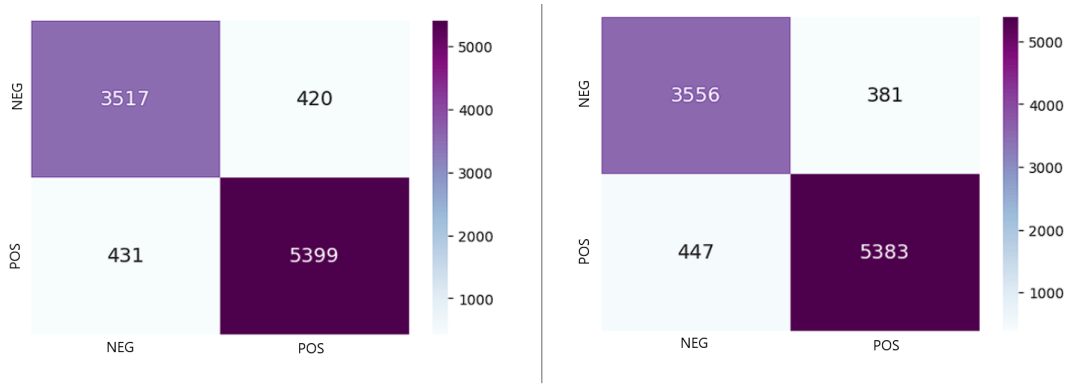

Figura 5.9: Matriz de confusión para una clasificación de sentimientos con 2 salidas con los modelos RoBERTa (izquierda) y roBERTuito (derecha) para el corpus de Twitter

Las matrices de confusión nos permiten ver como la confusión de ambas clases no es muy alta teniendo en cuenta el número de instancias totales pertenecientes a cada clase.

Resulta interesante ver como RoBERTuito reduce en 39 las instancias mal clasificadas de la clase negativa con respecto a RoBERTa, mientras aumenta en 16 las instancias mal clasificadas de polaridad positiva.

Aún así tenemos que para un mismo conjunto de evaluación RoBERTuito clasifica 23 instancias correctamente más que RoBERTa, por lo que sería el mejor modelo.

5.7.1.2. Análisis de tres polaridades: positiva, negativa y neutra

Los parámetros óptimos para los modelos obtenidos con wandb son los siguientes:

- RoBERTa: learning_rate = $4{,}751 \times 10^{-5}$, train_batch_size = 32, eval_batch_size = 64, num_train_epochs= 11
- RoBERTuito: learning_rate = $7{,}088 \times 10^{-5}$, train_batch_size = 128, eval_batch_size = 32, num_train_epochs= 6

En la tabla 5.2 vemos sus métricas para las distintas respuestas, donde se puede ver como los resultados han empeorado para cada una de las métricas como ya se sospechaba, ya que al aumentar el número de respuestas aumenta la confusión del clasificador.

Modelo	NEG			NEU			POS			macro-avg			
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	Acc
RoBERTa	0.751	0.795	0.772	0.737	0.756	0.746	0.869	0.813	0.839	0.787	0.788	0.786	0.788
RoBERTuito	0.752	0.815	0.782	0.785	0.701	0.740	0.830	0.859	0.843	0.789	0.792	0.788	0.792

Tabla 5.2: Resultados obtenidos en clasificación de sentimientos con 3 salidas para el corpus de Twitter

En este caso la diferencia de resultados entre la clase positiva y el resto es aún mayor, por lo que para 3 polaridades resulta más facil clasificar un texto positivo que cualquier otro.

La clase negativa y neutra obtienen métricas similares entre ellas. Es en la primera clase donde para todas las medidas gana el modelo de RoBERTuito, mientras que en el resto los resultados no son tan claros. Mientras que en la clase neutra es RoBERTa la que obtiene 2 de 3 medidas con

valores mayores, en la clase positiva ocurre al revés.

Al realizar el análisis de las medidas a nivel de clasificador es cuando tenemos que nuevamente es RoBERTuito el modelo que debe considerarse el mejor clasificador.

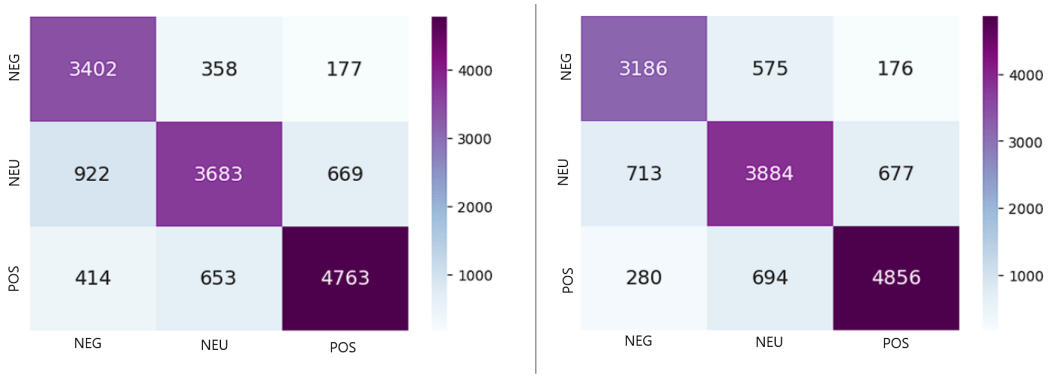

Figura 5.10: Matriz de confusión para una clasificación de sentimientos con 3 salidas con los modelos RoBERTa (izquierda) y roBERTuito (derecha) para el corpus de Twitter

En las matrices de confusión de la figura 5.10 observamos como sin duda es la nueva clase neutra la que mayor confusión introduce en ambos clasificadores por el número de instancias de dicha clase que se han clasificado como positivas o negativas.

RoBERTuito consigue reducir el número de instancias mal clasificadas tanto en la clase neutra como positiva. Sin embargo su confusión con textos negativos es mayor hasta en más de 200 textos con respecto a RoBERTa, de los cuales la mayoría son debidos a la nueva clase neutra.

Mientras que RoBERTa parece tener un sesgo mayor hacia la clase negativa al clasificar los textos neutros, RoBERTuito lo disminuye hasta niveles similares de confusión en ambas clases.

Aún así es importante comentar la importante pérdida de precisión al aumentar el número de salidas posibles del clasificador.

RoBERTuito tiene un accuracy 0.126 puntos mayor al clasificar con 2 clases que con 3, lo que supone un 15.9 % de mejora.

5.7.1.3. Comparación con un clasificador de redes neuronales

En esta sección cumpliremos uno de los objetivos del trabajo propuesto, en el que compararemos los mejores clasificadores para la tarea de análisis de polaridad, los cuales han sido RoBERTuito para dos y tres salidas, con los mejores clasificadores obtenidos en el trabajo fin de grado de Jesús Herrero [7].

Jesús utilizó un modelo híbrido de redes RNN y redes convolucionales donde la capa recurrente es una capa LSTM (long short-term memory) e incluyendo mejoras como la inclusión del concepto de ganancia de información, reduciendo el tamaño del corpus y utilizando un tamaño de vocabulario de 10.000 palabras.

Con dichas mejoras obtuvo mediante un experimento de validación cruzada un accuracy de 0.7586 para el clasificador de tres polaridades. En la tabla 5.3 se puede ver los resultados para

dicho clasificador.

Modelo	NEG			NEU			POS			macro-avg			Acc
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	
Híbrido	0.74	0.77	0.75	0.71	0.71	0.71	0.82	0.79	0.80	0.75	0.76	0.76	0.76
RoBERTuito	0.752	0.815	0.782	0.785	0.701	0.740	0.830	0.859	0.843	0.789	0.792	0.788	0.792

Tabla 5.3: Comparación de las métricas entre RoBERTuito y el clasificador híbrido después de las mejoras con tres polaridades

Realizando una comparativa con la tabla 5.2 ya analizada previamente obtenemos unas conclusiones similares, como que la polaridad neutra es la más confusión introduce al clasificador y que la polaridad positiva es la más sencilla de clasificar.

Realizando una comparativa uno a uno de las métricas para cada clase RoBERTuito obtiene mejores valores en todas las métricas salvo con el recall de la clase neutra.

Como consecuencia directa los valores macro del modelo de transformers analizado en este trabajo son mayores para todas las métricas de precisión.

RoBERTuito obtiene para esta tarea una precisión un 4 % mejor que el modelo híbrido.

Al aplicar el mismo modelo reduciendo el número de salidas a dos, Jesús obtuvo un accuracy de 0.907. Los resultados completos se encuentran en la tabla 5.4.

Modelo	NEG			POS			macro-avg			Acc
	P	R	F_1	P	R	F_1	P	R	F_1	
Híbrido	0.90	0.88	0.89	0.91	0.93	0.92	0.91	0.90	0.90	0.91
RoBERTuito	0.883	0.918	0.900	0.944	0.917	0.930	0.913	0.918	0.915	0.918

Tabla 5.4: Comparación de las métricas entre RoBERTuito y el clasificador híbrido después de las mejoras con dos polaridades

El análisis de resultados del clasificador RoBERTuito muestra valores de las métricas muy próximos entre sí, en la que dependiendo de la métrica obtiene mejores resultados un modelo u otro.

Revisando las métricas generales es RoBERTuito el modelo que vuelve a obtener mejores resultados, aunque de nuevo muy parejos respecto a los del modelo híbrido.

RoBERTuito logra una ligera mejora de accuracy del 1.2 % respecto al desarrollo de Jesús.

Concluimos por lo tanto que los modelos analizados en este trabajo han logrado superar en el problema de clasificación de textos para el análisis de polaridad a los propuestos con diferentes técnicas de deep learning, lo que muestra la potencia de los transformers en tareas de procesamiento del lenguaje natural.

5.7.2. Resultados de polaridad para el dataset de Twitch

A continuación se muestran los resultados de polaridad obtenidos en la clasificación del conjunto de datos de Twitch.

5.7.2.1. Análisis de 2 polaridades: positiva y negativa

Realizamos las pruebas con las siguientes configuraciones de parámetros óptimos obtenidos con wandb:

- RoBERTa: learning_rate = 6,775 $\times 10^{-5}$, train_batch_size = 128, eval_batch_size = 64, num_train_epochs= 15
- RoBERTuito: learning_rate = 9,517 $\times 10^{-5}$, train_batch_size = 64, eval_batch_size = 128, num_train_epochs= 7

Como se puede ver en la tabla 5.5 se ha producido una reducción general de las métricas de precisión en comparación al mismo experimento utilizando el conjunto de datos de Twitter.

Modelo	NEG			POS			macro-avg			
	P	R	F_1	P	R	F_1	P	R	F_1	Acc
RoBERTa	0.806	0.792	0.793	0.832	0.832	0.828	0.819	0.812	0.811	0.813
RoBERTuito	0.859	0.863	0.860	0.884	0.879	0.881	0.872	0.871	0.871	0.872

Tabla 5.5: Resultados obtenidos en clasificación de sentimientos con 2 salidas para el corpus de Twitch

Para este conjunto de datos hay diferencias notables entre ambos algoritmos, donde RoBERTuito sale beneficiado en ambas clases.

La tendencia a clasificar mejor los textos de polaridad positiva sigue presente en los resultados, lo que descarta la posibilidad de tratarse de la fuente de los datos y nos lleva a pensar que debido a las características del lenguaje al expresar un sentimiento positivo el lenguaje utilizado sea más expresivo y directo que en el caso opuesto, donde el sarcasmo puede confundir a los modelos.

El modelo RoBERTuito obtiene un accuracy un 7.25 % mejor que RoBERTa, lo que supone una diferencia considerable y por lo tanto se trata claramente del mejor clasificador de las tareas. Sin embargo el accuracy obtenido sigue siendo un 5.27 % peor que el obtenido con el conjunto de datos de Twitter.

Esto puede ser debido a las diferencias del conjunto de datos, ya que el lenguaje en entornos relacionados con videojuegos es muy habitual el uso de expresiones únicamente utilizadas en dichos entornos, además del uso frecuente de siglas y anglicismos por la gran influencia de streamers de habla inglesa.

En la figura 5.11 se encuentra la comparativa de matrices de confusión en la predicción de un subconjunto de los datos.

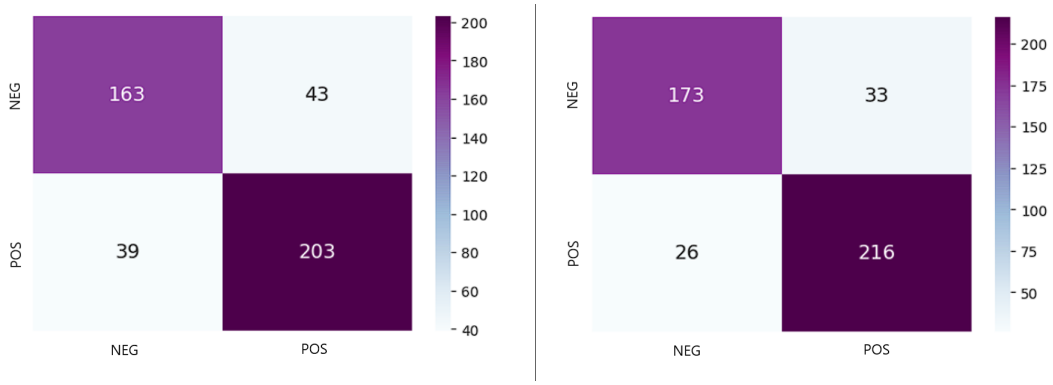

Figura 5.11: Matriz de confusión para una clasificación de sentimientos con 2 salidas con los modelos RoBERTa (izquierda) y roBERTuito (derecha) para el corpus de Twitch

Como ya se comentó en el análisis de la tabla de resultados la clase negativa parece ser la que mayor confusión produce.

Se observa como el modelo RoBERTuito clasifica correctamente un mayor número de textos independientemente de su polaridad.

5.7.2.2. Análisis de tres polaridades: positiva, negativa y neutra

Realizamos las pruebas con las siguientes configuraciones de parámetros:

- RoBERTa: learning_rate = $2{,}537 \times 10^{-5}$, train_batch_size = 128, eval_batch_size = 128, num_train_epochs = 10
- RoBERTuito: learning_rate = $9{,}553 \times 10^{-6}$, train_batch_size = 16, eval_batch_size = 16, num_train_epochs = 10

Como es de esperar a la vista de los resultados con dos polaridades, al añadir la posibilidad de una clase más se produce un descenso de las métricas de precisión en comparación al conjunto de datos de Twitter.

La pérdida no ha sido tan acentuada como en el caso de las salidas positiva y negativa. Sin embargo destaca la pobre capacidad predictiva de ambos modelos para la clase neutra, como puede verse en la tabla 5.6.

Modelo	NEG			NEU			POS			macro-avg			
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	Acc
RoBERTa	0.723	0.787	0.752	0.227	0.118	0.146	0.768	0.824	0.794	0.573	0.577	0.564	0.733
RoBERTuito	0.749	0.849	0.795	0.525	0.184	0.271	0.823	0.856	0.839	0.699	0.630	0.635	0.780

Tabla 5.6: Resultados obtenidos en clasificación de sentimientos con 3 salidas para el corpus de Twitch

Estas medidas tan bajas pueden deberse a la falta de balanceo del dataset, en el cual apenas un 10 % de las instancias corresponden a la clase indeterminada.

El clasificador sigue siendo más preciso al clasificar textos positivos en comparación al resto de clases. Aunque las métricas a nivel de modelo (macro-avg) son bastante bajas hay que tener en cuenta que se trata de la media aritmética de las métricas de cada clase, y dado que la clase

neutra ha obtenido unos resultados tan bajos, esto reduce la media general. Si ponderáramos los pesos con respecto al número de textos de cada clase estos valores serían bastante más elevados y parecidos a los resultados de accuracy.

El modelo RoBERTa obtiene un valor más bajo de accuracy que RoBERTuito, por lo que este último vuelve a ser considerado el mejor clasificador para la tarea en análisis de sentimientos con un accuracy de 0.78.

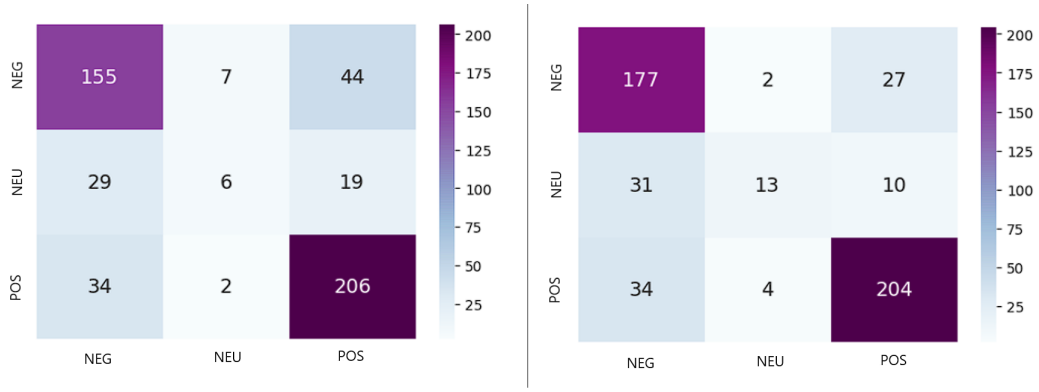

Figura 5.12: Matriz de confusión para una clasificación de sentimientos con 3 salidas con los modelos RoBERTa (izquierda) y roBERTuito (derecha) para el corpus de Twitch

En la comparativa de matrices de confusión de la figura 5.12 se puede ver el claramente el problema de balanceo del conjunto de datos y su relación con las métricas. Al tener tan pocos textos con clasificación neutra el modelo no ha podido aprender bien los patrones que le indiquen la relación con dicha clase.

Por ello se tiene que de 54 textos posibles son 48 y 41 los textos con polaridad neutra mal clasificados respectivamente. También se observa como aunque el modelo RoBERTuito logra reducir la cantidad de textos neutros mal clasificados, no reduce la confusión de ambas clases simultáneamente, si no que aumenta ligeramente el número de textos mal clasificados como negativos y reduce en mayor medida aquellos mal clasificados como positivos.

Obviamente la clase que mayor confusión introduce en relación al tamaño se trata de la clase neutra, seguido de la clase negativa, la cual mejora considerablemente al utilizar el modelo RoBERTuito. La clase positiva proporciona resultados interesantes ya que a diferencia de las otras dos clases no mejora en su clasificación con el modelo RoBERTuito, si no que empeora en 2 textos clasificados erróneamente como neutros para el mismo conjunto de datos de evaluación.

5.7.3. Resultados de polaridad para el dataset de salud mental

A continuación se muestran los resultados de polaridad obtenidos en la clasificación del conjunto de datos de salud mental.

5.7.3.1. Análisis de dos polaridades: positiva y negativa

Realizamos las pruebas con las siguientes configuraciones de parámetros óptimos obtenidos con wandb:

- RoBERTa: learning_rate = 9,677 $\times 10^{-5}$, train_batch_size = 128, eval_batch_size = 64, num_train_epochs = 11

- RoBERTuito: learning_rate = $8{,}448 \times 10^{-5}$, train_batch_size = 64, eval_batch_size = 64, num_train_epochs = 10

Para este conjunto de datos se obtienen los mejores valores de métricas de precisión de todas las pruebas realizadas hasta ahora, lo que nos lleva a pensar que el contenido de los textos a analizar está muy polarizado y por lo tanto es fácilmente separable.

Modelo	NEG			POS			macro-avg			
	P	R	F_1	P	R	F_1	P	R	F_1	Acc
RoBERTa	0.929	0.893	0.910	0.960	0.973	0.966	0.944	0.933	0.938	0.951
RoBERTuito	0.932	0.932	0.930	0.974	0.973	0.973	0.953	0.952	0.952	0.961

Tabla 5.7: Resultados obtenidos en clasificación de sentimientos con 2 salidas para el corpus de salud mental

Analizando los resultados de la tabla 5.7 se concluye una vez más que resulta más sencillo clasificar correctamente los textos de polaridad positiva por el fuerte sentimiento de su contenido en comparación a aquellos de polaridad negativa.

El modelo RoBERTuito es el que obtiene nuevamente los mejores resultados en ambas clases y consecuentemente en los resultados globales con un accuracy de 0.961. Esto tiene sentido al haber sido entrenado previamente con un corpus basado en tweets, lo que le beneficia al analizar conjuntos de datos obtenidos de Twitter como son el primero y este último.

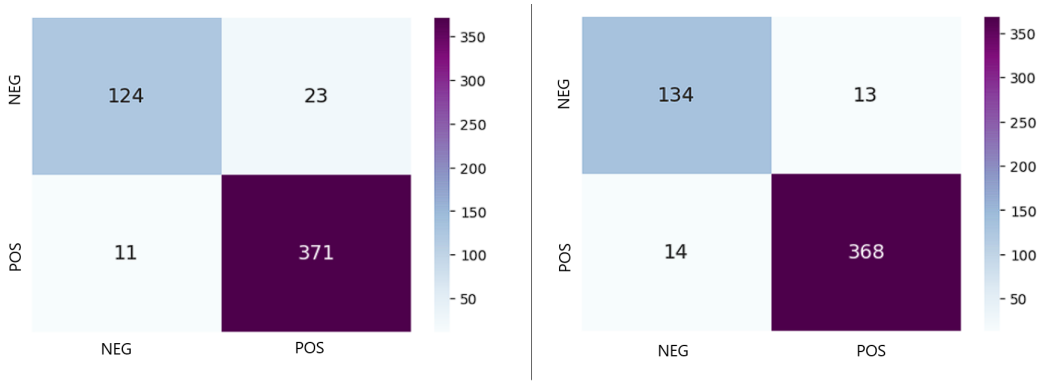

Figura 5.13: Matriz de confusión para una clasificación de sentimientos con 2 salidas con los modelos RoBERTa (izquierda) y roBERTuito (derecha) para el corpus de salud mental

La figura 5.13 nos muestra la comparativa de ambos modelos con respecto a la prueba de clasificación de un conjunto de test. En ella podemos ver como de forma similar al experimento anterior RoBERTuito logra mejorar la clasificación de textos de la clase negativa, la cual introduce una mayor confusión en la clasificación. Sin embargo en la clase positiva empeora la clasificación de textos, lo cual resulta curioso.

Aún con dicho empeoramiento en la clase positiva, la mejora en la clase negativa de la clasificación con RoBERTuito con respecto a RoBERTa hace que RoBERTuito obtenga mejores resultados y sea por lo tanto el mejor clasificador para el experimento.

5.7.3.2. Análisis de tres polaridades: positiva, negativa y neutra

Realizamos las pruebas con las siguientes configuraciones de parámetros:

- RoBERTa: learning_rate = 4,863×10^{-5}, train_batch_size = 64, eval_batch_size = 128, num_train_epochs = 13
- RoBERTuito: learning_rate = 8,759 × 10^{-5}, train_batch_size = 16, eval_batch_size = 32, num_train_epochs = 10

Para esta configuración seguimos obteniendo muy buenos resultados, incluso comparables en cuanto a precisión a los de otros datasets en su configuración de 2 salidas, donde es lógico lograr mejores métricas. Esto indica lo bien diferenciados que se encuentran los textos en cuando a polaridad en este conjunto de datos respecto a los anteriores.

En la tabla 5.8 se ve un claro empeoramiento en la clase indeterminada respecto a las polaridades extremas, lo cual a la vista de los resultados obtenidos con los otros conjuntos de datos nos lleva a la conclusión de que en general resulta más fácil clasificar un texto hacia valores extremos y opuestos como son positivo y negativo que a la difusa clase indeterminada.

También se observa con claridad los mejores resultados de las métricas de la clase positiva frente a la negativa, con una diferencia de al menos 0.5 puntos en precision, recall y F_1-score.

Modelo	NEG			NEU			POS			macro-avg			
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	Acc
RoBERTa	0.865	0.879	0.871	0.767	0.532	0.620	0.943	0.968	0.955	0.858	0.793	0.815	0.912
RoBERTuito	0.900	0.918	0.908	0.830	0.611	0.689	0.957	0.975	0.965	0.896	0.835	0.854	0.933

Tabla 5.8: Resultados obtenidos en clasificación de sentimientos con 3 salidas para el corpus de salud mental

Al igual que con dos posibles salidas con el conjunto de datos de salud mental RoBERTuito sale reforzado en los resultados de todas las clases, obteniendo finalmente una accuracy un 2.3 % mejor que el clasificador RoBERTa.

En la comparativa de la figura 5.14 se puede ver el bajo número de instancias mal clasificadas general.

Debido al mal balanceo del dataset, donde menos de un 10 % de instancias pertenecen a la clase difusa, no hay muchas instancias clasificadas en dicha clase. Sin embargo de 43 posibles solo se clasifican mal 17 y 11, por lo que en el caso de haber tenido un dataset mejor balanceado se podrían haber obtenido incluso mejores medidas.

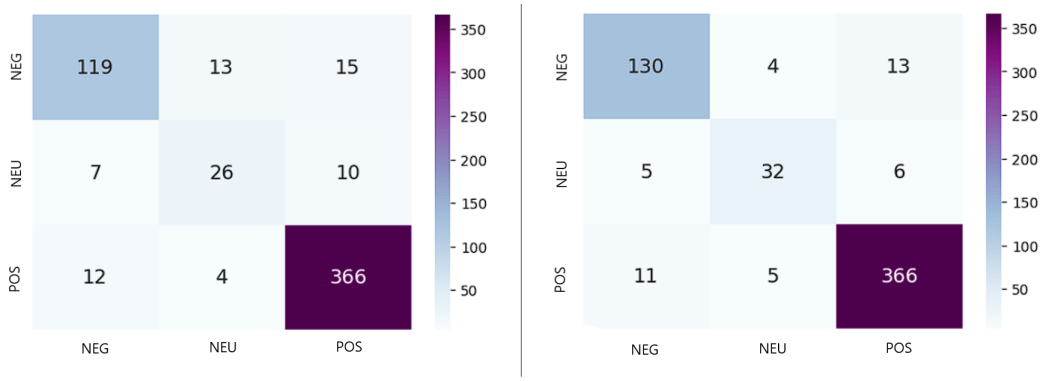

Figura 5.14: Matriz de confusión para una clasificación de sentimientos con 3 salidas con los modelos RoBERTa (izquierda) y roBERTuito (derecha) para el corpus de salud mental

Como es habitual, la clase que mayor confusión produce al clasificador es la neutra. Para el conjunto de datos de evaluación seleccionado RoBERTuito mejora la clasificación de los textos negativos y neutros, manteniendo igual el número de textos positivos mal clasificados respecto a RoBERTa.

Capítulo 6

Análisis emocional con modelos de BERT

6.1. Introducción

El análisis de emociones está fuertemente relacionado con el análisis de polaridad ya que al igual que en el capítulo anterior seguimos enfrentando un problema de clasificación multiclase simple a nivel de documento.

La diferencia entre ambos problemas radica en el número de valores que puede tomar la clase respuesta. Mientras que en análisis de polaridad como mucho podía haber tres valores en este capítulo el número aumenta hasta a siete, por lo que la correcta clasificación de una instancia se dificulta.

6.2. Análisis descriptivo de los dataset

6.2.1. Twitter

Se trata del conjunto de datos EmoEvent [53] en su variante en español, el cual es utilizado para el análisis de emociones a partir de mensajes de Twitter en base a diferentes eventos ocurridos.

Consta de 8.409 instancias con 3 entradas cada uno: el contenido del mensaje, la emoción del texto y si el texto se considera ofensivo. Este último campo se descarta ya que no entra dentro de los objetivos del estudio.

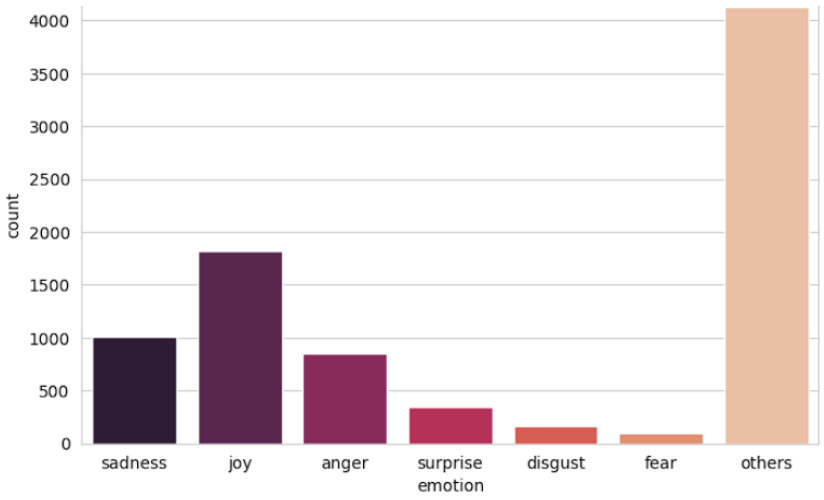

Figura 6.1: Conjunto de datos de Twitter (EmoEvent) para emociones

Las posibles respuestas se encuentran en la columna de emociones, que toman los valores de las seis emociones de Ekman, nombradas en honor al psicólogo estadounidense Paul Ekman, pionero en el estudio de las emociones y su expresión facial [54], y una última clase indeterminada.

Esta configuración en la respuesta es la más utilizada en el análisis de emociones con inteligencia artificial.

En la figura 6.1 podemos ver la distribución de los datos según su emoción. En datos concretos la clase sadness tiene 1009 instancias, joy 1815, anger 857, surprise 344, disgust 161, fear 96 y 4127 indeterminado.

Se trata por lo tanto de un dataset bastante desbalanceado en cuanto a la respuesta, acumulando casi la mitad en la única que no se trata de una emoción.

6.2.2. Twitch

Utilizamos el mismo conjunto de datos utilizado en el estudio lingüístico para el análisis de textos en el ámbito de los videojuegos, haciendo uso ahora de las etiquetas de resultado para el análisis de emociones.

Tenemos por lo tanto 2007 instancias repartidas en la variable respuesta según la figura 6.2 con 182 pertenecientes a la clase Decepción/Tristeza, 711 en Aprobación/Empatía/Confianza, 168 en Enfado/Ira, 268 en Interés/Anticipación/Hype, 246 en Desaprobación, 171 en Desinterés/Odio y 261 indeterminadas.

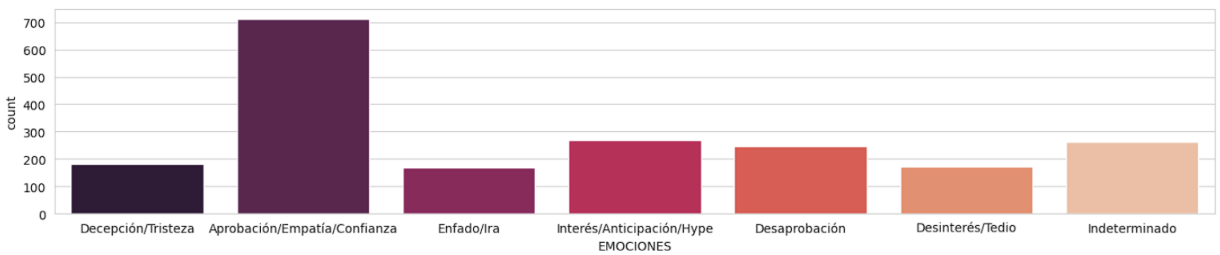

Figura 6.2: Conjunto de datos de Twitch para emociones

6.2.3. Salud mental

Se trata del conjunto de datos descrito en el apartado 5.2.3 para el estudio de la salud mental en relación a redes sociales.

El dataset contiene 640 textos clasificados con la emoción Amor/Admiración, 227 de Gratitud, 122 de Tristeza/Pena, 466 de Enfado/Desprecio/Burla, 659 de Comprensión/Empatía/Identificación y 173 indeterminados.

Uno de los textos no contenía información relativa a la emoción, por lo que se ha omitido.

En la figura 6.3 se encuentra la representación gráfica de la información presentada.

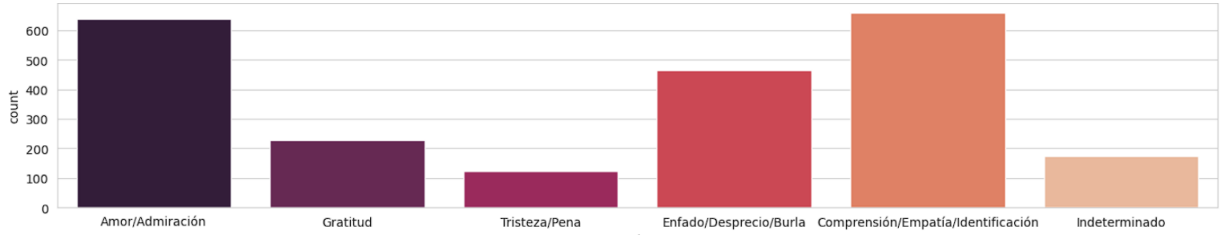

Figura 6.3: Conjunto de datos de salud mental para emociones

6.3. Modelos utilizados

Para este problema se vuelven a analizan dos modelos que permitan realizar una comparativa entre ellos:

6.3.1. Daveni

Se trata del modelo XML-RoBERTa [55], el mejor de un conjunto de modelos creados con el objetivo de mejorar los modelos de comprensión multiidioma mediante el entrenamiento no supervisado con grandes cantidades de texto multiidiomas.

Para su creación se partió de un modelo entrenado para la tarea de masked language modeling (MLM o modelado del lenguaje en español) para un solo idioma y se utilizó corpus nuevo basado en los datos de CommonCrawl [56] en 100 idiomas para su preentrenamiento.

Como resultado se obtuvo un modelo entrenado para 100 idiomas, logrando considerables mejoras en tareas de clasificación multiidioma, respuesta a preguntas y etiquetado de secuencias. Por ejemplo obtuvo una mejora de hasta el 23 % de accuracy en tareas de clasificación respecto al modelo mBERT, modelo BERT original con preentrenamiento multiidioma.

También se analiza la llamada maldición del multiidioma, por la que aunque hasta cierto punto se logra una mejora en tareas para idiomas con bajos recursos, es decir que con falta de datos para el aprendizaje automático [57], pasado dicho punto el comportamiento general tanto para idiomas con bajos recursos como el resto se verá reducido.

El modelo seleccionado del repositorio Hugging Face fue entrenado para la clasificación de emociones en textos por el usuario Daveni, por lo que al modelo se le llamará por dicho nombre.

Fue seleccionado debido a la expectación que supuso ver que fue presentado al evento EmoEvalES [58], parte del IberLEF [59] en su edición de 2021 logrando el primer premio en clasificación de emociones. [60]

6.3.2. RoBERTuito

Repite el modelo de pysentimiento presentado y analizado en la sección 5.3, ahora entrenado para la clasificación de emociones debido a su popularidad dentro de la comunidad Hugging Face y los buenos resultados aportados en el capítulo anterior. [61]

6.4. Resultados

A continuación se presentan los resultados obtenidos con los tres conjuntos de datos propuestos, siguiendo la misma metodología utilizada en el análisis de polaridad para su obtención.

6.4.1. Resultados de emociones para el dataset de Twitter

Realizamos las pruebas con las siguientes configuraciones de parámetros obtenidos con wandb:

- Daveni: learning_rate = $2{,}863 \times 10^{-5}$, train_batch_size = 128, eval_batch_size = 16, num_train_epochs = 12
- RoBERTuito: learning_rate = $2{,}143 \times 10^{-5}$, train_batch_size = 64, eval_batch_size = 64, num_train_epochs = 13

Con los que se obtienen los datos de la tabla 6.1, donde sorprenden los valores de la clase Disgust para el modelo Daveni, que son de exactamente 0 en las métricas, algo que no ha ocurrido en ninguna de las pruebas obtenidas hasta ahora y por lo tanto un resultado inusual.

La posibilidad de tener representación nula en alguno de los folds queda descartada ya que la división de todos los conjuntos de datos para las pruebas se realiza de forma estratificada sobre la clase respuesta para evitar este problema. Por lo tanto de momento se tiene que el clasificador del modelo Daveni no ha sido capaz de clasificar correctamente nignún texto de emoción Disgust en ninguno de los 10 folds de los que consta la prueba.

La misma clase presenta también valores muy bajos para el modelo RoBERTuito, y revisando los resultados para las 10 evaluaciones hay ocasiones en las que efectivamente RoBERTuito también obtiene 0 en las métricas, por lo que puede tratarse de un problema en la categorización del dataset.

Curiosamente el fenómeno se repite para la emoción Fear, pero en este caso sólo en el modelo RoBERTuito.

Emoción	Métrica	Daveni	RoBERTuito
Sadness	P	0.858	0.907
	R	0.826	0.869
	F_1	0.839	0.888
Joy	P	0.758	0.869
	R	0.786	0.838
	F_1	0.770	0.838
Anger	P	0.673	0.743
	R	0.781	0.888
	F_1	0.721	0.808
Surprise	P	0.554	0.719
	R	0.337	0.607
	F_1	0.415	0.655
Disgust	P	0.000	0.269
	R	0.000	0.056
	F_1	0.000	0.085
Fear	P	0.623	0.542
	R	0.544	0.488
	F_1	0.564	0.506
Other	P	0.826	0.876
	R	0.847	0.908
	F_1	0.836	0.892
macro-avg	P	0.613	0.704
	R	0.589	0.665
	F_1	0.592	0.669
Global	Acc	0.784	0.853

Tabla 6.1: Resultados obtenidos en clasificación de emociones para el corpus de Twitter

Realizando ahora un análisis general de los resultados tenemos que la clase Sadness es la que mejores resultados obtiene en ambos modelos, seguida de la clase indeterminada u Other. Surprise sería la que peores resultados obtiene, obviando las clases con resultados excepcionales comentados anteriormente. En esta clase RoBERTuito obtiene métricas generalmente mejores que Daveni, de forma más marcada que en el resto de clases.

Revisando los resultados macro de ambos modelos, así como el accuracy global del modelo se ve como RoBERTuito mejora en todas las métricas a Daveni, logrando un accuracy un 8.8 % mayor.

Podemos considerar por lo tanto a RoBERTuito como el mejor clasificador de emociones para el conjunto de datos de EmoEval.

Revisando las matrices de confusión la figura 6.4 se puede comprender lo ocurrido en la clase Disgust para los modelos, donde se ve como Daveni parece no ser capaz de clasificar ninguna de las instancias de dicha clase correctamente, y por ello obtendría valores de las métricas de 0 para la clasificación del conjunto de evaluación utilizado.

Esto puede deberse a la poca representación de la clase en el dataset, como ya vimos en la figura 6.1, lo que puede afectar al entrenamiento de los modelos.

RoBERTuito si que es capaz de clasificar 6 textos de 40 posibles como Disgust, donde la ma-

yoría son clasificados incorrectamente en Anger, al igual que ocurre con Daveni.

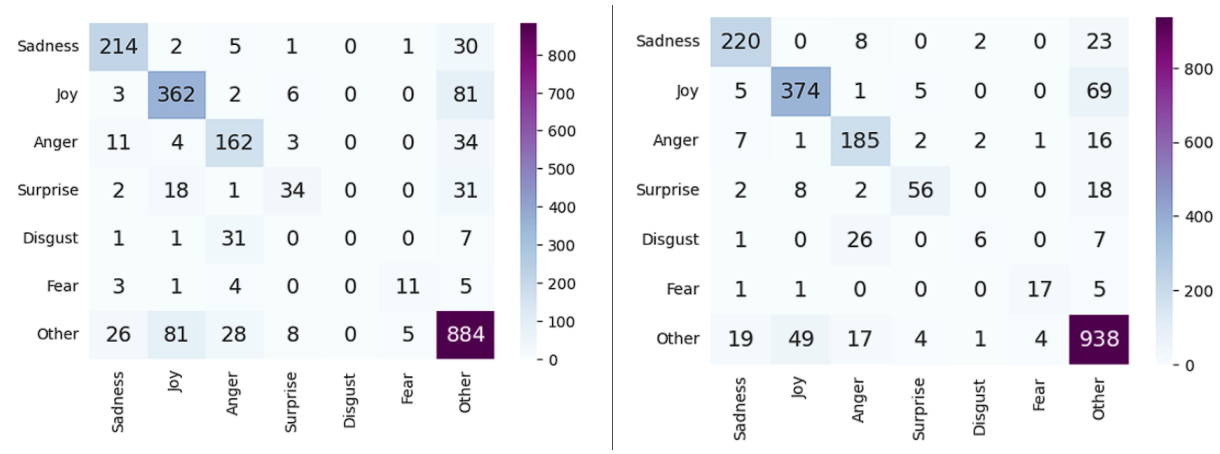

Figura 6.4: Matriz de confusión para una clasificación de emociones con los modelos Daveni (izquierda) y RoBERTuito (derecha) para el corpus de Twitter

Para el resto de las clases se ve claramente como en caso de clasificar mal el texto, lo más probable es que el clasificador lo incluya como texto indeterminado, probablemente debido a la gran representación de esta clase en el conjunto de datos, acumulando casi el 50 % de los textos en esta respuesta.

Revisando en profundidad la clase Other hay un comportamiento similar para ambos clasificadores, en el que en caso de clasificar mal un texto indeterminado sólo destacan la clase Joy como la más probable para ser clasificado y Disgust como la menos probable.

Para esta clase indeterminada RoBERTuito reduce en cerca de un 40 % las instancias mal clasificadas de la clase con la que más confunde. Para el resto de respuestas la confusión reducida es menor, pero igualmente notable.

Es por ello que al igual que con los resultados de las métricas vistos en la tabla 6.1 RoBERTuito es el mejor clasificador.

6.4.1.1. Comparativa con los resultados obtenidos por SVM

En este apartado realizaremos una comparativa con el estudio realizado por el equipo de investigación que creó el corpus EmoEvent, en el cual utilizaron un clasificador SVM (support vector machine) para la clasificación de emociones.

En concreto realizaremos la comparativa sobre los resultados en español, ya que el corpus contiene textos tanto en español como en inglés y realizan las pruebas en ambos idiomas. Al hacer estas pruebas de forma separada no hay inconvenientes en la comparativa.

En la comparativa utilizaremos el modelo RoBERTuito como representante de los modelos basados en transformers ya que es el que mejores resultados obtuvo de los dos modelos analizados.

Language	*joy*			*sadness*			*anger*			*fear*			*disgust*			*surprise*			*other*			*macro-avg*			
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	*Acc*
SP	0.60	0.49	0.54	0.79	0.63	0.70	0.55	0.34	0.42	0.63	0.36	0.46	0.21	0.02	0.03	0.39	0.12	0.19	0.64	0.84	0.73	0.54	0.40	0.44	0.64
EN	0.59	0.60	0.6	0.62	0.36	0.46	0.33	0.10	0.16	0.35	0.04	0.07	0.38	0.21	0.27	0.16	0.02	0.04	0.54	0.73	0.62	0.42	0.29	0.32	0.55

Figura 6.5: Resultados de las pruebas de clasificación de emociones (10-fold cross validation) con SVM [53]

En la figura 6.5 tenemos la tabla de resultados obtenida en el artículo, de la cual nos fijaremos en la fila SP la cual se refiere al conjunto de datos en español. El resto de siglas siguen la misma dinámica explicada anteriormente y aplicada a las tablas de resultados durante todo el informe.

Para el separador SVM tenemos como la clase Sadness obtiene los mejores resultados, mientras que es Disgust, al igual que ha ocurrido con los modelos de transformers, la que peores resultados obtiene. Tenemos por lo tanto una coincidencia para ambos modelos en cuanto a las clases más y menos fáciles de clasificar.

Realizando una comparativa directa entre ambos modelos tenemos que en ninguna de las clases el clasificador SVM supera a RoBERTuito para ninguna de las métricas analizadas.

Finalmente tenemos que RoBERTuito supera ampliamente al separador SVM en cuanto a accuracy global, con una mejora del 33.2 %, muy significativa. Esto vuelve a demostrar la importancia que han supuesto los nuevos modelos basados en transformers a los problemas NLP.

6.4.2. Resultados de emociones para el dataset de Twitch

Realizamos las pruebas con las siguientes configuraciones de parámetros óptimos obtenidos mediante la búsqueda de hiperparámetros con wandb:

- Daveni: learning_rate = 4,507 $\times 10^{-5}$, train_batch_size = 8, eval_batch_size = 32, num_train_epochs = 12

- RoBERTuito: learning_rate = 5,151 $\times 10^{-5}$, train_batch_size = 4, eval_batch_size = 16, num_train_epochs = 9

Como se ve en la tabla 6.2 las métricas de ambos modelos presentan valores muy bajos, incluso teniendo en cuenta los problemas del dataset anterior.

Emocion	Métrica	Daveni	RoBERTuito
Decepción/Tristeza	P	0.454	0.494
	R	0.362	0.419
	F_1	0.396	0.431
Aprobación/Empatía/Confianza	P	0.713	0.729
	R	0.818	0.812
	F_1	0.757	0.765
Enfado/Ira	P	0.527	0.536
	R	0.417	0.607
	F_1	0.454	0.563
Interés/Anticipación/Hype	P	0.692	0.728
	R	0.660	0.694
	F_1	0.673	0.702
Desaprobación	P	0.395	0.399
	R	0.420	0.465
	F_1	0.400	0.419
Desinterés/Tedio	P	0.535	0.523
	R	0.314	0.397
	F_1	0.354	0.440
Indiferente	P	0.466	0.539
	R	0.418	0.341
	F_1	0.425	0.398
macro-avg	P	0.540	0.564
	R	0.487	0.533
	F_1	0.494	0.531
Global	Acc	0.579	0.604

Tabla 6.2: Resultados obtenidos en clasificación de emociones para el corpus de Twitch

La clase que obtiene las peores métricas es Desaprobación, seguido de la clase Indiferente. La emoción que mejor clasifican ambos modelos se trata de Aprobación/Empatía/Confianza. Esto supone una diferencia interesante respecto al conjunto de datos anterior, ya que es una emoción positiva la que resulta más facil de clasificar para los modelos, de forma similar a lo ocurrido en el análisis de polaridad con la respuesta positiva.

Realizando una comparativa por clases es RoBERTuito el que obtiene por lo general mejores resultados en las métricas, salvo casos concretos como el recall de Aprobación/Empatía/Confianza. Es por ello que tanto en las métricas macro como en el accuracy global es el modelo que vuelve a resultar óptimo, con un 4.3 % de mejora respecto a Daveni.

Sin embargo estos resultados siguen resultando muy bajos, considerando que para un conjunto de datos tan desbalanceado como es el de EmoEvent se ha obtenido un accuracy de 0.853, mientras que en este es de 0.604.

Utilizando como referencia la figura 6.6 podemos explicar los pobres resultados como una gran confusión general en la clasificación en ambos modelos.

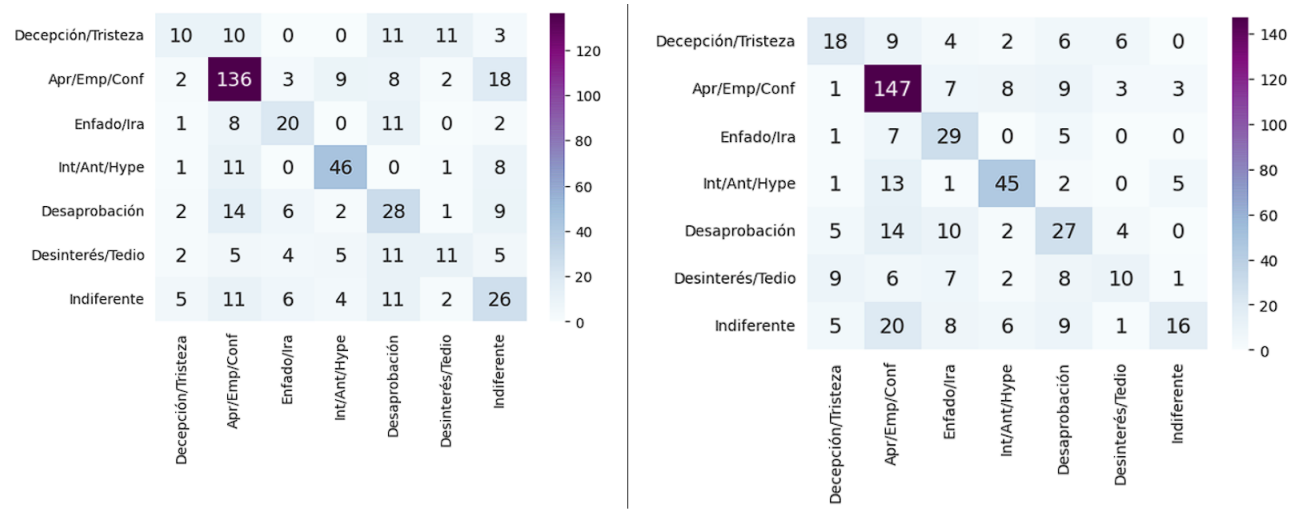

Figura 6.6: Matriz de confusión para una clasificación de emociones con los modelos Daveni (izquierda) y RoBERTuito (derecha) para el corpus de Twitch

Para el primer clasificador se clasifican mal más textos que bien para todas las emociones salvo Aprobación/Empatía/Confianza, respuesta mayoritaria dentro del dataset, e Interés/Anticipación/Hype.

No se puede establecer un patrón claro sobre las clases que más confusión produce. Solo los casos de Aprobación/Empatía/Confianza y Desaprobación muestran una tendencia a absorber la mayoría de textos mal clasificados pertenecientes a otras clases.

Para Aprobación/Empatía/Confianza es la clase indeterminada la que mayor número de instancias incorrectas absorbe. Estos resultados mejoran generalmente al utilizar RoBERTuito en favor de las tres primeras emociones. Sin embargo a partir de ahí el número de textos bien clasificados disminuye ligeramente. Para este modelo sigue siendo Aprobación/Empatía/Confianza la emoción que más confusión produce, lo que demuestra su gran representación dentro del conjunto de datos.

Tras una discusión de resultados se llegó a la conclusión de que el etiquetado del dataset podría ser el gran responsable de los malos resultados obtenidos.

Al tratar las respuestas como agrupaciones de emociones similares se ha podido producir un solapamiento entre ellas. Un ejemplo sería al expresar desaprobación, donde inherentemente se incluyen matices de enfado o desinterés en el contexto.

Es por ello que se utilizará el conjunto de datos de salud mental para contrastarlo, ya que se considera que las emociones consideradas como respuestas en ese dataset son más separables entre ellas.

6.4.3. Resultados de emociones para el dataset de salud mental

Realizamos las pruebas con las siguientes configuraciones de parámetros obtenidos con wandb:

- Daveni: learning_rate = $1{,}049 \times 10^{-5}$, train_batch_size = 8, eval_batch_size = 2, num_train_epochs = 12
- RoBERTuito: learning_rate = $7{,}557 \times 10^{-5}$, train_batch_size = 16, eval_batch_size = 16, num_train_epochs = 6

Obteniendo los datos representados en la tabla 6.3, los cuales son los mejores valores para el problema de clasificación de emociones en comparación con el resto de datasets.

Estos datos confirman por lo tanto la hipótesis sobre el conjunto de datos anterior, ya que ahora la respuesta que peores resultados obtiene es la indeterminada, mientras que para aquellos textos etiquetados con emociones los resultados están casi siempre por encima de 0.8.

También hay que tener en consideración que este conjunto de datos tiene seis respuestas posibles, cinco emociones y la clase indeterminada, frente a las siete de los conjuntos de EmoEval y Twitch. Un menor número de respuestas suele resultar en una clasificación mejor de los textos al tener menos posibilidades entre las que diferenciar.

Emocion	Métrica	Daveni	RoBERTuito
Amor/Admiración	P	0.906	0.931
	R	0.927	0.950
	F_1	0.916	0.940
Gratitud	P	0.941	0.908
	R	0.890	0.921
	F_1	0.914	0.913
Tristeza/Pena	P	0.814	0.837
	R	0.794	0.878
	F_1	0.793	0.851
Enfado/Desprecio/Burla	P	0.847	0.910
	R	0.858	0.932
	F_1	0.849	0.920
Comprensión/Empatía/Identificación	P	0.846	0.899
	R	0.886	0.885
	F_1	0.863	0.891
Indeterminado	P	0.701	0.795
	R	0.526	0.659
	F_1	0.574	0.709
macro-avg	P	0.842	0.880
	R	0.814	0.871
	F_1	0.818	0.871
Global	Acc	0.860	0.899

Tabla 6.3: Resultados obtenidos en clasificación de emociones para el corpus de salud mental

Tenemos que nuevamente es una emoción positiva, Amor/Admiración, la que mejores métricas obtiene para ambos modelos. Como ya dijimos es la clase indeterminada la que peores resultados obtiene. Obviando esta clase y centrándonos en las emociones puras es Tristeza/Pena la emoción con peores métricas, lo que concuerda con los resultados del conjunto de datos de Twitch donde la emoción negativa Desaprobación era la que por clasificaba.

El modelo RoBERTuito vuelve a ser mejor que Daveni en función de las métricas generales, obteniendo un accuracy de casi 0.9, un 4.5 % mejor que el modelo Daveni.

Podemos concluir que para el análisis de emociones RoBERTuito ha resultado ser superior para todas las pruebas propuestas, al igual que ocurrió en el capítulo de análisis de polaridad. Esto muestra la razón de su popularidad dentro de la comunidad Hugging Face, con cerca de 400.000

descargas en el último mes en su versión de polaridad, la más utilizada.

Revisando ahora la figura 6.7 se ve como la clasificación de los textos para ambos modelos es en general muy buena, donde la mayor confusión la producen las clases Amor/Admiración y Comprensión/Empatía/Identificación, las clases con un mayor número de textos dentro del dataset.

Para Comprensión/Empatía/Identificación es justamente Amor/Admiración la emoción que más confunde con diferencia la clasificación de sus textos. Esto no ocurre en el caso contrario.

Al igual que en casos anteriores al pasar de Daveni a RoBERTuito se observa un descenso del número de textos bien clasificados en alguna clase en beneficio de otras minoritarias en cuanto a representación en la muestra.

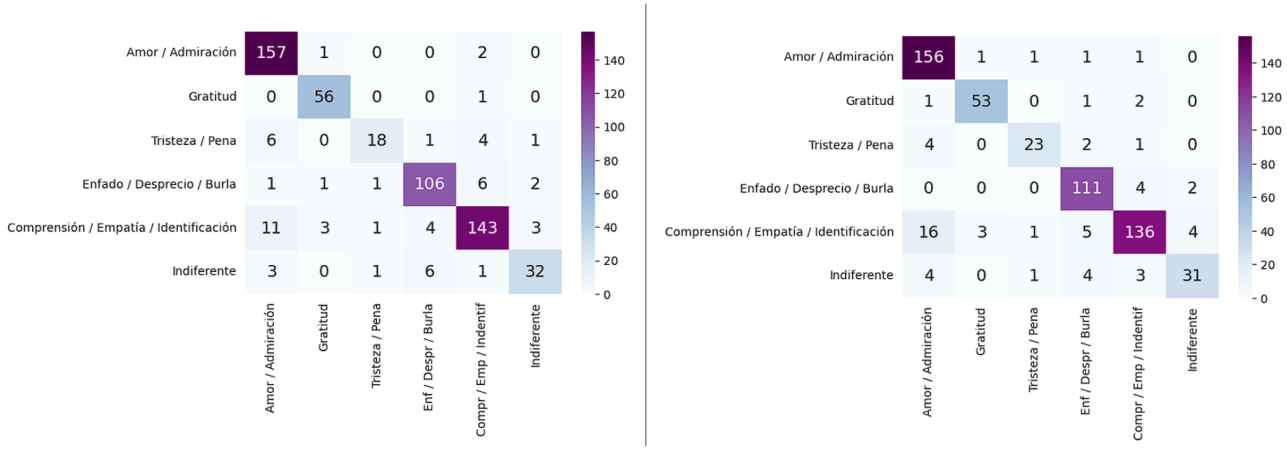

Figura 6.7: Matriz de confusión para una clasificación de emociones con los modelos Daveni (izquierda) y RoBERTuito (derecha) para el corpus de salud mental

RoBERTuito vuelve a ser por lo tanto el modelo que mejores resultados aporta, cosa que ha ocurrido para todos los conjuntos de datos y siendo este último, para el cual no se han observado problemas durante la ejecución de las pruebas, el que lo confirma.

6.5. Predicción de nuevos datos

Una vez entrenado el modelo sobre un conjunto de datos es posible realizar predicciones sobre nuevos datos. Existen dos aproximaciones para su ejecución:

La primera es utilizar la propia clase *trainer* con la que se ha realizado el entrenamiento con la función *evaluate* para obtener las predicciones del modelo.

La segunda es guardar los archivos de configuración del modelo y tokenizer obtenidos tras el entrenamiento de forma externa y utilizarlos junto a la clase *pipeline*, diseñada para permitir a los usuarios realizar pruebas de manera sencilla sobre los modelos de su repositorio.

Se ha creado una función predict siguiendo la segunda aproximación para los modelos entrenados con el conjunto de datos de salud mental, por haber presentado los mejores resultados para ambos problemas de clasificación.

Para su uso se debe especificar la tarea, entre *sentiment* y *emotion*, el modelo a usar, con las posibilidades *robertuito*, *roberta* y *daveni*, y el texto o lista de textos que se quieran clasificar.

Como se puede ver en la figura 6.8 dependiendo de la tarea de clasificación seleccionada se define el número de salidas del modelo y la configuración para interpretar dichas salidas, ya que del modelo se obtienen en forma numérica.

Después se crea el modelo a partir de los archivos de configuración guardados en la nube. Se han guardado los modelos de clasificación de polaridad para tres salidas y los de análisis de emociones con sus configuraciones particulares. Se modifica el nivel para el cual se generan mensajes de log ya que de lo contrario cada vez que se prediga algo aparecerían mensajes de aviso irrelevantes para esta tarea.

Por último se utiliza la clase *pipeline* para predecir sobre los datos. El parámetro top_k=1 indica que solo nos interesa obtener la respuesta con probabilidad máxima para cada uno de los textos a predecir. La función devolverá únicamente la respuesta para cada texto, almacenada en el campo *label*, sin incluir su probabilidad asociada.

```
#Prediction function
#name = "roberta", "robertuito" if task=="sentiment". "daveni", "robertuito" if task=="emotion"
#task = "sentiment", "emotion"
#data = string data or list of strings to predict from

#return = prediction or list of predictions
def predict(name, task, data):
  if(task=="emotion"):
    num_labels=6
    id2label = {0: "Amor/Admiración", 1: "Gratitud", 2: "Tristeza/Pena",
                3:"Enfado/Desprecio/Burla", 4:"Comprensión/Empatía/Identificación", 5:"Indiferente"}
    label2id = {"Amor/Admiración":0, "Gratitud":1, "Tristeza/Pena":2,
                "Enfado/Desprecio/Burla":3, "Comprensión/Empatía/Identificación":4, "Indiferente":5}
  else:
    num_labels=3
    id2label = {0: "NEG", 1: "NEU", 2: "POS"}
    label2id = {"NEG": 0, "NEU": 1, "POS": 2}

  logging.set_verbosity_error()
  model_path = "./drive/MyDrive/Colab Notebooks/TFG/Salud/model_"+name+"_"+task
  tok_path = "./drive/MyDrive/Colab Notebooks/TFG/Salud/tok_"+name+"_"+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)
  logging.set_verbosity_warning()
  return([i[0]["label"] for i in pipe(data)])
```

Figura 6.8: Función de predicción para la clasificación de textos utilizando los modelos entrenados con el corpus de salud mental

En la tabla 6.4 se muestran las predicciones del modelo roBERTuito para clasificación de emociones para un conjunto de textos de ejemplo mediante la ejecución de la sentencia de la figura 6.9.

```
predict(name="robertuito", task="emotion",
        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"])
```

Figura 6.9: Sentencia de código utilizada para obtener los resultados de la tabla 6.4

Texto	**Salida**
Te quiero	Amor/Admiración
Te odio	Enfado/Desprecio/Burla
Me das pena	Enfado/Desprecio/Burla
No tengo una opinión clara al respecto	Enfado/Desprecio/Burla
Siento profundamente tu reciente pérdida	Tristeza/Pena
Soy vegano porque siento que debo hacer algo por el bienestar animal	Comprensión/Empatía/Identificación

Tabla 6.4: Ejemplo de predicción de emociones obtenidas con la ejecución de la figura 6.9

Con la frase “Me das pena” se esperaba expresar pena, pero al tratarse de una expresión frecuentemente utilizada de forma despectiva el modelo lo clasifica como Enfado/Desprecio/Burla, lo que sirve como muestra de la importancia del contexto de un mensaje y la ambigüedad que supone la falta del medio oral en su transmisión.

Capítulo 7

Conclusiones y líneas futuras

7.1. Conclusiones

El crecimiento del deep learning en los últimos años ha afianzado su papel como el futuro del procesamiento de datos, no solo para las tareas NLP analizadas en este trabajo si no para una gran variedad en diversos sectores.

La realización de este trabajo me ha resultado de gran interés y me ha servido de mucho ya que he podido ampliar mis conocimientos acerca de técnicas de inteligencia artificial aprendidas durante mis estudios en el grado en Ingeniería Informática, así como el aprendizaje sobre tecnologías desconocidas para mí previamente como son los transformers y sus aplicaciones en modelos muy utilizados actualmente mediante productos conocidos por el público general, como puede ser OpenAI chatGPT.

El procesamiento natural del lenguaje es un campo en el cual se están destinando muchos recursos en investigación por sus potenciales beneficios hacia la industria, como el de automatizar tareas repetitivas y de poco requerimiento técnico.

Es por ello que todavía quedan por ver nuevos avances revolucionarios en este campo de trabajo como lo están siendo los modelos analizados en la actualidad.

A lo largo de este trabajo se han encontrado limitaciones propias del problema a tratar, como puede ser la dificultad en algunos casos de predecir correctamente un texto dependiendo de la emoción que quiere transmitir, debido a lo complicado que resulta crear relaciones y patrones acerca de la gran cantidad de información que pueden expresar las palabras, más aún teniendo en cuenta recursos expresivos como el sarcasmo o la gran diversidad del léxico en el lenguaje español.

También destacan los buenos resultados obtenidos, con ambos modelos analizados y en especial el modelo RoBERTuito, con resultados tan asombrosos como un 0.96 de accuracy obtenido con el conjunto de datos de salud mental para dos polaridades, lo que se acerca mucho a una puntuación perfecta.

Se adjunta en la bibliografía un enlace a un repositorio GitHub donde se encuentran los documentos generados durante este trabajo [62]

7.2. Líneas futuras

Como posibles líneas de trabajo futuras se plantea la creación o adaptación de un modelo transformers a un nivel más bajo, ya que al haber servido este trabajo como vía de entrada a la librería transformers no se ha podido realizar.

Esta tarea quizás deba realizarse de forma independiente a la librería utilizada ya que durante la fase de documentación no se encontró la posibilidad de crear un modelo totalmente de cero, permitiendo solo el uso de modelos ya preentrenados y la posibilidad de realizar un fine-tuning para adaptarlo al dominio de trabajo o modificar la tarea que realiza.

También se podrían investigar métodos para mejorar las predicciones de las clases neutra e indeterminadas, ya que al ser las menos polarizadas suponen un mayor reto en tareas de clasificación.

Capítulo 8

Anexos

8.1. Requisitos de computación

Uno de los condicionantes a la hora de realizar este trabajo ha sido el de los requisitos computacionales que supone el entrenamiento de modelos deep learning.

Para el entrenamiento de este tipo de modelos se puede utilizar la CPU del ordenador, pero es muy desaconsejable por requerir de tiempos de ejecución mucho más largos que al utilizar la GPU o tarjeta dedicada de vídeo.

Esto es debido a la diferencia de núcleos de procesamiento entre ambos componentes informáticos. Cada núcleo es capaz de realizar una tarea específica individualmente, con lo que aún teniendo en cuenta las tecnologías de computación paralela desarrolladas para suplir estas carencias, la diferencia de números continúa siendo abrumadora. Es por esto que los procesos relacionados con la inteligencia artificial se ejecutan generalmente con GPU [63].

Además es importante saber que Tensorflow [64] tiene requerimientos específicos para su uso con GPU. Uno de los más destacados es el uso de tarjetas gráficas NVIDIA [65] con soporte de núcleos CUDA (Compute Unified Device Architecture), una plataforma plataforma de computación paralela y un modelo de programación que permite incrementos dramáticos en el rendimiento de computación al aprovechar la potencia de la GPU, según una de las primeras entradas del blog de desarrolladores de NVIDIA [66].

Recientemente se está habilitando la posibilidad de realizar estas tareas con tarjetas gráficas de la compañía AMD [67]gracias a su plataforma de código libre ROCm [68], que ya cuenta con soporte para las librerías Tensorflow y PyTorch [69].

También es importante tener instalados en la tarjeta gráfica los drivers de instalación y los paquetes compatibles con tu versión de CUDA que admite tu tarjeta gráfica.

Por todas estas limitaciones resulta más aconsejable hacer uso de entornos de ejecución con opción al uso de GPUs para el trabajo con redes neuronales y similares.

8.2. Entornos de computación

Para el desarrollo de este trabajo se ha utilizado el entorno de computación en la nube de Google, Google Colab [13] debido a que ofrece de forma gratuita los recursos computacionales necesarios para el desarrollo del trabajo y fue recomendado por los tutores del TFG por los buenos

resultados en su uso en trabajos anteriores.

Ofrece una gráfica NVIDIA Tesla T4, 12.69 GB de memoria RAM y 107.72 GB de almacenamiento.

Una de sus limitaciones en su versión gratuita es el tiempo máximo permitido para realizar ejecuciones de código. Al tratarse de un entorno pensado para ejecución dinámica tiene un tiempo límite de 12 horas continuas. El problema es que según las necesidades de los usuarios pueden decidir desconectar una sesión activa, perdiendo los resultados obtenidos hasta entonces. Esto ha sucedido en alguna ocasión durante ejecuciones de código de hasta 4 horas, lo que retrasó el trabajo.

También está el hecho de que las GPU no siempre se encuentran disponibles para los usuarios sin plan de pago, por lo que hubo días en los que no se pudo trabajar.

Para no volver a sufrir estos problemas se pagó una suscripción de 2 meses al plan de pago Colab Pro, el cual ofrece mayores tiempos de ejecución, mayores capacidades de RAM y almacenamiento acceso a gráficas y CPUs más potentes hasta agotar las unidades de computación disponibles. Esta suscripción tiene un coste de 11.19€ mensuales.

Kaggle [70] es otra plataforma interesante que ofrece un entorno gratuito. Cuenta con una GPU NVIDIA TESLA P100, memoria RAM de 16GB y espacio de disco de 5GB. Esta capacidad de almacenamiento es muy limitada para la cantidad de archivos generados por la librería transformers al realizar tareas como el entrenamiento de modelos.

Bibliografía

[1] *Pandemia de COVID-19 en España - Wikipedia la enciclopedia libre*. URL: https://es.wikipedia.org/wiki/Pandemia_de_COVID-19_en_Espa%5C%C3%5C%B1a (visitado 15-05-2023).

[2] *Attention Is All You Need*. URL: https://arxiv.org/abs/1706.03762 (visitado 22-05-2023).

[3] *Procesamiento del lenguaje natural Qué es y por qué es importante - SAS Institute*. URL: https://www.sas.com/es_es/insights/analytics/what-is-natural-language-processing-nlp.html (visitado 10-06-2023).

[4] *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. URL: https://arxiv.org/abs/1810.04805 (visitado 22-05-2023).

[5] *Twitter*. URL: https://twitter.com/ (visitado 20-05-2023).

[6] *Twitch*. URL: https://www.twitch.tv/ (visitado 20-05-2023).

[7] *Análisis de sentimientos en Twitter mediante técnicas de Deep Learning*. URL: https://uvadoc.uva.es/handle/10324/57316 (visitado 07-06-2023).

[8] *Illustrated Guide to Transformers - Step by Step Explanation*. URL: https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0 (visitado 20-05-2023).

[9] *GanttProject*. URL: https://www.ganttproject.biz/ (visitado 20-05-2023).

[10] *Proyecto/Guía docente de la asignatura Trabajo Fin de Grado Mención Computación para el curso 2022-2023*. URL: https://albergueweb1.uva.es/guia_docente/uploads/2022/545/46978/1/Documento.pdf (visitado 20-05-2023).

[11] *Salario medio para Ingeniero De Datos en España, 2023*. URL: https://es.talent.com/salary?job=ingeniero+de+datos (visitado 20-05-2023).

[12] *Evolución del precio de la electricidad*. URL: https://www.ocu.org/vivienda-y-energia/gas-luz/informe/precio-luz (visitado 20-05-2023).

[13] *Google Colab*. URL: https://colab.research.google.com/ (visitado 20-05-2023).

[14] *Inteligencia artificial - Wikipedia la enciclopedia libre*. URL: https://es.wikipedia.org/wiki/Inteligencia_artificial (visitado 20-05-2023).

[15] *What is strong AI? IBM*. URL: https://www.ibm.com/topics/strong-ai#:~:text=Weak%5C%20AI%5C%2C%5C%20also%5C%20known%5C%20as,to%5C%20solve%5C%20for%5C%20new%5C%20problems. (visitado 21-05-2023).

[16] *MACHINE LEARNING HISTORY: THE COMPLETE TIMELINE - STARTECH UP*. URL: https://www.startechup.com/blog/machine-learning-history/ (visitado 21-05-2023).

[17] *OpenAI ChatGPT*. URL: https://openai.com/blog/chatgpt (visitado 20-05-2023).

[18] *Microsoft Vall-E*. URL: https://www.microsoft.com/en-us/research/project/vall-e/ (visitado 20-05-2023).

[19] *Midjourney*. URL: https://www.midjourney.com/home/ (visitado 20-05-2023).

[20] *Deepfakes being used in 'sextortion' scams, FBI warns*. URL: https://www.theregister.com/2023/06/08/ai_deepfakes_sextortion_fbi/ (visitado 13-05-2023).

[21] *Artists Are Suing Artificial Intelligence Companies and the Lawsuit Could Upend Legal Precedents Around Art*. URL: https://www.artnews.com/art-in-america/features/midjourney-ai-art-image-generators-lawsuit-1234665579/ (visitado 13-05-2023).

[22] *The AI act*. URL: https://artificialintelligenceact.eu/ (visitado 20-05-2023).

[23] *How AI is reshaping cancer diagnosis and treatment*. URL: https://www.labiotech.eu/opinion/ai-reshaping-cancer-diagnosis-treatment/ (visitado 13-05-2023).

[24] *SAP - ¿Qué es machine learning?* URL: https://www.sap.com/spain/products/artificial-intelligence/what-is-machine-learning.html (visitado 13-05-2023).

[25] *Machine Learning and Deep Learning Approaches for Brain Disease Diagnosis: Principles and Recent Advances*. URL: https://www.researchgate.net/publication/349643727_Machine_Learning_and_Deep_Learning_Approaches_for_Brain_Disease_Diagnosis_Principles_and_Recent_Advances (visitado 10-03-2023).

[26] *Artificial Intelligence vs. Machine Learning vs. Deep Learning: What's the Difference?* URL: https://www.sumologic.com/blog/machine-learning-deep-learning/ (visitado 10-03-2023).

[27] *An Empirical Analysis of Generative Adversarial Network Training Times with Varying Batch Sizes*. URL: https://www.researchgate.net/publication/344544069_An_Empirical_Analysis_of_Generative_Adversarial_Network_Training_Times_with_Varying_Batch_Sizes (visitado 10-03-2023).

[28] *Adam: A Method for Stochastic Optimization*. URL: https://arxiv.org/abs/1412.6980 (visitado 20-05-2023).

[29] *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. URL: https://arxiv.org/abs/1910.10683 (visitado 21-05-2023).

[30] *BookCorpus Dataset*. URL: https://paperswithcode.com/dataset/bookcorpus (visitado 20-05-2023).

[31] Thomas 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*. Online: Association for Computational Linguistics, oct. de 2020, págs. 38-45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[32] *Welcome to Python.org*. URL: https://www.python.org/ (visitado 20-05-2023).

[33] *Wandb*. URL: https://wandb.ai/site (visitado 25-05-2023).

[34] *The LaTex Project*. URL: https://www.latex-project.org/ (visitado 20-05-2023).

[35] *Overleaf, Editor de LaTex online*. URL: https://es.overleaf.com/ (visitado 20-05-2023).

[36] *Elon Musk compra Twitter: cómo se fraguó .el acuerdo más loco de la historia de Silicon Valley& qué puede cambiar en la red social*. URL: https://www.bbc.com/mundo/noticias-63429142 (visitado 20-05-2023).

[37] *Amazon*. URL: https://www.amazon.es/ (visitado 20-05-2023).

[38] *14 - Deep autoencoder-based automated brain tumor detection from MRI data*. URL: https://www.sciencedirect.com/science/article/abs/pii/B9780323911979000138 (visitado 10-03-2023).

[39] *TFM José Carlos Sobrino Sande*. URL: https://github.com/jcsobrino/TFM-Analisis_sentimientos_Twitter-UOC (visitado 15-05-2023).

[40] *Aprendizaje automático aplicado al análisis de sentimientos*. URL: http://portal.amelica.org/ameli/jatsRepo/339/3391369008/html/ (visitado 20-05-2023).

[41] Juan Manuel Pérez, Juan Carlos Giudici y Franco Luque. *pysentimiento: A Python Toolkit for Sentiment Analysis and SocialNLP tasks*. 2021. arXiv: 2106.09462 [cs.CL].

[42] *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. URL: https://arxiv.org/abs/1907.11692 (visitado 30-05-2023).

[43] *BERT: State of the Art NLP Model, Explained*. URL: https://www.kdnuggets.com/2018/12/bert-sota-nlp-model-explained.html#:~:text=Next%5C%20Sentence%5C%20Prediction%5C%20(NSP),sentence%5C%20in%5C%20the%5C%20original%5C%20document. (visitado 13-06-2023).

[44] *pysentimiento/roberta-es-sentiment - Hugging Face*. URL: https://huggingface.co/pysentimiento/roberta-es-sentiment (visitado 30-05-2023).

[45] *RoBERTuito: a pre-trained language model for social media text in Spanish*. URL: https://arxiv.org/abs/2111.09453 (visitado 13-06-2023).

[46] *BERTweet: A pre-trained language model for English Tweets*. URL: https://arxiv.org/abs/2005.10200 (visitado 13-06-2023).

[47] *pysentimiento/robertuito-sentiment-analysis - Hugging Face*. URL: https://huggingface.co/pysentimiento/robertuito-sentiment-analysis (visitado 30-05-2023).

[48] *Overview of TASS 2020: Introducing Emotion Detection*. URL: https://ceur-ws.org/Vol-2664/tass_overview.pdf (visitado 01-06-2023).

[49] *pysentimiento: A Python toolkit for Sentiment Analysis and Social NLP tasks*. URL: https://github.com/pysentimiento/pysentimiento (visitado 03-06-2023).

[50] *Tokenizer*. URL: https://huggingface.co/docs/transformers/main_classes/tokenizer (visitado 14-05-2023).

[51] *Token_type_ids*. URL: https://huggingface.co/docs/transformers/glossary#token-type-ids (visitado 14-05-2023).

[52] *Hyperparameter Search using Trainer API*. URL: https://huggingface.co/docs/transformers/hpo_train (visitado 25-05-2023).

[53] Flor Miriam Plaza-del-Arco et al. "EmoEvent: A Multilingual Emotion Corpus based on different Events". English. En: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, mayo de 2020, págs. 1492-1498. ISBN: 979-10-95546-34-4. URL: https://www.aclweb.org/anthology/2020.lrec-1.186.

[54] *Paul Ekman*. URL: https://es.wikipedia.org/wiki/Paul_Ekman (visitado 14-06-2023).

[55] *Unsupervised Cross-lingual Representation Learning at Scale*. URL: https://arxiv.org/abs/1911.02116 (visitado 30-05-2023).

[56] *Common Crawl*. URL: https://commoncrawl.org/ (visitado 14-06-2023).

[57] *A Quick Guide to Low-Resource NLP - MLOps Community*. URL: https://mlops.community/a-quick-guide-to-low-resource-nlp/#:~:text=Low%5C%2Dresource%5C%20languages%5C%20lack%5C%20data,can%5C%20also%5C%20be%C2%BA%5C%20low%5C%2Dresource. (visitado 14-06-2023).

[58] *EmoEvalEs@IberLEF 2021*. URL: https://competitions.codalab.org/competitions/28682 (visitado 05-06-2023).

[59] *IberLEF 2023*. URL: https://sites.google.com/view/iberlef-2023/home (visitado 05-06-2023).

[60] *daveni/twitter-xlm-roberta-emotion-es - Hugging Face*. URL: https://huggingface.co/daveni/twitter-xlm-roberta-emotion-es (visitado 30-05-2023).

[61] *pysentimiento/robertuito-emotion-analysis - Hugging Face*. URL: https://huggingface.co/pysentimiento/robertuito-emotion-analysis (visitado 30-05-2023).

[62] *javeste/BERTsentiment*. URL: https://github.com/javeste/BERTsentiment (visitado 20-06-2023).

[63] *Why GPU Can Process Image Much Faster than CPU? - E2E Networks*. URL: https://www.e2enetworks.com/blog/why-gpu-can-process-image-much-faster-than-cpu (visitado 14-03-2023).

[64] *Crea modelos de aprendizaje automático de nivel de producción con TensorFlow*. URL: https://www.tensorflow.org/?hl=es-419 (visitado 16-03-2023).

[65] *NVIDIA: Líder mundial en computación de inteligencia artificial*. URL: https://www.nvidia.com/es-es/ (visitado 16-03-2023).

[66] *Qué son los Nvidia CUDA Cores y cuál es su importancia - Profesional Review*. URL: https://www.profesionalreview.com/2018/10/09/que-son-nvidia-cuda-core/ (visitado 14-03-2023).

[67] *AMD — juntos avanzamos$_A$I*. URL: https://www.amd.com/es.html (visitado 16-03-2023).

[68] *ROCm™: Machine Learning — AMD*. URL: https://www.amd.com/en/graphics/servers-solutions-rocm-ml (visitado 16-03-2023).

[69] *PyTorch*. URL: https://pytorch.org/ (visitado 16-03-2023).

[70] *Kaggle: Your Machine Learning and Data Science Community*. URL: https://www.kaggle.com/ (visitado 10-03-2023).