



**Universidad
Europea**

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MASTER UNIVERSITARIO EN ANALISIS DE DATOS MASIVOS (BIG DATA)

TRABAJO FIN DE MÁSTER

CLASIFICACIÓN DE NOTICIAS MEDIANTE EL USO DE MODELOS

BASADOS EN TRANSFORMERS

GERARD MARTÍNEZ CAÑETE

Dirigido por

Ingeniero HUGO FERNÁNDEZ VISIER

CURSO 2022-2023

TÍTULO: CLASIFICACIÓN DE NOTICIAS MEDIANTE EL USO DE MODELOS BASADOS EN TRANSFORMERS

AUTOR: GERARD MARTÍNEZ CAÑETE

TITULACIÓN: MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS (BIG DATA)

DIRECTOR DEL PROYECTO: Ingeniero HUGO FERNÁNDEZ VISIER

FECHA: OCTUBRE DE 2023

RESUMEN

En este Trabajo Final de Máster, se aborda la clasificación de textos de noticias utilizando modelos de Procesamiento del Lenguaje Natural (NLP) basados en Transformers.

En primer lugar, se establece un marco teórico detallado en el que se analiza el estado del arte y sienta las bases para la experimentación práctica.

En una segunda fase, se emplean modelos preentrenados de Hugging Face y se realizan ajustes finos (fine-tuning) utilizando un conjunto de datos específico, obtenido y preparado expresamente para este proyecto. A continuación, se realiza un análisis y comparativa de los resultados obtenidos.

De las conclusiones se destacan tanto las ventajas como los desafíos inherentes al uso de Transformers en la clasificación de noticias.

La memoria concluye con una serie de recomendaciones y una propuesta de posibles líneas de investigación futura.

Palabras clave: Procesamiento del Lenguaje Natural (PLN), Transformers, Clasificación de noticias, Hugging Face, Modelos preentrenados, Fine-Tuning.

ABSTRACT

In this Master's Final Project, we address the classification of news texts using Natural Language Processing (NLP) models based on Transformers.

To begin with, a detailed theoretical framework is established, and the state-of-the-art is analyzed, which lays the basis for practical experimentation.

In a second phase, pre-trained Hugging Face models are employed, and fine-tuning is performed using a specific dataset, obtained, and prepared specifically for this project. Then an analysis and comparison of the obtained results is conducted.

The conclusions emphasize both the advantages and the challenges involved in the use of Transformers in news text classification.

The report concludes by providing recommendations and outlining potential lines for further research.

Key words: Natural Language Processing (NLP), Transformers, News Classification, Hugging Face, Pretrained Models, Fine-Tuning.

AGRADECIMIENTOS

Con la entrega de esta memoria y su correspondiente defensa finaliza este Máster en Análisis de Datos Masivos (Big Data) en la Universidad Europea de Madrid, un año repleto de desafíos que han requerido mucho esfuerzo y sacrificio, pero que también han aportado gran satisfacción y crecimiento personal y profesional.

Por esto mismo, quiero aprovechar este momento para agradecer a aquellas personas que me han acompañado en esta aventura haciendo que todo fuera más llevadero.

En primer lugar, quiero agradecer a Tamara, mi pareja y futura madre de nuestro hijo. Tu paciencia, amor y apoyo incondicional me han ayudado y motivado para seguir adelante. Gracias por estar a mi lado y por ayudarme a alcanzar cada una de las metas propuestas.

En segundo lugar, quiero agradecer a mi madre, que siempre me ha animado a progresar y a seguir adelante. De todo corazón, gracias.

En tercer lugar, quiero agradecer a mi tutor de este TFM Hugo Fernández Visier. Su paciencia, orientación y disposición han sido esenciales para la elaboración de este trabajo. Gracias por dedicar tanto tiempo, esfuerzo y conocimientos para guiarme en este proceso. Ha sido un verdadero placer y una suerte coincidir con alguien tan dispuesto y de quién poder aprender tanto.

A todos vosotros, GRACIAS.

Gerard

TABLA RESUMEN

	DATOS
Nombre y apellidos:	Gerard Martínez Cañete
Título del proyecto:	Clasificación de noticias mediante el uso de modelos basados en Transformers
Directores del proyecto:	Hugo Fernández Visier
El proyecto ha consistido en el desarrollo de una investigación o innovación:	SI
Objetivo general del proyecto:	Obtención de un modelo de Aprendizaje Automático para clasificación de noticias basado en Transformers

Índice

RESUMEN	3
ABSTRACT	4
TABLA RESUMEN	6
Capítulo 1. RESUMEN DEL PROYECTO.....	12
1.1 Contexto y justificación	12
1.2 Planteamiento del problema	12
1.3 Objetivos del proyecto	13
1.4 Resultados obtenidos	13
1.5 Estructura de la memoria	13
Capítulo 2. ANTECEDENTES / ESTADO DEL ARTE	15
2.1 Antecedentes.....	15
2.2 Estado del arte.....	27
2.3 Conclusiones del marco teórico	38
Capítulo 3. OBJETIVOS.....	39
3.1 Objetivos generales	39
3.2 Objetivos específicos	39
3.3 Beneficios del proyecto	39
Capítulo 4. DESARROLLO DEL PROYECTO	40
4.1 Planificación del proyecto	40
4.2 Descripción de la solución, metodologías y herramientas empleadas	40
4.3 Recursos requeridos	46
4.4 Presupuesto	47
4.5 Test y evaluación de modelos preentrenados	48
4.6 Test y evaluación modelos con Fine-tuning	50
4.7 Comparativa resultados modelos preentrenados vs fine-tuned	52
Capítulo 5. DISCUSIÓN	53
5.1 Resultados del proyecto	53
Capítulo 6. CONCLUSIONES.....	54
6.1 Conclusiones	54

6.2	Lecciones aprendidas	55
6.3	Sugerencias de mejora	55
Capítulo 7.	FUTURAS LÍNEAS DE TRABAJO	56
Capítulo 8.	REFERENCIAS.....	57
Capítulo 9.	ANEXOS	61
	ANEXO 1: Relación de archivos de código disponibles en el repositorio Github personal.	61
	ANEXO 2: Métricas utilizadas.....	62
	ANEXO 3: Hugging Face AutoTrain.....	63

Índice de Figuras

Ilustración 1- Imágenes generada por DALL-E. Elaboración propia	16
Ilustración 2- SVM: Proyección de un espacio bidimensional a un espacio tridimensional. Elaboración propia.	17
Ilustración 3- Arquitectura Red Neuronal Convolucional. Elaboración propia	18
Ilustración 4- Arquitectura Feed-forward Neural Network. Elaboración propia.....	19
Ilustración 5- Funcionamiento de una Red Neuronal Recurrente. Elaboración propia	20
Ilustración 6- Procesamiento de un texto por una RNN. Elaboración propia.	20
Ilustración 7- Arquitectura LSTM. Elaboración propia.....	21
Ilustración 8- Normalización, Tokenización, Lematización y eliminación de stop words. Elaboración propia	23
Ilustración 9- One-hot encoding. Elaboración propia.....	24
Ilustración 10- Relaciones semánticas Word2vec. Elaboración propia	25
Ilustración 11- Aprendizaje por transferencia. Elaboración propia.....	26
Ilustración 12- Ejemplo Seq2Seq. Elaboración propia	28
Ilustración 13- Arquitectura Transformers (Vaswani et al, 2017).	28
Ilustración 14- Arquitectura Transformer: Encoder-Decoder stack y subcapas. Elaboración propia a partir de ^[34]	29
Ilustración 15- Arquitectura Transformer: Capas Linear y Softmax. Elaboración propia	30
Ilustración 16- Tipos de atención. (Vaswani et al, 2017) ^[6]	30
Ilustración 17 - Aparición cronológica modelos NLP basados en Transformer. (Amatriain et al, 2023) ^[34]	32
Ilustración 18- Aparición cronológica y parámetros modelos NLP basados en Transformer. (Amatriain et al, 2023) ^[34]	32
Ilustración 19 - Agrupación por familias. (Amatriain et al, 2023) ^[34]	33
Ilustración 20- Transformer Pipeline para clasificación de textos. Fuente: https://github.com/somosnlp/recursos/blob/main/aplicaciones_de_los_transformers.ipynb	34
Ilustración 21- Hugging Face. Página principal. Elaboración propia.....	35
Ilustración 22- Hugging Face: Trending models, datasets and spaces. Elaboración propia.	35
Ilustración 23- Hugging Face: OpenLLM Leaderboard. Elaboración propia.	36
Ilustración 24- Hugging Face: AutoTrain. Elaboración propia.	36
Ilustración 25- Hugging Face: Página de un modelo. Elaboración propia.	37

Ilustración 26- Hugging Face: Instancia de AutoTrain Advanced. Elaboración propia	38
Ilustración 27- Diagrama de Gantt planificación del proyecto. Elaboración propia.	40
Ilustración 28- BigQuery detalles dataset url_analysis. Elaboración propia	41
Ilustración 29- BigQuery columnas metadatos dataset url_analysis. Elaboración propia	41
Ilustración 30- Número de noticias diarias publicadas en el último año. Elaboración propia	42
Ilustración 31.- Filtrado del Dataset de entrenamiento. Elaboración propia	43
Ilustración 32- Histograma: Número de palabras Train Dataset. Elaboración propia.....	43
Ilustración 33- Elementos por categoría en el Dataset. Elaboración propia	44
Ilustración 34 - Hiperparámetros fine-tuning. Elaboración propia	46
Ilustración 35- Matriz de confusión de bert-base-spanish-wwm-cased preentrenado. Elaboración propia	49
Ilustración 36 - Matriz de confusión de distilbert-base-uncased preentrenado. Elaboración propia	49
Ilustración 37 - Matriz de confusión de GPT2 preentrenado. Elaboración propia	49
Ilustración 38- Matriz de confusión bert-base-spanish-wwm-cased finetuned. Elaboración propia	51
Ilustración 39- Matriz de confusión BERT AutoTrain finetuned. Elaboración propia.....	51
Ilustración 40- Matriz de confusión DistilBERT finetuned. Elaboración propia.....	51
Ilustración 41 - Matriz de confusión GPT2 finetuned. Elaboración propia	51
Ilustración 42- AutoTrain. Página principal. Elaboración propia	63
Ilustración 43- AutoTrain. Configuración nuevo proyecto. Elaboración propia	63
Ilustración 44- AutoTrain. Selección de los datos. Elaboración propia	64
Ilustración 45 - AutoTrain. Configuración del entrenamiento. Elaboración propia	64
Ilustración 46 - AutoTrain. Entrenamiento del modelo. Elaboración propia.	65
Ilustración 47 - AutoTrain. Métricas del modelo tras Fine-tuning. Elaboración propia	65
Ilustración 48 - Hugging Face - Página del modelo. Elaboración propia	65
Ilustración 49- Hugging Face. Hosted inference API. Elaboración propia	65

Índice de Tablas

Tabla 1- Distribución tareas y tiempo de dedicación.	47
Tabla 2 - Requisitos de ejecución para los scripts y coste económico	47
Tabla 3- Métricas obtenidas modelos preentrenados.....	48
Tabla 4- Métricas obtenidas modelos con Fine-tuning	50
Tabla 5- Matrices de confusión modelos fine-tuned	51
Tabla 6 - Comparativa métricas BERTO preentrenado vs fine-tuned.....	52
Tabla 7- Comparativa métricas DistilBERT preentrenado vs fine-tuned	52
Tabla 8 - Comparativa métricas GPT2 preentrenado vs fine-tuned	52
Tabla 9 - Relación de scripts con código Python disponibles a través de Github.....	61

Capítulo 1. RESUMEN DEL PROYECTO

1.1 Contexto y justificación

El contexto en el que nos encontramos, una época en la que las noticias, fuentes y contenidos informativos proliferan en la web y redes sociales, llegando al punto de generar infodemia, un término aparecido tras la pandemia de COVID19 que se refiere al hecho de disponer de tal exceso de información que resulta confuso y abrumador.

Además, el acceso inmediato a la información proveniente de innumerables fuentes, algunas de las cuales de muy dudosa fiabilidad, plantea un grave problema en términos de organización, clasificación y verificación de su veracidad.

Ante esta situación, surge la principal pregunta que este proyecto pretende responder: ¿Cómo se pueden clasificar eficientemente las noticias aprovechando los modelos basados en Transformers en el ámbito del Procesamiento de Lenguaje Natural (NLP)?

1.2 Planteamiento del problema

La exposición a esta sobrecarga de información, con la propagación de noticias falsas en la red y la dificultad para detenerlas, hace necesaria la creación de sistemas capaces de organizar y clasificar las noticias. Por ello, y a pesar de que los modelos basados en arquitecturas Transformers aparecieron por primera vez en el año 2017, siguen siendo el estado del arte en procesamiento NLP.

Además de su capacidad para manejar dependencias a largo plazo y su flexibilidad para adaptarse a diferentes tareas, los Transformers permiten transfer-Learning (aprendizaje por transferencia), lo que posibilita la utilización de modelos previamente entrenados y ajustarlos posteriormente para una tarea específica mediante una técnica llamada fine-tuning.

Otro punto importante por considerar es la existencia de repositorios públicos de modelos preentrenados como The Hugging Face [\[1\]](#), que brindan a los usuarios comunes acceso a estas herramientas que han sido desarrolladas y entrenadas por empresas con una gran capacidad computacional y que, con unos pequeños ajustes, puedan ser adaptadas para realizar otras tareas ahorrándose el elevado coste que supone el entrenamiento completo de un modelo.

Este proyecto, por lo tanto, pretende abordar este problema en varias fases. En primer lugar, se pretende establecer una base teórica que permita comprenderlo. Seguidamente, se aspira a desarrollar una solución práctica a partir de la realización Fine-Tuning en alguno de los modelos preentrenados disponibles en The Hugging Face. Finalmente, se llevará a cabo una comparación de los resultados obtenidos antes y después de implementar estas modificaciones y se planteará una serie de líneas futuras de trabajo y mejoras.

1.3 Objetivos del proyecto

En el desarrollo de este proyecto se establece como objetivo general el desarrollo e implementación de un sistema de clasificación de noticias utilizando la tecnología de modelos basados en Transformers.

1.4 Resultados obtenidos

1. Memoria, fruto de la investigación del estado del arte de los modelos basados en Transformers, poniendo énfasis en las contribuciones recientes en el ámbito del procesamiento de lenguaje natural.
2. Modelos completamente funcionales con capacidad para clasificar textos de noticias dentro de un total de 9 categorías con una precisión superior al 90%.

1.5 Estructura de la memoria

La presente memoria se organiza en seis capítulos principales que siguen una secuencia lógica, partiendo de la presentación del problema, una exposición del marco teórico, el desarrollo práctico de la implementación de un modelo basado en Transformers para la clasificación de noticias y finalizando con las conclusiones y posibles líneas futuras:

- **Capítulo 1. Resumen del proyecto:** Se establece el contexto del trabajo, se identifica el problema a resolver, los objetivos generales y específicos y una breve descripción de la estructura de la memoria.
- **Capítulo 2. Antecedentes / Estado del arte:** En este capítulo se desarrolla toda la base teórica necesaria para comprender el trabajo realizado en este proyecto. Se introduce brevemente el concepto de Procesamiento del Lenguaje Natural (NLP), se hace una presentación de qué son los Transformers y cómo es su arquitectura. Además, se amplía esta información con un análisis en profundidad sobre el estado del arte en la materia, enfatizando la aplicación de los Transformers en la clasificación de textos y finalizando con una reflexión sobre el marco teórico presentado.
- **Capítulo 3. Objetivos:** Se detallan los objetivos del trabajo así como los beneficios que se pueden obtener tras su realización.
- **Capítulo 4. Desarrollo del proyecto:** En este capítulo se realiza el grueso del proyecto, es decir, se muestra la planificación seguida y se desarrolla todo el trabajo práctico necesario para la obtención de los modelos fine-tuned para la tarea de clasificación de noticias. También se exponen las métricas utilizadas para evaluar el desempeño de los modelos utilizados y se realiza un análisis detallado de los resultados obtenidos. Asimismo, se contrastan estos resultados con y sin la aplicación del “Fine-Tuning”, para así poder sacar conclusiones sobre el uso de esta técnica.
- **Capítulo 5. Discusión:**
- **Capítulo 6. Conclusiones:** En este capítulo se recogen las principales conclusiones del proyecto y se realiza una reflexión sobre las lecciones aprendidas durante el proceso.

- **Capítulo 7. Futuras líneas de trabajo:** Se proponen aquellas mejoras que podrían realizarse para mejorar los resultados, así como algunas de las posibles líneas de trabajo futuras en el ámbito de este proyecto.
- **Capítulo 8. Referencias:** Listado de las referencias y fuentes de información utilizadas para la realización de este proyecto.
- **Capítulo 9: Anexos:** Material extra que se ha considerado de interés incluir en este trabajo pero sin tener peso suficiente para aparecer en la memoria.

Capítulo 2. ANTECEDENTES / ESTADO DEL ARTE

2.1 Antecedentes

2.1.1 Introducción al Procesamiento del Lenguaje Natural (NLP)

El Procesamiento del Lenguaje Natural, NLP por sus siglas en inglés, es una rama de la Inteligencia Artificial y el Aprendizaje Automático. Su objetivo es que las máquinas tengan la capacidad de comprender, interpretar y generar el lenguaje de manera similar a cómo lo utilizamos los humanos.

Dentro del Procesamiento del Lenguaje Natural, se destacan dos enfoques o disciplinas:

- Por un lado, **NLP Generativa** que se centra en la creación de texto de manera autónoma por parte de las máquinas. Como su propio nombre indica, implica la generación de contenido coherente y significativo con la intención de imitar la capacidad humana de producir lenguaje escrito.
- Por otro lado, la **NLU (Natural Language Understanding)** se orienta hacia la comprensión del lenguaje humano. Su objetivo es lograr que las máquinas entiendan el texto y extraer su significado. Esto implica tareas como el análisis de sentimientos, la extracción de información o la clasificación de textos.

2.1.1.1 Historia del Procesamiento del Lenguaje Natural ^[2]

En la década de 1950 Alan Turing, sugirió lo que se conoce como “Test de Turing” y plantea la posibilidad de que una máquina sea capaz de entender y comunicarse utilizando el mismo lenguaje que los humanos hasta el punto en el que no es posible distinguir si el interlocutor es una máquina o un ser humano.

Desde entonces han aparecido numerosos chatbots de los cuales se destacan:

- **“ELIZA”**: un chatbot diseñado por Joseph Weizenbaum del MIT en el 1966 que pretendía imitar las conversaciones con un psicólogo.
- **“SHRDLU”** ^[3]: un programa desarrollado en LISP por Terry Winograd del MIT entre los años 1968-1970 en el que el usuario se encontraba en un entorno llamado “blocks world”, daba instrucciones mediante el uso del lenguaje natural y el programa las interpretaba y cumplía.
- **“Eugene Goostman”**: un chatbot desarrollado por los rusos Vladimir Veselov, Sergey Ulasenque y el ucraniano Eugene Demchenko en el año 2001. Este chatbot pretendía hacerse pasar por un niño ucraniano de 13 años. En el año 2014 estuvo a punto de superar el test de Turing en el Premio Loebner al convencer al 29% de los jueces que lo evaluaron.
- **“IBM Watson”** ^[4]: una plataforma de IA desarrollada por IBM desde el año 2006. Se hizo muy popular en el año 2011 cuando logró ganar un concurso de televisión llamado “Jeopardy!”, en el que demostró su capacidad para entender y responder preguntas en

lenguaje natural. También se ha utilizado en apoyo al diagnóstico médico y como back-end para chatbots de atención al cliente.

- **“BERT”**: Fue desarrollado por Google y su nombre corresponde a las siglas de **“Bidirectional Encoder Representations from Transformers”** o Representación de Codificador Bidireccional de Transformadores. Se trata del primer modelo de lenguaje basado en Transformers.
- **“GPT”**: Generative Pre-Trained Transformer. Desarrollado por OpenAI se trata de una familia de modelos de lenguaje basados en Transformers. En los últimos años se ha hecho famosa debido a que su herramienta ChatGPT está abierta al público de manera gratuita.
- **“DALL-E”**: También desarrollado por OpenAI, permite generar imágenes a partir de descripciones detalladas en lenguaje natural. En el momento de escribir estas líneas, se abría al público de manera gratuita a través de Microsoft Bing.



Ilustración 1- Imágenes generada por DALL-E. Elaboración propia

2.1.1.2 Aplicaciones del Procesamiento del Lenguaje Natural

Dada la evolución que ha tenido en los últimos años y la expectación generada, la utilización de la IA, y en particular de herramientas que hacen uso del NLP, hoy en día se pueden encontrar numerosas utilidades que hacen más fácil y cómodo el día a día de miles de personas. Algunos de los usos más extendidos son:

- **Asistentes Virtuales**: Los asistentes virtuales como Siri, Alexa y Google Assistant, entre otros, utilizan NLP para comprender y responder preguntas en lenguaje natural. Lo que implica la interacción con dispositivos electrónicos, la obtención de información o la realización de tareas con órdenes verbales.
- **Traducción Automática**: Plataformas como Google Translate o DeepL utilizan NLP para la traducción de textos entre distintos idiomas con una gran precisión y fiabilidad.

- **Servicios de Atención al Cliente:** Cada vez más empresas utilizan chatbots y otros sistemas de NLP para ofrecer a sus clientes atención de manera ininterrumpida, lo que permite agilizar la resolución de consultas, dudas y problemas.
- **Búsqueda de información:** Los motores de búsqueda como Google o Bing hacen uso de técnicas de NLP para mejorar los resultados obtenidos.
- **Generación de contenido:** Últimamente se ha expandido ^[5] el uso de NLP para la generación automática de cualquier tipo de contenido como noticias o redes sociales.
- **Análisis de sentimiento:** Promovido por el uso de redes sociales y las reseñas de los clientes, se utiliza NLP para analizar el sentimiento de los comentarios para comprender la opinión de los usuarios y actuar en consecuencia.

2.1.1.3 Modelos de Lenguaje

El objeto de este trabajo es el uso de Transformers para la clasificación de noticias, y los Transformers no aparecieron hasta el 2017 con el artículo “Attention Is All You Need” (Vaswani et al., 2017^[6]). Esta arquitectura supuso un cambio en el paradigma del Procesamiento Natural del Lenguaje (NLP) por las múltiples ventajas que supone su uso frente a otros modelos, sin embargo, también resulta de interés conocer los modelos utilizados con, porque hoy en día siguen siendo de vital importancia y se utilizan para otro tipo de tareas dentro de la Inteligencia Artificial y el Aprendizaje Automático.

Un modelo de lenguaje trata de predecir la siguiente palabra de un texto a partir de las anteriores. Para ello se pueden utilizar:

2.1.1.3.1 Modelos de lenguaje probabilísticos que utilizan técnicas estadísticas:

Un tipo de modelo de lenguaje probabilístico son las **Máquinas de Vectores de Soporte (SVM)**, que se utilizan en tareas de clasificación y regresión y se basan en el concepto de maximizar el margen entre las clases. Para ello, se representan las observaciones como puntos en un espacio de dimensión n , donde n es el número de características del conjunto de datos. El objetivo es hallar el hiperplano más óptimo posible que pueda clasificar correctamente los puntos en sus respectivas clases ^[7].

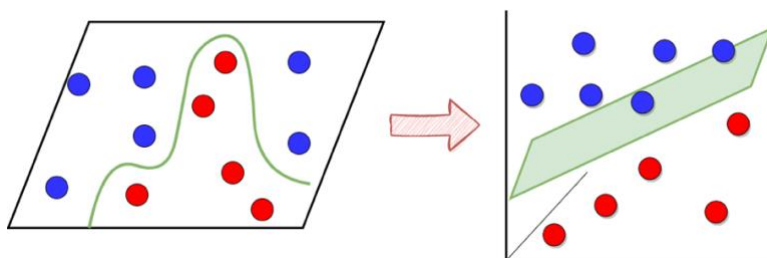


Ilustración 2- SVM: Proyección de un espacio bidimensional a un espacio tridimensional. Elaboración propia.

Estos modelos ofrecen un buen rendimiento en espacios de alta dimensionalidad; sin embargo cuando los datos no son linealmente separables, es necesario proyectar los datos en una dimensión superior para poder separarlos. Esto requiere el uso de funciones de kernel lo que aumenta la complejidad computacional.

Entre las características de los modelos de lenguaje probabilísticos encontramos:

- Tratan de predecir qué palabra será la n , sabiendo las $n-1$ anteriores.
- La distribución probabilística sólo tiene en cuenta las **palabras anteriores**.
- Cada probabilidad debe ser calculada y almacenada, con los consecuentes requisitos de cálculo y espacio de almacenaje. Por lo tanto, **no suelen escalar bien**.
- Es habitual que muchas palabras tengan la misma probabilidad, generalmente bajas, por lo que pueden surgir **problemas de escasez**.

2.1.1.3.2 Modelos de lenguaje basados en redes neuronales.

Las **Redes Neuronales** ^[8] están inspiradas en el funcionamiento del cerebro humano, y se componen por neuronas o nodos agrupadas en capas. Existen varios tipos de redes neuronales, entre ellas se destacan:

- **Redes Neuronales Convolucionales (CNN)** ^[9]

Las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) son un tipo de red neuronal ampliamente utilizado en el campo del reconocimiento de imágenes y tareas relacionadas con visión por computador. Por ello, su uso en el ámbito del Procesamiento del Lenguaje Natural (NLP) se limita principalmente al reconocimiento de caracteres para convertir imágenes en texto. A pesar de su uso sea más predominante en otros campos, se ha considerado necesario una breve explicación dada su importancia dentro del ámbito de la Inteligencia Artificial y el Aprendizaje Automático.

Su arquitectura se compone de varias capas y las principales son las siguientes:

- **Capa de entrada:** Esta primera capa recibe la imagen o dato de entrada.
- **Capas de convolución:** En estas capas se aplican los filtros (kernels) de convolución con el objetivo de detectar patrones.
- **Capas de agrupación o pooling:** En estas capas se reduce la dimensionalidad y cantidad de parámetros.
- **Capa totalmente conectada o perceptrón multicapa:** En esta última capa se realiza la clasificación final.
-

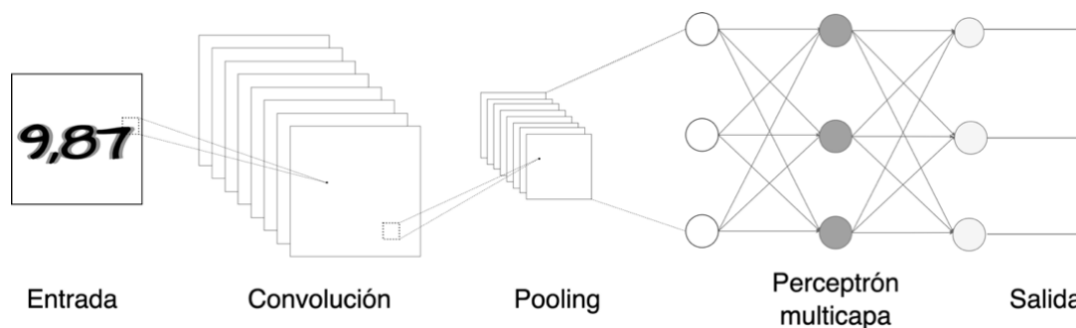


Ilustración 3- Arquitectura Red Neuronal Convolucional. Elaboración propia

Se debe remarcar que los datos de entrada utilizados por las Redes Neuronales Convolucionales son en forma de matriz, mientras que los utilizados en tareas de procesamiento del lenguaje son secuenciales. Por este motivo, entre otros, las tareas de NLP suelen abordarse utilizando otro tipo de redes neuronales, tal y como se exponen a continuación.

- **Feed-forward Neural Networks (FNN)** [\[10\]](#)

Se trata de un tipo de Red Neuronal de un solo sentido en el que no hay recurrencia. Los datos de entrada se obtienen a partir de la capa de entrada, en la que se utiliza un nodo para cada característica del dataset. Estos nodos de la capa de entrada se conectan con otros nodos ubicados en una o más capas ocultas cuya función principal es la transformación de las entradas, el aprendizaje de características abstractas y la identificación de patrones.

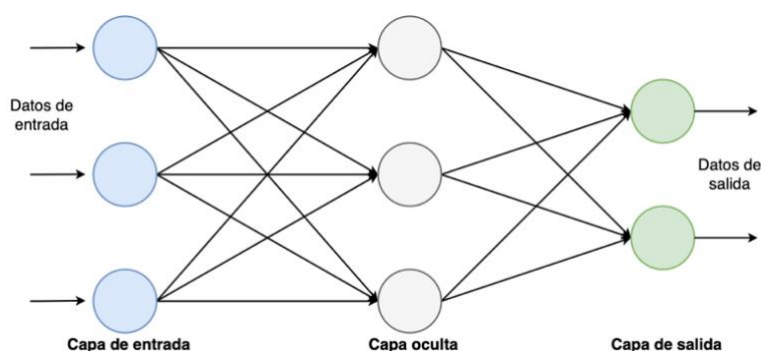


Ilustración 4- Arquitectura Feed-forward Neural Network. Elaboración propia

Las conexiones entre las neuronas están asociadas con pesos ponderados que sirven para controlar la contribución de cada neurona hacia la neurona de la capa siguiente a la que se conecta. Estos pesos se ajustan de manera iterativa durante el proceso de entrenamiento para dar más importancia a ciertas entradas y menos a otras.

La evaluación del error se lleva a cabo mediante una función denominada "**función de pérdida**," que se utiliza para medir el desempeño de la tarea. Por otro lado, la corrección del error se realiza mediante el **algoritmo de backpropagation**, que se encarga de ajustar los pesos y sesgos de la red neuronal en función de las discrepancias existentes entre los valores reales y las predicciones realizadas.

Sin embargo, este tipo de red neuronal tampoco resulta adecuado para tareas de Procesamiento de Lenguaje Natural (NLP) porque al no tener memoria, las predicciones de la próxima palabra en una secuencia de texto son pobres y poco precisas.

- **Redes Neuronales Recurrentes (RNN)** [\[11,12\]](#)

Las Redes Neuronales Recurrentes (RNN), son una arquitectura fundamental en el campo del NLP porque, a diferencia de las FNN, las RNN tienen conexiones retroalimentadas, es decir, tienen una memoria interna que les permite tener en cuenta los contextos anteriores en la secuencia de datos. Esto hace que las RNN sean especialmente adecuadas para tareas relacionadas con el NLP, ya que el contexto determina la comprensión de las palabras y las frases correctas en un texto.

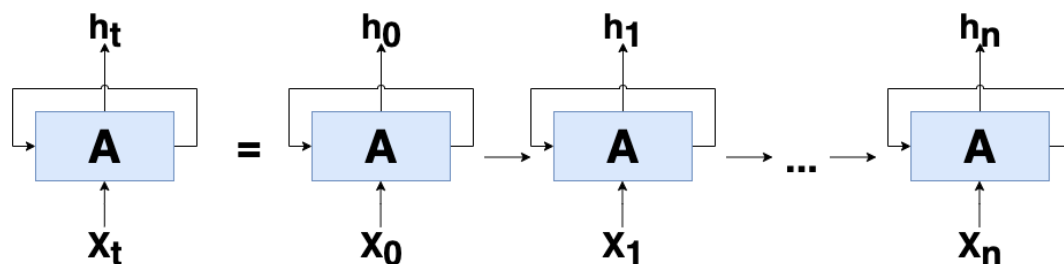


Ilustración 5- Funcionamiento de una Red Neuronal Recurrente. Elaboración propia

El funcionamiento de las RNN implica procesar los datos de entrada uno por uno, a medida que avanzan en la secuencia. La red toma la entrada actual y la combina con su estado interno anterior para generar una salida y una actualización de su estado interno.

Esto permite que las RNN mantengan una especie de "memoria" de las entradas previas, fundamental para tareas como la generación de texto, traducción automática y análisis de sentimientos.

Sin embargo, las RNN también tienen inconvenientes que dificultan la obtención de resultados precisos. Uno de estos inconvenientes son los problemas que presentan los gradientes ^[12], que son las derivadas parciales de la función de pérdida con respecto a los parámetros del modelo, y pueden suceder dos cosas:

- **Gradientes explosivos:** Durante el proceso de backpropagation se vuelven extremadamente grandes y haya problemas para converger o de inestabilidad numérica.
- **Gradientes desvanecientes (vanishing gradient):** En este caso, los gradientes se vuelven muy pequeños por lo que dificulta el aprendizaje, la convergencia se hace muy lenta o incluso nula y la información puede perderse a lo largo del backpropagation.

Otro problema derivado de estos es la dificultad para tratar con secuencias largas, ya que una RNN procesa las palabras secuencialmente y actualiza los pesos en función de las palabras anteriores.

Por ello, para tratar de determinar qué palabra irá en la posición n , sólo tiene en cuenta las $n-1$ palabras anteriores. Además, si se trata de un texto muy largo es posible que olvide alguna de las palabras iniciales o le dé un peso diferente al que le corresponde.

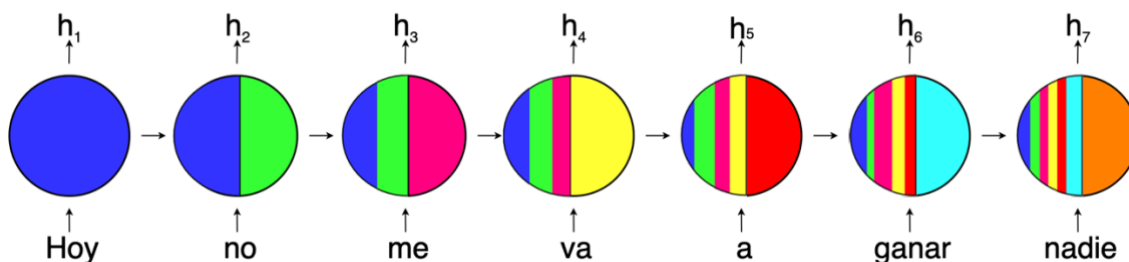


Ilustración 6- Procesamiento de un texto por una RNN. Elaboración propia.

En el ejemplo de la imagen, se muestra cómo las palabras van perdiendo importancia a medida que avanza la secuencia del texto y que llegado el punto, un modelo de lenguaje basado en RNN puede cometer errores al olvidar ciertas palabras que sean clave para el contexto de la frase.

Con el objetivo de abordar estas limitaciones, surgieron variantes más avanzadas basadas en RNN, como las Redes Neuronales de Memoria a Largo Plazo, **Long Short-Term Memory (LSTM)** o las Redes Neuronales con Atención, tal y como se verá a continuación.

En resumen, las RNN son una parte muy importante del NLP por su capacidad para modelar secuencias de datos, pero las desventajas con los gradientes y la retención de memoria a largo plazo hace que su uso sea limitado, sobre todo por la aparición de nuevas arquitecturas con resultados más sólidos y precisos.

▪ **Redes Neuronales de Memoria a Largo Plazo, Long Short-Term Memory (LSTM)** [\[13,33\]](#)

Las Redes LSTM, "Long Short-Term Memory" o "Memoria a Largo Plazo de Corto Plazo", son una variante especializada de las Redes Neuronales Recurrentes (RNN) diseñada específicamente para abordar el desafío de aprender dependencias a largo plazo en secuencias de datos. Fueron introducidas en el año 1997 por Hochreiter y Schmidhuber [\[14\]](#), y se han convertido en un componente fundamental en el campo del Procesamiento de Lenguaje Natural.

A diferencia de las RNN tradicionales, las LSTM resuelven el problema del "gradiente desvaneciente", lo que las hace ideales para tareas de NLP que requieren la comprensión de contextos largos y complejos en el lenguaje humano. Esto se logra gracias a una estructura interna de celdas y puertas que permite a la LSTM controlar y gestionar la información a medida que se procesan secuencias de texto.

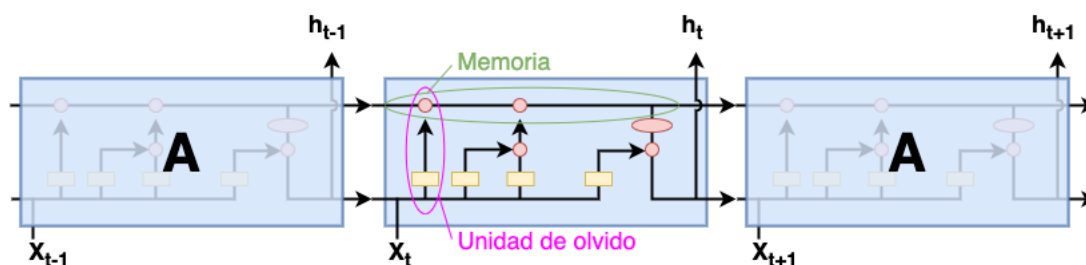


Ilustración 7- Arquitectura LSTM. Elaboración propia.

Las LSTM incorporan, principalmente, tres elementos que las distinguen de las RNN:

1. **Unidad o puerta de Olvido (Forget Gate):** Este elemento decide qué información previa en la celda de memoria debe olvidarse o mantenerse, permitiendo a la LSTM conservar información relevante a lo largo del tiempo.
2. **Puerta de Entrada (Input Gate):** La puerta de entrada determina qué nueva información se debe agregar a la celda de memoria, lo que permite a la LSTM capturar eventos o conceptos importantes en la secuencia.
3. **Memoria:** Se trata del componente de una unidad LSTM que almacena información a largo plazo y se modifica en función de la información que entra a través de la puerta de entrada y de la información que se mantiene u olvida a través de la puerta de olvido.

Con estas características, las LSTM han demostrado grandes resultados en aplicaciones de NLP como la traducción automática, el análisis de sentimientos, la generación de texto y muchas otras, mejorando la calidad y la capacidad de comprensión del procesamiento de lenguaje natural.

Sin embargo, las LSTM ciertas desventajas. Por un lado, tienen unos requisitos computacionales muy elevados y necesitan grandes cantidades de datos para el entrenamiento. Además, ya que los cálculos se realizan en serie, los tiempos de entrenamiento tienden a ser más largos que con otras arquitecturas.

2.1.1.4 Preprocesamiento de la entrada de datos

Cuando se inserta un texto en un modelo NLP es necesario realizar una serie de tareas para que el lenguaje humano sea comprendido por las máquinas. Este preprocesamiento de texto está formado por una serie de fases, no todas obligatorias y cuyo resultado impactará directamente en el desempeño del modelo.

Las fases principales de este proceso son las siguientes:

- **Normalización:** Transformación del texto a minúsculas y eliminación de signos de puntuación y demás símbolos (#, !, @...), así como de espacios extra, tabulaciones o saltos de línea.
- **Tokenización:** División del texto en palabras o incluso en unidades más pequeñas, como partes de éstas.
- Eliminación de **stopwords** ^[15]: Se eliminan aquellas palabras de uso común que no aportan significado relevante al texto como pueden ser ciertos artículos, pronombres de las que se abusa mucho al hablar y carecen de información.
- **Lematización:** Conversión de las palabras a su base o raíz, de modo que se pueda consolidar las distintas formas de una misma palabra (Género, número, conjugaciones...)
- **Representación de texto: Vectorización y/o Embeddings.** Se trata de la conversión del texto en vectores con valores numéricos a los que se le pueden aplicar modelos matemáticos para trabajar con estos datos.
- **Padding o truncamiento:** En función del tamaño del texto, si su longitud es inferior al tamaño de entrada permitido por el modelo, se añaden valores vacíos para completarlo. Por el contrario, si el tamaño es superior, el texto se trunca para adecuarlo al tamaño máximo.

En los siguientes subapartados se ampliará la explicación de las fases de tokenización, lematización y representación de texto.

2.1.1.4.1 Tokenización y lematización

Como bien se ha explicado en el apartado anterior, el texto de entrada de un modelo NLP debe ser procesado previamente y dos de las fases clave en este proceso son la tokenización y la lematización.

Por un lado, la **tokenización** es un proceso en el cual se convierte uso elementos, llamados **tokens**. Estos pueden ser palabras, fragmentos de palabras o incluso caracteres, dependiendo según el nivel de tokenización definido. Para realizar esta conversión se pueden aplicar ciertas normas como la de separar a partir de los espacios entre palabras, expresiones regulares o incluso directamente utilizando modelos de lenguaje entrenados para esta tarea y el idioma utilizado, lo cual puede ayudar a obtener una mejor tokenización

Por otro lado, la **lematización** es el proceso en el cual se reduce una palabra a su forma base o raíz, lo que en lingüística sería eliminar los morfemas de una palabra dejando únicamente el **lema**, de ahí el nombre de este proceso. Para realizar la lematización es necesario llevar a cabo un análisis morfológico de las palabras y eliminar aquellas inflexiones gramaticales, como el género y número o las conjugaciones y es importante el conocimiento del idioma en el que se realiza la tarea. Ejemplo: Aprendiendo → Aprender.

Existen librerías que llevan a cabo estas dos tareas de manera automatizada, entre ellas se destacan:

- **NLTK (Natural Language Toolkit):** “NLTK es una plataforma líder para crear programas Python para trabajar con datos del lenguaje humano. Proporciona interfaces fáciles de usar para más de 50 corpus y recursos léxicos como WordNet, junto con un conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico, contenedores para bibliotecas de PNL de potencia industrial, y un foro de discusión activo.” (NLTK Documentation, 2023 [\[16\]](#)).
- **SpaCy:** “spaCy es una biblioteca gratuita de código abierto para el procesamiento avanzado del lenguaje natural (NLP) en Python. spaCy está diseñado específicamente para uso en producción y ayuda a crear aplicaciones que procesan y “comprenden” grandes volúmenes de texto. Se puede utilizar para crear sistemas de extracción de información o de comprensión del lenguaje natural, o para preprocesar texto para un aprendizaje profundo.” (Spacy-101, 2023 [\[17\]](#)).
- **TensorFlow, PyTorch y Transformers:** Librerías de aprendizaje automático que disponen de herramientas y bibliotecas para el preprocesamiento de texto incluidas las tareas de tokenización y lematización.

```
Texto: Cada vez me gusta más este tema. Los Transformers son geniales!
Texto normalizado: cada vez me gusta más este tema los transformers son geniales
Tokens Transformers: ['cada', 'vez', 'me', 'gusta', 'más', 'este', 'tema', 'los', 'transform', '##ers', 'son', 'geniales']
Lematización: ['cada', 'vez', 'yo', 'gustar', 'más', 'este', 'tema', 'el', 'transform', '#', '#', 'ers', 'ser', 'genial']
Eliminación stop words: ['gustar', 'tema', 'transform', '#', '#', 'ers', 'genial']
```

Ilustración 8- Normalización, Tokenización, Lematización y eliminación de stop words. Elaboración propia

En la imagen anterior se muestra a partir de la ejecución de un [pequeño código](#) el proceso de la entrada de un texto que a partir de las librerías Transformers y spaCy se lleva a cabo la tokenización, lematización y finalmente se eliminan los stop words.

2.1.1.4.2 Representación de texto

La representación de texto se refiere a la conversión del texto en formatos que las máquinas puedan procesar, principalmente vectores o tensores. Es un paso clave en el NLP porque les permite poder trabajar con textos a partir de la aplicación de modelos matemáticos.

Existen numerosos tipos de representación de texto, de los cuales se destacan:

- **One-hot encoding** ^[18]

Dado un conjunto ordenado de n palabras, el vector de salida tendrá $n+1$ dimensiones y su codificación se realizará asignando un 1 a la posición que ocupe la palabra en el conjunto, dejando el resto a 0. Esta diferencia entre las n palabras y $n+1$ dimensiones del vector corresponden a que se añade una categoría UNK para poder codificar aquellas palabras que no estén en el conjunto n .

Sus principales inconvenientes son que no guarda relación entre las distintas palabras y que es un sistema que escala con dificultad ya que la dimensión de los vectores aumenta con cada palabra que se añade.

Palabra	One-hot encoding
Neurona	[1, 0, 0, 0]
Dato	[0, 1, 0, 0]
Aprender	[0, 0, 1, 0]
UNK	[0, 0, 0, 1]

Ilustración 9- One-hot encoding. Elaboración propia.

- **Bag of Words** ^[19]

Esta técnica convierte un texto en una matriz de términos y sus frecuencias, donde cada palabra se representa como una dimensión en un espacio vectorial y el valor en esa dimensión es la frecuencia con la que aparece la palabra en el texto. Es más efectiva que el One-hot encoding, pero no tiene en cuenta el orden en el que aparecen las palabras y no tiene en cuenta el contexto.

- **TF-IDF (Term Frequency-Inverse Document Frequency)** ^[20]

TF-IDF es un valor que mide la relevancia de una palabra para un documento dentro de una colección de documentos. Este valor se hace a partir de la multiplicación de dos métricas:

- **Term Frequency:** La frecuencia de la palabra en un documento.
- **Inverse Document Frequency:** La frecuencia de la palabra dentro del conjunto de documentos. Cuanto más frecuente sea, más cercano a 0 será este valor.

Al multiplicar estos dos números se obtiene la puntuación del TF-IDF de una palabra dentro de un documento y, cuanto mayor sea este valor, más importante se considera la palabra evaluada para ese documento en particular.

- **Embeddings** ^[21]

Estas técnicas transforman las palabras o textos en representaciones vectoriales y capturan tanto su significado semántico como la relación entre las palabras dentro del texto. Los resultados de este tipo de representación de texto son mejores que en las técnicas vistas hasta ahora. Existen numerosos tipos de embeddings, de los cuales se destacan:

- **Word2vec** ^[22]

En Word2vec la tabla de embedding tiene en cuenta para cada palabra, aquellas otras que tienden a aparecer de manera habitual cerca y para utilizar este tipo de codificación requiere que sea entrenado previamente.

Sin embargo, Word2vec no puede vectorizar palabras que no se ven durante el entrenamiento, por lo que no es posible manejar palabras no incluidas en el vocabulario y tampoco es capaz de solucionar la desambiguación, es decir, discernir entre palabras con distintos significados según el contexto.

Su estructura es lineal y las relaciones semánticas y sintácticas son relaciones lineales en el espacio vectorial:

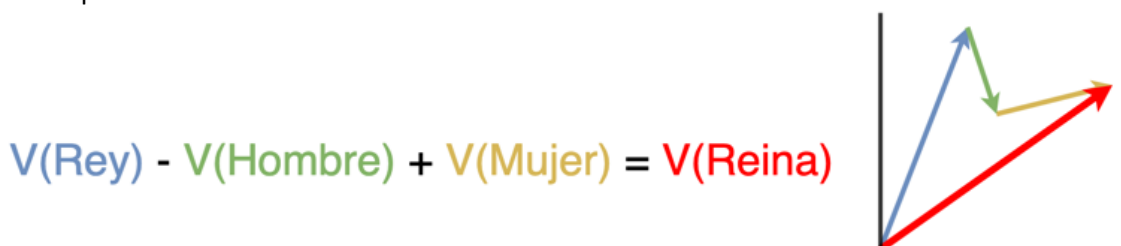


Ilustración 10- Relaciones semánticas Word2vec. Elaboración propia

Existen dos versiones de Word2vec: **Continuous Bag-of-words (CBOW)** y **SkipGram (SG)**. La primera versión predice la palabra central a partir de las palabras que la rodean y la segunda hace justo lo contrario: a partir de la palabra central, predice el contexto ^[22].

Existen múltiples extensiones de Word2Vec, como Doc2Vec donde cada documento al completo se asocia con un vector que captura su contenido semántico ^[23].

Otra extensión de Word2Vec es **FastText**, desarrollado por Facebook se diferencia de su predecesor porque tiene en cuenta las subpalabras o n-gramas de palabras y a partir de ello puede manejar palabras fuera del diccionario. Además es más rápido y sencillo de entrenar comparado con Word2Vec.

- **GloVe: Global Vectors for Word Representation** ^[24,25]

Desarrollado como un proyecto de código abierto en Stanford, el modelo se basa en observar la relación entre palabras a partir del cálculo de la frecuencia en que co-ocurren en un corpus de texto dado. De esta manera, cuando se vectoriza una palabra se tiene en cuenta todo el conjunto de datos que se utilizó para el entrenamiento en lugar de limitarse al texto de entrada.

- **ElMo: Embeddings from Language Models**^[26]

Se trata de un modelo de Word embeddings desarrollado por AllenNLP que utiliza redes LSTM bidireccionales. Su principal característica diferenciadora respecto a los demás modelos expuestos es que una misma palabra puede tener varias representaciones vectoriales en función del contexto en la que se encuentre. Esto permite diferenciar en aquellos casos donde las palabras pueden ser ambiguas.

2.1.1.5 Aprendizaje por transferencia

El entrenamiento de un modelo desde cero requiere gran capacidad computacional, grandes cantidades de datos y mucho tiempo de procesamiento. Esto se traduce en unos costes muy elevados que no todo el mundo se puede permitir. Afortunadamente, las técnicas de entrenamiento por transferencia, que permite la reutilización de modelos en tareas para las que no fueron desarrollados o entrenados.

El **aprendizaje por transferencia** ^[27,28,29] se basa en aprovechar el conocimiento adquirido durante entrenamiento previo en otra tarea. Se utiliza un modelo ya entrenado con un gran conjunto de datos como punto de partida para afrontar la tarea objetivo.

Esto conlleva grandes ventajas y beneficios como son unos requisitos más asequibles: se requiere menos potencia computacional, menores tiempo de entrenamiento, e incluso con una cantidad de datos reducida, se obtienen buenos resultados.

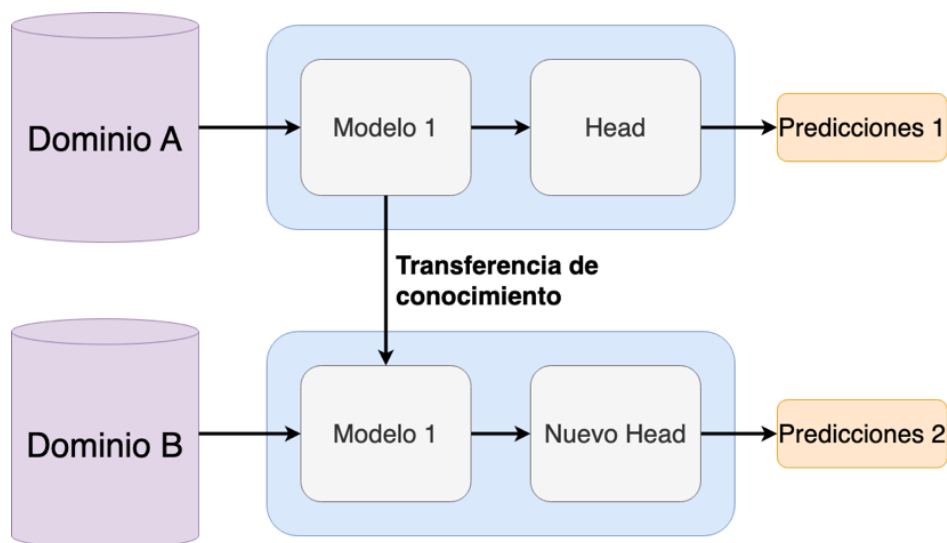


Ilustración 11- Aprendizaje por transferencia. Elaboración propia

El aprendizaje por transferencia sienta las bases para el **Fine-tuning**, una técnica fundamental para este trabajo, ya que a partir de un modelo preentrenado se puede afinar y optimizar el modelo para realizar una tarea específica, como la clasificación de textos de noticias.

2.2 Estado del arte

Los modelos basados en Transformers aparecieron en el año 2017 en el artículo "Attention Is All You Need" (Vaswani et al) ^[6] y en estos últimos 6 años, el desarrollo y expansión de la Inteligencia Artificial y el Procesamiento Natural del Lenguaje han sido revolucionarios ^[30].

Una de las características fundamentales de los Transformers es su capacidad para capturar relaciones en textos muy largos ^[31], lo que permite una mayor comprensión del contexto y de manera más efectiva que con otras arquitecturas utilizadas hasta la fecha. Para alcanzar este logro, se hace uso de mecanismos de atención, un componente clave que permiten al modelo enfocarse en partes específicas del texto y asignar diferentes pesos a las palabras según su relevancia.

Gracias a ello, los modelos basados en arquitecturas Transformers se han convertido en un elemento clave en tareas relacionadas con la traducción automática, ya que pueden aprender a traducir entre múltiples idiomas sin necesitar sistemas de alineación de frases, lo que ha simplificado los procesos y ha mejorado la calidad de las traducciones obtenidas.

En su lugar, los Transformers hacen uso de una técnica llamada "autoatención" o "self-attention", que les permite considerar todas las palabras en la secuencia de entrada al mismo tiempo y asignar diferentes pesos a cada una en función de su relevancia.

Sin embargo, la capacidad de los Transformers en materia de Procesamiento del Lenguaje Natural (NLP) no se limita a la traducción automática, sino que la capacidad para la comprensión y generación de texto coherente ha supuesto grandes avances en la generación del lenguaje natural y sus aplicaciones, como generación de noticias, narrativas o chatbots.

Además, la aparición de modelos de acceso libre como BERT, GPT o similares y los repositorios públicos como Hugging Face, ha supuesto la democratización ^[32] de la IA y el NLP, ya que han permitido que el público en general pueda utilizar modelos previamente entrenados y así aprovechar de las ventajas que ofrece esta tecnología.

El progreso en el desarrollo de la Inteligencia Artificial (IA) y el Procesamiento del Lenguaje Natural (NLP), con modelos como los Transformers, ha sido increíblemente rápido. Esta última década ha sido una revolución en este campo.

En este capítulo se presentarán y analizarán aquellas investigaciones, desarrollos y avances más relevantes y actuales sobre el Procesamiento Natural del Lenguaje en general y sobre la clasificación de noticias mediante el uso de modelos basados en Transformers.

2.2.1 Transformers

El artículo “Attention Is All You Need” (Vaswani et al, 2017) ^[6] introdujo los Transformers como una arquitectura secuencia-a-secuencia (Seq2Seq).

Se trata de un tipo red neuronal que convierte una secuencia de componentes en otra, en este caso, Transformers convierte una secuencia de palabras, como puede ser una frase o textos completos en otra.

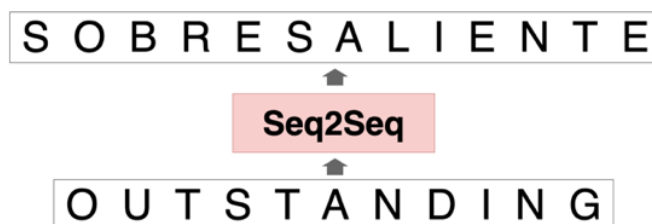


Ilustración 12- Ejemplo Seq2Seq. Elaboración propia

Uno de los motivos por lo que los Transformers resultan particularmente interesantes es la manera de procesar las entradas, ya que aprenden la frase al completo en paralelo, dejando atrás la computación secuencial utilizada por otros tipos de redes neuronales como RNN, CNN o LSTM, lo que permite establecer relaciones de largo alcance y determinar patrones complejos en el texto.

Para ello se utiliza la arquitectura que se observa en la siguiente imagen, extraída directamente del paper “Attention Is All You Need” (Vaswani et al, 2017) ^[6]:

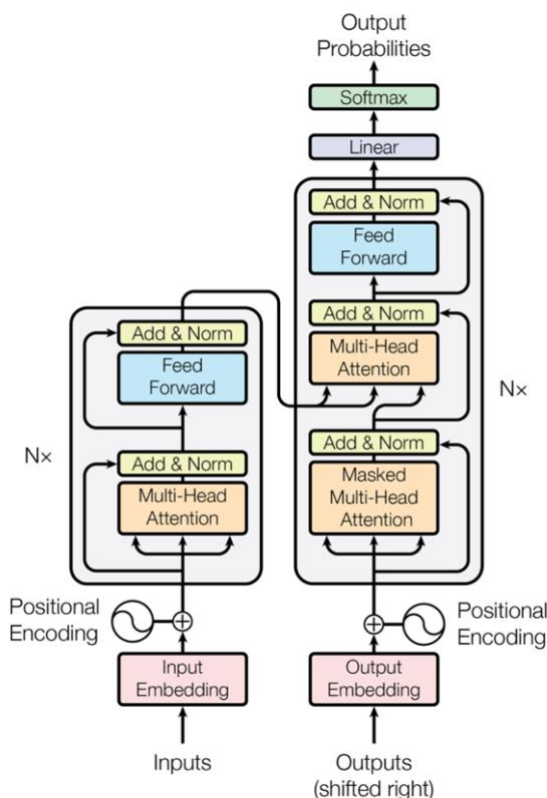


Ilustración 13- Arquitectura Transformers (Vaswani et al, 2017).

2.2.1.1 Funcionamiento de los Transformers

A partir de la imagen 13, donde se muestra la arquitectura al completo, se pueden diferenciar dos elementos principales de un Transformer: el **Encoder** y el **Decoder** ^[34].

Ambos componentes, encoders y decoders, son realmente una agrupación apilada de éstos con el mismo número de elementos y con la misma estructura interna con la única diferencia de los pesos, que varían de uno a otro.

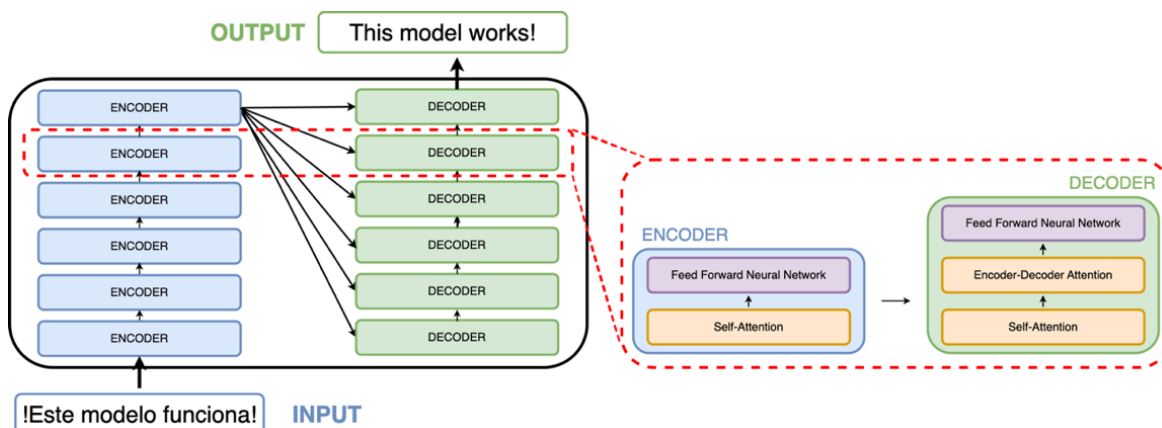


Ilustración 14- Arquitectura Transformer: Encoder-Decoder stack y subcapas. Elaboración propia a partir de ^[34]

Dentro de cada Encoder hay dos subcapas: una capa de atención que ayuda al codificador para tener en cuenta otras palabras de la frase mientras codifica una palabra en concreto. El proceso de atención se explica en el apartado 2.2.2.

En el interior del Decoder también están estas dos capas, sin embargo, entre ellas hay una capa de atención extra, que ayuda al Decoder a centrarse en los puntos más importantes del input.

A continuación del Decoder, se añaden dos capas:

- **Linear layer** (Capa Lineal): Se trata de una capa densa o totalmente conectada y transforma el vector producido por el Decoder en uno mucho mayor llamado vector de logits. Este vector es del tamaño de todas las palabras que conoce, es decir del tamaño vocabulario de salida y a cada celda le asigna la puntuación que le corresponde a cada palabra.
- **Capa Softmax**: Es una capa de activación que se encarga de transformar las puntuaciones asignadas por la Lineal layer en probabilidades, todas positivas y cuya suma da 1.0. De este modo, se escoge la celda con mayor probabilidad y la palabra asociada a esta se convierte en el output.

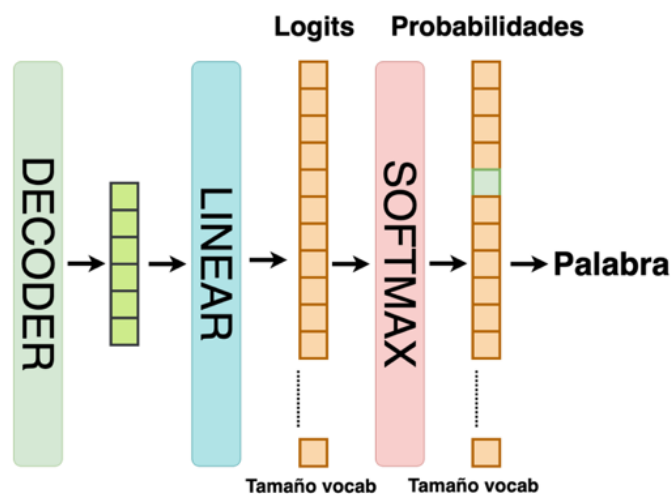


Ilustración 15- Arquitectura Transformer: Capas Linear y Softmax. Elaboración propia

2.2.2 Mecanismo de atención

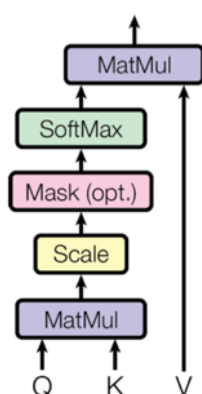
Una de las principales diferencias entre las redes neuronales vistas en el apartado 2.1.1.3.2 y el modelo Transformers es lo que se conoce como **atención**.

Se trata un mecanismo que permite al modelo centrarse en las partes importantes a la vez que tiene en cuenta el contexto, para así distinguir cuando una misma palabra tiene diferentes significados.

Este mecanismo de atención ha supuesto grandes mejoras en el rendimiento y resultados de numerosas tareas relacionadas con el NLP, como la traducción, donde el orden de las palabras puede variar en distintos idiomas o a la hora de interpretar textos complejos con palabras que tienen múltiples significados.

Su funcionamiento se basa en una función matemática que al recibir un conjunto de vectores realiza un mapeo entre una consulta y un conjunto clave-valor a una salida [\[35\]](#).

Scaled Dot-Product Attention



Multi-Head Attention

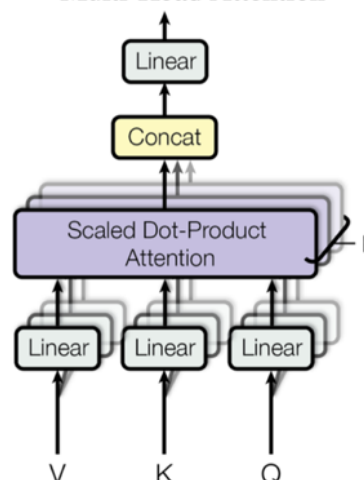


Ilustración 16- Tipos de atención. (Vaswani et al, 2017) [\[6\]](#)

Tipos de atención ^[36]:

Existen distintos tipos de mecanismos de atención en función de dónde se utiliza dentro de un modelo o en qué partes de la entrada se aplica el foco de atención. Según cada caso se puede encontrar los siguientes tipos:

- **Atención Generalizada:** Cuando el mecanismo de atención comprueba cada elemento de la entrada y lo compara con su correspondiente salida. De esta comparación se obtiene una puntuación que servirá para escoger el elemento al que prestar atención.
- **Self-attention:** cada vector decide la atención que presta al resto en función de la similitud entre vectores. De este modo, para calcular la atención que debe prestar un vector respecto a los demás, cuanto más parecidos sean, más importantes se considerarán y más atención se prestará.
- **Multi-head Attention:** Se trata de aplicar el mecanismo de atención múltiples veces obteniendo distintas salidas en distintas capas. Estas salidas se concatenan formando un vector cuya dimensión es la suma de las dimensiones de todas las capas y el resultado se obtiene aplicando una capa lineal para reducir la dimensionalidad final. Con este tipo de atención se consigue una combinación de las puntuaciones de atención de todas las “cabezas de atención”.
- **Atención Aditiva:** Realiza cálculos de puntuaciones para decidir qué parte del input es más importante y una vez tiene todas las puntuaciones las suma para determinar dónde centrará la atención.
- **Atención Global:** Se trata de una mejora de la atención aditiva ya que en lugar de realizar suma de puntuaciones, utiliza un modelo multiplicativo. Este tipo de atención puede centrarse en el conjunto completo de la entrada y por ello se ha mostrado de gran utilidad para traducciones por su precisión.
- **Hard Attention:** Cada vector del output atiende únicamente a un solo vector del input, en particular, el que está en su misma posición. Esto se consigue modificando los pesos para que la matriz o vector de atención sea todo ceros excepto a la posición a la que se pretende atender.
- **Soft Attention:** En este caso, cada vector del output atiende a todos los vectores del input, de modo que cada salida sea una combinación de todas las entradas, siempre en función de los pesos definidos previamente.

2.2.3 Modelos destacados en el NLP basados en Transformer

Con la publicación del artículo “Attention Is All You Need” (Vaswani et al, 2017) ^[6], el uso de modelos basados en la arquitectura Transformer ha proliferado de una manera sorprendente en el ámbito del Deep Learning (DL) y del NLP.

Los primeros modelos en surgir basados en Transformer fueron **GPT** de OpenAI y **BERT** de Google en el 2018. A lo largo de estos años han ido actualizándose a la par que han aparecido nuevas versiones y variantes, como se muestra en la ilustración 17.



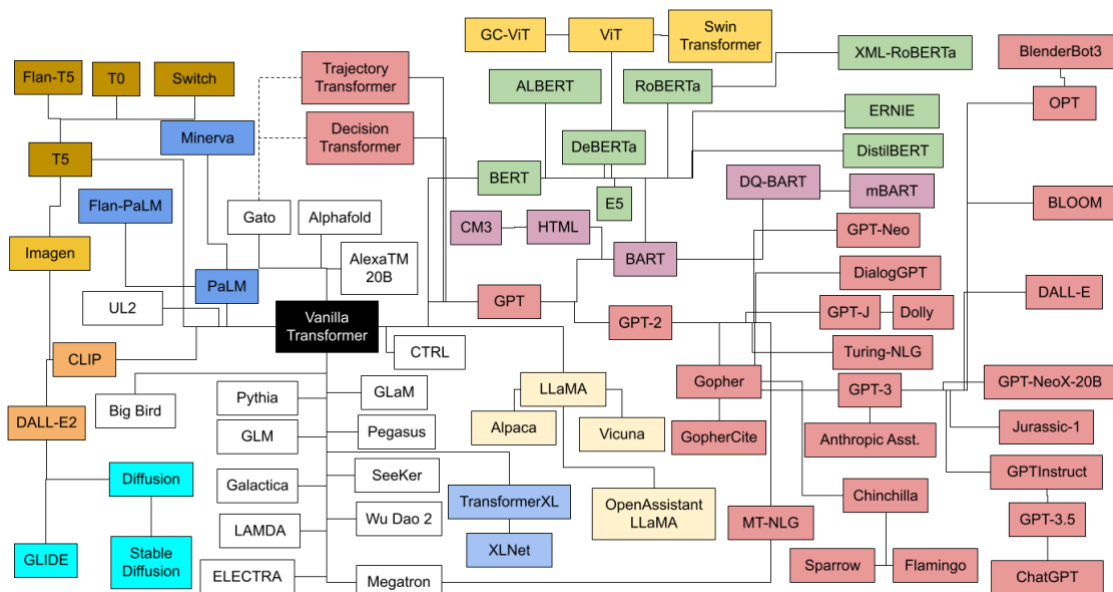


Ilustración 19 - Agrupación por familias. (Amatriain et al, 2023) [34]

Además de los modelos ya publicados, se espera que en los próximos meses se anuncien y se pongan a disposición del público general los siguientes modelos:

- Google Deepmind **Gemini**: Se trata de un modelo multimodal previsto para finales de 2023[37].
- OpenAI **GPT5**: La nueva versión de GPT está en entrenamiento y su publicación se espera para abril de 2024[38].

2.2.4 Transformers en la clasificación de textos

Los modelos basados en la arquitectura Transformers tienen un gran desempeño en tareas de clasificación de texto debido a sus características, en particular:

- **Atención de largo alcance**: La utilización de mecanismos de atención, en particular *multi-head attention*, permite capturar las relaciones entre palabras en textos largos y sin importar su posición. Esto es especialmente importante en cuanto a la desambiguación de palabras con múltiples significados.
- **Modelado de secuencias y procesamiento en paralelo**: Están diseñados para el manejo de secuencias de datos, adecuado para tareas de NLP, pero además, facilitan su procesamiento en paralelo, lo que permite entrenamientos más rápidos.
- **Preentrenamiento y aprendizaje por transferencia**: Los modelos basados en Transformers pueden ser preentrenados y después, mediante técnicas de aprendizaje por transferencia, permiten adaptarlos para realizar otras tareas diferentes para las que inicialmente fueron desarrollados. Esto permite aprovechar LLM de acceso público como BERT.

Estas características han permitido que los modelos basados en Transformers sean los más utilizados a la hora de realizar tareas de clasificación de texto, por su gran rendimiento y simplicidad a la hora de realizar *fine-tuning* y ponerlo en producción.

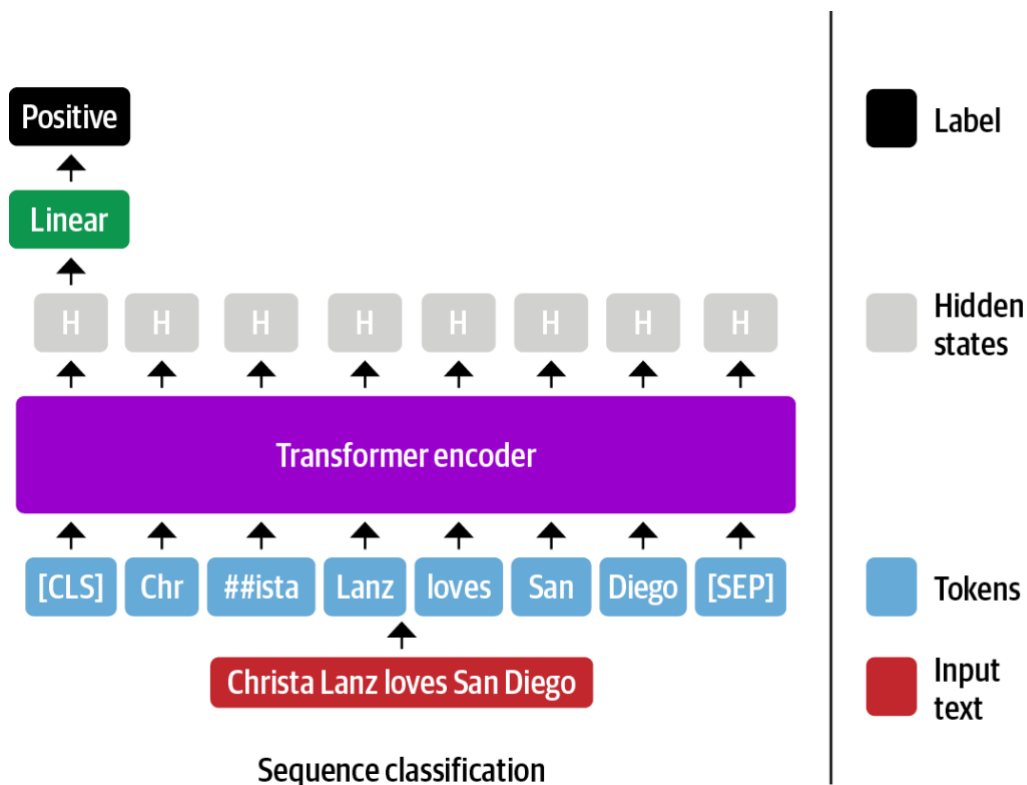


Ilustración 20- Transformer Pipeline para clasificación de textos. Fuente: https://github.com/somosnlp/recursos/blob/main/aplicaciones_de_los_transformers.ipynb

La arquitectura Transformer además permite crear un pipeline que envuelva toda la arquitectura, abstrayendo al programador de lo que ocurre en su interior y simplificando enormemente su uso.

2.2.5 Hugging Face

Hugging Face ^[1] es una empresa estadounidense pero de origen francés que empezó en el 2016 como un chatbot web orientado al público adolescente. Sin embargo, poco después evolucionó hacia una plataforma que pone a disposición de los usuarios herramientas para el desarrollo de modelos de Aprendizaje Automático basado en tecnologías de código abierto. Desde entonces, a lo largo de 4 series de financiación ^[39] ha logrado captar \$396M y cuenta como inversores empresas como Google, IBM, Nvidia, Amazon o Qualcomm.

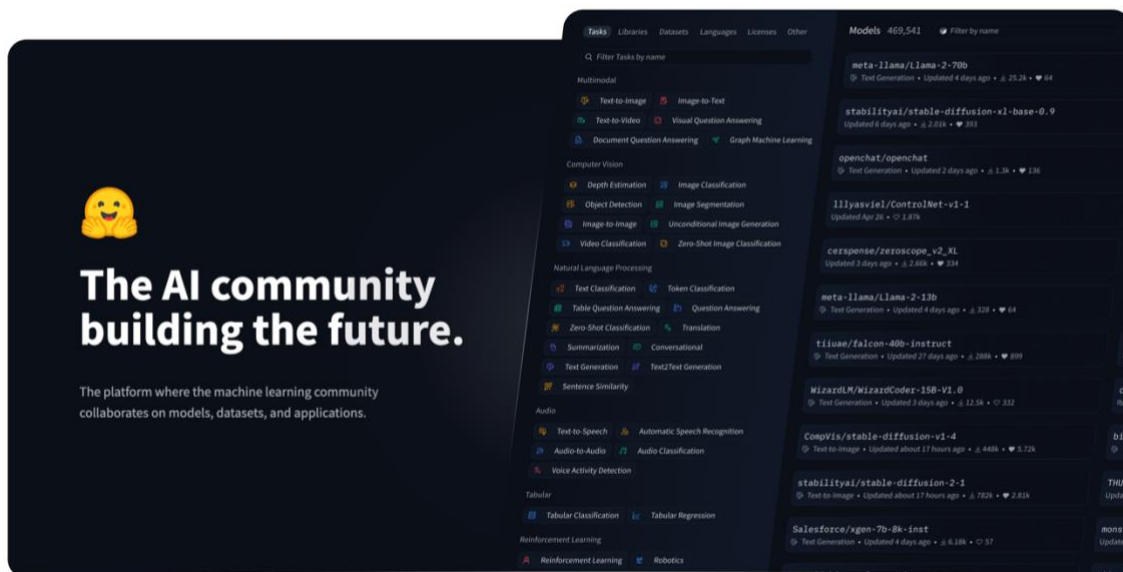


Ilustración 21- Hugging Face. Página principal. Elaboración propia

Algunas de estas empresas y organizaciones publican en esta plataforma sus datasets y modelos preentrenados, los ponen a disposición del público en general y promueven su uso de manera gratuita. Esto ha colaborado a que Hugging Face se convierta en todo un referente para todo aquél interesado en la Inteligencia Artificial y el Aprendizaje Automático.

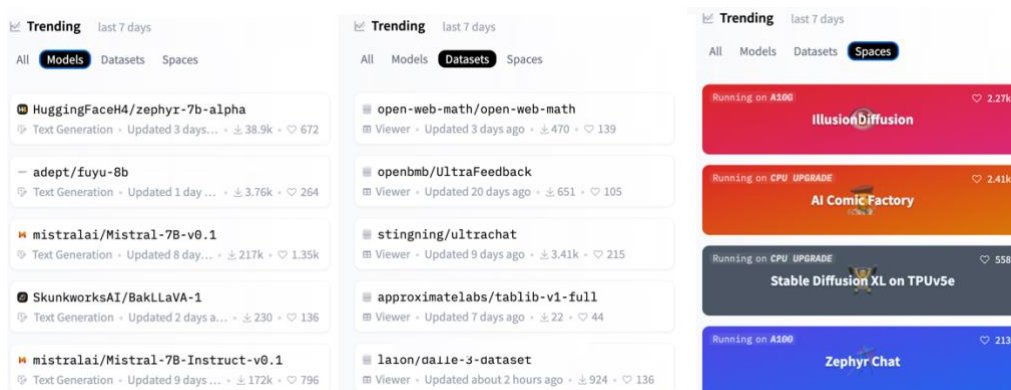


Ilustración 22- Hugging Face: Trending models, datasets and spaces. Elaboración propia.

El acceso a estos recursos supone una gran ayuda para aquellos desarrolladores, investigadores y entusiastas que no tienen los medios suficientes para llevar a cabo entrenamientos tan exigentes como los que demandan algunos modelos como BERT, LLAMA o GPT, pues los requisitos para el entrenamiento son muy exigentes y costosos.

Pero además, Hugging Face ha logrado construir una comunidad alrededor de su plataforma donde los usuarios comparten las modificaciones y ajustes que realizan a los modelos base en el Hugging Face HUB, e incluso compiten para obtener las mejores métricas en tareas concretas [40].

En la siguiente imagen se muestra una competición activa donde los usuarios, ya sean individuales u organizaciones compiten por lograr un modelo LLM con las mejores métricas posibles.

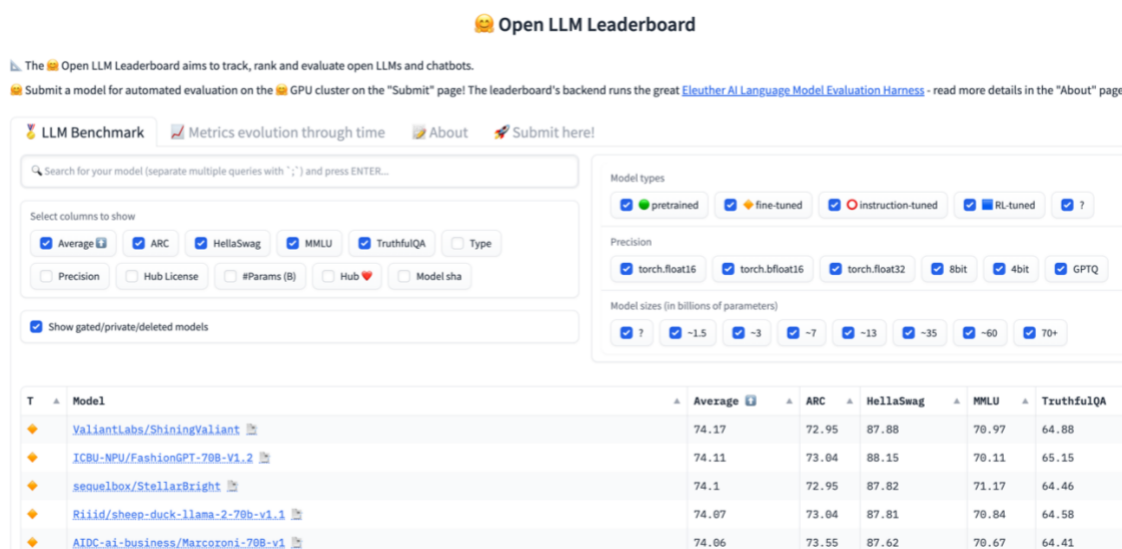


Ilustración 23- Hugging Face: Open LLM Leaderboard. Elaboración propia.

Otra herramienta que destacar es el AutoTrain, que permite realizar Fine-Tuning sin necesidad de escribir una línea de código. Tan sólo asignarle un nombre al proyecto, elegir la tarea objetivo, basada en el tipo de dato y escoger el modelo base o incluso dejar que la propia herramienta lo elija de manera automática. A continuación, basta con subir el dataset de entrenamiento y la propia herramienta se encargará del resto.

The screenshot shows the Hugging Face AutoTrain interface. It starts with a 'New project' section where users can 'Select a task, language, and how many models you want to train. You will prepare data in the next step.' Below this, there's a 'Project name' field with the value 'news-classification-2'. The 'Task' section has three options: 'Vision', 'Text' (selected), and 'Tabular'. Under 'Text', there are three sub-options: 'Text Classification (Binary)', 'Text Classification (Multi-class)' (selected), and 'Token Classification'. Below these, there are two more options: 'Question Answering (Extractive)' and 'Question Answering (Generative)'. On the right, there's a 'Model choice' section with two options: 'Automatic' and 'Manual' (selected). Below this, there's a 'Selected Model' dropdown showing 'dccuchile/bert-base-spanish-wwm-cased'. A note at the bottom states: 'Only models compatible with the transformers library and the fill-mask task can be used for this AutoTrain project.'

Ilustración 24- Hugging Face: AutoTrain. Elaboración propia.

Además, la plataforma facilita las métricas de desempeño obtenidas y pone a disposición del usuario un endpoint para poder utilizar modelo entrenado e incluso, su despliegue.

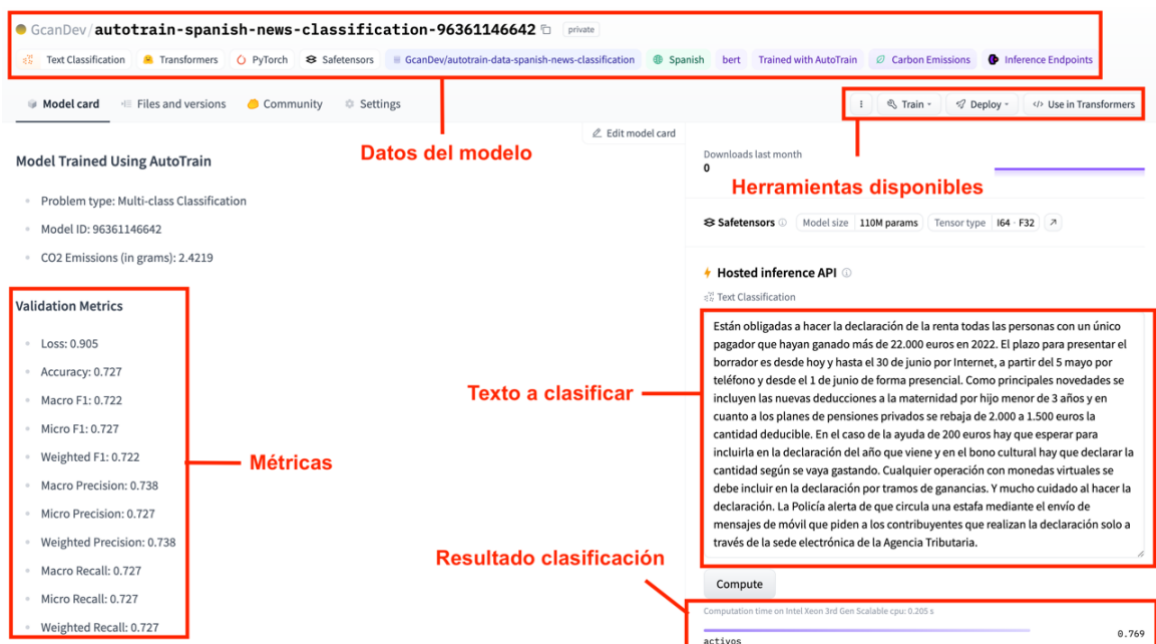


Ilustración 25- Hugging Face: Página de un modelo. Elaboración propia.

La imagen anterior muestra la página de un modelo entrenado con AutoTrain, que no difiere de los modelos compartidos por los usuarios independientemente de cómo hayan sido entrenados. Lo que resulta de interés es que en una sola vista está disponible toda la información necesaria sobre el modelo e incluso una pequeña API para probarlo y con la cual se obtiene la predicción y la puntuación recibida.

AutoTrain permite realizar este entrenamiento de forma gratuita para un único modelo por proyecto y con un dataset de hasta 3.000 registros. Sin embargo, la modalidad de pago por uso permite utilizar datasets sin límite de tamaño y que la herramienta escoja hasta 10 modelos distintos a la vez, les realice un Fine-Tuning y presente como modelo definitivo aquel que haya obtenido mejor puntuación.

En el caso de este proyecto, realizar esta tarea de auto-entrenamiento con 10 modelos y el dataset utilizado con 26.000 registros suponía un coste total de 89\$. Si se tiene en cuenta que este servicio lo pueden usar empresas para entrenar modelos que después pongan en producción y el ahorro en horas de trabajo, es un importe muy bajo.

Otra opción disponible es AutoTrain Advanced, que permite desplegar pods tipo Docker o similares en máquinas virtuales con distintas características y precios por hora, y realizar en ellas las tareas de Fine-Tuning tan intensivas como considere el usuario. Esta herramienta es similar a las que ofrecen los grandes proveedores cloud, pero tiene la ventaja que abstrae al usuario por completo de cualquier tarea relacionada con la programación.

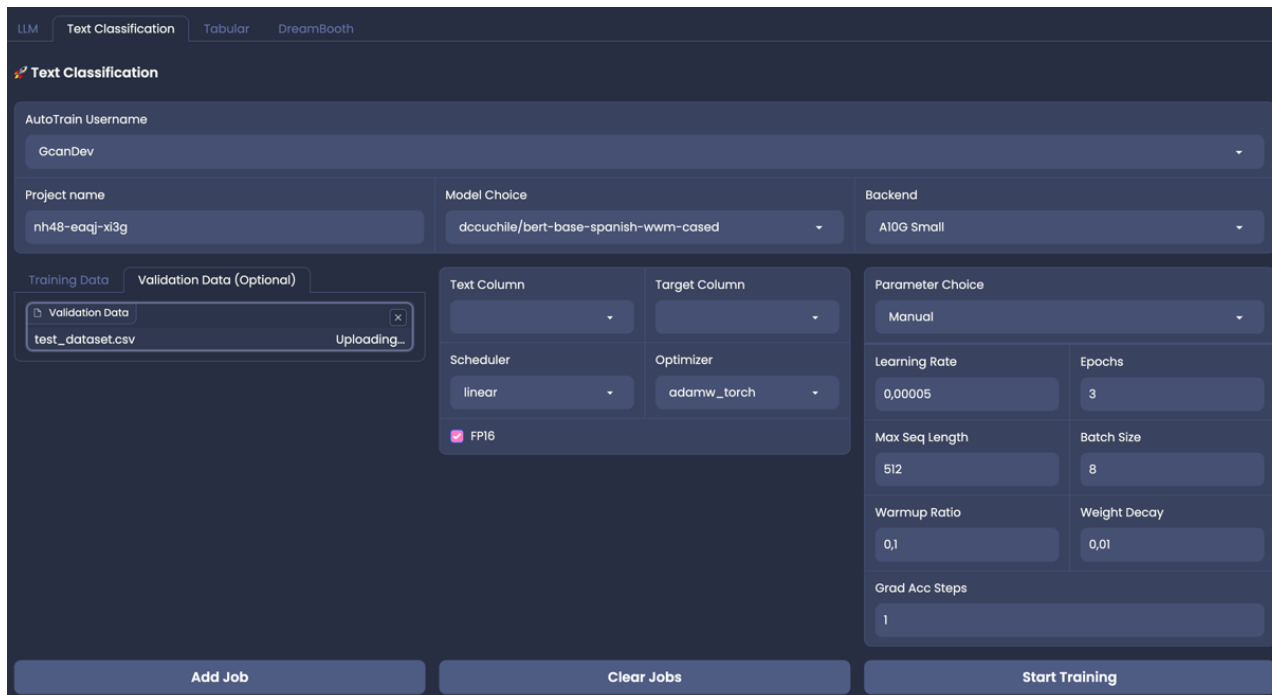


Ilustración 26- Hugging Face: Instancia de AutoTrain Advanced. Elaboración propia.

2.3 Conclusiones del marco teórico

Este capítulo ha tratado en conjunto el ámbito del NLP desde una breve reseña histórica sobre los orígenes de esta disciplina hasta la aparición de últimos modelos publicados y aquellos que están por venir.

La intención de este capítulo ha sido proporcionar una base teórica para poder llegar a entender el funcionamiento de la arquitectura Transformer, partiendo de la definición de los tipos de modelos de lenguaje, las formas disponibles para codificar las palabras de lenguaje natural a números para así aplicar modelos matemáticos e incluso el aprendizaje por transferencia, un concepto fundamental para la parte práctica de este trabajo.

Una vez presentados estos conceptos básicos, se ha introducido la arquitectura Transformer y los distintos mecanismos de atención, responsables de obtener unos resultados mejores que las tecnologías y modelos utilizados hasta entonces.

Además, se ha presentado la plataforma Hugging Face y los numerosos servicios que ofrece, como su HUB a través del cual se descargarán los modelos preentrenados para realizar Fine-tuning y abordar la tarea objetivo de este trabajo.

Capítulo 3. OBJETIVOS

3.1 Objetivos generales

El objetivo general del presente trabajo consiste en realizar una investigación exhaustiva en el ámbito del NLP con el objetivo principal de ampliar los conocimientos sobre la materia y en particular con todo aquello relacionado con los modelos basados en Transformers.

Además, se pretende obtener al menos un modelo que sea capaz de clasificar eficazmente los textos de noticias entre varias categorías.

3.2 Objetivos específicos

Asimismo, a lo largo de la elaboración de la presente memoria se establecen los siguientes objetivos específicos:

1. Investigar a fondo el estado del arte de los modelos basados en Transformers, poniendo énfasis en las contribuciones recientes en el ámbito del procesamiento de lenguaje natural.
2. Evaluar y comparar diferentes modelos basados en Transformers, determinando sus ventajas y limitaciones específicas para tareas relacionadas con la clasificación de textos de noticias.
3. Implementar y realizar un Fine-Tuning el modelo seleccionado utilizando un conjunto de datos representativo y actual que garantice su adaptabilidad a distintas fuentes y tipos de contenido.
4. Analizar las implicaciones prácticas de los modelos propuestos e identificar las áreas de mejora y adaptabilidad con el objetivo de obtener una solución efectiva y duradera.

3.3 Beneficios del proyecto

La integración del producto final, un modelo de clasificación de noticias basado en Transformers, en un entorno de producción, permitiría la automatización de las tareas de clasificación de noticias permitiendo así ahorro de tiempo humano y el consecuente aumento de la productividad del equipo.

Esto puede resultar tremendamente útil, por ejemplo, en aquellas empresas u organizaciones interesadas en analizar el contenido de noticias web.

Capítulo 4. DESARROLLO DEL PROYECTO

4.1 Planificación del proyecto

El presente trabajo se organiza en tres fases bien diferenciadas tal y como se muestra en el siguiente diagrama de Gantt. Además, se ha tenido en cuenta el periodo vacacional del mes de agosto. La temporización de dichos periodos es la siguiente:

- **Fase 1 – Documentación e investigación:** Del 11/07/2023 al 11/08/2023.
- **Vacaciones:** Del 14/08/2023 al 01/09/2023.
- **Fase 2 – Implementación de la solución:** Del 04/09/2023 al 15/09/2023.
- **Fase 3 – Extracción de resultados y elaboración de la memoria:** Del 18/09/2023 al 12/10/2023.

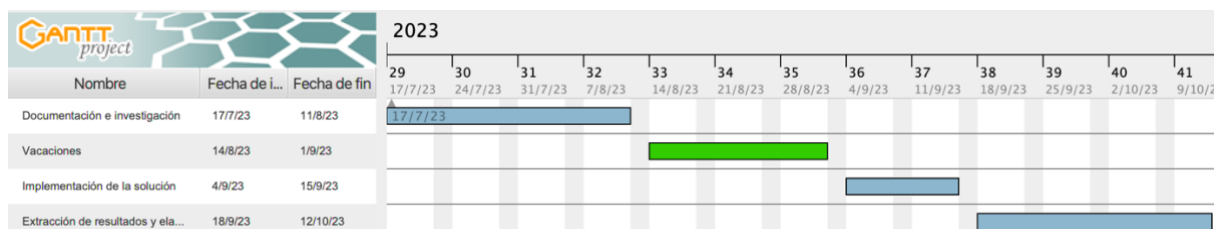


Ilustración 27- Diagrama de Gantt planificación del proyecto. Elaboración propia.

4.2 Descripción de la solución, metodologías y herramientas empleadas

4.2.1 Preparación de los datos

4.2.1.1 Introducción al Dataset Utilizado

Actualmente estoy cursando prácticas extracurriculares en el contexto de este Máster en Prensa Ibérica, uno de los principales grupos editoriales de España. Entre otros medios, cuenta con 26 periódicos, 61 crónicas locales y, en el ámbito digital, sus páginas web tienen más de 27 millones visitantes únicos y casi 600 millones de páginas vistas al mes.

La empresa dispone de un DataLake basado en BigQuery de Google Cloud Platorm con una gran cantidad datos de distinta índole. Entre ellos se encuentra un dataset de 34Gb con más de 3.300.000 registros de noticias.

ESQUEMA	DETALLES	PREVIEW	LINAJE
Particionada por	DAY		
Particionada en el campo	publish_date		
Vencimiento de la partición	Las particiones no vencen		
Filtro de partición	Obligatorio		
Agrupado en clústeres por	site url_type is_premium is_amp		

Información de almacenamiento	
Cantidad de filas	10,126,340
Cantidad de particiones	1,352
Total de bytes lógicos	34.08 GB
Bytes lógicos activos	5.49 GB
Bytes lógicos a largo plazo	28.6 GB
Total de bytes físicos	6.81 GB
Bytes físicos activos	2.85 GB
Bytes físicos a largo plazo	3.97 GB
Bytes físicos de viajes en el tiempo	2.09 GB

Ilustración 28- BigQuery detalles dataset url_analysis. Elaboración propia

Estos registros tienen más de 100 columnas de metadatos, con elementos como texto de la noticia, la categoría, url, sitio web, números de páginas vistas generadas, tiempo de visualización generado e incluso impacto en redes sociales.

ESQUEMA	DETALLES	PREVIEW	LINAJE
<input type="checkbox"/> Nombre del campo	Tipo	Modo	
<input type="checkbox"/> publish_date	DATE	NULLABLE	
<input type="checkbox"/> url	STRING	NULLABLE	
<input type="checkbox"/> site	STRING	NULLABLE	
<input type="checkbox"/> last_operation_timestamp	TIMESTAMP	NULLABLE	
<input type="checkbox"/> id_hashed_url	STRING	NULLABLE	
<input type="checkbox"/> id_content	STRING	NULLABLE	
<input type="checkbox"/> id_site_news	STRING	NULLABLE	
<input type="checkbox"/> canonical_url	STRING	NULLABLE	
<input type="checkbox"/> url_type	STRING	NULLABLE	
<input type="checkbox"/> id_source	STRING	NULLABLE	
<input type="checkbox"/> full_domain	STRING	NULLABLE	
<input type="checkbox"/> url_section	STRING	NULLABLE	
<input type="checkbox"/> url_subsection	STRING	NULLABLE	
<input type="checkbox"/> publisher_internal_source	STRING	NULLABLE	
<input type="checkbox"/> url_multimedia	STRING	NULLABLE	
<input type="checkbox"/> url_publishing_user	STRING	NULLABLE	
<input type="checkbox"/> url_category	STRING	NULLABLE	
<input type="checkbox"/> article_headband	STRING	NULLABLE	
<input type="checkbox"/> authors	STRING	NULLABLE	
<input type="checkbox"/> body			
<input type="checkbox"/> body_length			
<input type="checkbox"/> body_word_count			
<input type="checkbox"/> paragraph_count			
<input type="checkbox"/> title			
<input type="checkbox"/> title_word_count			
<input type="checkbox"/> description			
<input type="checkbox"/> description_word_count			
<input type="checkbox"/> heading			
<input type="checkbox"/> heading_word_count			
<input type="checkbox"/> keywords			
<input type="checkbox"/> entities			
<input type="checkbox"/> robots_meta			
<input type="checkbox"/> url_commercial_type			
<input type="checkbox"/> signature_cms			
<input type="checkbox"/> author_cms			
<input type="checkbox"/> authors_global			
<input type="checkbox"/> authors_global_type			
<input type="checkbox"/> thumbnail_url			
<input type="checkbox"/> section_name			
<input type="checkbox"/> news_author			
<input type="checkbox"/> author_url			
<input type="checkbox"/> content_generator_group			
<input type="checkbox"/> video_author			
<input type="checkbox"/> video_origin			
<input type="checkbox"/> video_title			
<input type="checkbox"/> video_type			
<input type="checkbox"/> url_status			
<input type="checkbox"/> news_origin_category			
<input type="checkbox"/> url_secondary_section			
<input type="checkbox"/> create_timestamp			
<input type="checkbox"/> modified_timestamp			
<input type="checkbox"/> publish_timestamp			
<input type="checkbox"/> first_view_timestamp			
<input type="checkbox"/> first_view_date			
<input type="checkbox"/> analysis_date			
<input type="checkbox"/> is_premium			
<input type="checkbox"/> is_behind_registration_wall			
<input type="checkbox"/> is_afondo			
<input type="checkbox"/> is_mas			
<input type="checkbox"/> is_rc			
<input type="checkbox"/> is_republished			
<input type="checkbox"/> is_amp			
<input type="checkbox"/> content_classification			
<input type="checkbox"/> url_recirculation_classification			
<input type="checkbox"/> original_site			
<input type="checkbox"/> frontpage_first_date			
<input type="checkbox"/> google_first_date			
<input type="checkbox"/> first_day_pub_browsers			
<input type="checkbox"/> first_day_pub_page_views			
<input type="checkbox"/> three_days_pub_page_views			
<input type="checkbox"/> three_days_pub_seo_page_views			
<input type="checkbox"/> three_days_pub_browsers			
<input type="checkbox"/> three_days_pub_seo_browsers			
<input type="checkbox"/> num_sites_published			
<input type="checkbox"/> frontpage_hours_1_day			
<input type="checkbox"/> frontpage_hours_3_days			
<input type="checkbox"/> frontpage_min_position			
<input type="checkbox"/> frontpage_max_position			
<input type="checkbox"/> frontpage_views_1_day			
<input type="checkbox"/> frontpage_views_3_days			
<input type="checkbox"/> frontpage_browser_views_1_day			
<input type="checkbox"/> frontpage_browser_views_3_days			
<input type="checkbox"/> frontpage_clicks_1_day			
<input type="checkbox"/> frontpage_clicks_3_days			
<input type="checkbox"/> frontpage_first_hour			
<input type="checkbox"/> google_web_impressions_3_day			
<input type="checkbox"/> google_web_clicks_3_day			
<input type="checkbox"/> google_web_avg_position_3_day			
<input type="checkbox"/> promoted_in_push_notification			
<input type="checkbox"/> promoted_in_newsletter			
<input type="checkbox"/> is_passed_seo_audit			
<input type="checkbox"/> seo_audit_scoring			
<input type="checkbox"/> site_news_scoring			
<input type="checkbox"/> original_site			
<input type="checkbox"/> has_timeline			
<input type="checkbox"/> internal_tag			
<input type="checkbox"/> infographics			
<input type="checkbox"/> social_networks_exposure			
<input type="checkbox"/> promoted_in_push_notification			
<input type="checkbox"/> promoted_in_newsletter			
<input type="checkbox"/> is_passed_seo_audit			
<input type="checkbox"/> seo_audit_scoring			
<input type="checkbox"/> site_news_scoring			
<input type="checkbox"/> cms_city			
<input type="checkbox"/> cms_province			
<input type="checkbox"/> cms_region			
<input type="checkbox"/> cms_tag			
<input type="checkbox"/> url_sub_subsection			
<input type="checkbox"/> social_networks_views			

Ilustración 29- BigQuery columnas metadatos dataset url_analysis. Elaboración propia

Los datos de la tabla anterior permiten realizar toda clase de análisis sobre las noticias del Grupo Editorial, como, por ejemplo, el siguiente gráfico:

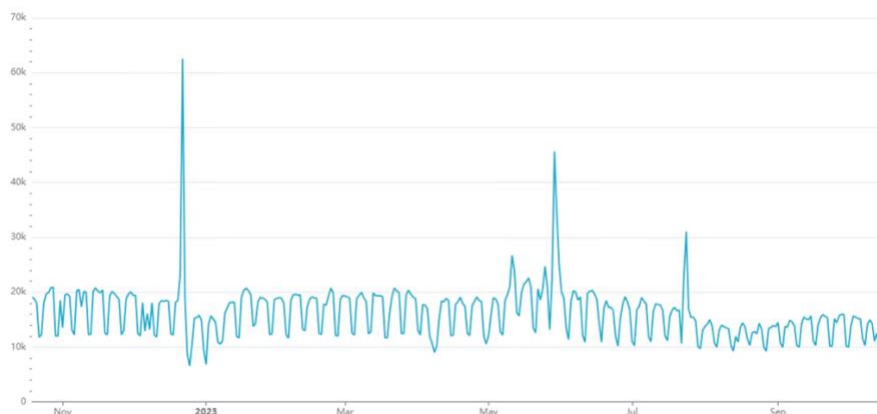


Ilustración 30- Número de noticias diarias publicadas en el último año. Elaboración propia

En él, se muestran el número de noticias diarias publicadas en el último año y se observa un claro patrón cíclico correspondiente a los días entre semana con una bajada de la actividad durante los fines de semana. También se observan 3 grandes momentos a lo largo del año, que corresponden con los días 21/12/2023 (Lotería de Navidad), 29/05/2023 (Pedro Sánchez convoca elecciones) y 24/07/2023 (Día posterior a las Elecciones Generales).

Éste es sólo un ejemplo de lo que se puede realizar con los datos contenidos en la tabla utilizada y en el caso de este proyecto, se utilizarán los textos de las noticias y sus categorías para realizar Fine-Tuning en un modelo basado en Transformers preentrenado.

4.2.1.2 Adquisición y limpieza del dataset

Actualmente Google Cloud tiene una limitación a la hora de exportar datos a partir de BigQuery desde su consola web. En caso de que los datos recuperados con la consulta SQL superen 1Gb, solo se permite la exportación hacia otra tabla BigQuery o a un bucket de almacenamiento.

Por este motivo, se ha desarrollado un pequeño ETL en Python cuyo código se adjunta en el repositorio de Github, accesible desde el [Anexo 1](#), que lanza la consulta SQL, recupera los datos como Dataframe de Pandas y lo almacena en local en un archivo CSV con un tamaño de 11.57Gb.

Como el problema al que hace frente el modelo es de clasificación, el entrenamiento es supervisado, por ello, los datos utilizados para esta tarea deben estar etiquetados. En este caso, como únicamente tienen una etiqueta, la estructura de un dataset para realizar Fine-tuning debe ser una lista de diccionarios cuyo contenido sea:

```
{ 'label': 'etiqueta',  
  'text': 'texto cuya categoría corresponde a la etiqueta' }
```

Por este motivo, el dataset utilizado para el Fine-tuning de un modelo basado en transformers debe tener únicamente dos columnas: label y text. Sin embargo, por comodidad, se decidió descargar algunas columnas más de la tabla de BigQuery con la intención de poder utilizarlas como filtros para reducir el dataset final.

```

1 df1 = df[~df['site'].isin(['regio7', 'diaridegirona', 'levante-emv',
2 | 'mallorcazeitung', 'farodevigo', 'laopinioncoruna'])]
3 keep_category = ["vida y estilo", "cultura", "politica",
4 | "economia", "deportes", "gastronomia",
5 | "ciencia", "tecnologia", "esquelas"]
6 df1 = df1[df1['url_subsection'].isin(keep_category)]
7 df1 = df1[~df1['url'].str.contains('val.|amp')]
8 df1['length'] = df1['body'].apply(lambda x: len(str(x).split()))
9 df1 = df1[(df1['length'] <=800) & (df1['length'] >=80)]
10 df1 = df1.drop_duplicates(subset=['body'])
11 df2 = df1[['body', 'url_subsection']].copy()
12 df2.rename(columns={'body': 'text', 'url_subsection': 'label'}, inplace=True)
13 df2 = df2.groupby('label').apply(lambda x: x.sample(n=1900, replace=True, random_state=1)
14 | if len(x) >= 1900 else x).reset_index(drop=True)

```

Ilustración 31.- Filtrado del Dataset de entrenamiento. Elaboración propia

Del conjunto total de datos se han descartado aquellos medios locales que publican noticias en otras lenguas, como catalán, valenciano, gallego o alemán, para asegurar que todas las noticias sean en castellano.

También se ha observado que las categorías de la columna `url_category` eran demasiado genéricas, por lo que se ha decidido utilizar la columna `url_subsection` que ofrece categorías mejor definidas.

Otra característica escogida para el dataset ha sido que el texto tenga más de 80 palabras y menos de 800. Esto se decidió porque modelos como BERT tienen limitado el tamaño del input a valores próximos a 500 tokens. De esta manera, los textos utilizados para el entrenamiento tienen un tamaño similar para evitar que existan tamaños muy dispares entre los textos de distintas categorías.

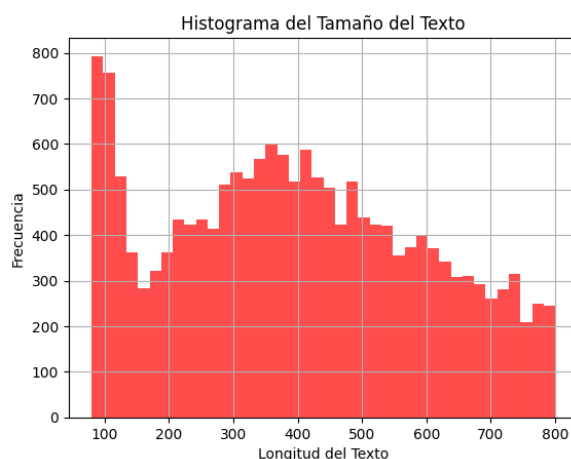


Ilustración 32- Histograma: Número de palabras Train Dataset. Elaboración propia

Finalmente, el filtrado explicado arriba y eliminar los registros duplicados, se ha decidido limitar a un total de 9 categorías distintas y 1700 textos para cada una de ellas en el dataset de entrenamiento y 200 para el de test. De este modo, los datos están balanceados y se evita que ninguna categoría tenga más peso que otra a la hora de entrenar los modelos.

<code>1 train_df['label'].value_counts()</code>		<code>1 test_df['label'].value_counts()</code>	
label		label	
ciencia	1700	ciencia	200
cultura	1700	cultura	200
deportes	1700	deportes	200
economia	1700	economia	200
esquelas	1700	esquelas	200
gastronomia	1700	gastronomia	200
politica	1700	politica	200
tecnologia	1700	tecnologia	200
vida y estilo	1700	vida y estilo	200
Name: count, dtype: int64		Name: count, dtype: int64	

Ilustración 33- Elementos por categoría en el Dataset. Elaboración propia

4.2.2 Selección de Modelos de Transformers preentrenados de Hugging Face

Como se ha visto anteriormente, el repositorio Hugging Face permite disponer de muchos modelos preentrenados para adaptarlos a otras tareas a través de Fine-tuning.

De entre aquellos disponibles en el Hub, se ha decidido utilizar los siguientes modelos:

- **dccuchile/bert-base-spanish-wwm-cased**

Desarrollado por el Data Science Institute de la Universidad de Chile, se basa en la arquitectura BERT (Bidirectional Encoder Representations from Transformers) y se caracteriza por ser un modelo de lenguaje preentrenado en español, que distingue entre mayúsculas y minúsculas. Se puede utilizar en múltiples tareas de NLP como clasificación, QA o análisis de sentimientos.

- **DistilBERT**

Desarrollado por Hugging Face, es una versión más ligera de BERT, ya que tiene un 40% menos de parámetros que el modelo base. Por este motivo es más eficiente en cuanto a memoria y velocidad, manteniendo un rendimiento es cercano al modelo original. Sus posibles usos son los mismos que que pueden darse al BERT base, pero al ser más ligero, se puede ejecutar en entornos con recursos limitados.

- **GPT2**

Desarrollado por OpenAI, está basado en el modelo Transformer y se trata de un modelo de lenguaje de propósito general. Existen las versiones 3.5 y 4, pero en Hugging Face sólo se está disponible esta versión. A pesar de ello, tiene 1.5 mil millones de parámetros y es un modelo muy eficiente a la hora de generar texto.

4.2.3 Fine-tuning

4.2.3.1 Justificación y objetivos

En el apartado 2.1.1.5 se introdujo el concepto de aprendizaje por transferencia y cómo esta metodología asentaba las bases para el Fine-tuning, ya que, gracias a ello, se puede ajustar un modelo de modo que realice una tarea para la que no fue desarrollado inicialmente.

En este caso se han escogido tres modelos preentrenados cada uno para una tarea distinta:

- BERT español y distilBERT diseñados para tareas de FILL MASK.
- GPT2 diseñado para generación de texto.

Sin embargo, estos tres modelos disponen de métodos específicos para clasificación. En particular, los métodos BertForSequenceClassification, DistilBertForSequenceClassification y GPT2ForSequenceClassification disponibles en la librería Transformers de Hugging Face.

Otro factor a tener en cuenta es la potencia computacional y el tiempo requerido en función de ésta. Por poner un ejemplo, en una de las pruebas se realizó un Fine-tuning en un equipo sin GPU y tardó 45 horas 50 minutos para tan sólo 3 épocas. Por otro lado, los fine-tuning realizados a través de Google Colab con GPU Nvidia V100 tardaron menos de 2 horas en completar 10 épocas para el mismo modelo y con el mismo dataset.

Estos datos muestran cómo los requisitos son muy inferiores a los que se necesitarían para entrenar un modelo desde cero, pero aun así, debe tenerse en cuenta el tiempo de entrenamiento de los modelos.

En cuanto al objetivo del Fine-tuning es la obtención de un modelo que sea capaz de clasificar correctamente los textos de las noticias en las categorías establecidas.

4.2.3.2 Implementación del fine-tuning

La implementación del Fine-tuning en los modelos preentrenados se ha hecho mediante las clases Trainer y TrainingArguments de la librería Transformers [41] de Hugging Face.

En ellas se definen los hiperparámetros del modelo, como el número de épocas, batch size, la métrica sobre la que se pretende optimizar el modelo o si se desea generar checkpoints durante el entrenamiento.

```
training_args = TrainingArguments(  
    output_dir='./results',  
    num_train_epochs=10,  
    per_device_train_batch_size=25,  
    per_device_eval_batch_size=25,  
    warmup_steps=300,  
    weight_decay=0.01,  
    logging_dir='./logs',  
    logging_steps=200,  
    evaluation_strategy="steps",  
    eval_steps=100,  
    save_total_limit=2,  
    load_best_model_at_end=True,  
    metric_for_best_model='accuracy',  
    greater_is_better=True,  
    do_train=True,  
    do_eval=True,  
    do_predict=True  
)
```

Ilustración 34 - Hiperparámetros fine-tuning. Elaboración propia

Los códigos relativos al Fine-tuning de cada modelo pueden consultarse a través del repositorio personal de Github accesible desde el Anexo 1, y han sido desarrollados para ejecutarse a través de Google Colab Pro con aceleración de hardware por GPU. En BERT y DistilBERT se ha ejecutado el entrenamiento durante 10 épocas y el batch size se ha modificado en función de la GPU disponible en ese momento. En cambio, dadas las características de GPT2 se configuraron en 3 épocas y con un batchsize de 4, ya que sus requisitos de memoria son mucho mayores.

Como alternativa también se realizó un *fine-tuning* a través de la aplicación AutoTrain de HuggingFace para investigar su funcionamiento y observar sus resultados. En este caso se marcó la opción de selección automática del modelo base. El proceso está descrito en el [ANEXO 3: Hugging Face AutoTrain](#).

4.3 Recursos requeridos

Para el desarrollo de este proyecto han sido necesarias una serie de herramientas que han permitido alcanzar los objetivos propuestos. Entre ellas se destacan:

- **Virtual Studio Code:** Entorno de desarrollo para programar los scripts.
- **Github:** Repositorio online para control de versiones y compartir los scripts.
- **Google Cloud Storage:** Almacenamiento cloud de los archivos.
- **Big Query:** Herramienta de consulta de base de datos de Google Cloud para recuperar los datos almacenados en del Data Lake.
- **Google Colab Pro:** Plataforma de ejecución de cuadernos tipo Jupyter Notebook con capacidad de **aceleración GPU** para realizar el Fine-tuning.

4.4 Presupuesto

El presente proyecto se ha realizado partiendo de la siguiente planificación de tareas y su correspondiente asignación de tiempo:

TAREA	Tiempo de dedicación
Investigación sobre la materia	70h
Implementación	30h
Redacción de la memoria	50h
TOTAL	150h

Tabla 1- Distribución tareas y tiempo de dedicación.

La ejecución de los scripts se ha hecho a través de la plataforma Google Colab Pro, donde una suscripción mensual con 100 unidades de procesamiento tiene un coste de 11,19€.

1 hora de ejecución de una máquina virtual con Nvidia V100 tiene un coste de 5,45 unidades de procesamiento y para una Nvidia A100 son alrededor de 15 unidades por hora, lo que supone unos 0,60€ para la V100 y 1,50€ para la A100.

SCRIPT	Tiempo	Coste/hora	Coste
tokenization_lemmatization_example.ipynb	10m	-	-
extract_data_from_bigquery.py	10m	-	-
extract_data_from_full_dataset_9cat.ipynb	10m	-	-
test_bert_base_model.ipynb	2h	0,60€	1,20€
test_distilbert_base_model.ipynb	2h	0,60€	1,20€
test_gpt2_base_model.ipynb	2h	0,60€	1,20€
finetune_bert.ipynb	3h	0,60€	1,80€
finetune_distilbert.ipynb	3h	0,60€	1,80€
finetune_gpt2.ipynb	3,5h	0,60€	2,10€
Test_bert_autotuned.ipynb	2h	0,60€	1,20€
TOTAL	18h	-	10,50€

Tabla 2 - Requisitos de ejecución para los scripts y coste económico

Teniendo en cuenta los precios de estos dispositivos y el rendimiento que supone poder utilizarlos, acceder a estos dispositivos y aprovechar la aceleración por hardware disponible en los frameworks de Pytorch y Transformers ha permitido reducir considerablemente el tiempo de entrenamiento.

Una primera prueba de un Fine-tuning en local 45 horas y 50 minutos para únicamente 3 épocas. Por otro lado, utilizando una máquina virtual de Google Colab con GPU Nvidia V100 redujo este tiempo a tan sólo 3 horas para 10 épocas.

4.5 Test y evaluación de modelos preentrenados

A partir de un notebook de Jupyter se utilizaron los tres modelos seleccionados en el apartado anterior para realizar predicciones sobre el dataset de test, el cual incluye 200 registros para cada una de las 9 categorías. Los resultados de las clasificaciones generadas se muestran en la siguiente tabla:

Modelo	Accuracy	Precision	Recall	F1Score
BERT Spanish (BERTO) dccuchile/bert-base-spanish-wwm-cased	0.097777	0.021378	0.097777	0.026859
DistilBERT (distilbert-base-uncased)	0.111666	0.046514	0.111666	0.037020
GPT2	0.111111	0.058149	0.111111	0.030885

Tabla 3- Métricas obtenidas modelos preentrenados

Como era de esperar, las métricas de los 3 modelos escogidos son muy bajas ya que no fueron diseñados ni entrenados específicamente para la tarea de clasificación sobre la cual fueron evaluados.

En particular, cada uno de los tres modelos evaluados clasifica la mayoría de las noticias en una única categoría, aunque diferente para cada modelo. Esto explica por qué la accuracy es cercana al 10%. Dado que hay 9 categorías distintas, si casi todos los casos se clasifican en una sola categoría, habrá alrededor de un 11% de casos correctamente clasificados. Sin embargo, esto no tiene nada que ver con el buen funcionamiento del modelo sino con la distribución estadística de los resultados. Estos resultados tan bajos pueden observarse mejor con las matrices de confusión. En ellas, se observa claramente que las clasificaciones se efectúan prácticamente al azar:

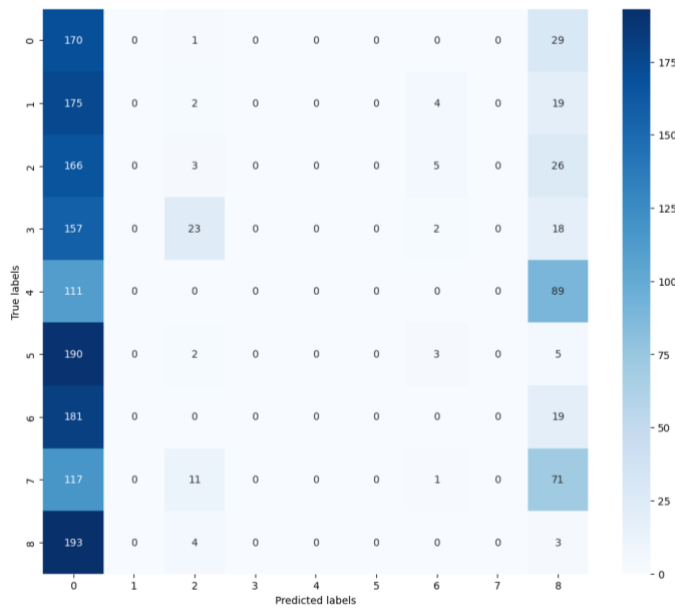


Ilustración 35- Matriz de confusión de bert-base-spanish-wwm-cased preentrenado. Elaboración propia

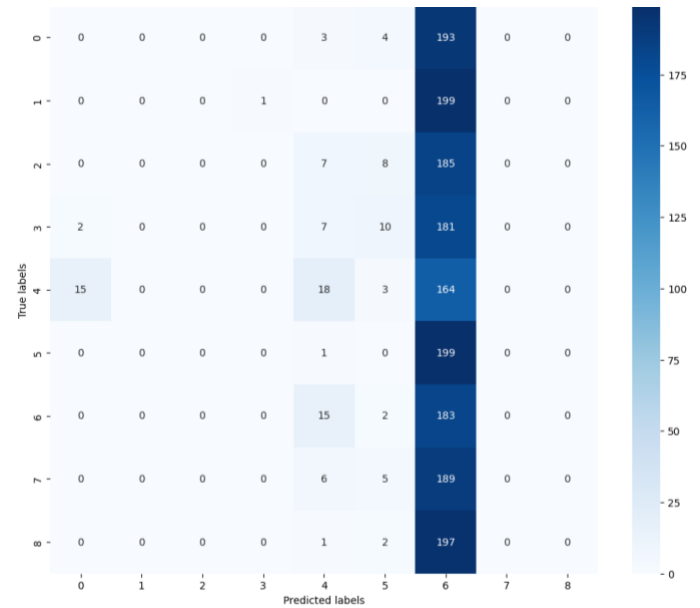


Ilustración 36 - Matriz de confusión de distilbert-base-uncased preentrenado. Elaboración propia

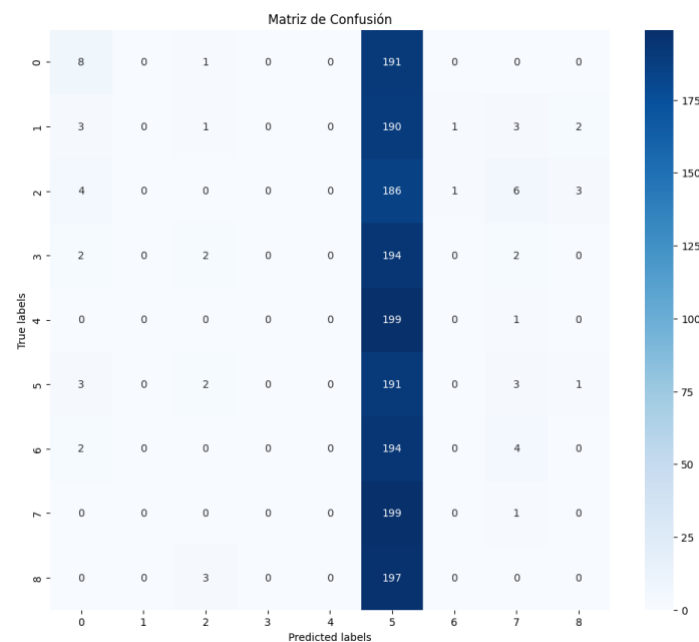


Ilustración 37 - Matriz de confusión de GPT2 preentrenado. Elaboración propia

En el caso de dccuchile/bert-base-spanish-wwm-cased, el modelo clasifica todos los textos en la categoría ciencia, para el modelo DistilBERT en política y para GPT2 en gastronomía.

4.6 Test y evaluación modelos con Fine-tuning

Los resultados después de realizar Fine-tuning con un dataset de 1700 registros para cada una de las 9 categorías son muy buenos.

Todos los modelos superan el 90% en las cuatro métricas, un muy buen resultado a pesar de utilizar un dataset relativamente pequeño y tener 9 categorías distintas para la clasificación.

Además, el modelo que obtiene unas mejores métricas es BERTO, con valores por encima del 96%, un resultado muy por encima de lo esperado, pero comprensible ya que el modelo estaba preentrenado en castellano, mientras que los demás estaban preentrenados en inglés.

Por el contrario, el modelo que peores valores obtiene es el que ha sido entrenado de manera automática por AutoTrain de Hugging Face, y que, a pesar de tardar menos de 10 minutos en el Fine-tuning y que el dataset de entrenamiento sólo tenía 330 registros por categoría, el resultado es bastante satisfactorio:

Modelo	Accuracy	Precision	Recall	F1Score
BERT Spanish (BERTO) dccuchile/bert-base-spanish-wwm-cased	0.966111	0.966551	0.966111	0.966235
BERT AutoTrain	0.900555	0.909563	0.900555	0.901149
DistilBERT (distilbert-base-uncased)	0.947777	0.948672	0.947777	0.947982
GPT2	0.935555	0.935573	0.935555	0.9352418

Tabla 4- Métricas obtenidas modelos con Fine-tuning

En cuanto a las matrices de confusión, como las clasificaciones han sido tan precisas, es difícil sacar conclusiones u observar patrones a partir de los errores:

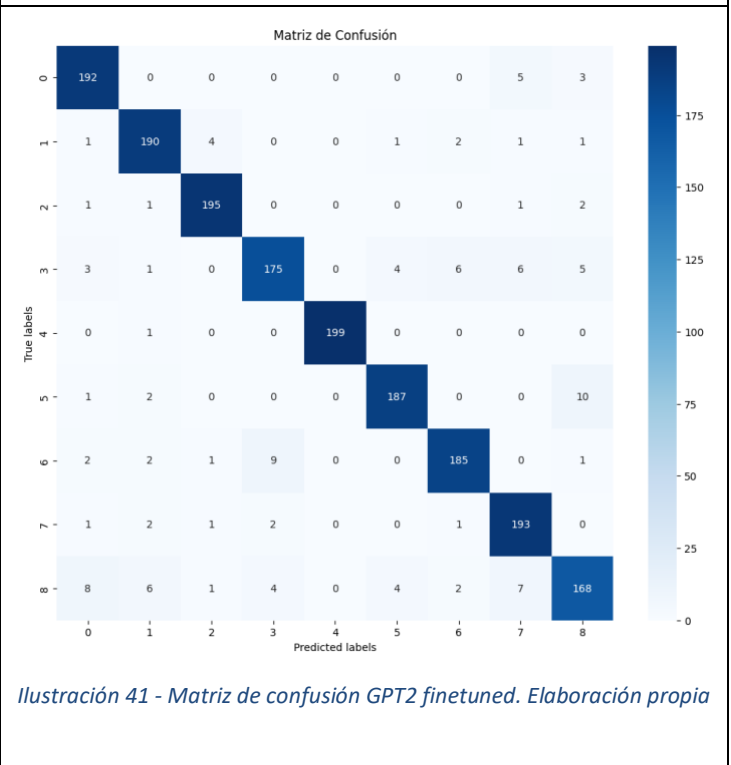
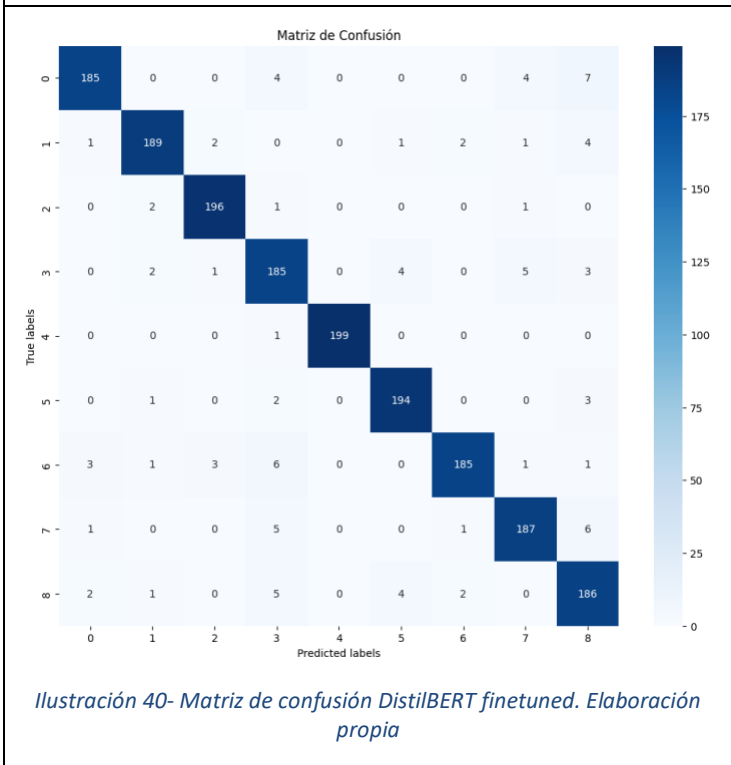
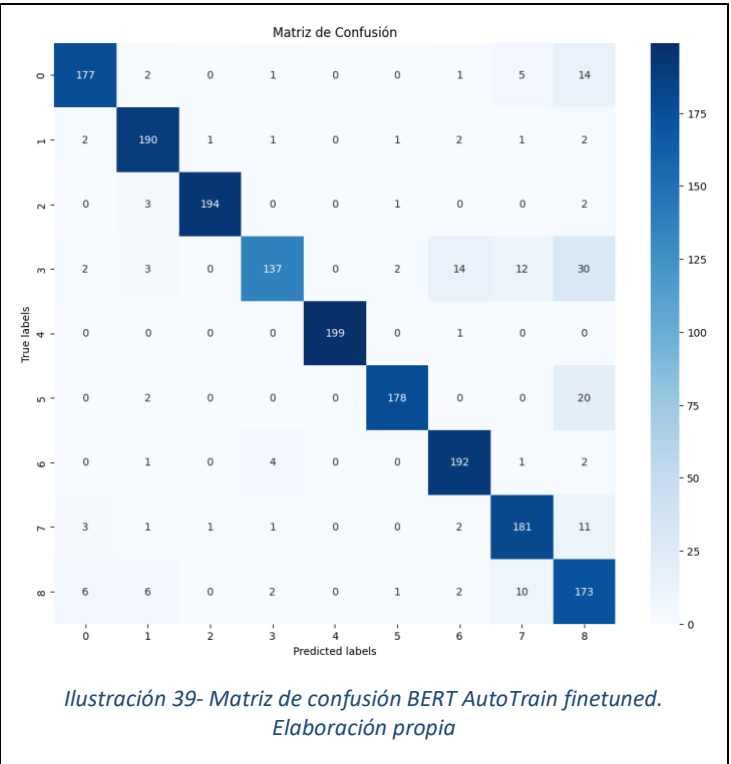
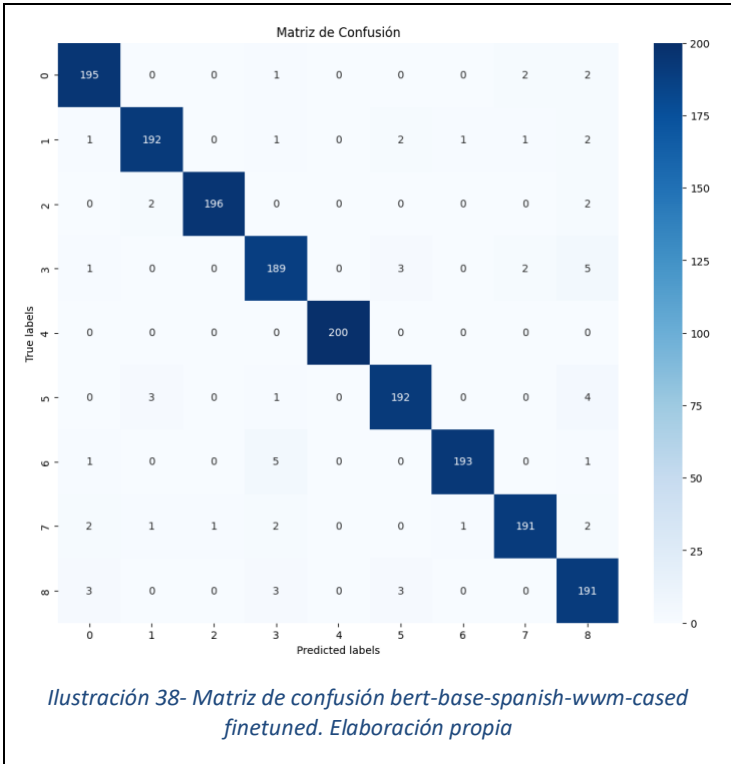


Tabla 5- Matrices de confusión modelos fine-tuned

4.7 Comparativa resultados modelos preentrenados vs fine-tuned

Una vez realizadas las predicciones y calculadas las métricas de los modelos antes y después de realizar Fine-tuning, se procede a comparar los resultados obtenidos.

En las siguientes tablas se muestra para cada modelo los valores de cada métrica antes y después junto con la diferencia obtenida. En todos los casos se observa una mejoría considerable y se obtiene un rendimiento muy bueno.

BERT Spanish (BERTO) dccuchile/bert-base-spanish-wwm-cased				
	Accuracy	Precision	Recall	F1Score
Modelo base	0.097777	0.021378	0.097777	0.026859
<i>Fine-tuned</i>	0.966111	0.966551	0.966111	0.966235
Diferencia	+0.868334	+0.945173	+0.868334	+0.939376

Tabla 6 - Comparativa métricas BERTO preentrenado vs fine-tuned.

DistilBERT (distilbert-base-uncased)				
	Accuracy	Precision	Recall	F1Score
Modelo base	0.111666	0.046514	0.111666	0.037020
<i>Fine-tuned</i>	0.947777	0.948672	0.947777	0.947982
Diferencia	+0.836111	+0.902158	+0.836111	+0.910962

Tabla 7 - Comparativa métricas DistilBERT preentrenado vs fine-tuned

GPT2				
	Accuracy	Precision	Recall	F1Score
Modelo base	0.111111	0.058149	0.111111	0.030885
<i>Fine-tuned</i>	0.935555	0.935573	0.935555	0.935241
Diferencia	+0.824444	+0.877424	+0.824444	+0.904356

Tabla 8 - Comparativa métricas GPT2 preentrenado vs fine-tuned

En particular, el modelo BERT (**dccuchile/bert-base-spanish-wwm-cased**) es el que inicialmente tuvo un peor rendimiento, pero también es el que, después de realizarle Fine-tuning, ha obtenido los mejores resultados.

Estos datos demuestran lo que se había expuesto en la parte teórica, donde se explicaba como gracias al aprendizaje por transferencia es posible aprovechar un modelo preentrenado para una tarea inicial y, después del Fine-tuning, poder utilizarlo para otras tareas con grandes resultados y con unos requisitos mucho menor es que haciéndolo desde cero.

Capítulo 5. DISCUSIÓN

5.1 Resultados del proyecto

En relación con los objetivos específicos para este trabajo, se ha obtenido los siguientes resultados:

A partir de la investigación realizada sobre el estado del arte en NLP y Transformers, se desarrolló la presente memoria en la que se han integrado las contribuciones y avances más relevantes en este campo.

En este sentido, se destaca el peso que ha adquirido la arquitectura Transformers debido a los increíbles resultados que ofrecen los modelos que la utilizan con la realización del fine-tuning.

Por este motivo, en la parte práctica se seleccionaron varios modelos preentrenados basados en Transformers para realizar pruebas de funcionamiento en tareas de clasificación de textos de noticias.

Seguidamente, en cada uno de los modelos escogidos se implementó un Fine-Tuning a partir de un dataset preparado específicamente para la tarea objetivo. Con las métricas obtenidas en cada una de las fases, se pudo comprobar de manera empírica la enorme mejora que supone utilizar esta técnica y los costes tan bajos que conlleva.

Tal y como se muestra en las tablas de 6 a 8, la mejora del modelo preentrenado al fine-tuned supone un incremento de la precisión superior al 80%, lo cual no deja dudas a los resultados que aporta esta técnica.

Capítulo 6. CONCLUSIONES

6.1 Conclusiones

En este trabajo se ha podido experimentar con una pequeña parte de lo que ofrece la arquitectura Transformer. Gracias a la utilización de modelos preentrenados y la realización de Fine-tuning con un dataset relativamente pequeño, se ha conseguido adaptar los modelos para una nueva tarea con una precisión superior al 90% en todos los casos, siendo la mejor de ellas cercana al 97%.

Los frameworks de Pytorch, TensorFlow y Transformers permiten realizar grandes tareas con un tiempo de implementación muy pequeño, costes muy reducidos y unos resultados excepcionales.

Además, la facilidad de la utilización de estas librerías y la puesta a punto de los modelos no tienen nada que ver con la cantidad de conceptos y procesos que ocurren detrás y de lo que el programador no necesita ser consciente.

Esta abstracción es lo que permite al desarrollador centrarse en qué quiere lograr sin necesidad de saber exactamente lo que ocurre en cada paso. Personalmente, opino que esto es un factor fundamental para la completa expansión de los modelos de DL y NLP en entornos de producción en empresas de todo tipo, ya que en estos entornos lo que realmente importa es que cada una de las piezas, cada uno de los códigos, funcione y que lo haga bien, sin importar demasiado en cómo lo hace.

Por otro lado, la expansión de los entornos Cloud y la posibilidad de acceder a máquinas virtuales con hardware de última generación con modalidad de pago por uso ha permitido que cualquier persona con interés pueda permitirse entrenar un modelo por muy poco dinero.

Haciendo una comparativa rápida, las GPUs Nvidia A100 solo están disponibles en un equipo tipo servidor vendido exclusivamente por Nvidia con 8 GPUs de estas y por un precio de 200.000€, una inversión muy elevada incluso para muchas empresas. Sin embargo, utilizar una instancia de máquina virtual con este hardware puede suponer algo más de un euro por hora, un bajo coste para los ahorros de tiempo que suponen.

Para finalizar, como se ha podido ver en el apartado de estado del arte, el ámbito del NLP ha evolucionado mucho en los últimos años y su expansión ha provocado un aumento en la actividad relacionada con este campo, por lo que es de esperar que siga avanzando a pasos agigantados.

6.2 Lecciones aprendidas

La elaboración de este Trabajo Final de Máster ha permitido adquirir numerosas lecciones y conocimientos, de las cuales se destacan:

- **Profundización en NLP y Transformers:** Este proyecto ha exigido llevar a cabo un proceso de documentación muy exhaustivo sobre la materia, teniendo que buscar entre numerosas fuentes de información para llegar a comprender tanto el uso como el funcionamiento de los modelos basados en Transformers, proporcionando una base sólida para futuras investigaciones y proyectos en esta rama.
- **Manejo de grandes volúmenes de datos:** La preparación y limpieza del dataset ha sido una fase clave para los resultados obtenidos en la realización del Fine-tuning. En este trabajo se han tenido que tratar los datos adecuadamente para poder utilizarlos con éxito en el Fine-tuning de los modelos.
- **Experiencia práctica con herramientas y librerías de ML:** La utilización de Pytorch, Tensorflow y Transformers han aportado nuevos conocimientos prácticos sobre la implementación de modelos de aprendizaje automático con las herramientas más novedosas disponibles en la actualidad.
- **Experiencia práctica con entornos Cloud:** La ejecución del código desarrollado se hizo en Google Colab, sin embargo, se investigaron las diferentes opciones disponibles y las herramientas que ofrece cada uno de los principales proveedores (Azure, AWS y GCP), pero se descartaron por quedar demasiado lejos del alcance del proyecto. A pesar de ello, se ha podido poner en práctica parte de lo aprendido en la asignatura de Arquitecturas Cloud de este Máster.
- **Importancia de la experimentación y realización de pruebas:** La selección de los modelos y la realización del Fine-tuning ha supuesto una gran oportunidad para probar distintas configuraciones y parámetros para encontrar soluciones válidas.

6.3 Sugerencias de mejora

Una vez finalizado el trabajo y a pesar de haber obtenido unos resultados muy buenos, siempre hay espacio para la mejora y por ello se presenta una serie de posibles acciones encaminadas a mejorar:

- **Ampliación del dataset:** El dataset utilizado contenía 1900 registros para cada una de las 9 categorías. Sería recomendable ampliar este dataset para poder realizar un mejor entrenamiento en la tarea de clasificación y obtener aún mejores resultados en la clasificación de noticias.
- **Experimentación con distintos modelos:** Este trabajo se ha centrado en la utilización de modelos basados en Transformers, pero sería interesante ampliar el abanico y utilizar modelos basados en otras arquitecturas como LSTM y poder comparar los resultados.

Capítulo 7. FUTURAS LÍNEAS DE TRABAJO

El presente trabajo ha logrado alcanzar los objetivos marcados al inicio, sin embargo, del mismo modo que siempre hay margen para la mejora, un proyecto de estas características siempre puede ampliarse si existe la motivación y se dispone de los recursos necesarios.

Por este motivo, se sugieren las siguientes líneas futuras para ampliar el trabajo aquí realizado:

- **Exploración de nuevos modelos y arquitecturas:** Seguir con la investigación y experimentación sobre nuevos modelos y arquitecturas en NLP.
- **Integración de los modelos con otros idiomas:** Adaptar los modelos para que sean capaces de realizar las tareas en otros idiomas.
- **Ampliación del número de categorías para clasificación:** Las categorías definidas en los modelos de este trabajo han sido reducidas a un total de 9. Una posible línea futura podría ampliar el número de categorías y que el modelo fuera capaz de trabajar con un mayor abanico de textos de noticias.
- **Automatización de la clasificación:** Integrar el modelo de clasificación con la publicación de noticias para realizar la tarea de manera automática.

Capítulo 8. REFERENCIAS

- 1- Hugging Face. (2023). En Wikipedia. https://es.wikipedia.org/wiki/Hugging_Face Consultado el 9 de julio de 2023.
- 2- Foote, K. D. (2023, 6 de julio). A Brief History of Natural Language Processing. DataVersity. <https://www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/> Consultado el 01 de agosto de 2023.
- 3- Winograd, T. (s.f.). SHRDLU. Stanford University. <http://hci.stanford.edu/~winograd/shrdlu/> Consultado el 01 de agosto de 2023.
- 4- IBM Research. (2012, 19 de febrero). Watson and the Jeopardy! Challenge [Video]. YouTube. <https://www.youtube.com/watch?v=P18EdAKuC1U> Consultado el 01 de agosto de 2023.
- 5- Cantor, M. (2023, 8 de mayo). Nearly 50 news websites are 'AI-generated', a study says. Would I be able to tell? The Guardian. <https://www.theguardian.com/technology/2023/may/08/ai-generated-news-websites-study> Consultado el 01 de agosto de 2023.
- 6- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems* (p./pp. 5998--6008), . <https://arxiv.org/abs/1706.03762> Consultado el 01 de agosto de 2023.
- 7- Saxena, S. (2021, 12 de marzo). Beginner's Guide to Support Vector Machine (SVM). Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/03/beginners-guide-to-support-vector-machine-svm/> Consultado el 01 de agosto de 2023.
- 8- Sotaquirá, M. (2018, 3 de septiembre). ¿Qué es una Red Neuronal? Codificando Bits. <https://www.codificandobits.com/blog/que-es-una-red-neuronal/> Consultado el 01 de agosto de 2023.
- 9- The Click Reader. (s.f.). Building a Convolutional Neural Network. <https://www.theclickreader.com/building-a-convolutional-neural-network/> Consultado el 05 de agosto de 2023.
- 10- Red neuronal prealimentada. (2022, 24 de octubre). En Wikipedia. https://es.wikipedia.org/wiki/Red_neuronal_prealimentada Consultado el 06 de agosto de 2023.
- 11- Sotaquirá, M. (2019, 8 de junio). Introducción a las Redes Neuronales Recurrentes. Codificando Bits. <https://www.codificandobits.com/blog/introduccion-redes-neuronales-recurrentes> Consultado el 06 de agosto de 2023.
- 12- Torres, J. (2019, 22 de septiembre). Redes Neuronales Recurrentes. En Deep Learning – Introducción práctica con Keras (SEGUNDA PARTE). Contenido abierto del libro. Kindle Direct Publishing. ISBN 978-1-687-47399-8. Colección WATCH THIS SPACE – Barcelona (Libro 6). <https://torres.ai/redes-neuronales-recurrentes/> Consultado el 06 de agosto de 2023.
- 13- Olah, C. (2015, 27 de agosto). Understanding LSTM Networks. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> Consultado el 06 de agosto de 2023.
- 14- Hochreiter, S., & Schmidhuber, J. (1997). LONG SHORT-TERM MEMORY. *Neural Computation*, 9(8), 1735-1780. <http://www.bioinf.jku.at/publications/older/2604.pdf> Consultado el 06 de agosto de 2023.

- 15- Matias C., M. (2021, 15 de mayo). Palabras Vacías en Español (stop words) ft Python. Medium. <https://cr0wg4n.medium.com/palabras-vac%C3%ADas-en-espa%C3%B1ol-stop-words-ft-python-3117e52d2bff> Consultado el 07 de agosto de 2023.
- 16- NLTK Documentation. (2023). <https://www.nltk.org/> Consultado el 10 de agosto de 2023.
- 17- SpaCy Documentation (2023). <https://spacy.io/usage/spacy-101> Consultado el 10 de agosto de 2023.
- 18- Grandury, M. (2021). Word embeddings. NLP de cero a cien. [Documento PDF]. GitHub. https://github.com/somosnlp/nlp-de-cero-a-cien/blob/main/1_word_embeddings/word_embeddings.pdf Consultado el 10 de agosto de 2023.
- 19- Uzila, A. (2022, 30 de agosto). All You Need to Know About Bag of Words and Word2Vec — Text Feature Extraction: Why Word2Vec is better, and why it's not good enough. Towards Data Science. <https://towardsdatascience.com/all-you-need-to-know-about-bag-of-words-and-word2vec-text-feature-extraction-e386d9ed84aa> Consultado el 22 de agosto de 2023.
- 20- Stecanella, B. (2019, 10 de mayo). Understanding TF-IDF: A Simple Introduction. MonkeyLearn. <https://monkeylearn.com/blog/what-is-tf-idf/> Consultado el 22 de agosto de 2023.
- 21- García Ferrero, I. (2018). Estudio de Word Embeddings y métodos de generación de Meta Embeddings (Proyecto de Fin de Grado). Universidad del País Vasco. Disponible en https://addi.ehu.es/bitstream/handle/10810/29088/MemoriaTFG_IkerGarciaFerrero.pdf Consultado el 22 de agosto de 2023.
- 22- Tovar, J. C. (2023, 24 de julio). ¿Qué es Word2Vec, GloVe y Fasttext? Foro de Huawei. <https://forum.huawei.com/enterprise/es/%C2%BFqu%C3%A9-es-word2vec-glove-y-fasttext/thread/683598187732025344-667212895009779712?author=667212676360708115> Consultado el 22 de agosto de 2023.
- 23- Shperber, G. (2017, 26 de julio). A gentle introduction to Doc2Vec. Wisio. <https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e> Consultado el 22 de agosto de 2023.
- 24- Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Disponible en <https://nlp.stanford.edu/pubs/glove.pdf> Consultado el 22 de agosto de 2023.
- 25- GloVe. (2023, 4 de julio). En Wikipedia. <https://es.wikipedia.org/wiki/GloVe> Consultado el 22 de agosto de 2023.
- 26- Joshi, P. (2022, 23 de junio). A Step-by-Step NLP Guide to Learn ELMo for Extracting Features from Text. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/> Consultado el 22 de agosto de 2023.
- 27- Sebastian Ruder, "Transfer Learning - Machine Learning's Next Frontier". <http://ruder.io/transfer-learning/>, 2017. Consultado el 10 de septiembre de 2023.
- 28- Tunstall, L. (2021). Aprendizaje por transferencia NLP de cero a cien. [Documento PDF]. GitHub. https://github.com/somosnlp/nlp-de-cero-a-cien/blob/main/4_transformers_aprendizaje_por_transferencia/aprendizaje_por_transferencia.pdf Consultado el 10 de septiembre de 2023.

- 29- Calvo, J. (2020, 23 de agosto). Aprendizaje por transferencia: NLP. European Valley. <https://www.europeanvalley.es/noticias/aprendizaje-por-transferencia-nlp/> Consultado el 10 de septiembre de 2023.
- 30- Vaca, A. (2022). Transformers en Procesamiento del Lenguaje Natural. Instituto de Ingeniería del Conocimiento, Universidad Autónoma de Madrid. <https://www.iic.uam.es/innovacion/transformers-en-procesamiento-del-lenguaje-natural/> Consultado el 05 de julio de 2023.
- 31- Uszkoreit, J. (2017, 31 de agosto). Transformer: A Novel Neural Network Architecture for Language Understanding. Blog de Google. <https://blog.research.google/2017/08/transformer-novel-neural-network.html> Consultado el 05 de julio de 2023.
- 32- Sudmann, A. (Ed.). (2019). The Democratization of Artificial Intelligence: Net Politics in the Era of Learning Algorithms. Disponible en https://library.oapen.org/bitstream/id/61810f6f-bf40-404d-b688-61e9ac3405a4/external_content.pdf Consultado el 05 de septiembre de 2023.
- 33- Kumar T, S. (2021, 22 de junio). Natural Language Processing – Sentiment Analysis using LSTM. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/06/natural-language-processing-sentiment-analysis-using-lstm/> Consultado el 20 de septiembre de 2023.
- 34- Alammar, J. (2018, 27 de junio). The Illustrated Transformer. <http://jalammar.github.io/illustrated-transformer/> Consultado el 20 de julio de 2023.
- 35- Amatriain, X., Sankar, A., Bing, J., Bodigutla, P. K., Hazen, T. J., & Kazi, M. (2023). Transformer models: an introduction and catalog. Disponible en <https://arxiv.org/abs/2302.07730> Consultado el 05 de septiembre de 2023.
- 36- Sachdeva, N. (2023, 29 de agosto). What is Attention Mechanism in Deep Learning? Insights & Blogs Around Software Engineering. <https://insights.daffodilsw.com/blog/what-is-the-attention-mechanism-in-deep-learning> Consultado el 05 de septiembre de 2023.
- 37- Diario Expansión (2023, 28 de junio). Gemini: la nueva inteligencia artificial de Google que desafía a ChatGPT.. <https://www.expansion.com/economia-digital/2023/06/28/649c012de5fdeac7448b465f.html> Consultado el 05 de octubre de 2023.
- 38- Thompson, A. D. (2023). GPT5. LifeArchitect.ai. <https://lifearchitect.ai/gpt-5/> Consultado el 05 de octubre de 2023.
- 39- Dealroom.co. (2023). Hugging Face. Consultado el 10 de septiembre de 2023. https://app.dealroom.co/companies/hugging_face
- 40- Hugging Face. (2023). Competitions. <https://huggingface.co/competitions/> Consultado el 10 de septiembre de 2023.
- 41- Hugging Face. (2023). Fine-tune a pretrained model. <https://huggingface.co/docs/transformers/training> Consultado el 10 de septiembre de 2023.
- 42- Díaz, R. Métricas de Clasificación. The Machine Learners. <https://www.themachinelearners.com/metricas-de-clasificacion/> Consultado el 10 de septiembre de 2023.

- 43- Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. Disponible en <https://arxiv.org/abs/1602.02410> Consultado el 05 de julio de 2023.
- 44- Hanslo, R. (2021, 1 de noviembre). Deep Learning Transformer Architecture for Named Entity Recognition on Low Resourced Languages: State of the art results. Disponible en <https://arxiv.org/abs/2111.00830> Consultado el 05 de julio de 2023.
- 45- Kaiser, Ł. (2017, 4 de octubre). Attention is all you need; Attentional Neural Network Models [Video]. Pi School. https://www.youtube.com/watch?v=rBCqQTEfxvg&ab_channel=PiSchool Consultado el 05 de julio de 2023.
- 46- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv. <https://arxiv.org/pdf/1802.05365.pdf> Consultado el 22 de agosto de 2023.
- 47- Rodrawangpai, B., & Daungjaiboon, W. (2022). Improving text classification with transformers and layer normalization. Machine Learning with Applications. <https://doi.org/10.1016/j.mlwa.2022.100403> Consultado el 22 de agosto de 2023.
- 48- Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. arXiv. <https://arxiv.org/pdf/1801.06146v5.pdf> Consultado el 20 de septiembre de 2023.
- 49- Mayo, M. (2018, 3 de mayo). Preprocesamiento de datos de texto: un tutorial en Python. Medium. <https://medium.com/datos-y-ciencia/preprocesamiento-de-datos-de-texto-un-tutorial-en-python-5db5620f1767> Consultado el 10 de septiembre de 2023.

Capítulo 9. ANEXOS

ANEXO 1: Relación de archivos de código disponibles en el repositorio Github personal.

Los códigos utilizados para las distintas tareas realizadas a lo largo de la presente memoria están disponibles de manera pública en el [proyecto TFM en mi repositorio personal de Github](#).

La siguiente tabla muestra una relación al completo de todos los scripts desarrollados, el apartado de la memoria para el cual se desarrolló y su ubicación en el repositorio:

APARTADO MEMORIA	RUTA	SCRIPT
2.1.4.1	TFM_Transformers/	tokenization_lemmatization_example.ipynb
3.2	TFM_Transformers/ETL	extract_data_from_bigquery.py
3.2	TFM_Transformers/ETL	extract_data_from_full_dataset_9cat.ipynb
4.5	TFM_Transformers/TEST/BASE	test_bert_base_model.ipynb
4.5	TFM_Transformers/TEST/BASE	test_distilbert_base_model.ipynb
4.5	TFM_Transformers/TEST/BASE	test_gpt2_base_model.ipynb
4.6	TFM_Transformers/FineTune	finetune_bert.ipynb
4.6	TFM_Transformers/FineTune	finetune_distilbert.ipynb
4.6	TFM_Transformers/FineTune	finetune_gpt2.ipynb
4.6	TFM_Transformers/TEST/FINETUNED	Test_bert_autotuned.ipynb

Tabla 9 - Relación de scripts con código Python disponibles a través de Github

ANEXO 2: Métricas utilizadas

Para la evaluación de los modelos se han utilizado las siguientes métricas ^[42]:

- **Accuracy:** Se trata de la proporción entre el número de predicciones correctas y el total de predicciones. $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- **Precision:** Proporción de los verdaderos positivos entre todos los positivos predichos. $Precision = \frac{TP}{TP+FP}$
- **Recall:** Ratio de verdaderos positivos entre todos los positivos. $Recall = \frac{TP}{TP+FN}$
- **F1 Score:** Combina precisión y recall: $F1 = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}$
- **Curva ROC:** Se trata de una gráfica que representa el porcentaje de verdaderos positivos.
- **AUC (Area Under Curve):** A partir de la gráfica de la curva ROC se puede calcular el área bajo la curva cuyos valores se encuentran entre 0 y 1 y equivale a una función de densidad.
- **Matriz de confusión:** Se trata de una matriz donde las filas representan los valores reales de cada clase y las columnas los valores de las predicciones del modelo. Esta métrica permite ver fácilmente si el modelo clasifica correctamente o si existe confusión entre clases.

En las métricas anteriores, se han utilizado las abreviaturas:

- **TP (True Positive):** Valores que el modelo clasifica positivos y su valor real es positivo.
- **TN (True Negative):** Valores que el modelo clasifica negativos y su valor real es negativo.
- **FP (False Positive):** Valores que el modelo clasifica positivos y su valor real es negativo.
- **FN (False Negative):** Valores que el modelo clasifica negativos y su valor real es positivo.

ANEXO 3: Hugging Face AutoTrain

A modo de investigación se ha querido probar la herramienta que ofrece Hugging Face llamada AutoTrain. Esta herramienta permite realizar fine-tuning a un modelo preentrenado sin necesidad de escribir una sola línea de código.

Para acceder a ella, tan sólo hay que pulsar en el menú superior de la plataforma en Solutions>AutoTrain.

A continuación, se mostrará la página principal de la herramienta y permite escoger entre dos opciones:

- **Create new Project:** Versión sencilla que es la que se ha probado en este caso.
- **Try AutoTrain Advanced:** Permite desplegar la herramienta en un contenedor en una máquina virtual bajo demanda. Más costoso pero ofrece mayor personalización.

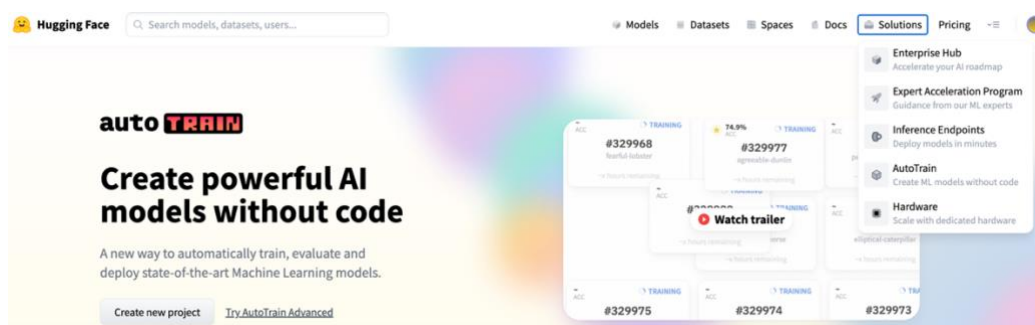


Ilustración 42- AutoTrain. Página principal. Elaboración propia

Una vez se pulsa en “Create new project” la herramienta pedirá que escojamos un nombre para el proyecto, la tarea objetivo, el idioma y, además, AutoTrain permite escoger manualmente el modelo preentrenado sobre el que trabajar o lo escoge de manera automática:

New project

Select a task, language, and how many models you want to train. You will prepare data in the next step.

Project name
eg: imdb-sentiment-analysis

Task
Vision Text Tabular

Text Classification (Binary) [checked]
Text Classification (Multi-class)
Token Classification
Question Answering (Extractive)
Translation

Summarization
Text Regression

Text Classification (Binary) is the task of classifying texts into two distinct groups.

Model choice
Automatic [selected] Manual

Recommended. AutoTrain will automatically pick the best models for this task.

Language
Spanish [selected]

Create project Cancel

Ilustración 43- AutoTrain. Configuración nuevo proyecto. Elaboración propia

A continuación, hay que escoger los datos utilizados para el entrenamiento. En este caso se ha subido una versión reducida del dataset de entrenamiento utilizado para el Fine-tuning en los demás modelos. En caso de superar los 3.000 registros en el dataset, el entrenamiento conllevará un coste económico:

The screenshot shows the AutoTrain interface. At the top, there's a header with 'auto TRAIN BETA', a dropdown menu set to 'spanish-new-classification-9categories', and tabs for 'Text Classification (Multi-class)', 'Spanish', and 'Training'. On the left, there are three sidebar options: 'Data' (selected), 'Trainings', and 'Metrics'. The main area is titled 'Auto-Split' and shows a dataset named '9cat_mini_dataset.csv' with a status of 'DONE' and 'Updated 3 minutes ago'. Below this, it says 'Type: file' and 'text: text target: label'. A note mentions that data files are hosted on a private Hugging Face repository. Below the Auto-Split section, there's a 'Select number of model candidates' section with a note: 'Training more models is more expensive, but it increases the probability of finding one that performs greatly.' There are four buttons: '1 model', '3 models', '5 models', and '10 models' (which is selected). Below this is the 'Training data' section with a table showing data usage. The table has two columns: 'Data split' and 'Rows'. It lists '9cat_mini_dataset.csv' as a 'file' with 2,970 rows. The total is 2,970 / 3,000. Below the table is the 'Training Cost' section with a note: 'Calculated according to the data size and the number of models trained.' It shows a yellow box with the text: '\$27 extra training fee. You have 9 more models than the allowed amount for your account. There will be a \$27 extra training fee. You will be charged using the payment method attached to your account. To train more models, try AutoTrain Advanced.' At the bottom, there is a 'Start models training' button.

AutoTrain BETA

spanish-new-classification-9categories Text Classification (Multi-class) Spanish Training

Data

Trainings

Metrics

Auto-Split

9cat_mini_dataset.csv

DONE • Updated 3 minutes ago

Type: file

text: text target: label

Your data files are hosted on a private [Hugging Face repository](#).

Ilustración 44- AutoTrain. Selección de los datos. Elaboración propia

Una vez escogido o cargado el dataset, en el apartado Training la herramienta permite entrenar hasta 10 modelos diferentes para obtener el mejor resultado posible. Entrenar un modelo con menos de 3.000 filas es gratuito, sin embargo, ampliar estas limitaciones conlleva cierto coste, por lo que, en casos que se vaya a realizar un Fine-tuning con un dataset muy grande, es más conveniente utilizar otro sistema como el “AutoTrain Advanced” que cobra por hora de ejecución o cualquier otro proveedor Cloud.

Select number of model candidates

Training more models is more expensive, but it increases the probability of finding one that performs greatly.

1 model 3 models 5 models 10 models

Training data

Data usage for this project

Data split	Rows
9cat_mini_dataset.csv file	2,970
Total	2,970 / 3,000

Training Cost

Calculated according to the data size and the number of models trained.

\$27 extra training fee

You have 9 more models than the allowed amount for your account.

There will be a \$27 extra training fee. You will be charged using the payment method attached to your account.

To train more models, try AutoTrain Advanced.

Start models training

Ilustración 45 - AutoTrain. Configuración del entrenamiento. Elaboración propia

Una vez escogidas todas las opciones, el entrenamiento, empieza de manera automática sin necesidad de más intervención por parte del usuario.

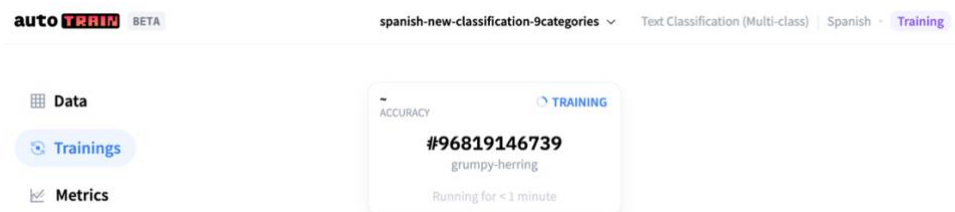


Ilustración 46 - AutoTrain. Entrenamiento del modelo. Elaboración propia.

Tras apenas 10 minutos de espera, *AutoTrain* ofrece un resumen de las distintas métricas de desempeño del modelo y el modelo está listo para usarse a través de la propia plataforma de Hugging Face.

Tal y como se muestra en la siguiente imagen, con un dataset 2970 elementos se ha obtenido una precisión del 92,42% un gran resultado teniendo en cuenta el esfuerzo que ha requerido.

#96819146739 ✓ DONE
grumpy-herring

Accuracy	
92.42%	
Loss	Accuracy
0.2609	0.9242
F1_macro	F1_micro
0.9252	0.9242
F1_weighted	Precision_macro
0.9252	0.9307
Precision_micro	Precision_weighted
0.9242	0.9307
Recall_macro	Recall_micro
0.9242	0.9242
Recall_weighted	
0.9242	

Ilustración 47 - AutoTrain. Métricas del modelo tras Fine-tuning. Elaboración propia



Ilustración 48 - Hugging Face - Página del modelo. Elaboración propia



Ilustración 49- Hugging Face. Hosted inference API. Elaboración propia

Finalmente, en la página del modelo, permite a cualquier usuario probarlo y ver los resultados de la clasificación.

[PÁGINA INTENCIONADAMENTE EN BLANCO]