TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica Superior de Jaén

Tecnologías del lenguaje para tratar problemas de desinformación en entornos periodísticos

Autor: Cristian Ibáñez Bautista

Grado: Ingeniería Informática

Director:
Departamento del director:
Fecha: 20/06/2024

Licencia CC

CREA

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Manuel Carlos Díaz Galiano y María Teresa Martín Valdivia, tutores del Proyecto Fin de Grado titulado: **Tecnologías del lenguaje para tratar problemas de desinformación en entornos periodísticos**, que presenta Cristian Ibáñez Bautista, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, 2 Julio de 2024

El alumno: Los tutores:

Firmado por IBAÑEZ BAUTISTA, CRISTIAN (FIRMA) el día 02/07/2024 con un certificado emitido

Firmado digitalmente por DIAZ GALIANO MANUEL CARLOS - ***3299** el día 03/07/2024

MARTIN VALDIVIA MARIA TERESA - 26006595N

Firmado digitalmente por MARTIN VALDIVIA MARIA TERESA - 26006595N
Fecha: 2024.07.02 12:52:13 +02'00'

Cristian Ibáñez Bautista

Manuel Carlos Díaz Galiano

María Teresa Martín Valdivia

Agradecimientos

Me gustaría poder dar las gracias a todas aquellas personas que han hecho posible que este trabajo de fin de grado haya podido realizarse con gran éxito .

Entre ellos me gustaría destacar la oportunidad recibida por los profesores Doña María Teresa Martín Valdivia y Don Manuel Carlos Díaz Galiano , los cuales me han guiado durante el desarrollo del mismo , y también me gustaría agradecer el apoyo de otros miembros del grupo de investigación de SINAÍ , como Sergio Stoia.

Índice

1. Especificación del proyecto

1.1. Introducción

Cada vez más, las tecnologías abarcan un mayor número de ramas en desarrollo e investigación, sobretodo en campos en los cuales una mayor cantidad de usuarios acceden constantemente y se alimentan de su contenido, como bien puede ser el ámbito del periodismo o de la información compartida, por ello se requiere de una mayor investigación para poder evitar en la mayor medida posible cualquier tipo de acto nocivo o dañino que pueda perjudicar estas aplicaciones de la tecnología, por ello que este trabajo de fin de grado se enfocará en el tratamiento de la desinformación en estos entornos.

Este trabajo se basará en un proyecto de investigación sobre la base de los conocimientos del procesamiento del lenguaje natural,los cuales tienen cada vez una mayor evidencia científica gracias al auge de la inteligencia artificial. Estos modelos nos permitirán desarrollar sistemas computacionales eficientes y capaces de automatizar el proceso de detección de estos actos dañinos sobre noticias de la actualidad mayormente basadas en las redes sociales, ya que se caracterizan por tener un gran potencial de extensión y propagación por todo usuario de las redes sociales.

1.2. Motivación

Hoy en día los diferentes usuarios, pasan gran parte del día rodeados de sistemas tecnológicos tanto móviles como fijos, los cuales nos transmiten información a través de cualquier aplicación o red social, en diferentes webs de opiniones de distintos usuarios, o incluso muchas marcas de periodismo optan ya por un formato digital, lo cual hace que llegue a una mayor cantidad de usuarios. Esto mismo determina la importancia de este proyecto, el gran número de usuarios que puede llegar a obtener la información de una noticia redactada por una persona, que puede tener una gran repercusión.

La aparición de este nuevo modo de periodismo tecnológico, consigue que se pueda llegar a crear una sociedad en el que la gran parte de la población tenga

acceso a toda la información actual, por lo que sería más fácil la comunicación de diferentes noticias con cierta preocupación o una mayor gravedad, pero y si esta noticia no fuera del todo cierta o tuviera una parte de ella que contuviese información falsa. Al igual que una noticia verídica esta sería captada por toda esta población tecnológica y podría causar desinformación entre todos sus usuarios, o lo que podría causar incluso más daño según el contexto, es decir, podrían ser manipulados mediante una noticia con fines específicos.

Estas “fake news” pueden ejercer una gran problemática en la sociedad de la información en la que vivimos, y cualquier persona con acceso a estas aplicaciones o a diferentes medios tecnológicos de transmisión de información tendría en sus manos una peligrosa fuente de poder sobre la opinión y las creencias de los usuarios, sobre todo en algunos campos como puede ser la política, conflictos de guerra o incluso en temas relacionados con la salud y enfermedades.

Este ámbito dentro de la sociedad de la información causa un gran revuelo debido a la preocupación de las personas o usuarios de todo tipos de tecnologías que entran en contacto con cualquier tipo de información.

Esta preocupación podemos verla reflejada en diferentes estudios realizados a lo largo de los años (ver Figura 1.1) los cuales nos muestran como desde el inicio de la era de la computación distintas preocupaciones relacionadas por el temor de los usuarios hacia diversos aspectos sensibles que los sistemas tecnológicos pueden llegar a ocasionar.

La motivación de este proyecto junto a todo lo nombrado anteriormente, puede destacar por la ignorancia de los propios usuarios que aumentan aún más el potencial de estas fake news, es decir, como se muestra en el estudio mostrado en la revista *Nature* **[1]**, podemos observar cómo las personas afirman que era de máxima importancia que cada uno de los usuarios debían de compartir solo información precisa y verídica dentro de las redes sociales. Esto llevó a los investigadores a plantear la hipótesis de que hay una gran posibilidad de que muchos de los usuarios de la red, difundan información errónea o falsa, pero esta de un modo no intencionado.

Esto nos lleva a aumentar más si cabe la preocupación de las personas por este problema que persiste a lo largo de la historia.

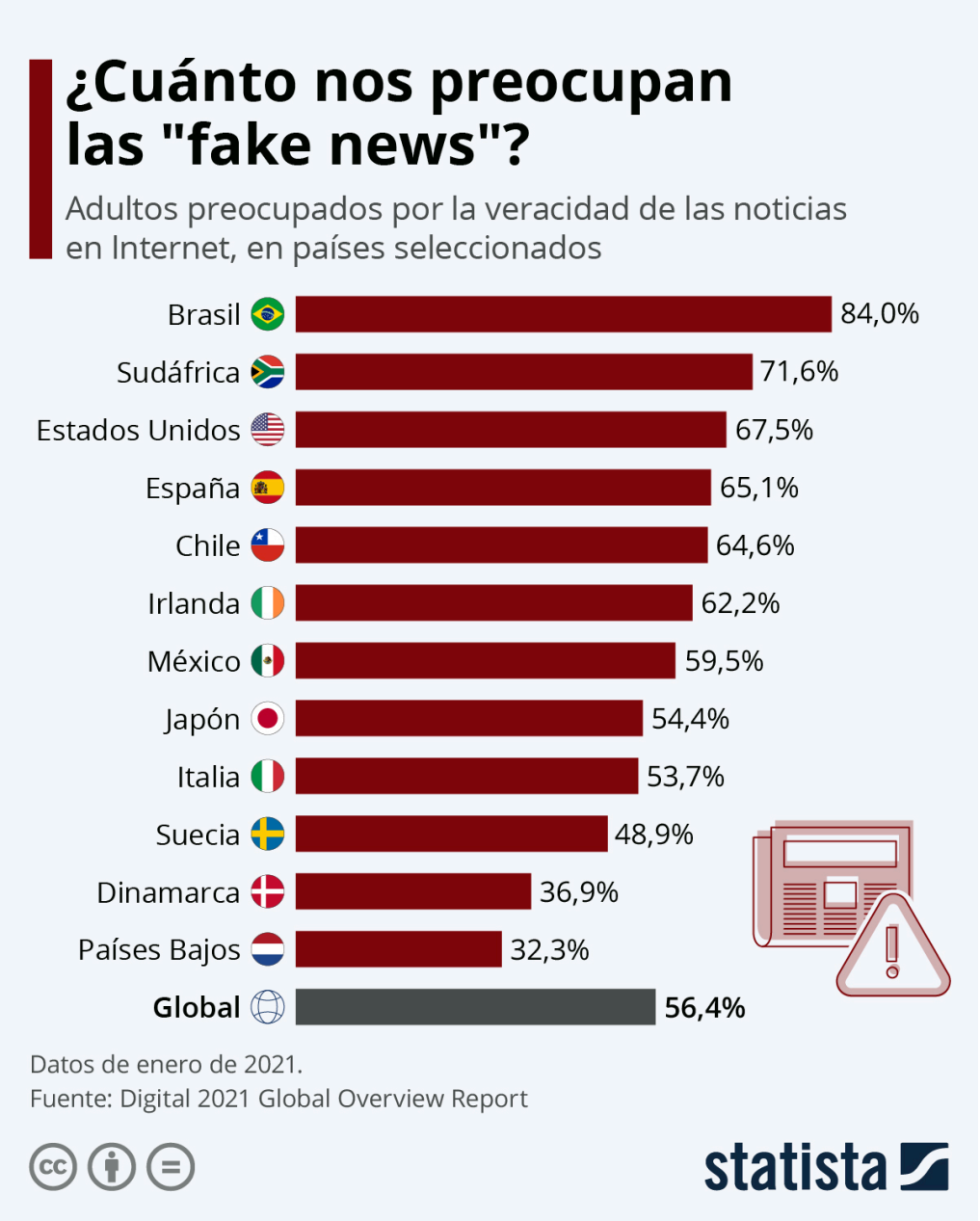

Figura 1.1: Estadísticas preocupación mundial (Fuente: Gráfico: La amenaza de las "fake news" | Statista)

1.3. Estado del arte

Las fake news o las noticias falsas, pueden parecer relativamente recientes por el hecho de su relación con la tecnología, pero esto no es así , es decir las fake news datan desde el inicio de la transmisión de información, hasta la antigua Roma con diferentes ejemplos de información falseada hasta nuestros días.

Después de diferentes hitos de noticias falsas, tanto en la edad Media como en la edad Moderna, no es hasta 1835 cuando se cometen las primeras transmisiones de fake news sobre el uso de tecnologías con “La gran mentira de la luna” **[2]** (ver figura 1.2), en la cual se divulgó sobre un acontecimiento que se ha considerado como el hallazgo más significativo de la historia.

THE SUN.

NEW YORK, TUESDAY MORNING, AUGUST 25, 1835.

GREAT ASTRONOMICAL DISCOVERIES, LATELY MADE BY SIR JOHN HERSCHEL, L.L.D. F.R.S. &c. At the Cape of Good Hope.

Figura 1.2: Primera noticia falsa “La gran mentira de la luna” (Fuente:El gran fraude de la Luna (1835) – NeoTeo)

Posteriormente dieron lugar otros de los ejemplos de mayor importancia como la transmisión de diferentes falsedades, esta vez mediante radios, durante la Primera y Segunda guerra mundial en 1917 y 1938, llamada la guerra de la desinformación, con la que los diferentes bandos manipulaban las opiniones de las personas, las cuales carecían de medios óptimos para poder contrastar la veracidad de estas noticias.

Después de este intercambio de noticias falsas en las guerras que jugaron un papel fundamental en ella, no es hasta la década de los 90 cuando comienzan a surgir diferentes tipos de “fake news” como bulos o rumores, a través de los primeros sitios webs, debido a la popularización de internet en el inicio de esta época. Sin embargo, no es hasta la década de 2010 cuando se comienza a aplicar el procesamiento del lenguaje natural en este ámbito de detección de fake news, con un desarrollo muy notable desde las primeras investigaciones con técnicas de “PLN” o “NLP” para identificar noticias falsas, hasta el 2020 que comienzan los desarrollos

de diferentes modelos más avanzados como BERT y GPT que consiguen obtener grandes resultados con una cantidad de avances bastante significativa.

Para poder observar más de cerca el desarrollo de procesamiento del lenguaje natural dentro de esta rama de investigación sobre las fake news, podemos indagar acerca de muchos de los proyectos ya desarrollados a lo largo de los años **[23]**.

Entre ellos podemos destacar algunos últimos estudios realizados por la gran pandemia, como el desarrollado por el instituto de ingeniería del conocimiento, el cual trataba de detectar noticias falsas en entornos periodísticos acerca de la COVID-19 **[25],** las cuales tenían y actualmente todavía contienen un gran impacto sobre la opinión pública. Esta investigación se encargó de realizar algunos re-entrenamientos de modelos con más de 500.000 noticias de hasta 400 fuentes distintas lo que proporcionaba al modelo una mayor variedad entre ellas y un aprendizaje mayor.

Al igual que esta investigación han surgido muchas otras desde el comienzo del uso de modelos del lenguaje que adquieren un gran potencial dentro de esta rama facilitando su desarrollo, como bien pueden ser:

- **“The factual”:** realizada en 2016, la cual se desarrolló una extensión de nuestro navegador que tenía la capacidad de analizar la credibilidad de más de 10.000 historias diarias, para ello se involucra en una investigación de distintas características de la propia información como fuentes, autor…etc
- **“Logically”:** La cual constaba de una aplicación móvil y al igual que la anterior también de una extensión del navegador, que ofrecían distintos verificadores de hechos e imágenes, analizando tanto afirmaciones, opciones incluso eventos puntuales. Esto era posible ya que el modelo desarrollado era capaz de rastrear y monitorear en tiempo real más de 1 millón de webs y plataformas de traspaso de información, evaluando la veracidad de cada una de ellas.
- Otra de las más importantes de los últimos años es “Check by Meedan” que fue una herramienta desarrollada por la propia organización en 2019 y se encargaba de la verificación de hechos en colaboración con grandes empresas como Whatsapp y Facebook que como sabemos son dos de las mayores redes sociales actualmente existentes y buscan también la eliminación de la desinformación por medio de sí mismas **[26]**.

1.4. Definición del Objetivo

Este proyecto se inicia con el objetivo de analizar y desarrollar sistemas computacionalmente eficaces y eficientes, que sean capaces de realizar la función de detectar noticias falsas, o las llamadas “Fake news” para poder evadir la malversación de las mismas evitando así conflictos y problemas con mayores riesgos que pueden derivar de este problema.

Esta rama de investigación está siendo muy demandada, por lo que se trata de un problema mayor, que como hemos observado cada vez obtiene un mayor nivel de inquietud entre los usuarios, es por ello que nuestro objetivo se basa en un término tan concreto como las noticias calificadas como falsas, pero a la vez un tema tan extenso que prácticamente afecta a la totalidad de la población o más específicamente a todos los usuarios aplicaciones basadas en traspaso de información, ya que como podemos observar en la gráfica (ver Figura 1.3), los usuarios pasan una larga parte del día conectados a las redes sociales, lo que aumenta la probabilidad de encontrar noticias falsas.

Figura 1.3: Tiempo medio empleado en redes sociales (Fuente:[Gráfico: La adicción a las redes sociales en el mundo | Statista]( ))

Por lo tanto para poder construir estos sistemas debemos de ayudarnos de herramientas basadas en el conocimiento del lenguaje natural, para poder hacer entender al sistema computacional sobre los conocimientos necesarios para realizar esta tarea, por ello hablamos del procesamiento del lenguaje natural, el cual guiará el camino hacia un sistema autónomo que sea capaz de realizar esta tarea sin problemas y con unos resultados óptimos.

1.5. Checkthat!

Para resolver este problema de la desinformación generada por las fake news, trataremos de desarrollar e implementar herramientas para resolverlo, estas herramientas se basarán en modelos de aprendizaje automático. Estos modelos serán evaluados y conseguiremos establecer aquellos con mejores resultados, y estos serán utilizados para participar en una competición llamada Check That, la cual es desarrollada por la campaña anual del CLEF 2024 **[3]**, esto es una organización que se encarga de realizar una evaluación sistemática de diferentes sistemas de las tecnologías y el acceso a la información. Esta competición consta de diferentes tareas en las que diferentes usuarios podrán presentar su proyecto desarrollado para obtener los mejores resultados acerca de cada uno de los desafíos establecidos.

En cuanto a nuestro proyecto vamos a participar en la tarea 1 y en la tarea 2. La tarea 1 se desarrolla sobre el idioma español y consiste en el desarrollo de una herramienta que sea capaz de determinar si es necesario la verificación de un tweet o transcripción, para establecer si contiene rasgos característicos de una noticia falsa. En cuanto a la forma de evaluar se deberá desarrollar un modelo de clasificación binaria en la que diferenciaremos entre la clase 0 (No) si no necesita verificación y 1 (Yes) sí sería conveniente analizar el tweet por que puede llegar a ser una noticia falsa.

En segundo lugar, la tarea 2 al igual que la tarea 1 sin embargo esta será desarrollada en inglés para tratar diferentes noticias, pero en este caso nos basamos en la determinación de la objetividad o subjetividad de una noticia, es decir una clasificación binaria, en la que diferenciamos 0 (Objetivo) y 1 (Subjetividad) si presenta o no rasgos que determinan si una oración presenta objetividad o subjetividad **[3]**.

Figura 1.4: Logo competición CLEF+Checkthat! (Fuente: CheckThat!)

1.6. Herramientas Utilizadas

En esta sección se presentarán las herramientas, aplicaciones y servicios necesarios para poder realizar este proyecto, cada una de ellas contiene la funcionalidad necesaria para poder completar diferentes tareas a lo largo del desarrollo de nuestro proyecto.

1.6.1. Python

Python es un lenguaje de programación que está aumentando su índice de popularidad por su gran potencia y facilidad, usable para una gran diversidad de problemas, sobre todo los orientados a la inteligencia artificial o al procesamiento del lenguaje natural.

Este lenguaje es caracterizado por su tipado dinámico, gracias al cual sus variables pueden tomar diferentes valores a lo largo de todo el programa, al igual que con el inmenso número de bibliotecas ya incorporadas en el mismo lenguaje , las cuales facilitan e introducen un aumento de la eficacia en la programación. Python contiene librerías, como transformers, Torch, o incluso diferentes herramientas de procesamiento del texto , que nos facilitaran mucho el desarrollo del trabajo **[4]**.

1.6.2. Visual Studio Code

Visual Studio Code o VSCode, es entorno de programación o editor de código para programadores de acceso gratuito, código abierto y multiplataforma, este ha sido desarrollado por Microsoft, el cual ha conseguido crear una herramienta la cual gracias a sus extensas características ha conseguido ser una de las herramientas más utilizadas por todo tipo de programadores.

Entre sus características podemos destacar:

- Gestión eficiente de proyectos y carpetas de trabajo
- Integración con control de versiones, lo que ofrece una posibilidad de integrar con repositorios para su almacenamiento y control del código.
- Extensibilidad, una de las características más reveladoras, la capacidad de instalar un enorme conjunto de extensiones que nos ayudarán a realizar infinidad de tareas, con distintas tecnologías y variedad de lenguajes.
- Potentes herramientas tanto de depuración como de pruebas, lo que hace de ella un uso eficiente que ayuda al programador a completar sus objetivos con una mayor agilidad.

1.6.3. Google Colab

Google Colaboratory o google colab, es una plataforma creada y dirigida por google la cual proporciona una herramienta gratuita, el cual consta de un entorno de cuaderno de tipo Jupyter que nos permite codificar y ejecutar nuestro código en el lenguaje anteriormente mencionado “Python” y este es alojado en la nube.

Esta herramienta como su propio nombre indica ha sido desarrollada para los trabajos en equipos, es decir un entorno en el cual se puede colaborar entre varios usuarios, facilitando la comunicación entre ellos. Otra de las características por las que destaca este entorno de trabajo es por sus recursos informáticos, entre los que encontramos la CPU, GPU y TPU, gracias a estos recursos podemos llegar a ejecutar programas y códigos que requieren de un alto nivel de procesamiento, reduciendo así los tiempos de ejecución, esto facilitará nuestro trabajo, ya que nuestro proyecto requiere de una alta tasa de procesamiento debido al uso de modelos de aprendizaje automático.

Figura 1.5: Logo google colaboratory (Fuente:Google Colab)

1.6.4. Hugging face

Hugging face es una empresa tecnológica que se encarga del desarrollo de herramientas y servicios los cuales son orientados a la inteligencia artificial, orientados a la rama del procesamiento del lenguaje natural. Su principal enfoque se centra en el desarrollo de modelos de aprendizaje automático, así como el almacenamiento de los mismos.

Esta plataforma, se ha vuelto muy popular debido a que se ha desarrollado como una plataforma de código abierto, la cual permite a todo tipo de usuarios y desarrolladores, acceder a esta plataforma e interactuar con ella almacenando sus propios modelos o incluso utilizar diferentes modelos ya guardados en esta plataforma, para sus propias tareas. Entre las distintas empresas que desarrollan modelos en Hugging face encontramos algunas como google (Bert, T5, Pegasus), Microsoft (DeBERTa, UniLM, ProphetNet) o incluso amazon (DistilBERT, Electra, GPT-3).

Esta plataforma no es una herramienta como tal, pero gracias a ella hemos podido tener acceso libre a cada uno de los modelos que hemos utilizado para hacer nuestro propio desarrollo y entrenamiento, además hemos almacenado en ella para que puedan trabajar con los modelos más usuarios, los distintos modelos pre entrenados, lo que facilita mucho la comunicación y traspaso de información entre los distintos clientes de la plataforma, por lo que aumenta la eficiencia de los trabajos cooperativos.

1.6.5. Drive

El almacenamiento es un recurso muy valioso debido a la cantidad de datos que se necesitan almacenar, por ello que se requieren plataformas para almacenar información heterogénea, de esto es lo que se encarga la plataforma Drive.

Google drive es una plataforma gratuita de almacenamiento de archivos que provienen de distintas fuentes. En esta plataforma aparte de su característica principal de almacenamiento, también tiene la funcionalidad de crear distintos tipos de documentos, hojas de cálculo, presentaciones, formularios…

Esta herramienta nos proporciona un total de 15 Gigabytes de almacenamiento , por lo que no es un recurso ilimitado, si queremos hacer un uso gratuito, es decir si es posible acceder a una mayor cantidad de recursos pero tendría costes asociados.

Drive es una de las herramientas más utilizadas por los usuarios hoy en día, y por ello nosotros vamos a tomar beneficio de ella en el desarrollo de nuestros modelos, ya que almacenaremos tanto el contenido necesario para la construcción de nuestros modelos, como el almacenaje de los propios modelos previamente entrenados.

1.6.6. Twitter

Twitter como todos conocemos es una red social gratuita que permite la comunicación entre todos los usuarios de forma muy rápida y sencilla. Esto como veremos más adelante es una de las causas por la que nuestro proyecto se desarrollará sobre esta aplicación.

Twitter fundado en 2006 cuenta con una de las cifras más altas de usuarios registrados en su plataforma, esta permite al usuario compartir diferentes tipos de contenidos, como opiniones, gustos, iniciar conversaciones privadas, o incluso debatir acerca de un tema de interés. Estos mensajes llamados tuits, son fragmentos de textos no con más de 280 carácteres, permitiendo emoticonos y otro tipo de caracteres especiales.

Como se puede observar twitter, es una red social con bastante libertad para el usuario, es por ello que cualquier persona con unas intenciones dañinas podría realizar cualquier trabajo sin mucho problema. Además a diferencia de otras redes sociales, twitter carece de un mecanismo de control de usuarios, es decir, un usuario

puede registrarse anónimamente sin tener que introducir datos personales, lo que dificulta la búsqueda de responsabilidades ante cualquier acto malévolo.

2. Análisis

En este apartado del proyecto se va a realizar un análisis del proyecto realizado para las dos tareas, con un enfoque inicial de cómo se va a orientar la realización del mismo. Para poder realizar el posterior desarrollo del proyecto debemos de comentar primero los campos de investigación que vamos a utilizar en el mismo para poder introducir el contexto de cada uno de los elementos de nuestro trabajo.

Todos ellos nacen de la propia rama de la inteligencia artificial y poco a poco se va desarrollando y dando lugar a otras formas de sistemas informáticos inteligentes (ver Figura 2.1).

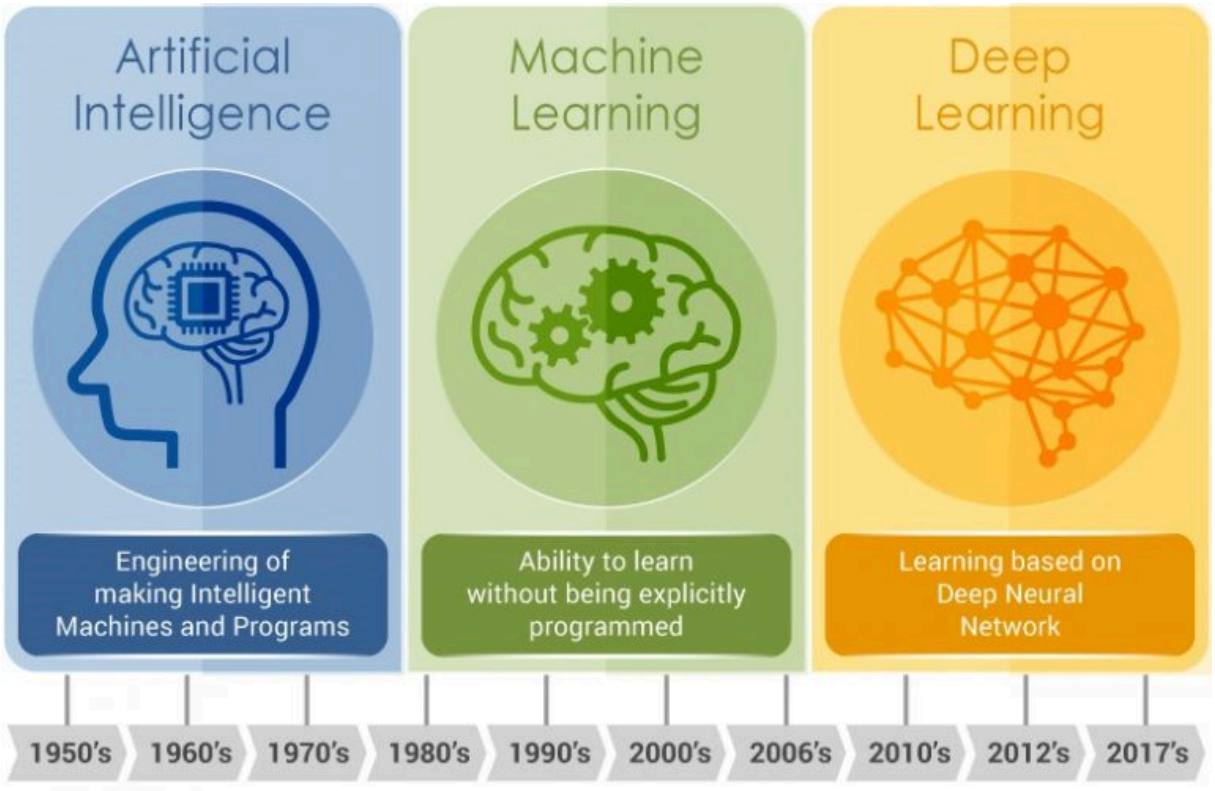

Figura 2.1: Historia de la IA (Fuente:El aprendizaje IA)

2.1. Procesamiento del Lenguaje natural

El procesamiento del lenguaje natural surge como una rama de investigación derivada del campo de la inteligencia artificial, esta rama se encarga del estudio y el desarrollo de la interacción de las personas con las máquinas, es decir trata de realizar o mejorar la comunicación entre ambas partes mediante el uso del conocimiento del lenguaje natural .

El procesamiento del lenguaje natural no es algo nuevo que haya surgido en los últimos años sino que tiene una gran trayectoria a lo largo de la historia. Nació en 1950 como una rama derivada de la inteligencia artificial unida a la lingüística de ahí su tratamiento en distintas ramas de la investigación, con el objetivo de estudiar problemas de generación y comprensión automática del lenguaje humano es decir del lenguaje natural.

Anteriormente ya se había hablado de diferentes trabajos relacionados con el procesamiento del lenguaje natural, no es hasta los años 50 cuando Alan Turing publicó el artículo con el que comenzó a hablarse de Inteligencia artificial. En él podemos ver lo que ahora en la actualidad llamamos la prueba de turing **[22]**(ver Figura 2.2).

La prueba de Turing se encarga de demostrar la inteligencia de las máquinas en un marco experimental y de investigación. Alan propuso este test junto a un matemático, informático y un especializado en lógica, en el que decía que durante el transcurso de una comunicación entre una persona y una máquina, la máquina pasaba el test si la persona no era capaz de distinguir si se está comunicando con otra persona o un sistema informático.

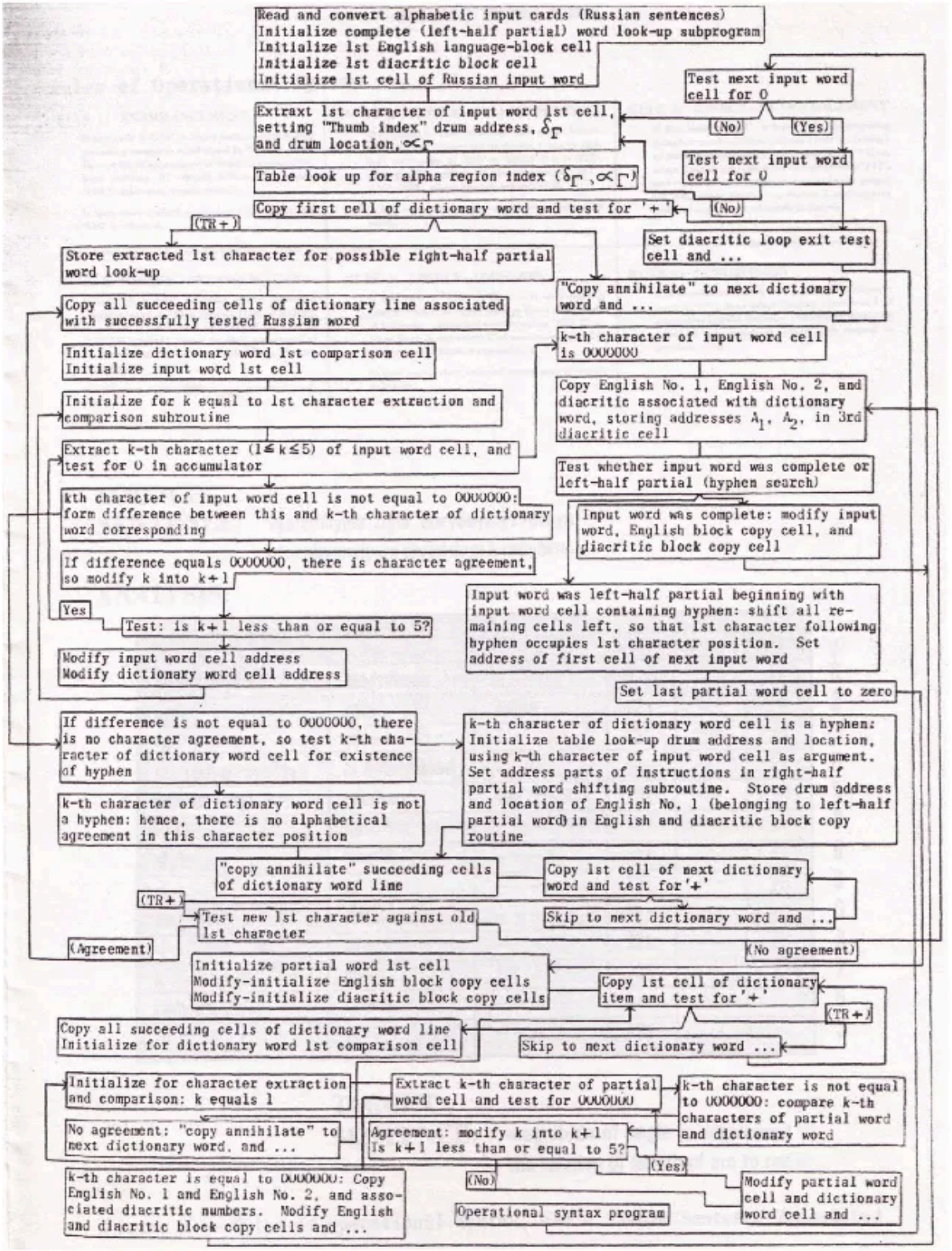

Figura 2.2: Periódico donde se publicó el test de Turing (Fuente: Test de Turing 1950)

2.1.1. Multidisciplinar

Como es obvio necesitamos de personas con un conocimiento en informática suficiente para poder manejar la programación de estas inteligencias artificiales, pero también es necesario incluir diferentes disciplinas para poder completar su funcionamiento, entre ellos se destacan:

Lingüistas: Como bien sabemos que nuestra tarea se trata de establecer una comunicación del humano con la máquina, es necesario el tratamiento por parte de lingüistas que son capaces de comprender a la perfección el lenguaje humano y sean capaces de establecer el modelo lingüístico para que en base a él, los ingenieros informáticos sean capaces de implementar el código siendo eficiente y funcional.

Matemáticos: Como hemos nombrado en el apartado anterior, es necesario el desarrollo de modelos matemáticos para poder entendernos con los ordenadores, por eso es importante el trabajo de la disciplina matemática.

Psicólogos: En cuanto a la profesión que estudia la psicología está tomando una gran importancia sobre todo en proyectos como nuestro Trabajo de fin de grado en el que es necesario un estudio de la subjetividad de las palabras para analizar más allá de un análisis lingüístico y profundizar en un análisis de sentimientos, en tareas como análisis de la toxicidad, ironía o incluso humor.

2.1.2. ¿Para qué sirve el PLN?

El procesamiento del lenguaje natural está en pleno auge debido a la explosión de la inteligencia artificial en el mundo actual, muchas personas no tienen el conocimiento necesario para comprender cómo la inteligencia artificial es capaz de funcionar de manera que es capaz de interaccionar con el usuario en el mismo lenguaje que le proporciona.

Gracias al procesamiento del lenguaje natural las máquinas de hoy en día son capaces de comprender la entrada que le proporciona el usuario y es capaz de procesarla y generar una salida en respuesta a esta, pero, ¿Cómo es capaz de entender el lenguaje humano? De esto es lo que se encarga la rama del procesamiento del lenguaje natural, de realizar los pasos necesarios para poder entregar como entrada una serie de datos y que el ordenador sea capaz de entenderlos y poder procesar información en este mismo lenguaje, para que al

establecer respuestas el usuario humano sea capaz de entender y así realizar una comunicación entre hombre-máquina más natural y eficiente.

Para poder "enseñar" a las máquinas el lenguaje humano debemos de realizar un proceso de modelización matemática ya que los ordenadores sólo entienden de bytes y dígitos por ello que necesitamos de otras disciplinas para poder realizar esta tarea.

Este desarrollo tecnológico nos acerca cada vez más a un sistema informático más similar al humano lo que hace que el miedo de las personas acerca de este tema, sobre la inteligencia artificial aumenta por momentos.

Los sistemas tienden a un desarrollo independiente y cada vez más alejado del control humano. Este miedo podemos verlo reflejado en la siguiente imagen con el llamado valle inquietante el cual trata que la familiaridad de las personas cae en picado cuando el parecido a una persona aumenta **[14]** (ver Figura 2.3).

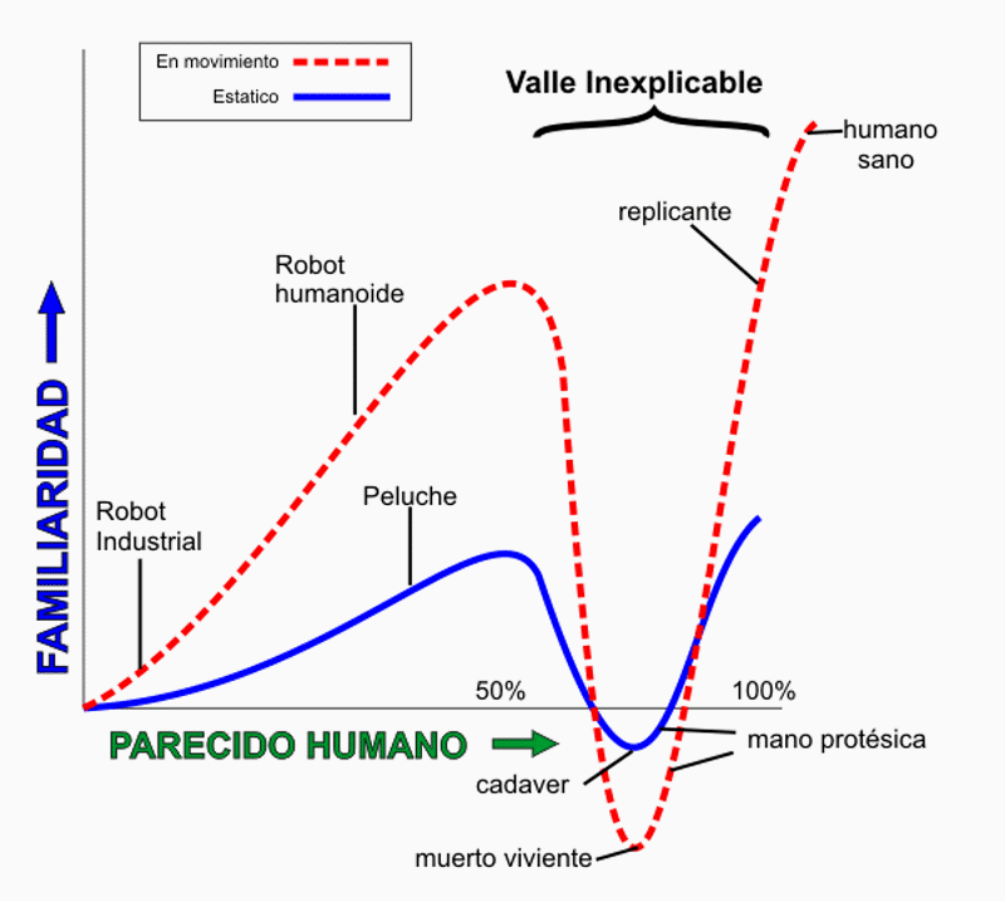

Figura 2.3: Valle inquietante (Fuente: [El miedo que nos generan los robots que parecen personas: el fenómeno del valle inquietante]( ))

2.2. Machine learning

Este modelo de aprendizaje de los sistemas computacionales, aunque puede llegar a parecer moderno, surgió hace bastantes años, desde 1950, en el momento que Alan Turing creó el "Test de Turing" **[9]**, donde la máquina debía engañar al propio humano. Junto a este hito del machine learning también cabe destacar el desarrollo del primer algoritmo capaz de aprender desarrollado por Arthur Samuel en 1952 **[10]**, el que trataba de una partida de damas donde la inteligencia cada vez era más avanzada, es decir desarrollaba su aprendizaje continuamente.

El machine learning, como su propio nombre indica, se trata de un aprendizaje automático, este comprende una de las ramas más importantes dentro de la inteligencia artificial que permite que las máquinas adquieran un aprendizaje óptimo, el cual las lleve a realizar distintas tareas con buenos resultados.

A modo de definición el machine learning es una aplicación de distintas prácticas de inteligencia artificial, para convertir un sistema computacional, en un sistema capaz de reconocer patrones y convertir un conjunto de datos en un programa que es capaz de adquirir e inferir nuevo conocimiento a partir del mismo.

Gracias a este reconocimiento de patrones, hace la diferencia con distintas técnicas aplicadas anteriormente, es decir, esto le permite a estos sistemas desarrollar la capacidad de adaptarse a los nuevos cambios que van surgiendo en los conjuntos de datos presentados como entrada a nuestro sistema **[13]**.

Como hemos dicho estos algoritmos trabajan buscando una serie de patrones e información en todo tipo de datos. Replicar el comportamiento humano en un sistema computacional es bastante complicado, pero es lo que se busca en estos algoritmos, aunque no cuentan con la capacidad de razonar y aprender, estos tratan de simular esa experiencia en la medida de lo posible.

Entre estos algoritmos podemos diferenciar dos tipos principales :

Aprendizaje supervisado: Este tipo de aprendizaje se caracteriza principalmente en la forma en la que se produce el entrenamiento de sus algoritmos, en este caso en el entrenamiento de proporciona un conjunto de datos etiquetado por lo que el algoritmo tendrá consciencia sobre la clasificación que contiene cada una de las partes del conjunto de entrenamiento. Por lo que este tipo de aprendizaje tiene un seguimiento, es decir su entrenamiento va guiado gracias a las características implicadas como entrada.

Aprendizaje no supervisado: A diferencia del aprendizaje anterior, como se puede deducir de su nombre, el entrenamiento que realiza este tipo de algoritmos se basa en datos no etiquetados, por lo que estos se centran en conseguir encontrar ciertos patrones específicos que se encuentran dentro del conjunto de datos de entrenamiento.

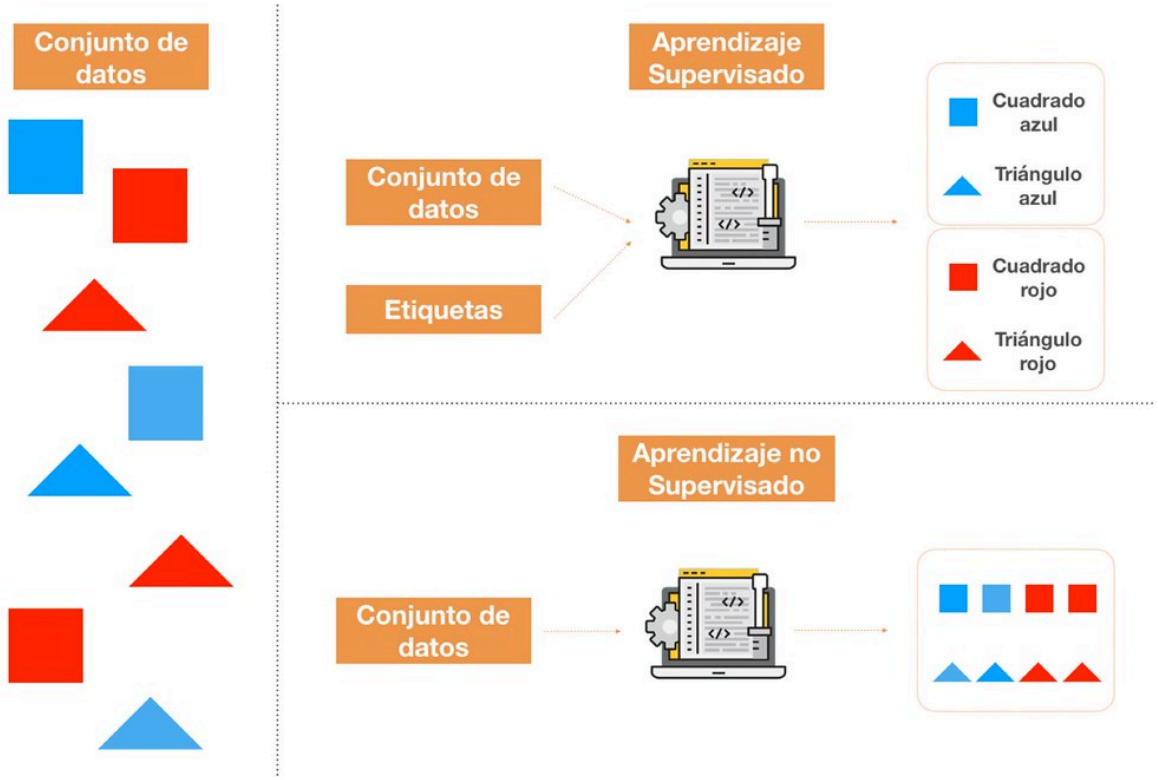

Figura 2.4: Tipos de algoritmos aprendizaje automático (Fuente:Diferencia entre aprendizaje supervisado y no supervisado)

Como podemos observar en la imagen anterior (ver Figura 2.4), vemos cómo a partir del cambio en el conjunto de entrada surge la diferencia entre estos, en el aprendizaje supervisado las figuras contienen su definición en forma de etiqueta por lo que el modelo no debe pararse a entender de qué se trata cada figura y las diferencias entre las clases, mientras que en el otro caso, en el aprendizaje no supervisado este solo cuenta con las figuras en la imagen y no contiene esa definición o etiqueta que le simplifica el trabajo de entrenamiento, por lo que es el propio modelo que tiene que establecer la diferencia entre clases por si mismo.

2.3. Deep learning

Deep learning es un término usualmente introducido dentro del machine learning, sin embargo aunque el objetivo es relativamente similar, se trata de dos conceptos bastante diferentes.

Como primera definición de los modelos computacionales de Deep Learning, podemos hablar del desarrollo de sistemas que tratan de imitar las características arquitectónicas del sistema nervioso de un humano. Esto permite que dentro de todo el sistema al completo haya diferentes redes de procesamiento y que cada una de ellas sea capaz de especializarse en distintas características de los datos de entrenamiento **[11]**.

Deep learning por lo tanto es una aproximación de la propia percepción humana, es decir, la inteligencia artificial cada vez se enfoca más en el desarrollo de sistemas que tengan mayor relación con el propio humano, pero a la vez se construyen sistemas mayormente orientados al aprendizaje no supervisado, es decir sin requerir intervención humana, lo cual resulta un poco contradictorio. Esto provoca que los sistemas computacionales cada vez más se alejan del aprendizaje mediante el humano y se basan más en un desarrollo propio.

Una de las diferencias más notables entre machine learning deep learning ocurre en el proceso de aprendizaje, como observamos en la imagen (ver Figura 2.5), es la fase de extracción de características, en la cual extraemos la información más valiosa de las secuencias de texto para transmitirlas a nuestros modelos, por lo tanto a diferencia de machine learning en la cual es el propio programador el que debe de extraer las características codificando, en deep learning el propio modelo ya es capaz de extraer las características por sí mismo lo que facilita mucho el proceso, además al tratar el propio modelo la información que va a utilizar para su entrenamiento el nivel de error del mismo se reduce, ya que se elimina el tratamiento humano del mismo.

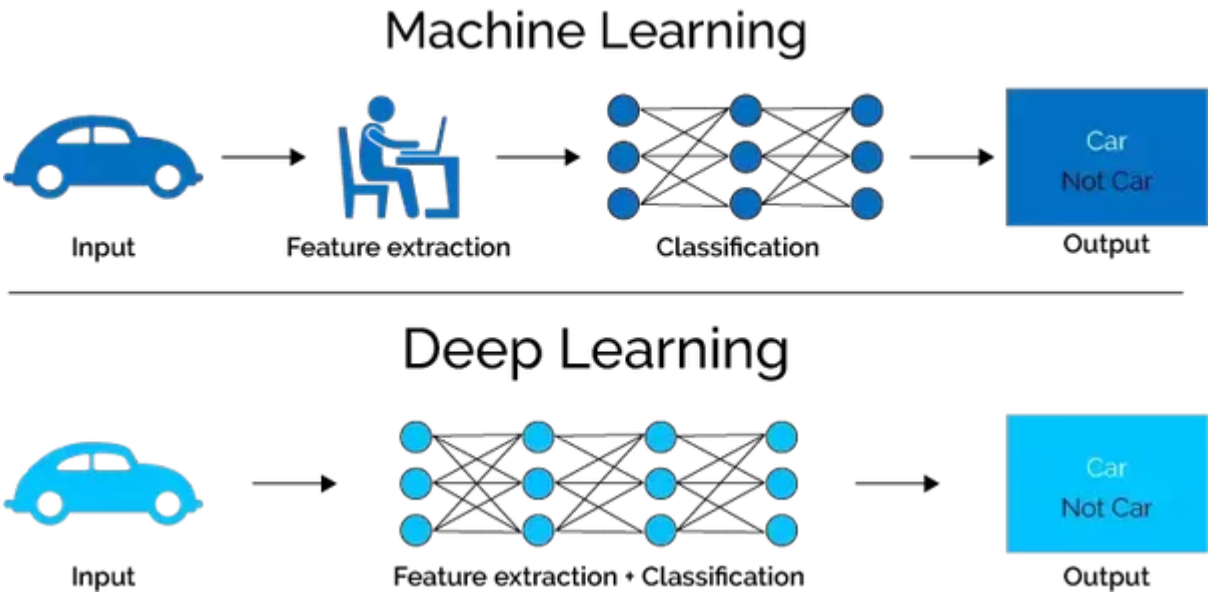

Figura 2.5: Fases de machine learning y deep learning (Fuente: El aprendizaje profundo (deep learning) en la optimización de estructuras)

2.3.1. Transformers

El concepto de transformer viene definido como un modelo de aprendizaje profundo para el propio procesamiento del del lenguaje natural y por su nombre como bien indica, funcionan con mecanismos de atención basado en transformadores.

Esto es muy reciente ya que no es hasta 2017 cuando se acuñó este término, gracias a los investigadores de la rama de inteligencia artificial de Google.

Los transformer están construidos con una red neuronal basada en codificador-decodificador, y al contrario que otras redes como las redes recurrentes o las redes convolucionales, estos no requieren de una secuencia fija de entrada de datos ya que son capaces de utilizar entradas de distintas longitudes.

Dentro de la arquitectura de los transformers hay diferentes tipos de innovaciones destacables como los embeddings posicionales, que son representaciones matriciales estáticas de los integrantes del texto, es decir cada una de las palabras de la secuencia de texto se encuentra codificada en un vector, y en este caso está incluida la posición de la palabra que representa. Esto permite al algoritmo conocer en cualquier momento la posición relativa de cada token o palabra de la secuencia.

Esto junto a otras características han demostrado una mejora de los transformers con respecto a otros modelos de desarrollo en la inteligencia artificial como podemos observar en la imagen (ver Figura 2.6).

Con los transformer podemos trabajar de dos formas, mejor dicho en dos fases distintas:

- En la primera de estas hablamos del **"Pre-training" o pre entrenamiento**, en la cual se trata de la fase de aprendizaje del modelo, en la que obtenemos como va a ser la estructura del lenguaje del modelo de forma general, adquiriendo así conocimiento general sobre las palabras de entrada.
- Seguidamente pasamos a hablar de un término más utilizado en las propias investigaciones, el "Fine-tuning", este, una vez entrenados los modelos, se realiza una fase de aprendizaje especial, ya que ahora en vez de entrenar al modelo de una manera más general, se trata de hacer una tarea más específica sobre el y adaptar el propio modelo a tareas más concretas hacia las que queramos orientar a nuestro propio transformer.

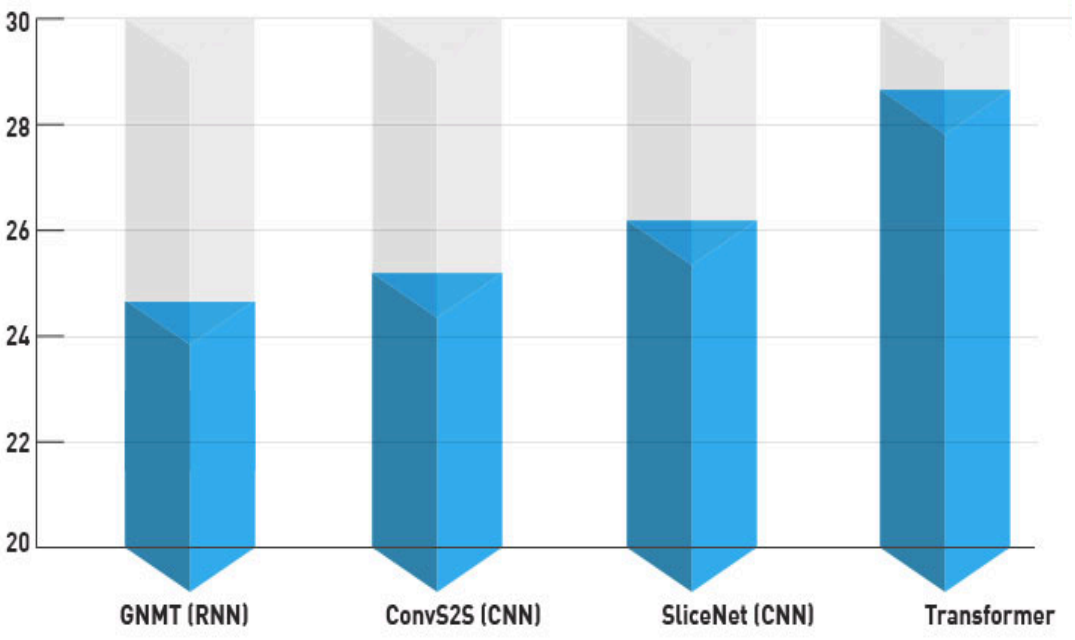

Figura 2.6: Utilización de las distintos tipos de redes neuronales

(Fuente:Transformers en Procesamiento del Lenguaje Natural - IIC)

2.4. Importancia del Proyecto

Podríamos preguntarnos, hasta qué punto este trabajo es necesario o si realmente afecta a la actualidad. Pero nada más lejos de la realidad este proyecto adquiere una gran importancia, ya que está teniendo un crecimiento proporcional a las redes sociales, es decir un crecimiento exponencial, por lo que es de suma importancia.

Las redes sociales se pueden ver de distintos modos, originalmente estas aplicaciones se concibieron por las personas como algo muy novedoso que facilitaba la comunicación entre cualquier usuario de una manera muy sencilla y al alcance de prácticamente toda la población que tuviera acceso a internet. Pero al igual que cualquier término tiene otra cara no tan beneficiosa para los usuarios, como puede ser la fuente de poder que adquiere cualquier persona con estas redes sociales en sus manos. Estas personas que tuviesen cualquier tipo de intención de realizar acciones dañinas hacia la población podían hacerlo sin ningún tipo de problema ni impedimento sobre ello.

Figura 2.7: Historia de las redes sociales (Fuente:Historia de las Redes Sociales: Guía cronológica | Sephor)

Como podemos ver en la imagen anterior (ver Figura 2.7), la importancia de las tecnologías comenzó a tomar mayor relevancia al establecer la red de internet como pública en 1991, pero no es hasta 1997 con el primer intento de aplicación capaz de realizar la acción de compartir información entre usuarios con la primera aplicación SixDegrees **[5]** . Esta aplicación fundada por Andrew Weinreich, la cual basándose en la teoría de los "seis grados" de ahí su nombre, en la que dice que una persona está conectada con otra del planeta mediante una conexión de 6 vínculos de relación, esta teoría fue establecida por el académico estadounidense Stanley Milgram, y gracias a ella se creó esta primera red social de la historia, la cual constaba de una especie de "directorio electrónico" la cual conectaba a cada usuario con las personas que conocía, y adicionalmente con todas las personas conocidas de su propio amigo y así sucesivamente (ver Figura 2.8). Pero no es hasta 2004 cuando se da el verdadero crecimiento exponencial de estas hasta el día de hoy, gracias a la red social de Facebook,Twitter o Tuenti **[12]** (ver Figura 2.1).

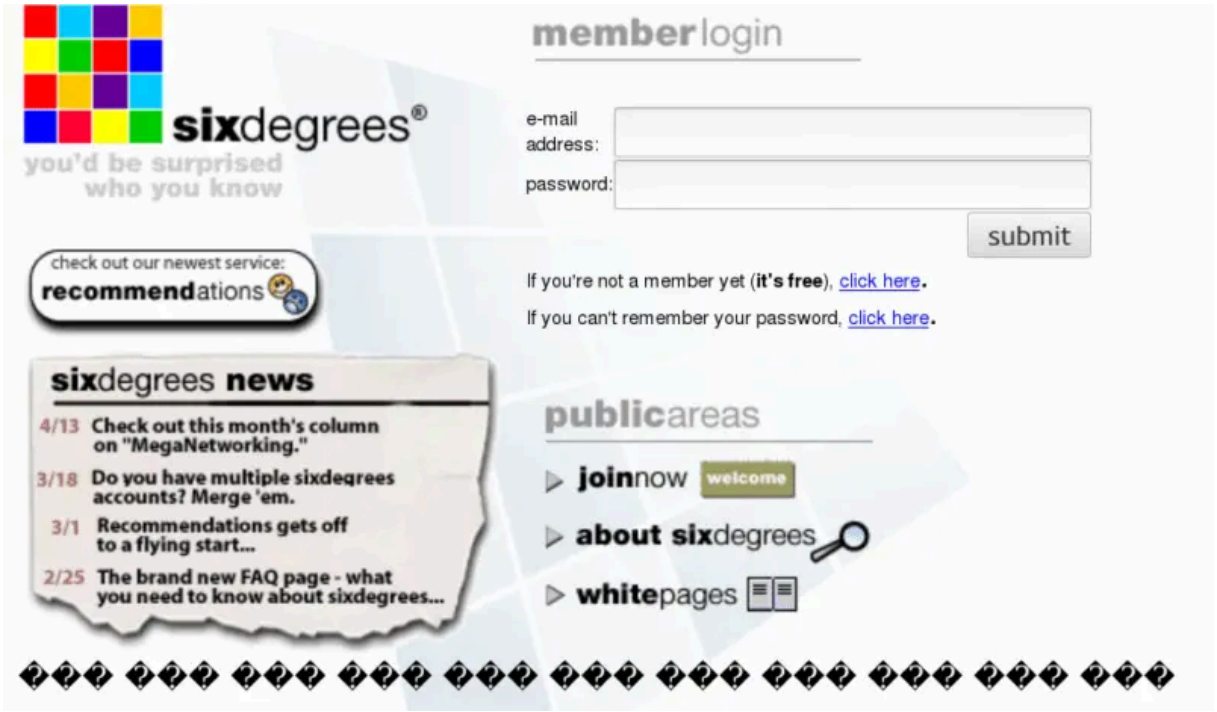

Figura 2.8: Primera red social de la historia (Fuente:Six Degrees | Cómo fue y quién creó la primera red social de internet)

Las redes sociales son capaces de comunicar una información al resto del mundo con la característica de su rápida propagación con todos los usuarios, esto

hace que la gran parte de la población esté siempre actualizada con todo tipo de noticias.

Esto es de gran ayuda pero si esta información no es cierta, o contiene fragmentos falsos que lo que buscan es un descontrol y compartir una información manipulada que hará crear controversia entre las opiniones de diferentes personas, esto es lo que conocemos como “Fake news”.

2.5. Fake news

Como hemos nombrado anteriormente las redes sociales hoy en día al igual que otros medios periodísticos o cualquier entorno de compartición de información entre usuarios, tienen una gran influencia en la información de cada usuario, por ello que utilizado con fines malévolos puede llegar a ser un hecho de gran importancia. Estas fake news o noticias falsas, son informaciones manipuladas o simplemente falsas que se propagan por internet con el fin de crear un clima de desinformación **[8]**, engañar , estafar o incluso manipular a los usuarios con cualquier fin, incluso fines políticos.

Esto no es algo nuevo sino que se ha arrastrado a lo largo de la historia, podríamos decir que uno de sus inicios fue con la invención de la imprenta, lo que condujo a la primera gran farsa periodística como “El gran engaño de la luna” en 1835 **[2]** o incluso en el uso político en la primera guerra mundial, utilizando la propaganda falsa para convencer a la opinión pública junto con los países neutrales en esta guerra.

La aparición de estas fake news no es algo puntual que puede ocurrir en algunos medios de transmisión de información, sino que es algo que está más presente de lo que nosotros creemos o percibimos .

Figura 2.9: Frecuencia de fake news (Fuente:¿Hay demasiadas noticias falsas?)

Según estudios de la OCU **[6]** que han investigado el porcentaje de las noticias de las que lee cada usuario pertenecen a informaciones falsas (ver Figura 2.9). Este estudio muestra cómo el 56% de las personas detectan estas fake news con alta frecuencia. Si a este porcentaje de personas que son capaces de afirmar esto, les añadimos aquellas noticias que el usuario de manera no intencionada adquiere su información, la frecuencia de aparición de estas noticias falsas es demasiado alta.

2.5.1. Twitter y las fake news

Las redes sociales han crecido mucho desde sus inicios, por ello que sus capacidades han aumentado gracias a diversas actualizaciones que han ido mejorando sus características, por lo que al igual que en otros aspectos, la cantidad de información que se transmite a través de estas aplicaciones ha crecido exponencialmente.

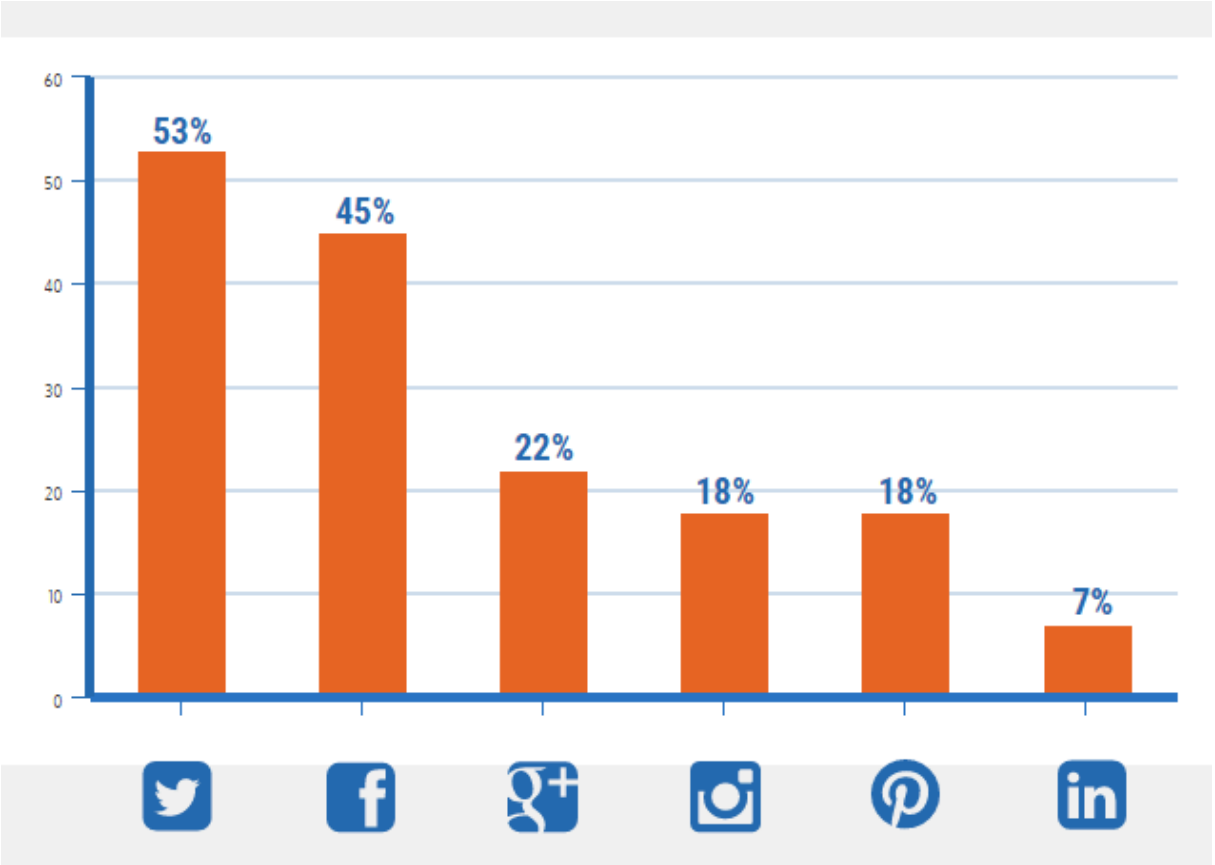

Figura 2.10: Demanda de las redes sociales más importantes (Fuente:Las redes sociales más utilizadas por las empresas)

Todas las redes sociales desde el inicio de las mismas han tenido un gran recibimiento por parte de los usuarios teniendo en cuenta todos los sectores desde usuarios privados hasta empresas u organizaciones. Pero algunas de ellas han tenido una mayor demanda por parte de los usuarios (ver Figura 2.10) como bien se muestra en esta imagen es el caso de twitter.

La red social twitter es una de las más antiguas y con mayor afluencia a lo largo de la historia, es por ello que muchas de las investigaciones de hoy en día se decantan por ella para su investigación .

A pesar de su larga trayectoria twitter no ha dejado de ser usado por millones de usuarios en todo el mundo (ver Figura 2.11) y como podemos observar en diferentes estudios como el realizado por RTVE podemos ver como las noticias en falsas en twitter se propagan hasta un 70% más rápido que las verídicas lo cual aumenta el riesgo debido al contenido de la aplicación **[24]**.

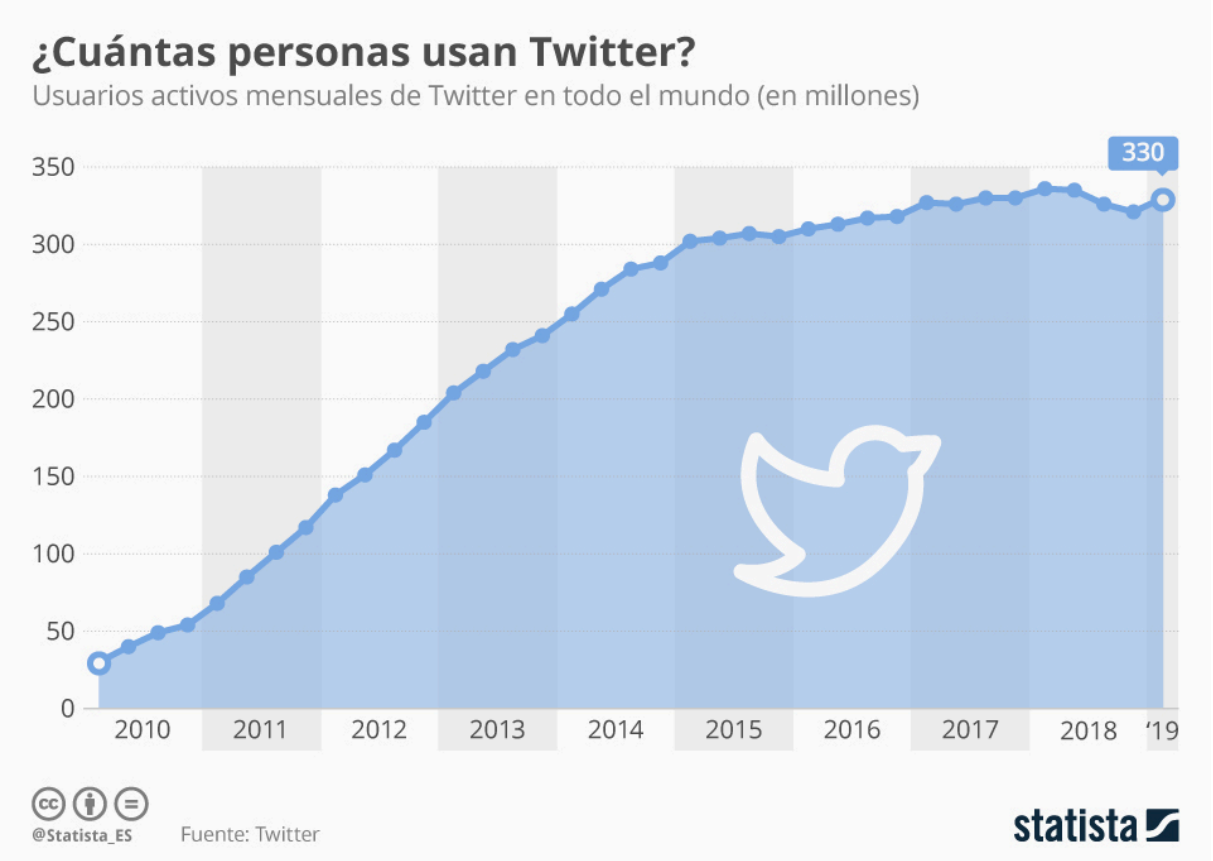

Figura 2.11: Usuarios de twitter en los últimos años (Fuente:Gráfico: Usuarios de twitter | Statista)

Twitter a parte de su temprana fundación destaca por varios factores determinantes, los cuales no solo incrementan la facilidad y el número de usuarios dentro de la misma si no que también favorece a la aparición de fake news dentro la misma, entre estos factores destacamos.

- **Opiniones**: Twitter es una red social que permite publicar información en diferentes formatos multimedia, por lo que podemos publicar textos cortos publicando nuestra opinión sobre alguna cuestión o incluso respondiendo a otra opinión propia de otro usuario, con lo que tendremos información bastante subjetiva centrada en la opinión de un

usuario, por lo que con esta información, distintos usuarios pueden malinterpretar incluso formar a partir de la misma otra publicación con información distorsionada que originará una o varias “fake news”.

- **Mensajes cortos**: Esta red social se basa en la publicación de elementos pequeños y noticias cortas (máximo de 280) caracteres lo que provoca que la información no sea completa, con mensajes simplificados que pueden llevar a engaños, e información falsa o incompleta.
- **Viralidad**: Una de las características distintivas de twitter es la capacidad de realizar retweets, es decir simplemente compartir información que otro usuario ha compartido . Esto hace que una misma información, sea compartida por gran cantidad de usuarios puedan visualizar un contenido que ha creado un usuario, por lo que la información llega a transmitirse entre los usuarios a una gran velocidad.
- **Responsabilidades**: Por último una característica negativa de twitter, el anonimato, hace que la búsqueda de responsabilidades hacia un usuario el cual comete alguna infracción o simplemente se ha detectado que la información que ha publicado no es verdadera o produce algún daño, podría no ser encontrado ni repercutir en ningún aspecto, ya que a diferencia de otras redes sociales, twitter carece de un mecanismo identificación de todos los usuarios, es decir, un usuario puede publicar una información sin la necesidad de exponer ningún tipo de información personal.

Todas estas características pueden haber favorecido a decantarse a diferentes entidades por centrar su estudio en esta red social, como por ejemplo al propio CLEF 2024 **[3]**.

2.6. Definición de estrategias

Después de un breve análisis teórico sobre los apartados más relevantes a los que nos enfrentamos con este proyecto, continuamos con un enfoque a seguir para el desarrollo de nuestro proyecto. Como bien indicaba la competición en la que vamos a participar, debemos de desarrollar un modelo para dos tareas de

clasificación binaria, por lo que sería buena idea investigar, desarrollar y evaluar modelos tanto de aprendizaje automático como de aprendizaje profundo.

Primeramente se hará un breve análisis del conjunto de datos de entrenamiento que le vamos a proporcionar a nuestros modelos, para poder realizar las transformaciones necesarias para obtener los mejores resultados posibles. Una vez trabajado con el conjunto de datos de entrenamiento, pasamos a desarrollar los primeros modelos de aprendizaje automático, en los que usaremos distintos algoritmos de aprendizaje como pueden ser Support Vector Machine o Stochastic Gradient Descent y evaluaremos sus resultados.

Posteriormente comprobados todos los resultados con estos modelos, pasaremos a trabajar con modelos de aprendizaje profundo los cuales generalmente en un alto porcentaje de tareas de PLN obtienen mejores resultados, los llamados “Transformer”, entre ellos destacaremos el uso e investigación de alguno de ellos, nosotros basaremos especialmente nuestro proyecto en las pruebas y desarrollo de modelos derivados del propio BERT, el cual veremos más detenidamente durante el transcurso del trabajo, como por ejemplo RoBERTuito o incluso Facebook/Roberta-base (ver Figura 2.12).

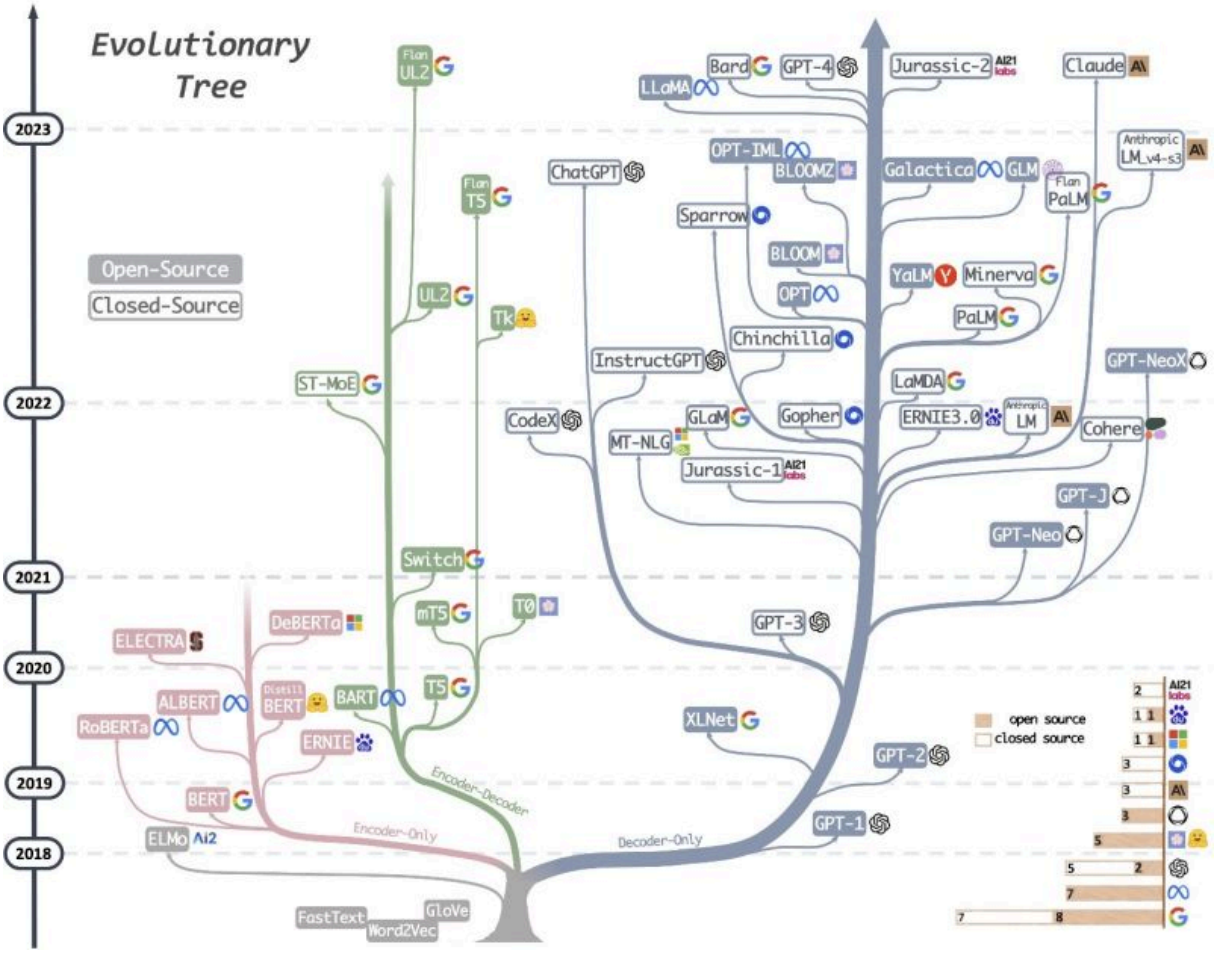

Figura 2.12: Árbol genealógico de transformer (Fuente: The Practical Guides for Large Language Models)

Por último realizaremos un análisis de todos los resultados obtenidos, a partir de los mismos planteamos diferentes hipótesis las cuales podrán realizar una mejora en nuestro modelo, mejorando su eficacia en cada uno de ellos, y finalizar con el modelo o los modelos que consigan resolver los problemas planteados de la mejor manera posible.

2.7. Definición de tareas

Una vez definido el enfoque que va a llevar nuestro proyecto, pasamos a definir las tareas necesarias para llevarlo a cabo y completar el objetivo. Para ello vamos a dividir el proyecto en 3 fases fundamentales y una fase final de conclusión para el devenir del mismo.

En cada una de las fases se irán desarrollando las dos tareas consecutivamente para poder concluir ambas para la fecha estimada. En la primera fase destacaremos el inicio del proyecto con el planteamiento del mismo y el desarrollo de los modelos basados en el aprendizaje automático, seguidamente en la fase 2 pasamos directamente al desarrollo de los modelos óptimos por definición como son los transformers y por último en la fase 3 ponemos en común todos los modelos desarrollados y tratamos de escoger y enfocarnos en la mejora del mejor para cada tarea. Finalmente estableceremos una fase análitica en el que se concluirá con la presentación de nuestros mejores modelos.

En esta fase 1 las tareas a realizar son:

- **Establecimiento de objetivos**: En esta fase hicimos un planteamiento inicial sobre las tareas que nos disponíamos a desarrollar, en ella concretamos la forma en la que íbamos a trabajar y las herramientas que íbamos a utilizar.
- **Solicitud y aceptación**: Para comenzar el desarrollo del proyecto, primeramente solicitamos la aceptación de nuestra participación en el CLEF 2024 **[3]**.
- **Recolección de los datos**: Una vez aceptados dentro de la competición, ya teníamos acceso a todos los recursos necesarios para ello, por lo que accedimos y almacenamos toda la información posible acerca de las dos tareas.
- **Preprocesamiento del texto**: Como bien hemos comentado anteriormente es necesario el preprocesamiento de todos los conjuntos de datos adquiridos en la anterior tarea para poder realizar el entrenamiento de los modelos.
- **Desarrollo de los algoritmos de aprendizaje automático**: Una vez reunido todo lo necesario procedemos con el entrenamiento de los modelos.
- **Almacenamiento de resultados**: Una vez entrenados y desarrollados los modelos con los algoritmos, almacenamos cada uno de los resultados obtenidos.

Una vez concluida la fase 1 pasamos a la fase 2 o fase de aprendizaje profundo :

- **Búsqueda de los modelos de aprendizaje profundo**: Para poder obtener los mejores resultados,dedicamos parte del tiempo en la búsqueda de los mejores modelos para estas tareas.

- **Realización de pruebas**: Una vez encontrado los modelos teóricamente más adecuados realizamos pruebas sencillas para centrarnos en los mejores.
- **Entrenamiento de modelos de aprendizaje profundo**: En última instancia de esta fase implementamos el código y realizamos el entrenamiento de los modelos de aprendizaje profundo seleccionados.
- **Almacenamiento de información y resultados**: Una vez realizado el entrenamiento de los modelos, almacenamos la información relacionada con cada modelo y los resultados obtenidos con cada uno de ellos.

Posteriormente al desarrollo de todos los modelos pasamos a la fase 3 o fase de puesta en común:

- **Análisis de resultados de la fase 1**: Ponemos en común los resultados de todos los algoritmos de aprendizaje automático de ambas tareas y seleccionamos los mejores.
- **Análisis de resultados de la fase 2**: Ponemos en común cada uno de los modelos entrenados de aprendizaje profundo y seleccionamos el mejor de ellos .
- **Selección del mejor modelo**: Por último unimos las dos tareas anteriores y seleccionamos el mejor de todos ellos, para cada una de las tareas.

Fase final de conclusión y análisis de resultados :

- **Definición de posibles mejoras**: Estudiamos las posibles mejoras que se le pueden aplicar a los dos modelos seleccionados.
- **Aplicación de mejoras**: Una vez estudiadas pasamos a la implementación de las mismas, y su posterior evaluación para conservar estas mejoras o volver al modelo anterior.
- **Conclusiones y entrega de modelos**: Una vez realizadas todas las mejoras posibles, analizamos resultados y entregamos el mejor modelo para la competición.

2.8. Distribución de tiempos del proyecto

En este diagrama está reflejada la distribución de tiempos empleada en la consecución final del proyecto, cada una de las tareas expuestas anteriormente están representadas.

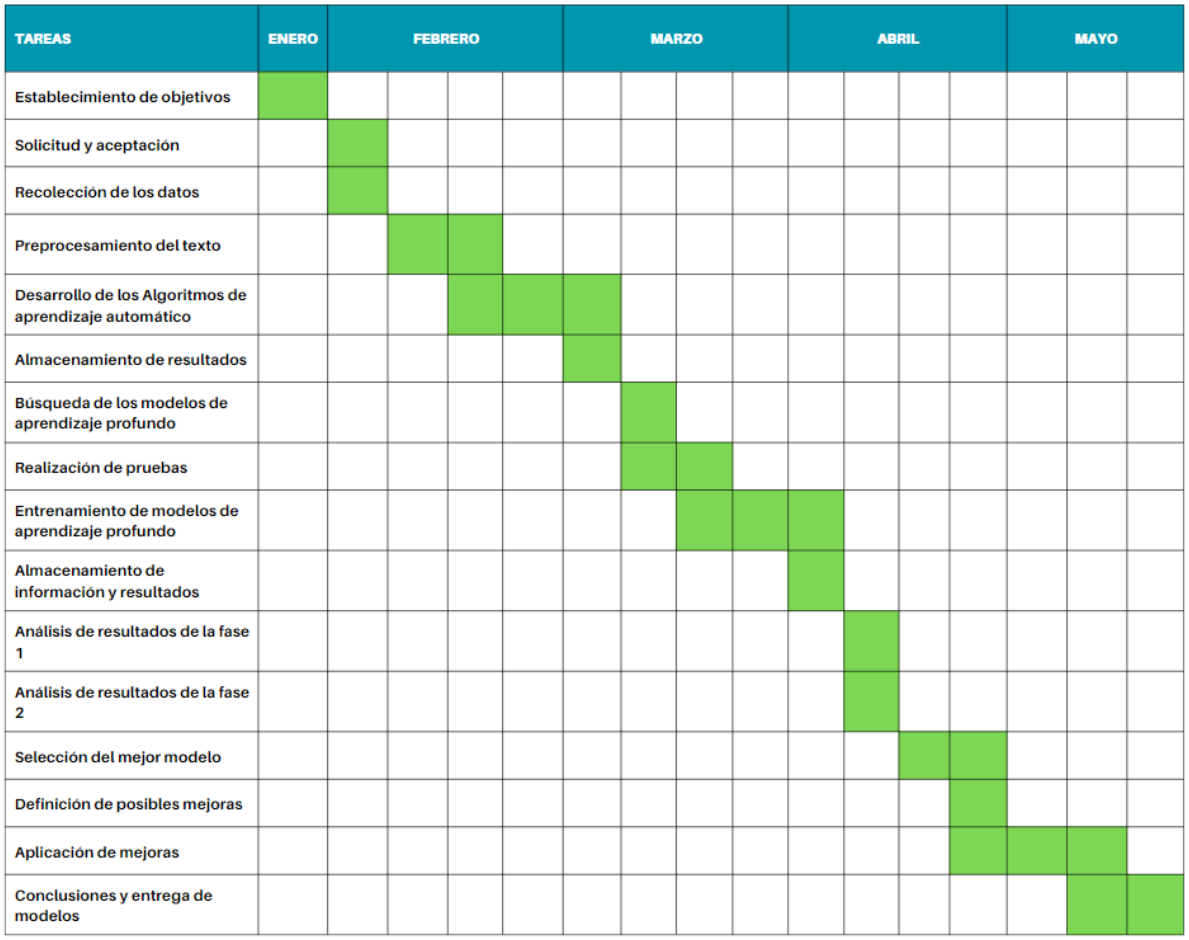

Figura 2.13: Diagrama de tiempos (Fuente:Elaboración propia)

3. Desarrollo

Después de un primer análisis sobre todas las tecnologías y herramientas que vamos a utilizar pasamos a realizar el desarrollo de nuestra investigación la cual seguirá la secuencia mostrada anteriormente. Primeramente realizaremos un tratamiento de los datos y seguidamente pasaremos al desarrollo de nuestros

modelos siguiendo tanto un enfoque con machine learning con algoritmos de aprendizaje automático, como con deep learning o algoritmos de aprendizaje profundo.

3.1. Corpus Utilizados - Tarea 1

Para poder realizar este proyecto orientado hacia la tarea 1, necesitamos de información lingüística, para poder aplicar el conocimiento del lenguaje natural en el proceso de desarrollo de sistemas computacionales. Entre estos recursos podemos encontrar, diccionarios, lexicones o incluso ontologías para marcar unas reglas de compartición de información **[7]**, pero nosotros nos vamos a basar en diferentes corpus que nos ayudarán en el entrenamiento de nuestros modelos de aprendizaje.

Estos corpus contienen distintas noticias que se han obtenido de la red social de twitter y están compuestos por 4 campos, entre las que se encuentran:

- **Id_label**: El cual contiene un identificador único que tendrá asignada cada noticia para poder realizar tratamientos sobre el mismo corpus sin perder ningún tipo de información.
- **Tweet_url**: En este campo se almacena el enlace de la noticia, por el cual podremos verificar si es necesario la publicación real a través de la aplicación, en nuestro trabajo este campo no será de mayor importancia.
- **Tweet_text**: Este contiene todo el contenido de la noticia lo cual serán los datos que trataremos mayormente para dárselos como entrada a nuestros modelos, después de un preprocesamiento de los mismos.
- **Class_label**: Con este campo que será de gran ayuda para el entrenamiento, podremos darle como dato de entrada el valor de la clase a la que pertenece, bien sea No (0) o Yes (1) si se trata de una noticia que requiere la verificación por tener indicios de ser una noticia falsa o no.

En cuanto a características específicas de los corpus utilizados constan de una cantidad de 19948 noticias, en el caso de los tweets que cuenta con 9948 noticias, las cuales gracias a las etiquetas podemos ver un cierto desequilibrio entre las dos clases que se presentan, las etiquetadas con la clase 0 contiene cerca de un 75% de las noticias del corpus, mientras que en la clase contraria solo presenta un total de un 25% de estas (ver Figura 3.1). En el caso de las transcripciones o textos de debate se presenta un aún mayor desequilibrio con casi un 94% de las noticias

clasificadas con la clase 0 (No) y el 6% restante en la clase positiva 1 (Yes). Estas noticias presentan algunas variaciones con respecto al texto simple, como puede ser, variedad de idioma, presencia de emoticonos e incluso presencia de enlaces a otros sitios webs, las cuales serán analizadas posteriormente para el preprocesamiento del texto.

Yes	0,252010
No	0,747989

Figura 3.1: División de tweets Tarea 1 (Fuente:Elaboración propia)

Una vez se transmita como entrada el corpus nombrado tendremos adicionalmente dos corpus, un conjunto de evaluación que consta exactamente de 1033 noticias y otro conjunto de test con 5001 noticias, con los cuales podremos evaluar la eficiencia durante el entrenamiento de nuestro modelo y a continuación establecer unos resultados finales sobre el modelo.

3.2. Corpus Utilizados - Tarea 2

En el segundo caso, es decir para la tarea 2 contamos con la misma cantidad de corpus necesarios para el desarrollo y evaluación de la tarea, es decir, contamos con 3 corpus diferentes, el conjunto de entrenamiento con un total de 832 oraciones de las cuales al igual que en la tarea anterior este presentaba un gran desequilibrio incluso en esta tarea adicionalmente, casi la totalidad de las secuencias de textos pertenecían a la clase 0, el fichero de evaluación con 221 y por último el fichero de test o evaluación final con 244 oraciones. A diferencia del corpus anterior este presenta noticias pero no tomados de la red social de twitter, si no de otras fuentes de información.

Yes	0,061500
No	0,938500

Figura 3.2: División de tweets Tarea 2 (Fuente: Elaboración propia)

Al igual que el corpus anterior se presentan diferentes columnas que contienen la información necesaria para el desarrollo de nuestros modelos:

- **Id**: Esta contiene un identificador único para diferenciar entre todo el conjunto de noticias, estos están compuestos por más de 20 carácteres distinguiendo entre letras y números.
- **Sentence**: Esta columna está formada por el contenido de cada una de las oraciones que será analizado, tratado y enviado al modelo para cada una de las fases de entrenamiento, evaluación y testeo.
- **Class_label**: En este campo tenemos definidas las etiquetas que presentan a que clase del problema pertenece bien sea OBJ (0) o SUBJ (1) si se trata de una noticia que contiene indicios de subjetividad o no, cada oración para ir ayudando en el entrenamiento del modelo y poder obtener los resultados del modelo.
- **Solved_conflict**: Por último tenemos una clase adicional, simplemente con la clasificación de la oración según sea False (Objetiva) o True (Subjetiva), siendo una relación coherente con la columna anterior de la clasificación de etiquetas, es decir presentando los mismos resultados que la clase anterior pero con distintas etiquetas.

3.3. Desarrollo de modelos de aprendizaje automático

Como bien está referenciado anteriormente vamos a desarrollar como herramientas para ello, modelos de aprendizaje automático que sean capaces de clasificar eficientemente cada una de las noticias tanto si necesitan verificación por la posible falsedad de la noticia o si esta misma presenta síntomas de subjetividad. Primeramente vamos a desarrollar y evaluar herramientas a partir de algoritmos de aprendizaje automático. Estos algoritmos se conocen como los algoritmos clásicos,

ya que estos fueron de los primeros y más utilizados en los inicios del machine learning.

Para poder realizar el proceso de entrenamiento de nuestro modelo debemos de realizar una serie de pasos para tratar el conjunto de datos de entrenamiento y así favorecer el aprendizaje de nuestro modelo (ver Figura 3.3).

Figura 3.3: Proceso de preprocesamiento (Fuente: Cómo crear nubes de palabras a partir de un texto utilizando técnicas de PLN)

3.3.1. Tokenización

Cuando hablamos de la tokenización, nos referimos al proceso previo al preprocesamiento del texto que es utilizado en tareas del procesamiento del lenguaje natural, en el cual pasamos a dividir el contenido de cada secuencia de texto o parte del conjunto de datos que posteriormente será tratado por nuestro algoritmo o modelo de aprendizaje, en diferentes unidades mínimas lingüísticas, en las cuales en la mayoría de los casos se dividen en palabras.

Al igual que otros muchos procesos que se realizan en tareas de PLN, este proceso ya tiene diferentes librerías implementadas en el lenguaje python, que facilitan la tokenización. Entre ellos podemos destacar algunos como “Treebank Word tokenizer” que se trata del tokenizador por defecto o “Word Punct Tokenizer” el cual separa los signos de puntuación pero los incluye en tokens separados.

En el caso de nuestro proyecto nos basamos en el uso del tokenizador utilizaremos el Treebank Word tokenizer **[15]**, evitaremos así distintos problemas que

pueden derivar de no usar el tokenizador por defecto, y a partir de los tokens generados, enviamos cada uno de ellos al preprocesamiento correspondiente.

Este apartado es de los primeros procesos que realizamos en nuestro proyecto, por lo que es importante que este cumpla con sus requisitos, para evitar futuros problemas.

3.3.2. Preprocesamiento del texto - Tarea 1

Estos modelos de aprendizaje automático requieren un tratamiento del texto especial, para poder recoger y aprender la mayor información posible acerca de los datos de entrenamiento. Cuando hablamos del preprocesamiento del texto, nos referimos a la tarea de realizar una serie de modificaciones al texto, para construir una entrada al modelo más limpia y simplificada. Por ello, una vez realizada la tokenización del conjunto de datos correspondiente, es decir cada una de sus secuencias de texto, pasamos a analizar los posibles inconvenientes que puede contener estos datos según la tarea que se desea realizar.

En cuanto a la tarea 1, como los corpus que vamos a utilizar se trata de diferentes noticias o tweets, de la red social twitter, sabemos que este requiere un preprocesamiento especial debido a su composición de caracteres y a otros factores dependientes de los corpus utilizados, entre estas características encontramos:

- **Contenido del corpus**: En esta característica hablamos de los distintos carácteres que puede presentar cada una de las noticias, entre ellos destacamos la presencia de emoticonos, los cuales nuestros algoritmos de aprendizaje automático no tienen la capacidad de analizarlos y pueden presentar irregularidades en su aprendizaje, por ello que debemos de traducirlos a texto para poder tratarlos como tales y que tengan un valor dentro de la oración, para ello utilizamos una librería ya implementada “Demojize” que nos facilitará la traducción de estas. También encontramos distintos carácteres que no son alfanuméricos los cuales no aportan valor. Como podemos ver en la imagen (ver Figura 3.4) encontramos la traducción que realiza esta librería, desde el propio emoticono a su definición textual. Esta librería tiene su propio traductor de idiomas por lo que directamente era capaz de determinar la definición del sticker en el idioma requerido por el usuario, en este caso nosotros.

```
# import emoji module
import emoji


print(emoji.demojize("😀"))
print(emoji.demojize("😜"))
print(emoji.demojize("🤐"))
```

```
:grinning_face_with_big_eyes:
:winking_face_with_tongue:
:zipper-mouth_face:
```

Figura 3.4:Ejemplo de funcionamiento de la librería demojize (Fuente: https://www.geeksforgeeks.org/python-program-to-print-emojis/)

- **Idioma de los tweets**: También hemos encontrado la presencia de un conjunto notable de diferentes noticias las cuales se encuentran en otro idioma diferente al castellano, en catalán, por lo que el modelo las trataría de una manera especial al resto.
- **Enlaces en tweets**: Cada noticia presenta su enlace procedente de la aplicación de twitter, excepto las transcripciones. Pero aparte de estos enlaces, los tweets dentro de su contenido contienen enlaces los cuales llevan a otro sitio web relacionado con la noticia la cual lleva asociada, pero esto no nos aporta ningún valor al entrenamiento por lo que todos los enlaces serán transformados a la palabra “URL” para establecer una forma común acerca de su tratamiento.
- **Nombres o usuarios**: Al igual que encontramos distintos enlaces, también encontramos distintas referencias a usuarios o personas, las cuales tampoco aportan valor al entrenamiento del modelo, por ello seguimos el mismo proceso, transformamos cada uno de los nombres a una forma común, en este caso “USER”.
- Una vez realizado estos procedimientos específicos para los tweets pasamos a realizar una serie de transformaciones comunes a muchos problemas dentro del procesamiento del lenguaje natural, entre ellos la eliminación de palabras vacías y stemming.

Para observar adecuadamente un ejemplo de preprocesamiento de un tweet (ver Figura 3.5), donde podemos observar cómo se traducen los emoticonos gracias a la librería además se complementa en este caso con la reducción de palabras y eliminación de caracteres que no sean alfanumérico, cabe destacar que nuestro procesamiento es ligeramente distinto a este ejemplo como bien se ha establecido anteriormente.

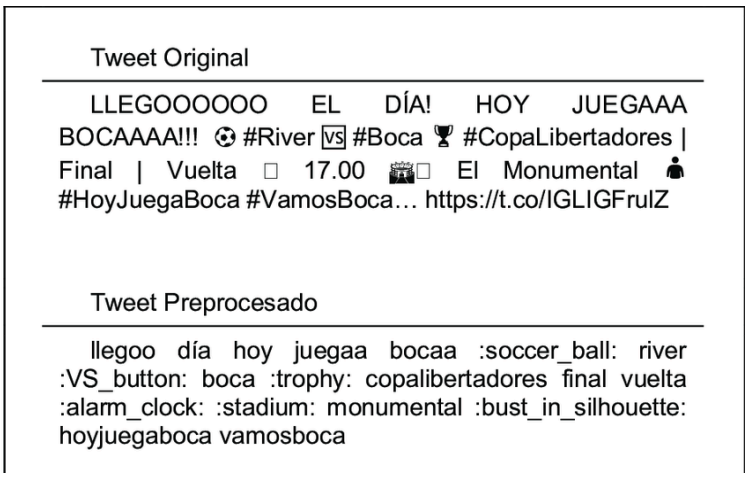

Figura 3.5:Ejemplo de preprocesamiento de un tweet
(Fuente:https://www.researchgate.net/figure/Figura-2-Ejemplo-preprocesamiento-de-tweet_fig1_340415256)

3.3.3. Preprocesamiento del texto - Tarea 2

En el caso de la tarea 2, el procesamiento del texto es mucho más sencillo que en la tarea anterior, ya que el contenido de los textos, sólo contiene carácteres alfanuméricos, sin emoticonos, ni otros carácteres que puedan perjudicar el entrenamiento del modelo. Esto facilita mucho la realización de esta tarea y nos ayuda a centrar nuestra atención directamente a los procedimientos generales como la eliminación de palabras vacías y la realización de stemming.

3.3.4. Palabras vacías o “Stopwords”

En este proceso nos basamos en la eliminación de las palabras vacías de un texto, es decir una palabra vacía es aquella que no aporta ningún significado potencialmente significativo en el análisis de un texto. Entre estas palabras podemos destacar la aparición de artículos, preposiciones y pronombres. Este conjunto de palabras son conocidas mayormente por “Stopwords”, y para identificarlas podemos

usar una librería de python la cual contiene todas las palabras vacías de cada idioma y simplemente debemos de analizar el texto buscando en cada token estas palabras y eliminarlas, lo cual reducirá entre un 30% y un 40% la cantidad de token a analizar por nuestro modelo, por lo que aumentaremos su rendimiento.

3.3.5. Stemming vs Lematización

Otro proceso muy común dentro del procesamiento del lenguaje natural son los procedimientos de lematización o stemming los cuales tienen un objetivo similar transformar cada palabra a un formato común (ver Figura 3.6). En cuanto la lematización se trata de reducir las palabras o token de la oración a su lema, mientras tanto el proceso de stemming el cual se trata de reducir cada token a su raíz o stem. Nosotros para ambas tareas nos hemos decantado por el proceso de stemming ya que es capaz de realizar un enfoque más general y reducir aún más el tamaño del vocabulario utilizado.

Word	Stemming	Lemmatization
information	inform	information
informative	inform	informative
computers	comput	computer
feet	feet	foot

Figura 3.6:Ejemplo de stemming y lematización
(Fuente:https://seonorth.ca/es/nlp/stemming-and-lemmatization/)

3.3.6. Vectorización

La vectorización es un proceso que se realiza antes de enviar los datos ya preprocesados a nuestro modelo para realizar el aprendizaje. Este trata de realizar una transformación de los datos de la secuencia de texto en vectores numéricos, entre los algoritmos más destacados encontramos incrustación de palabras o TF-IDF el cual este último es el que nosotros utilizaremos para nuestras tareas.

La vectorización tiene un función importante dentro del preprocesamiento del texto ya que puede ayudar a capturar las características de cada token o cada

palabra transformada a vector, entre ellas las características semánticas y sintácticas.

TF-IDF:

Este tipo de proceso de vectorización destaca por su funcionamiento en el cual aumenta proporcionalmente el número de veces que la palabra aparece en el documento y a la vez consigue contrarrestar este valor con el número de documentos en los que aparece. Así obtenemos que palabras destacadas que se repiten muchas veces pero en pocos documentos tendrán mayor valor que otras palabras que se presenten en todos los documentos.

Word	TF		IDF	TFIDF	
	Doc 1	Doc 2		Doc 1	Doc 2
Text	1/4	1/6	log (2/2) = 0	0	0
Processing	1/4	1/6	log (2/2) =0	0	0
Is	1/4	1/6	log (2/2) =0	0	0
Necessary	1/4	1/6	log (2/2) =0	0	0
And	0/4	1/6	log (2/1) =0.3	0	0.05
Important	0/4	1/6	log (2/1) =0.3	0	0.05

Figura 3.7:Ejemplo de vectorización TF-IDF (Fuente: FB-vectorización-bolsa-palabras-vs-vectorización-tfidf)

3.4. Algoritmos de aprendizaje automático

Estos algoritmos como su propio nombre indica, son aquellos que son capaces de realizar un aprendizaje, es decir son capaces de mejorar automáticamente, estos van obteniendo un mayor aprendizaje gracias a la experiencia. Estos algoritmos exploran, analizan y aprenden a partir de un conjunto de datos que se le transmiten como entrada, y son capaces de aprender patrones específicos y son capaces de proporcionar predicciones a partir de su entrenamiento. Estos son algoritmos de machine learning los cuales a lo largo de los años han ido evolucionando y surgiendo nuevos de ellos como SVM o SGD los cuales han dado bastantes expectativas a la inteligencia artificial por sus buenos resultados en diferentes campos.

3.4.1. SVM

El algoritmo “Support vector machine” o conocido como el algoritmo SVM, es un algoritmo de aprendizaje supervisado, es decir su entrenamiento contiene las etiquetas correspondientes de las distintas partes del conjunto de datos, este se utiliza en una gran variedad de problemas basados en la clasificación y regresión orientadas a tareas del procesamiento del lenguaje natural.

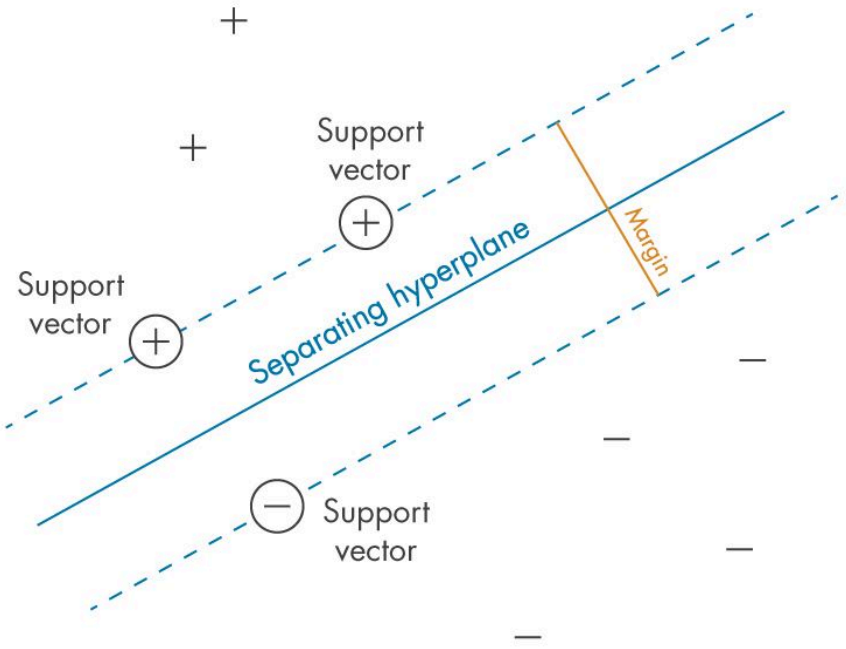

Figura 3.8:Procesamiento del algoritmo SVM (Fuente:Support Vector Machine (SVM) - MATLAB & Simulink)

El objetivo del SVM trata de encontrar un hiperplano que sea capaz de dividir y separar de la forma más eficiente posible dos clases predefinidas y diferentes de datos. Para encontrar la la mejor forma de separarlos se busca encontrar el hiperplano con el margen más amplio entre las dos clases. Este margen podemos definirlo como la anchura máxima que el algoritmo permite a la región paralela al hiperplano que no tiene puntos de datos interiores **[17]**.

Como ventajas de este algoritmo podemos encontrar la facilidad para obtener buenos resultados en problemas con espacios de alta dimensionalidad y junto con ello son capaces de realizar una gestión muy eficiente de la memoria al utilizar únicamente un subconjunto de puntos en la función de decisión. Por el contrario encontramos algunas desventajas como la posibilidad de sobreentrenamiento de

nuestro modelo si el número de características es mayor que el número de muestras, o la dependencia de su eficacia al tamaño del datasets, ya que si este es muy grande puede resultar muy lento.

3.4.2. SGD

El algoritmo de descenso de gradiente estocástico es un algoritmo de optimización, es decir es un modelo que busca encontrar la mejor opción. La función principal es minimizar la función de costo al ajustar los parámetros del modelo en cada iteración, en función del conjunto de datos de entrenamiento. Para calcular la actualización de parámetros este se basa en pequeñas muestras aleatorias de los datos de entrenamiento, para actualizar los parámetros iterativamente. Con esto conseguimos acercarnos más rápidamente hacia un óptimo local como se observa en la imagen (ver Figura 3.9).

Este algoritmo se utiliza para problemas de aprendizaje automático a gran escala y dispersos, es decir es capaz de tratar problemas con gran variedad de ejemplos de entrenamiento y gran cantidad de características distintas entre ellas **[18]**.

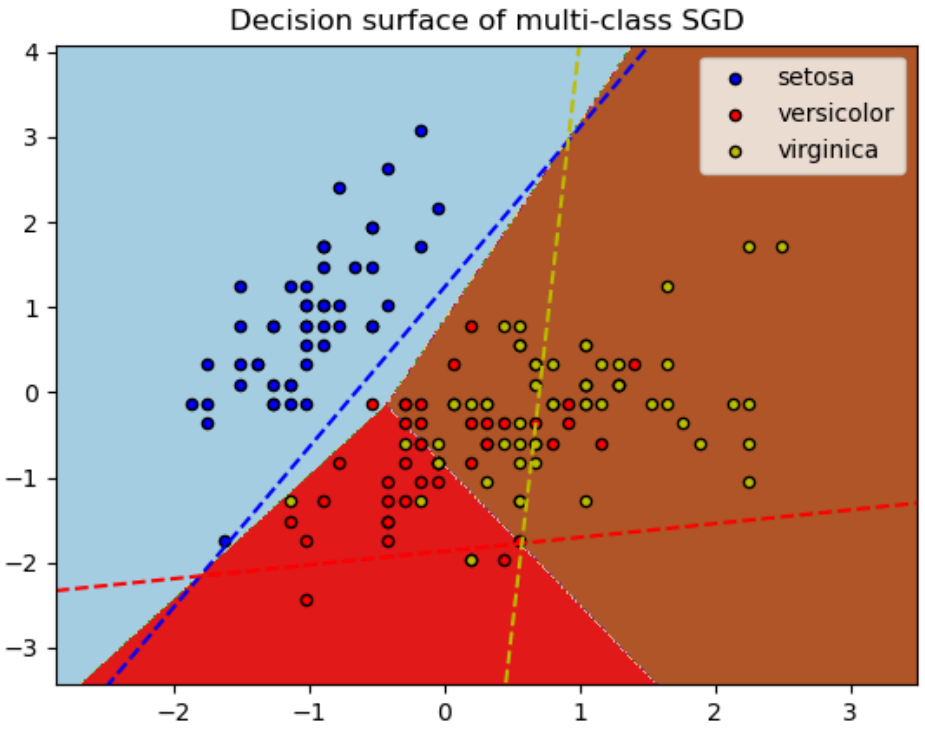

Figura 3.9: Procesamiento del algoritmo SGD(Fuente:1.5. Descenso de Gradiente Estocástico — documentación de scikit-learn)

Este algoritmo no se familiariza con ningún tipo de modelo de aprendizaje automático, ya que destaca por ser básicamente una técnica de optimización, lo cual lo reduce a una forma específica de entrenar a un modelo eficientemente. Entre las ventajas destacables de este algoritmo encontramos su alta eficiencia y su facilidad de implementación ya que es una técnica que permite una gran cantidad de oportunidades para el ajuste del código. Sin embargo como es normal también presenta algunas desventajas como la necesidad de establecer una serie de hiper parámetros como el parámetro de regularización y el número de iteraciones, o la sensibilidad del mismo al escalamiento de características, es decir, el ajuste de los datos de entrada de manera que sus valores se encuentren dentro de un rango comparable, en una escala similar.

3.4.3. Random Forest

El algoritmo random forest es un algoritmo de aprendizaje supervisado, que es usado mayormente al igual que SVM para tareas de clasificación y regresión.

Como su propio nombre indica el algoritmo de random forest, se basa en árboles, en este caso en árboles de decisión.

Para completar cada nodo de decisión del árbol se van respondiendo diferentes cuestiones para llegar a una conclusión final si pertenece o no pertenece a una de las clases predefinidas, es decir, un árbol de decisiones es una herramienta de apoyo a las decisiones, y en cada una de sus ramas se va mostrando cuáles son las posibles consecuencias de la elección de cada una de sus opciones (ver Figura 3.10).

En cuanto al formato del algoritmo se establecen inicialmente como parámetro el número de árboles que van a ser utilizados durante el entrenamiento, en este caso existe una relación la cual indica que a mayor número de árboles, el resultado del algoritmo tendrá una mayor precisión..

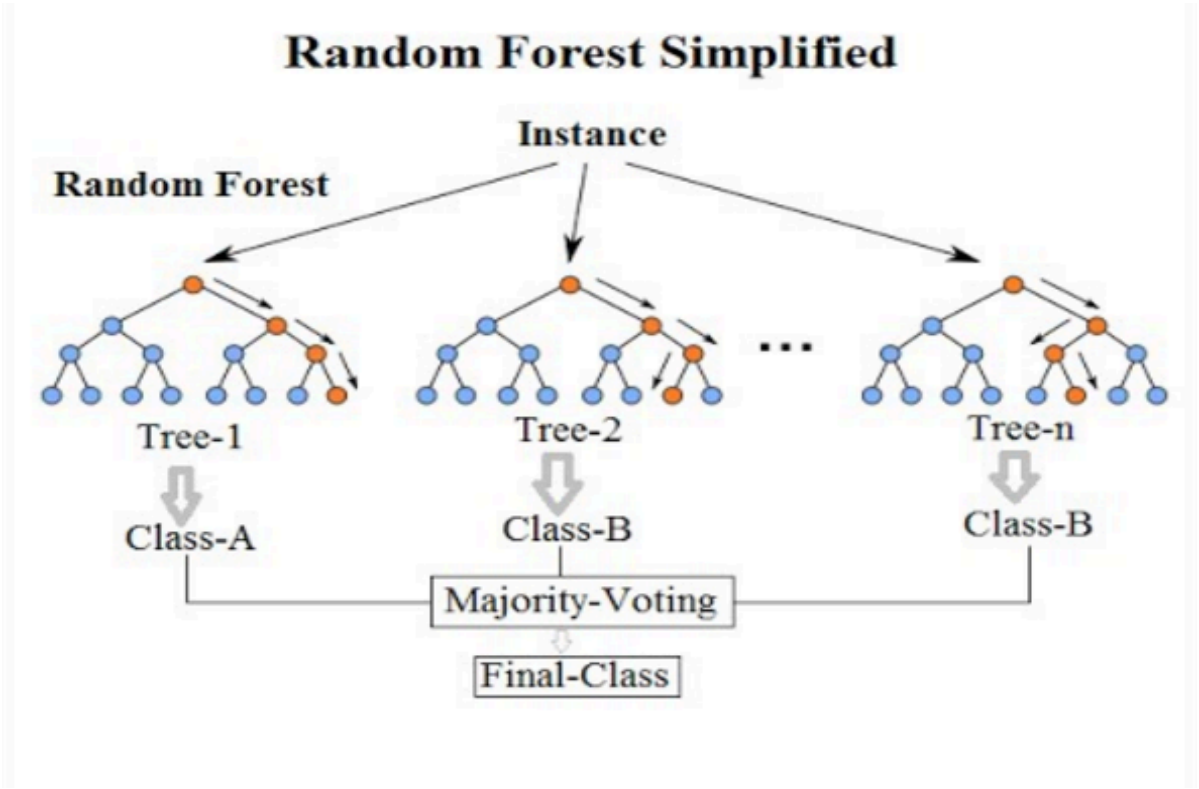

Figura 3.10:Procesamiento del algoritmo Random Forest (Fuente:Random forest - Bosques Aleatorios)

Todos estos árboles (ver Figura 3.8) se centran en un subconjunto de los datos de entrada del algoritmo y da un resultado acerca de sus datos, todos los árboles combinan sus resultados y se da una decisión final con la clase que mayor número de árboles han dado como solución **[19]**.

Gracias a este algoritmo podemos observar una ventaja predominante con respecto al resto de ellos como la eliminación del sobreajuste en la mayoría de los casos, si el número de árboles es el adecuado.

3.4.4. Logistic Regression

El algoritmo de regresión logística, es un modelo el cual está orientado de una manera más empírica, es decir se centra en una forma matemática para encontrar relaciones entre dos factores de datos. Este algoritmo está más orientado a tareas de clasificación a diferencia de la regresión lineal.

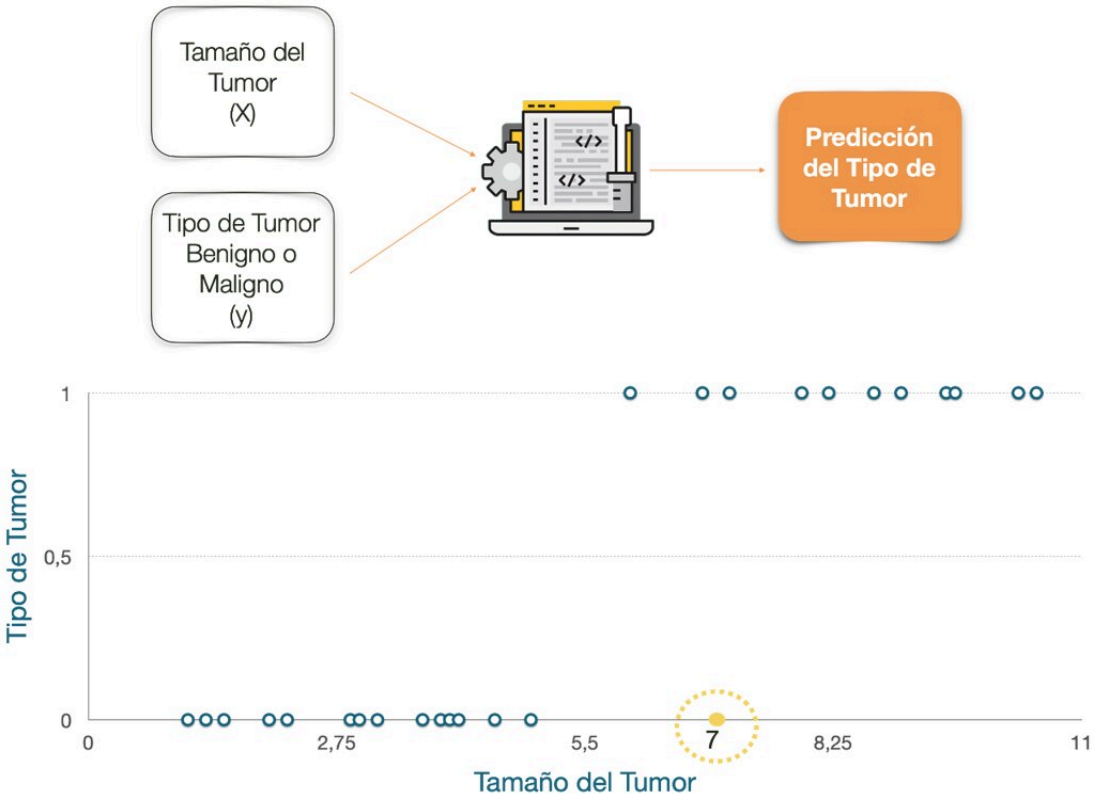

Figura 3.11:Procesamiento del algoritmo Logistic Regression (Fuente:Regresión Logística - Teoría - Aprende IA)

Este algoritmo es uno de los algoritmos que más se utilizan en tareas de clasificación y a la vez es uno de los algoritmos más simples y fáciles de implementar. Para su funcionamiento interno (ver Figura 3.11), lo que realiza es estimar una relación entre la variable dependiente con el resto de variables independientes del problema, es decir, estima la probabilidad de una respuesta binaria basada en una o varias variables independientes **[20]**.

La razón de porque este algoritmo es bastante utilizado es por que a pesar de que los algoritmos de aprendizaje profundo tengan mejores resultados, este es también muy eficiente y no requiere demasiados recursos computacionales, lo que lo hace ideal para problemas en los que podamos sacrificar un poco la eficiencia y obtener mayor facilidad y un tiempo de respuesta menor que con otros modelos, como por ejemplo detección de spam, predicción de compras de productos o incluso en predicción de enfermedades como la diabetes.

3.5. Desarrollo de modelos de aprendizaje Profundo

Una vez realizados los estudios pertinentes sobre los diferentes algoritmos de aprendizaje automático, pasamos a realizar pruebas con estos algoritmos de aprendizaje profundo que generalmente superan en su mayoría a todos o casi todos los algoritmos anteriores.

En cuanto a los modelos utilizados destacamos que todos ellos han sido tomados desde la web de hugging face ya que gracias a su facilidad de implementación podemos probar gran cantidad de modelos y analizar cada uno de ellos.

Los modelos de aprendizaje profundo como su propio nombre indica se basan en la técnica nombrada anteriormente “Deep Learning”. Estos modelos se basan en arquitecturas de red neuronal de varias capas, en las que en cada una de ellas se encuentran distintas unidades de procesamiento que se encargaran de evaluar y ayudar a tomar decisiones dentro del proceso como podemos observar en la imagen (ver Figura 3.12).

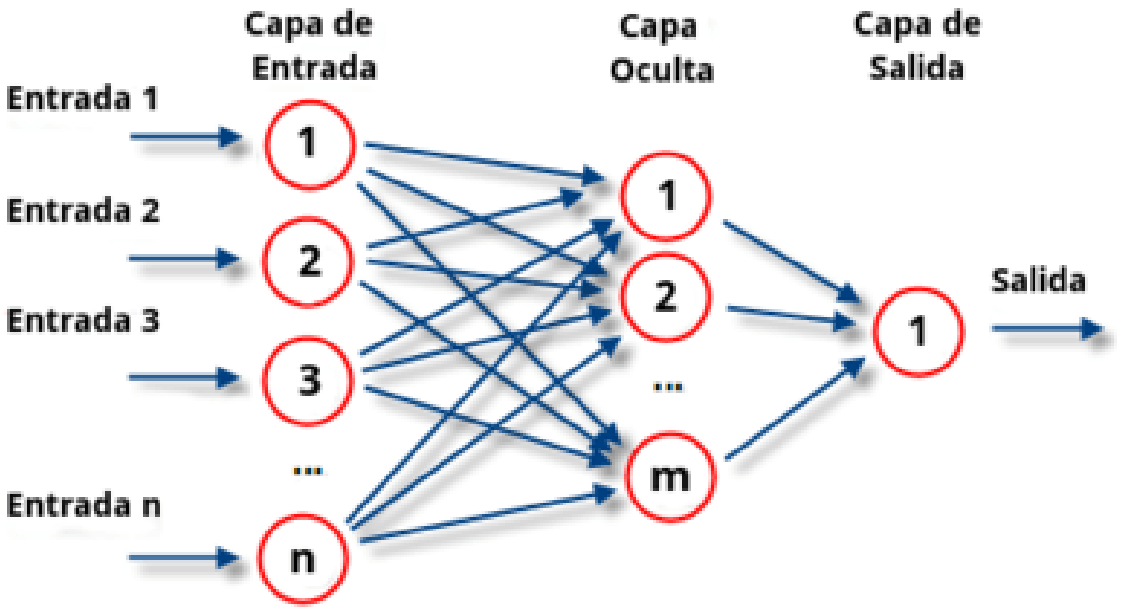

Figura 3.12: Funcionamiento de la red neuronal (Fuente: ¿Qué es el Deep Learning? | SmartPanel)

En cuanto a los algoritmos que nosotros vamos a basar, proceden de un mismo modelo inicial, al cual como hemos explicado anteriormente le suceden diferentes mejoras u optimizaciones que le han hecho derivar en otros modelos más especializados y algunos de ellos con mejores resultados que el mismo.

3.5.1. BERT

BERT es uno de los modelos más avanzados a día de hoy del procesamiento del lenguaje natural, un modelo lingüístico basado en redes neuronales que ha sido desarrollado por Jacob Devlin junto a la empresa de nivel mundial, Google en 2018.

La principal característica innovadora de este modelo es la aplicación de un entrenamiento bidireccional del transformer, esto era diferencial debido a los anteriores modelos que procesaban la secuencia de texto de izquierda a derecha. Los resultados obtenidos después de la evaluación del modelo determinaron que al realizar el enfoque bidireccional se obtiene entrenamiento más profundo y una mayor comprensión del contexto. A diferencia de otros modelos que leen la entrada de datos secuencialmente, este al recibir la entrada la lee completamente, es decir todas las palabras a la vez (ver Figura 3.13) .

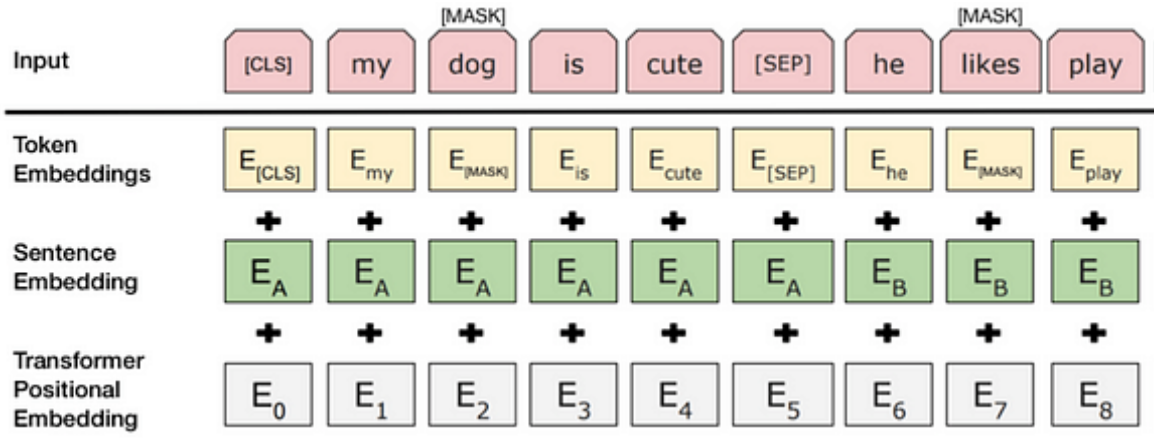

Figura 3.13: Funcionamiento de entrada del modelo BERT (Fuente: Todo sobre BERT, el modelo lingüístico más avanzado para NLP a día de hoy)

BERT utiliza el procesamiento de un transformer que como sabemos utiliza el mecanismo de atención el cual aprende las relaciones contextuales entre las distintas palabras o tokens de una secuencia de texto (ver Figura 3.14).

Antes de introducir esta secuencia de texto de entrada de las palabras, el 15% de estas se sustituyen por una máscara, y el modelo trata de predecir el valor original de estas, basándose en el contexto en el que se encuentran estas, es decir el conjunto restante de la secuencia de texto formada por las palabras no enmascaradas **[16]**.

Este modelo lo utilizaremos para hacer las pruebas oportunas en nuestro desarrollo del modelo para la tarea 2 en inglés.

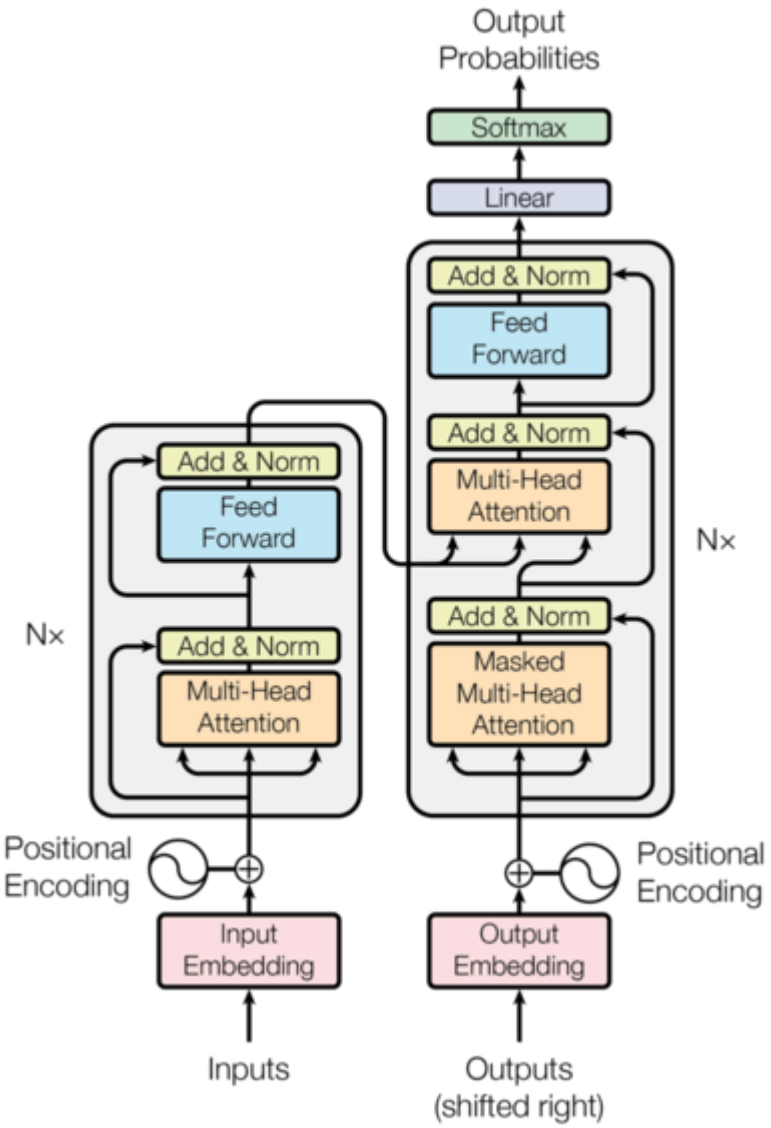

Figura 3.14: Funcionamiento por etapas del modelo BERT (Fuente: ¿Cómo funcionan los Transformers? en Español | Aprende Machine Learning)

En cuanto a características más profundas del modelo BERT podemos destacar su composición por 12 capas diferentes y 12 mecanismos de atención, con más de 100 millones de parámetros.

3.5.2. RoBERTa base BNE

Pero este modelo presenta un leve problema para nuestra tarea a resolver, y es que este modelo ha sido desarrollado para el idioma del inglés por lo que presenta malos resultados para conjuntos de datos en español. Gracias a un grupo de investigadores del Barcelona Supercomputing Center (BSC) se ha creado una nueva versión de este modelo preparada para este idioma cuyo nombre es "PlanTL-GOB-ES/roberta-base-bne", el cual utilizaremos para la primera tarea **[28]**.

Este modelo no presenta diferencia alguna en ninguna de sus características restantes internas ni externas, simplemente ha sido entrenado con un corpus en español, por lo que está orientado a este tipo de tareas.

3.5.3. Facebook/AI roBERTa base

Este, es un modelo desarrollado para el tratamiento de datos en el idioma inglés, es decir ha sido entrenado con un gran corpus basado en este idioma y busca un objetivo principal basado en el modelado del lenguaje enmascarado, es decir al igual que hace su predecesor BERT, toma la oración que se le transmite como entrada y escoge aleatoriamente un 15% del total de las palabras que la forman y las enmascara, posteriormente vuelve a ejecutar toda la oración y trata de predecir las palabras enmascaradas. De esta manera el modelo va aprendiendo la representación interna de las oraciones para que posteriormente el modelo aprenda a extraer características útiles de las mismas.

Este modelo ha sido pre entrenado de forma auto supervisada es decir entrenado solo con textos sin procesar y sin ser etiquetados por ningún humano.

3.5.4. RoBERTuito Sentimental Analysis

Este modelo encontrado en hugging face trata de un entrenamiento realizado con un corpus formado por un conjunto de más de 5000 tweets en el idioma español, por lo que resulta bastante adecuado para nuestra tarea 1 de nuestro proyecto. Este modelo destaca por la capacidad de realizar un análisis de sentimiento, es decir,

puede establecer si una oración dada contiene algún matiz sentimental, para ello realiza una clasificación multiclase con las etiquetas POS,NEG,NEU, las cuales referencian a las etiquetas positivo (POS) para aquellas que sí presentan características relacionadas con sentimientos, negativo (NEG) cuando no presentan y por último NEU(Neutro) es decir que presenta un contenido el cual podría representarse con cualquiera de las dos etiquetas anteriores, es decir no presenta los suficientes valores acerca de los sentimientos para poder clasificar esa secuencia de texto.

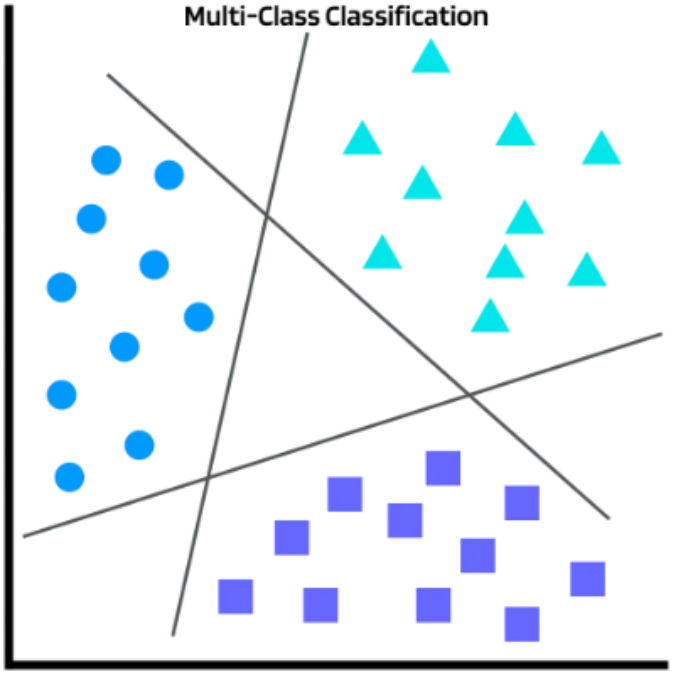

Figura 3.15: Clasificación multiclase (Fuente: 4 tipos de tareas de clasificación en el Machine learning - eLearning Actual)

Multiclase:

Como podemos observar en la imagen (ver Figura 3.15) la clasificación multiclase la cual trata de establecer una distinción entre al menos dos clases, y asignarle a cada una de las secuencias que debemos de clasificar una de las clases. Cuando hablamos del término multiclase generalmente se trata de más de 2 clases para diferenciarse de la clasificación original. Entre los algoritmos que pueden utilizar esta clasificación destacamos k-vecinos más próximos, Ingenuo Bayes, junto a los anteriormente mencionados como SVM y Random Forest.

3.5.5. Twitter-bert-base-sentimientos

Este modelo es muy similar al anterior nombrado anteriormente, está basado en el modelo roBERTa, y ha sido entrenado con 58 millones de tweets y el cual está principalmente enfocado para el análisis de sentimiento.

Es decir este modelo se utiliza para realizar una clasificación multiclase con 3 etiquetas, 0: Negativo; 1: Neutro; 2: Positivo, y respecto al modelo de roBERTuito este es un modelo adecuado al inglés por lo que será perfecto para nuestra tarea 2.

3.6. Selección de modelos

Una vez comentada la parte teórica de nuestros modelos desarrollados, pasamos a nombrar los resultados obtenidos en cada uno de ellos, para al realizar un análisis conjunto podamos obtener aquel que presente mejores resultados. Para poder evaluar estos modelos vamos a utilizar una serie de métricas de evaluación, estás métricas son en general la base de la evaluación de los modelos actuales del procesamiento del lenguaje natural, ya que son capaces de analizar y evaluar el modelo, sin la presencia de posibles medidas engañosas, que finalmente no presenten los resultados esperados .

3.7. Métricas de evaluación

Entre las métricas con las que vamos a evaluar nuestro modelo son :

- Como bien podemos observar en la imagen mostrada (ver Figura 3.16), vemos la llamada matriz de confusión la cual consta de una matriz de valores entre los que almacena los TN (True negative) valores tomados por el modelo como negativos y bien predecidos por el mismo, los FP (False positive) valores tomados como positivos pero con una predicción errónea del mismo. En contraposición a estos dos valores también obtenemos los FN (False negative) valores tomados erróneamente por el modelo como negativos y por último los TP (True positive) valores predecidos adecuadamente por nuestro modelo como valor de la clase positiva.

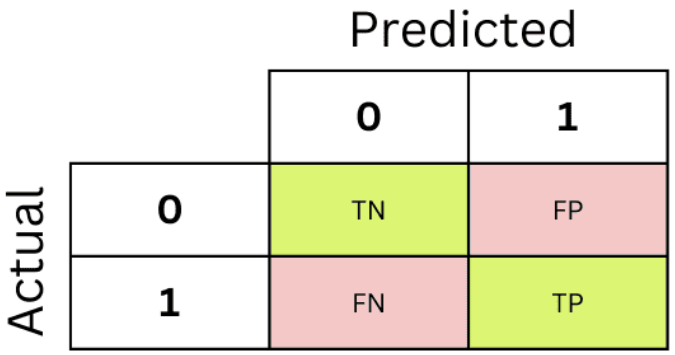

Figura 3.16: Tabla de confusión (Fuente: Confusion Matrix in Machine learning)

Esta matriz de valores es la llamada matriz de confusión que muestra resumidamente los resultados de la predicción obtenida por nuestro modelo.

- ***Precision***: La precisión es una forma de evaluar el modelo en torno a los valores que el sistema ha clasificado correctamente, es decir, se calcula dividiendo los TP (True positive) entre la suma de los TP más los FP (False positive).
- ***Recall***: La cobertura o en inglés llamada “Recall” es una forma de evaluar el modelo en torno a los valores que el sistema ha clasificado correctamente, es decir, se calcula dividiendo los TP (True positive) entre la suma de los TP más los FN (False negative).

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

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

Figura 3.17: Fórmulas de Precision y Recall (Fuente:Métricas De Evaluación De Modelos En El Aprendizaje Automático)

Las dos medidas anteriores pueden llegar a ser poco fiables si el conjunto de los datos presenta un desequilibrio entre las clases, para ello se utiliza la media más general y segura:

- ***F-score o F1***: La métrica de evaluación F1 es aquella que combina la precisión y el recall, proporcionando una medida que puede determinar el rendimiento del modelo en general.

$$F_1 = \frac{2PR}{P + R}$$

Figura 3.18: Fórmulas de F-score (Fuente: ¿Qué es una puntuación de F1? - Academia EITCA)

Para poder tener aún una mejor referencia sobre la efectividad de los datos realizamos también una media entre todas las clases, esta es llamada la medida “macro” y esto nos dará una mejor representación de los resultados de estos modelos del lenguaje.

3.8. Task 1

3.8.1. Modelos de aprendizaje automático

Dentro de estos algoritmos de aprendizaje automático hemos realizado las pruebas oportunas con aquellos nombrados en el apartado de desarrollo de los algoritmos, a continuación se exponen los resultados obtenidos en referencia a las 3 métricas nombradas en el apartado anterior, para cada una de ellas hemos establecido la diferenciación entre los resultados de cada métrica obtenida para cada clase de la misma clasificación.

Tablas de resultados :

Modelo	No_precision	No_Recall	No_F-score
SVM	0.956	1	0.977
RandomForest	0.906	0.994	0.948
Logistic Regression	0.916	0.989	0.951
SGD	0.917	0.988	0.951

Tabla 3.1: Resultados Métricas Clase No Algoritmos aprendizaje automático (Fuente: Elaboración propia)

Modelo	Yes_precision	Yes_Recall	Yes_F-score
SVM	1	0.597	0.747
RandomForest	0.676	0.094	0.165
Logistic Regression	0.684	0.204	0.314
SGD	0.683	0.212	0.323

Tabla 3.2: Resultados Métricas Clase Si Algoritmos aprendizaje automático (Fuente: Elaboración propia)

Modelo	macro_precision	macro_recall	macro_F-score
SVM	0.978	0.798	0.862
RandomForest	0.791	0.544	0.557
Logistic Regression	0.800	0.596	0.633
SGD	0.800	0.600	0.637

Tabla 3.3: Resultados Métricas Macro Algoritmos aprendizaje automático (Fuente: Elaboración propia)

Como observamos en esta tabla de resultados de estos algoritmos de aprendizaje automático, los resultados de los mismos no tienen una gran consistencia, no son malos resultados para estos tipos de algoritmos ya que es cierto que como veremos posteriormente los resultados según la métrica a valorar más detenidamente cómo es la macro_f1, veremos que los resultados llegan incluso a acercarse en algunos casos, sobre todo en el algoritmo SVM. Este mismo no llegó a ser explotado mayormente ya que tras realizar otras evaluaciones sobre otros conjuntos de datos las cuales no fueron igual de efectivas, además para determinar el porqué de este resultado podemos fijarnos en la condición del desequilibrio dentro del conjunto de datos de entrenamiento ya que el modelo ha podido seleccionar siempre una única clase y tener ese porcentaje de acierto el cual es muy similar al porcentaje de noticias etiquetadas con la clase 0.

3.8.2. Modelos de aprendizaje profundo

Una vez evaluados los modelos de aprendizaje automático pasamos a ver los resultados de los modelos de aprendizaje profundo, estas pruebas finalmente se realizaron con los modelos de facebook/IA roBERTa base, RoBERTa base BNE y roBERTuito ya que eran los modelos que más relación presentaban con nuestra tarea de detección de fake news. Para cada una de las pruebas realizamos la ejecución de cada modelo un número de épocas determinado, en cada caso se establecen las necesarias para ver la evolución del modelo con respecto al entrenamiento.

En cuanto a las pruebas del primer modelo que probamos fue el transformer RoBERTa base BNE:

RoBERTa base BNE	No_precision	No_Recall	No_F-score
Epoch 1	0.859	1	0.924
Epoch 2	0.859	1	0.924
Epoch 3	0.939	0.971	0.957
Epoch 4	0.953	0.950	0.951
Epoch 5	0.945	0.959	0.952

Tabla 3.4: Resultados Métricas Clase No Algoritmos RoBERTa base BNE (Fuente: Elaboración propia)

RoBERTa base BNE	Yes_precision	Yes_Recall	Yes_F-score
Epoch 1	0.0	0.0	0.0
Epoch 2	0.0	0.0	0.0
Epoch 3	0.801	0.617	0.699
Epoch 4	0.703	0.715	0.709
Epoch 5	0.726	0.661	0.692

Tabla 3.5: Resultados Métricas Clase Yes Algoritmos RoBERTa base BNE (Fuente: Elaboración propia)

RoBERTa base BNE	macro_precision	macro_recall	macro_F-score
Epoch 1	0.429	0.511	0.462
Epoch 2	0.429	0.511	0.462
Epoch 3	0.873	0.796	0.828
Epoch 4	0.828	0.832	0.829
Epoch 5	0.836	0.810	0.812

Tabla 3.6: Resultados Métricas Macros Algoritmos RoBERTa base BNE (Fuente: Elaboración propia)

Como observamos estos resultados son bastante decentes en cuanto a efectividad, y aunque las dos primeras épocas difieran de esos buenos resultados debido a cualquier problema o alguna variación en las primeras fases de entrenamiento del modelo, se ve una gran consistencia en el resto de épocas del entrenamiento de este modelo. Los valores obtenidos gracias a las pruebas y el entrenamiento de este modelo ascienden hasta un 0.829 de macro_F1 lo cual es un resultado gratamente positivo para nuestra tarea. Por ello será seleccionado posteriormente para continuar nuestro desarrollo y posible mejora del mismo.
Seguidamente pasamos a evaluar el modelo de facebook/IA roBERTa base:

Facebook/AI	macro_precision	macro_recall	macro_F-score
Epoch 1	0.461	0.457	0.459
Epoch 2	0.635	0.630	0.632
Epoch 3	0.630	0.626	0.628
Epoch 4	0.620	0.616	0.619
Epoch 5	0.595	0.591	0.593

Tabla 3.7: Resultados Métricas macros Algoritmos facebook/IA roBERTa base (Fuente: Elaboración propia)

Estos resultados obtenidos en primera instancia nos obligó a descartar este modelo del lenguaje ya que los resultados con respecto a los obtenidos anteriormente con el modelo BERT con base en español fueron bastante peores en casi todas las métricas de evaluación y sobre todo en la más importante en cuanto a nuestra competición, la macro _f1. Esto no quiere decir que este modelo no sea válido o sus resultados no presenten eficiencia, simplemente que para esta tarea de clasificación el modelo no es capaz de aprender lo suficiente acerca de las relaciones entre los token de las oraciones y por lo tanto no sea eficaz para nuestra tarea.

Por último probamos el roBERTuito:

RoBERTuito	macro_precision	macro_recall	macro_F-score
Epoch 1	0.696	0.630	0.590
Epoch 2	0.770	0.770	0.770
Epoch 3	0.747	0.749	0.748
Epoch 4	0.745	0.741	0.739
Epoch 5	0.739	0.729	0.723

Tabla 3.8: Resultados Métricas macros Algoritmos RoBERTuito (Fuente: Elaboración propia)

Este modelo al estar basado en la misma tarea para la cual estamos investigando y desarrollando, era de esperar que los resultados que tuviésemos fuesen bastante buenos, sin embargo, no es suficiente para alcanzar los grandes resultados obtenidos por el transformer RoBERTa base BNE.

3.9. Task 2

3.9.1. Modelos de aprendizaje automático

Después de realizar el análisis de los resultados para la tarea 1, pasamos al análisis de la tarea 2, el cual como hemos realizado anteriormente, primeramente nos basaremos en el desarrollo e investigación de los modelos basados en los algoritmos de aprendizaje automático. Como hemos visto en algunos casos, cualquiera de ellos pueden presentar un buen resultado, bien sea por algún tipo de

relación entre las palabras o simplemente que el modelo aprenda y se especializa demasiado en una de las clases de la tarea, por lo que deberemos estudiar el porqué de los resultados tanto por su buena eficiencia o por lo contrario por su mala eficiencia.

Modelo	SUBJ_precision	SUBJ_Recall	SUBJ_F-score
SVM	0.752	0.047	0.088
Random Forest	0.633	0.354	0.454
Logistic Regression	0.75	0.070	0.129
SGD	0.626	0.448	0.522

Tabla 3.9: Resultados Métricas clase SUBJ Algoritmos Aprendizaje automático(Fuente: Elaboración propia)

Modelo	OBJ_precision	OBJ_Recall	OBJ_F-score
SVM	0.485	0.982	0.649
Random Forest	0.523	0.775	0.625
Logistic Regression	0.489	0.974	0.651
SGD	0.539	0.706	0.611

Tabla 3.10: Resultados Métricas clase OBJ Algoritmos Aprendizaje automático(Fuente: Elaboración propia)

Modelo	macro_precision	macro_recall	macro_F-score
SVM	0.617	0.515	0.369
Random Forest	0.578	0.565	0.539
Logistic Regression	0.619	0.522	0.390
SGD	0.582	0.577	0.567

Tabla 3.11: Resultados Métricas Macro Algoritmos Aprendizaje automático (Fuente: Elaboración propia)

Tal y como se aprecia en estas tablas de datos los resultados de estos algoritmos son bastante peores que en la tarea anterior, que como hemos nombrado en el análisis anterior junto a estos resultados, podemos confirmar que los resultados de SVM y Logistic Regression de la tarea anterior no son muy fiables, debido al desequilibrio de las clases en el conjunto de datos, por lo que en esta tarea posiblemente ocurriese lo contrario y el modelo le da una mayor importancia a la clase minoritaria por ello que la medida “SUBJ_Recall” es prácticamente 0. Esto no ocurre para los algoritmos de Random forest y SGD los cuales parece que mantienen una constancia en cuanto a los dos conjuntos de datos, los otros dos presentan una gran caída en su porcentaje de acierto.

3.9.2. Modelos de aprendizaje profundo

Para esta tarea simplificamos las pruebas en cuanto a la diferencias entre modelos, por lo que nos centramos más en la investigación de mejora de los resultados de estos dos modelos. Nos centramos principalmente en la evaluación de los modelos de BERT que como hemos podido observar en los resultados de la tarea 1 sabemos que serán muy similares a esta por tratar al igual que la anterior de una clasificación de dos clases.

Primeramente evaluamos el modelo que antes nos ha dado los mejores resultados para la tarea 1, BERT pero en este caso no nos basamos en el modelo derivado para el español (Base bne) sino que utilizamos el modelo principal de BERT entrenado con conjuntos de datos en inglés.

BERT	SUBJ_precision	SUBJ_Recall	SUBJ_F-score
Epoch 1	0.632	0.764	0.692
Epoch 2	0.566	0.924	0.702
Epoch 3	0.778	0.764	0.771
Epoch 4	0.733	0.754	0.744
Epoch 5	0.699	0.811	0.751

Tabla 3.12: Resultados Métricas clase SUBJ aprendizaje profundo BERT (Fuente: Elaboración propia)

BERT	OBJ_precision	OBJ_Recall	OBJ_F-score
Epoch 1	0.725	0.584	0.647
Epoch 2	0.826	0.336	0.477
Epoch 3	0.782	0.796	0.789
Epoch 4	0.763	0.743	0.744
Epoch 5	0.791	0.672	0.727

Tabla 3.13: Resultados Métricas clase OBJ aprendizaje profundo BERT (Fuente: Elaboración propia)

BERT	macro_precision	macro_recall	macro_F-score
Epoch 1	0.679	0.674	0.668
Epoch 2	0.696	0.630	0.590
Epoch 3	0.780	0.782	0.780
Epoch 4	0.748	0.749	0.748
Epoch 5	0.745	0.741	0.739

Tabla 3.14: Resultados Métricas Macro aprendizaje profundo BERT (Fuente: Elaboración propia)

Como podemos observar los resultados son un poco más bajos que en la tarea anterior, ya que para cada clasificación se puede observar una gran variabilidad entre los resultados, por ello que cada modelo puede ser más eficaz en unas tareas que en otras.

Ahora pasamos a almacenar los resultados iniciales para nuestro modelo entrenado ya para una tarea de subjetividad y objetividad, como es twitter-roberta-base-sentimiento.

twitter-roberta-base-sentimiento	macro_precision	macro_recall	macro_F-score
Epoch 1	0.588	0.688	0.634
Epoch 2	0.740	0.790	0.780
Epoch 3	0.692	0.761	0.742
Epoch 4	0.655	0.735	0.713
Epoch 5	0.680	0.740	0.723

Tabla 3.15: Resultados Métricas Macro aprendizaje profundo BERT (Fuente: Elaboración propia)

3.10. Estrategias de mejora

Una vez realizado cada una de las pruebas oportunas con cada uno de los modelos que mejores resultados proporcionaban en tareas similares, pasamos a realizar un análisis posterior, para evaluar qué mejoras se podrían establecer a cada uno de los modelos.

Como hemos comentado los algoritmos de aprendizaje profundo presentan resultados muy mejorados respecto a los de aprendizaje automático por lo que nos centraremos en estos.

A partir de aquí pasamos al planteamiento de hipótesis y posteriormente al realizamiento de las pruebas oportunas para comprobar la eficacia de cada una de ellas para su adquisición o descarte de las mismas.

3.10.1. Aumentación de datos

Una primera estrategia de mejora, podríamos basarnos en que estos modelos supuestamente mejoran gracias a la experiencia, por lo que si aumentamos esa experiencia podríamos obtener mejores resultados. Para ello realizamos una aumentación de datos.

Para realizar esta aumentación de datos necesitamos conjuntos de datos con la misma naturaleza que los conjuntos de datos de entrenamiento actuales. Para poder conseguir esta premisa, conseguimos los corpus en otros idiomas para la misma tarea en la que participamos, pero disponible para otros idiomas, y eran datos distintos por lo que era justo lo necesario para esta mejora.

Para ello los traducimos usando herramientas que nombraremos en próximos apartados y una vez obtenidos otros corpus traducidos tanto a la tarea 1 como a la tarea 2, realizamos la aumentación de datos en cada uno de los conjuntos de entrenamiento.

Para la tarea 1 con un corpus inicial de 19949 de oraciones en español, le sumamos las 12051 secuencias de texto en árabe más las 22501 de inglés, ambos corpus traducidos al español, con lo que recopilamos más de 50000 oraciones para entrenar nuestro modelo.

Mientras tanto para la segunda tarea teníamos un corpus inicial de 831, y a ellas les sumamos las 1186 oraciones provenientes del idioma árabe traducidas al inglés, obteniendo un total de más de 2000 oraciones, una gran diferencia entre corpus aumentados entre las dos tareas, que se refleja en la gráfica (ver Figura 3.16).

3.10.2. Traducción de corpus

Como bien hemos hecho referencia en el apartado anterior para poder realizar la aumentación de datos, con datos de la competición del CLEF en otros idiomas era necesaria la traducción, para ello requerimos de una herramienta capaz de traducir una columna de los distintos corpus, la columna en la que encontramos las sentencias de texto que se trasladarán al modelo como entrada.

Investigando entre los distintos modelos de hugging face, encontramos varios que eran capaces de traducir grandes cantidades de texto.

Después de la lectura de varios de ellos nos decantamos por un modelo que tenía distintas variedades, es decir, distintos modelos que eran capaces de traducir entre varios idiomas, por lo cual era apropiado para las dos tareas.

El modelo del que hablamos es “Helsinki-NLP” que según el modelo derivado que tomemos traducía a un idioma u a otro.

Para la tarea 1 la cual requiere del texto en español utilizamos:

- **Helsinki-NLP/opus-mt-en-es**: Este modelo traduce el texto con idioma origen inglés hacia el idioma destino español.
- **Helsinki-NLP/opus-mt-ar-es**: Este modelo traduce el texto con idioma origen árabe hacia el idioma destino español.

Para la tarea 2 la cual requiere del texto en inglés utilizamos:

- **Helsinki-NLP/opus-mt-ar-en**: Este modelo traduce el texto con idioma origen árabe hacia el idioma destino inglés.

Posteriormente continuando con el apartado anterior de la aumentación de datos en la cual utilizamos esta estrategia de traducción de textos, observamos que los bajos resultados en la eficiencia de los modelos podría deberse a esta tarea, en efecto, tras investigar detenidamente la traducción de los corpus, nos dimos cuenta que pese a que los modelos presentaban todas las palabras bien traducidas al lenguaje, no estaban correctamente establecidas, por lo que se presentaban algunas diferencias entre las traducciones entre distintas lenguas.

Como podemos apreciar en la imagen (ver Figura 3.19) la traducción de esta secuencia de texto no es completamente correcta, por lo que al igual que esta se presentan muchas otras, lo que provoca que esta mala relación entre las palabras o tokens de la secuencia al ser traducida, se transforme en un aprendizaje más bajo de nuestro modelo en ambas tareas.

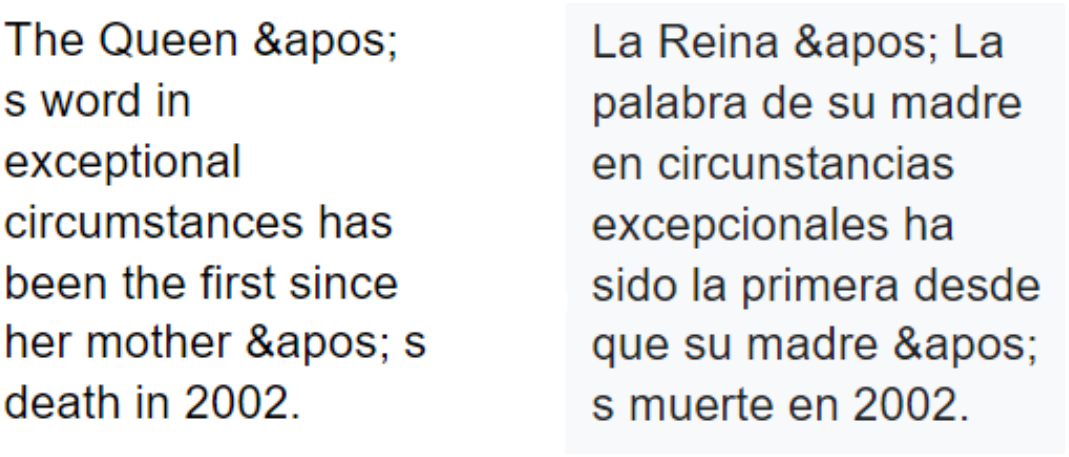

Figura 3.19: Ejemplo de mala traducción en los corpus utilizados para aumentación (Fuente: Elaboración propia)

3.10.3. Búsqueda de polaridad en Tarea-2

Como sabemos la segunda tarea en la que participamos cuenta con el desarrollo de sistemas capaces de detectar la subjetividad de una oración. Esta tarea es bastante abstracta por lo que otras muchas aplicaciones buscan un enfoque bastante similar, como la detección de la polaridad de un texto. Para realizar esta búsqueda de la polaridad usaremos un modelo ya nombrado anteriormente como es twitter-roberta-base-sentimiento el cual va a realizar un primer análisis de la secuencia de texto y establecer una etiqueta asociando el valor objetivo, subjetivo o neutro a todas las secuencias de entrada, una vez establecida estas etiquetas a todas las secuencias del texto introducimos estas clasificaciones en forma de tokens <POS>, <NEG> y <NEU> al principio de cada oración.

Con esto conseguimos que el corpus de entrenamiento tenga una mayor información en este caso una etiqueta que puede dar una referencia al siguiente modelo que sea entrenado con este corpus, por lo que este modelo tendrá una mayor cantidad de información para su aprendizaje con respecto a otros modelos.

3.10.4. Búsqueda de hiper parámetros

Posteriormente al intento de mejorar con la aumentación de datos que vimos que no era suficiente para mejorar los resultados, buscamos otra estrategia en la que nos basamos en el ajuste óptimo de los parámetros de entrenamiento de los modelos.

Para ello introducimos en el código una función que es capaz de obtener el mejor valor de los parámetros solicitados, estableciendo manualmente un rango de búsqueda con el mínimo valor y el máximo de cada parámetro. En cuanto a los parámetros que vamos a ajustar según esta búsqueda encontramos:

- **Batch Size**: Encontramos la definición del batch size como el tamaño del lote de los datos que transmite a través de la red durante el entrenamiento de nuestro modelo. Este a mayor tamaño del mismo ofrece un mayor rendimiento computacional, sin embargo hace un mayor uso de la memoria.
- **Learning rate**: El learning rate o tasa de aprendizaje se encarga de realizar el ajuste oportuno del tamaño de los pasos que va realizando el algoritmo al ajustar los pesos del modelo. Con una mayor tasa de aprendizaje obtenemos una convergencia más rápida mientras que si este es más bajo, tendremos un menor riesgo de poder caer en un mínimo global.
- **Número de épocas**: Por último el número de épocas o número de epoch simplemente se basa en el número de veces que el algoritmo durante el entrenamiento recorre todo el conjunto de datos de entrada. Esto puede provocar dos extremos durante el entrenamiento, bien que el algoritmo produzca un subajuste, es decir, no aprenda lo suficiente, o un sobreajuste o sobreentrenamiento, nuestro modelo aprende a demasiado bajo nivel los detalles y no generaliza bien los datos.

4. Análisis de resultados de los modelos

Una vez obtenido los primeros resultados de todos los tipos de algoritmos y modelos, analizado cuáles fueron los mejores resultados entre ellos, pasamos a la aplicación de las hipótesis o estrategias planteadas anteriormente, y junto a los primeros resultados analizar todos los modelos obtenidos y seleccionar posteriormente el mejor de ellos para cada una de las tareas.

4.1. Análisis Tarea-1

Una vez seleccionado el modelo del lenguaje basado en RoBERTa base BNE, el cual nos ha proporcionado los mejores resultados con respecto al resto de

modelos y algoritmos, pasamos a tratar de mejorar aún más los resultados de predicción con las estrategias de mejora.

Primeramente pasamos a realizar una **aumentación de datos** como bien hemos nombrado anteriormente con los conjuntos de datos de la misma competición pero derivada a otro idioma, para ello utilizamos los modelos de traducción **Helsinki-NLP/opus-mt-en-es** y **Helsinki-NLP/opus-mt-ar-es** para traducir tanto el conjunto inglés y el árabe y realizamos un entrenamiento con la unión de los 3 conjunto de datos de entrenamiento.

Tras realizar la evaluación del nuevo modelo vemos que los resultados no son los esperados, tras realizar un aumento de la experiencia del modelo, vemos como a lo largo del entrenamiento presenta síntomas de un sobreentrenamiento por lo que la eficiencia del modelo bajo no muy significativamente, pero no mejora qué es lo que buscábamos.

Para nuestra sorpresa que al aumentar los datos y por lo tanto aumentar la experiencia, supuestamente debería mejorar los resultados, pero esto no fue así, es más los resultados empeoraron en ambos modelos.

Como es fácil de entender los modelos durante el entrenamiento llega un momento que dejan de aprender e incluso pueden empeorar sus resultados, es por ello que con tal cantidad de muestras de nuestro entrenamiento de la tarea 1, era fácil la aparición de sobreentrenamiento.

El resultado obtenido con este modelo teniendo en cuenta el máximo valor obtenido sobre la métrica de macro _F1 es: **0.823**

A continuación aplicamos otra de las técnicas planteadas que sería la búsqueda de hiper parámetros gracias a la función mencionada anteriormente que consigue establecer cada uno de ellos.

En este caso la técnica aplicada si mejora los resultados de una manera notable como podemos observar en las tablas de resultados (ver Tabla 4.1,4.2,4.3), ya que gracias a esta búsqueda conseguimos acercarnos al entrenamiento más óptimo posible que podemos realizar con este modelo.

RoBERTa base BNE	No_precision	No_Recall	No_F-score
Época 1	0.947	0.970	0.958
Época 2	0.948	0.971	0.959
Época 3	0.966	0.922	0.943
Época 4	0.946	0.964	0.955
Época 5	0.933	0.974	0.953

Tabla 4.1: Resultados Métricas clase No BERT mejorado (Fuente: Elaboración propia)

RoBERTa base BNE	Yes_precision	Yes_Recall	Yes_F-score
Época 1	0.814	0.706	0.756
Época 2	0.824	0.713	0.764
Época 3	0.662	0.824	0.734
Época 4	0.785	0.706	0.744
Época 5	0.819	0.623	0.708

Tabla 4.2: Resultados Métricas clase Yes BERT mejorado (Fuente: Elaboración propia)

RoBERTa base BNE	macro_precision	macro_recall	macro_F-score
Época 1	0.880	0.838	0.857
Época 2	0.886	0.842	0.862
Época 3	0.814	0.873	0.839
Época 4	0.866	0.835	0.849
Época 5	0.876	0.798	0.830

Tabla 4.3: Resultados Métricas Macro BERT mejorado (Fuente: Elaboración propia)

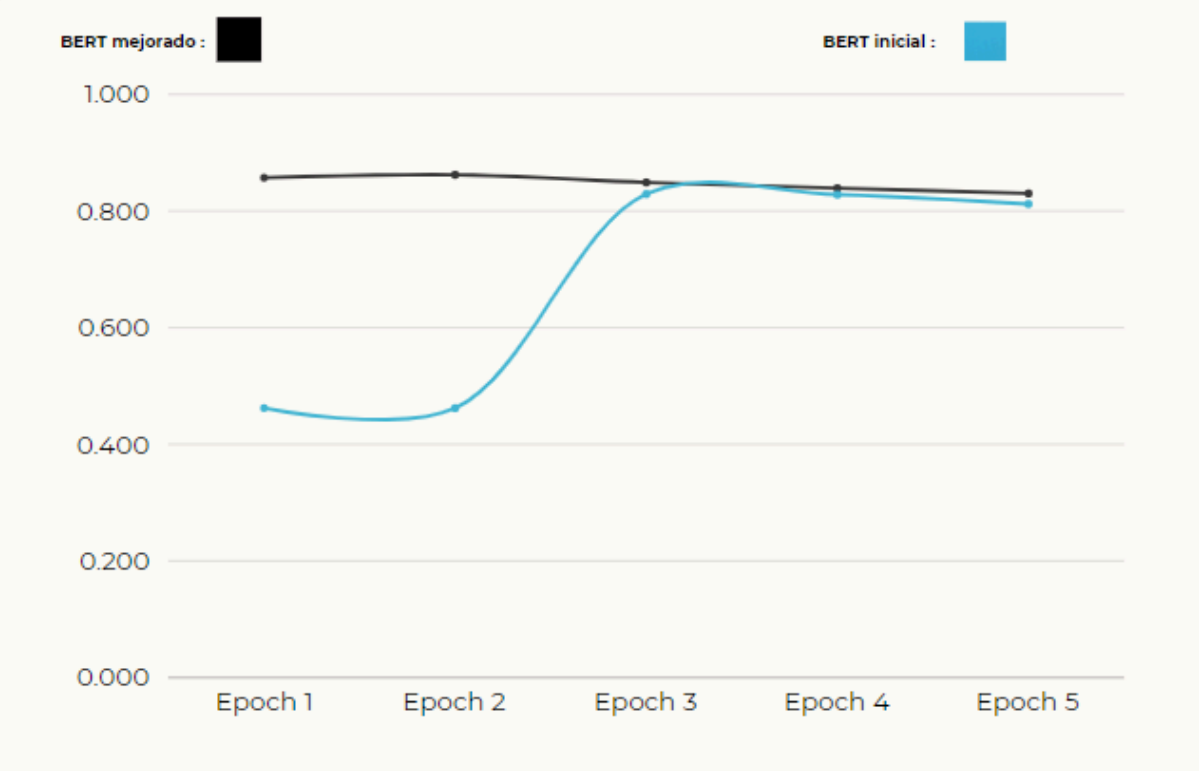

Figura 4.1: Gráfica de comparación de modelo BERT con estrategia de búsqueda hiper parámetros tarea 1 (Fuente: Elaboración propia)

Basándonos en la métrica más relevante del modelo y la solicitada por la competición en la que participamos, la macro_F1, vemos como la aplicación de la técnica a partir de un modelo con un máximo de 0.829 obtenemos hasta un resultado de 0.862, lo cual es una mejora bastante notable, lo cual nos deja un modelo competitivo para presentar .

Como hemos podido observar en el gráfico (ver Figura 4.1), el mejor resultado obtenido en el modelo inicial, es inferior en todas y cada una de las épocas entrenadas y evaluadas con el modelo mejorado, por lo que la mejora del mismo es bastante notable.

Mejores parámetros Tarea 1

Como sabemos el entrenamiento de estos modelos puede presentar distintos tipos de parámetros para definir cómo va a ser el desarrollo del aprendizaje de nuestro modelo, por ello vamos a buscar el mejor ajuste de los siguientes parámetros para la tarea 1 los cuales después de varias ejecuciones realizando esta búsqueda del mejor de cada uno de los parámetros encontrados son los siguientes:

- **Batch size ideal**: El tamaño del batch size ideal para nuestro modelo en la tarea 1, es 8, esto marca un equilibrio entre el malgasto de memoria necesaria para ello, y la velocidad de transmitir los lotes, por lo que fue el mejor tamaño encontrado.
- **Learning rate ideal**: El learning rate fue el parámetro que más variabilidad ganabamos respecto al resultado, por lo que fue importante la búsqueda del mismo, en el cual su mejor valor es 3.93454e-06.
- Por último el **número de épocas ideal** para no llegar al sobreentrenamiento fue 12, para ello realizamos una serie de ajustes en el código para que en el momento que los resultados obtenidos comenzaban a empeorar el algoritmo para el entrenamiento y se quedaba con ese número total de épocas.

Como hemos mencionado anteriormente estos parámetros han sido obtenidos gracias a la ejecución de varias épocas en las que en cada una de ellas se ha variado dentro del rango establecido de cada uno de los parámetros que hemos optimizado.

4.1.1. Overfitting

Como definición de sobreentrenamiento o overfitting, podemos entender que el modelo está sobreentrenado cuando el modelo funciona extremadamente bien con los datos de entrenamiento, pero tiene serias dificultades con los datos de evaluación o testeo, esto quiere decir que el modelo se ha ajustado demasiado al conjunto de entrenamiento en específico y posteriormente no es capaz de generalizar acerca de otros modelos (ver Figura 4.2).

Esto deriva en un mal aprendizaje del modelo ya que los resultados de este sobre cualquier otro conjunto de datos que no sea su conjunto de entrenamiento va a presentar una menor eficiencia. Este también se puede definir como el término opuesto del “Underfitting”, el cual se trata de que el modelo en vez de ajustarse demasiado a nuestro conjunto de entrenamiento, no llega a ajustarse a él, es decir, el modelo no tiene la eficacia suficiente para poder capturar las relaciones entre los distintos tokens o palabras de entrada del modelo[20].

Para poder detectar el sobreajuste en los modelos podemos fijarnos en una de las características explícitas del entrenamiento del modelo, “Training loss”.

- **Pérdida de entrenamiento o “Training loss”**: En el momento del entrenamiento de nuestro modelo llegará un punto en el que la pérdida o la

tasa de pérdida tanto para el entrenamiento como para la evaluación será muy alta y podremos observar el momento de crecimiento de esta para detectar cuando el modelo comienza a sobreentrenar.

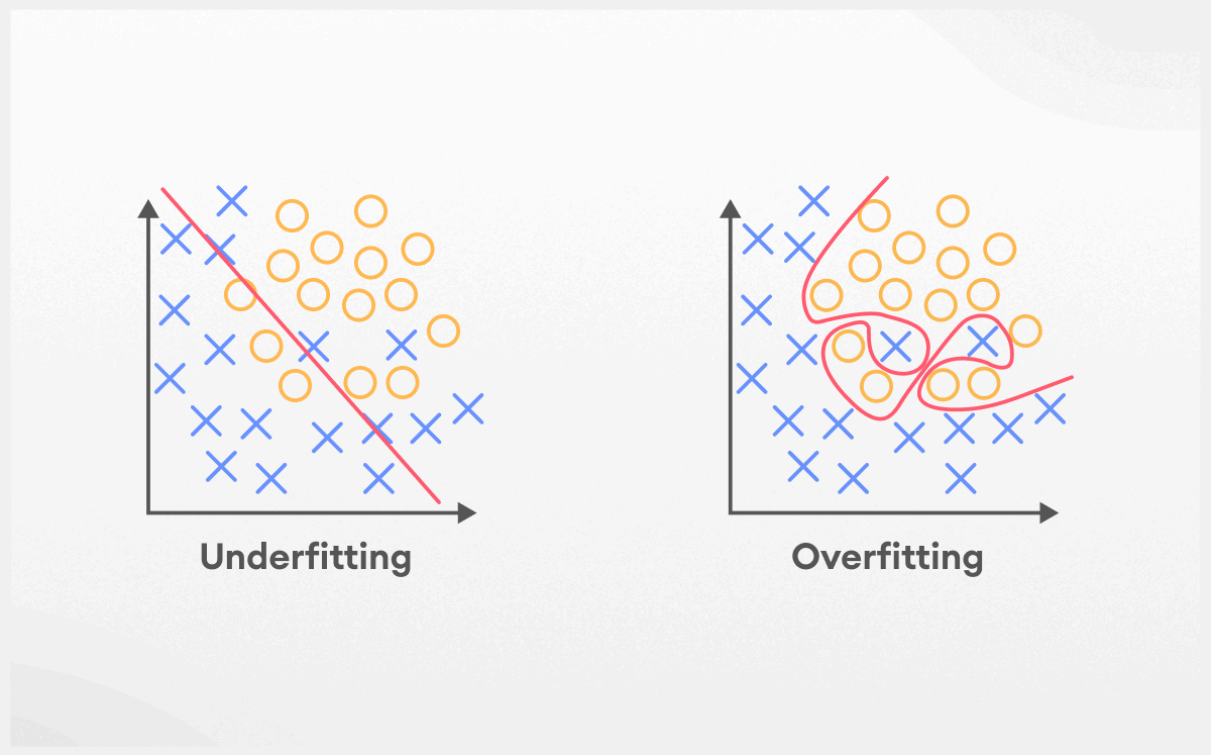

Figura 4.2: Significado gráfico de overfitting y underfitting (Fuente: Overfitting and underfitting in machine learning | SuperAnnotate)

4.2. Análisis Tarea-2

Una vez terminado el análisis anterior, comenzamos a realizar el análisis de la tarea 2 el cual seguirá pasos similares a esta. Comenzamos tomando el modelo basado en el transformar BERT esta vez el original, el cual nos ha proporcionado en instancias iniciales grandes resultados acerca de nuestro objetivo.

Comenzamos primeramente con la primera posibilidad de mejora que planteamos anteriormente como es la aumentación de datos gracias a los corpus traducidos, para esta tarea solo utilizaremos un conjunto de datos adicional, para evitar que suceda como en la tarea anterior y suceda un sobreentrenamiento de nuestro modelo. Para ello utilizamos el modelo previamente encontrado como **Helsinki-NLP/opus-mt-ar-en**, el cual realizará la traducción del corpus utilizado para la tarea 2 en arabe y lo traduce al inglés para poder entrenar a nuestro modelo.

Esta vez el modelo no estaba tan sobrecargado como para la tarea 1, sin embargo los resultados seguían sin mejorar, con el mejor resultado obtenido con esta técnica de un 0.774 , seguía sin superar el 0.780 obtenido con el modelo actual.

Por lo que una vez más esta técnica no está siendo competente en nuestros modelos, debido a este sobreentrenamiento o “overfitting”.

Sin embargo esta conclusión no es del todo completa, debido a que posteriormente al evaluar la aumentación de datos en la tarea 2, nos damos cuenta que los resultados también empeoran, y como hemos visto en la imagen anterior (ver Figura 3.15), la diferencia entre los datos es enorme (ver Figura 4.3), por lo que no debería de producir este sobreentrenamiento en el segundo modelo.

Por lo cual esta bajada en la eficiencia de los modelos puede deberse a otros factores, en concreto planteamos la posibilidad de una mala traducción de nuestros corpus, lo cual observaremos a continuación.

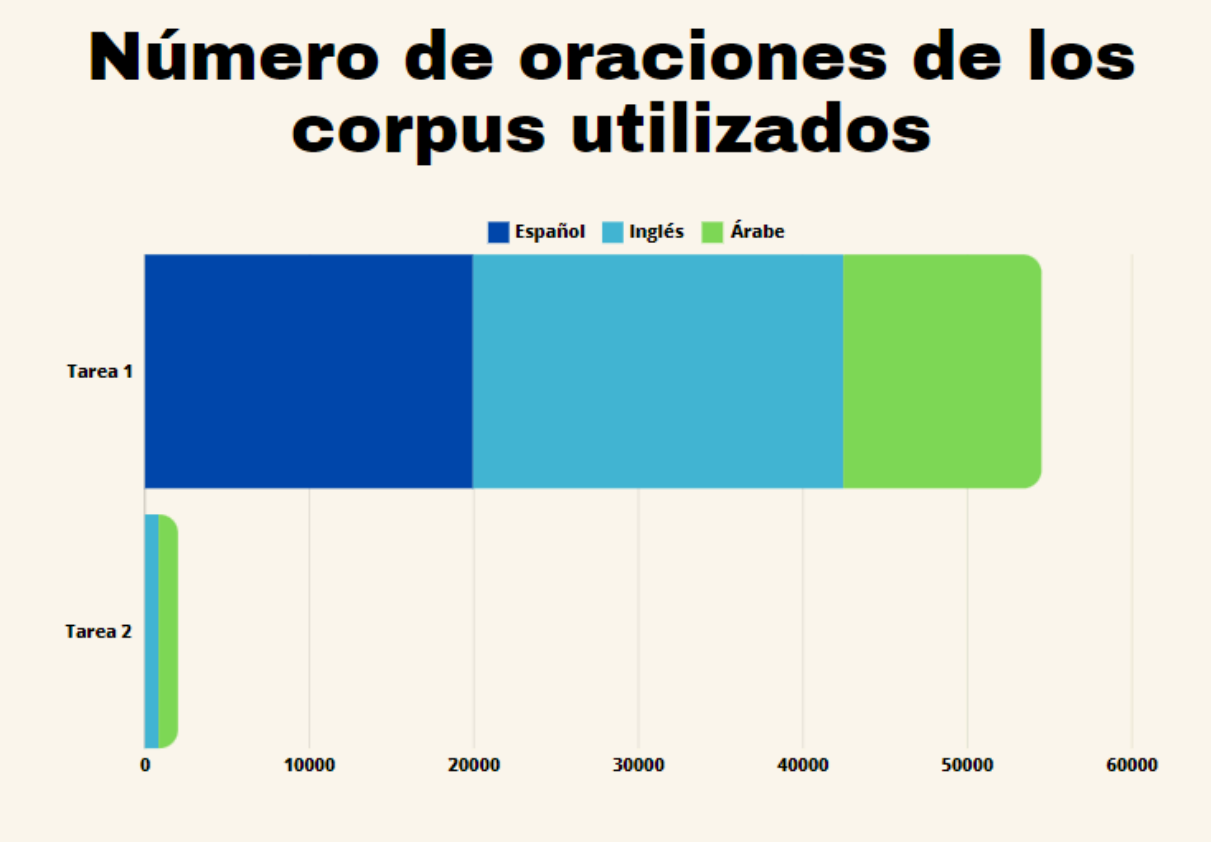

Figura 4.3: Comparación de número de secuencias de texto de los corpus utilizados (Fuente: Elaboración propia)

Como segunda mejora vamos a tratar de aplicar un aumento de la información que le añadimos a nuestro modelo. Para ello vamos a realizar una

búsqueda de la polaridad del texto ya que al tratarse de una tarea de clasificación sobre la subjetividad del mismo, la polaridad tiene una gran relación con ello.

Por lo que para aplicar esta técnica vamos a realizar un primer entrenamiento con el modelo de twitter-roberta-base-sentimiento con el conjunto de datos, este determinará a cada oración con una etiqueta bien sea positivo (presenta subjetividad), negativo (presenta objetividad) o neutro (si presenta las dos indiscriminadamente), una vez terminado, estas etiquetas de clasificación se añadirán al conjunto de datos de entrenamiento para que el modelo de entrenamiento posterior, en nuestro caso BERT tenga un token de referencia acerca de la clasificación del modelo anterior.

Con esta técnica los resultados fueron:

BERT+twitter-roberta-base-sentimiento	macro_precision	macro_recall	macro_F-score
Época 1	0.609	0.538	0.440
Época 2	0.730	0.730	0.731
Época 3	0.660	0.640	0.625
Época 4	0.645	0.638	0.631
Época 5	0.656	0.651	0.645

Tabla 4.4: Resultados Métricas Macro BERT más twitter-roberta-base-sentimiento(Fuente: Elaboración propia)

Como podemos observar los resultados en cuanto al modelo aplicado únicamente es casi tan eficaz como nuestro modelo por referencia BERT, sin embargo al unirlos, lo cual podríamos entender que los resultados incrementarían, sin embargo son bastante peores teniendo en cuenta que como mejor resultado encontramos 0.731 de macro_f1 quedándose muy atrás del resultado óptimo.

Analizando el resultado podemos ver que este es muy cercano al resultado anterior del entrenamiento propio del modelo twitter-roberta-base-sentimiento, por lo que nos lleva a pensar que nuestro modelo simplemente se deja guiar por la etiqueta creada en cada secuencia de texto, es decir finalmente obtendrá el resultado obtenido por este modelo, por lo que esta técnica no es de mucha utilidad.

Por último, al igual que en la tarea anterior, pasamos a la búsqueda de hiper parámetros la cual anteriormente nos ha llevado a encontrar esos buenos resultados.

Para ello repetimos la operación anterior estableciendo un código que sea capaz de buscar en cada ejecución el parámetro adecuado entre un rango especificado de entrada en cada uno de ellos.

BERT	No_precision	No_Recall	No_F-score
Época 1	0.660	0.660	0.689
Época 2	0.8	0.679	0.734
Época 3	0.811	0.771	0.762
Época 4	0.781	0.811	0.796
Época 5	0.801	0.801	0.801

Tabla 4.5: Resultados Métricas clase No BERT con búsqueda de hiper parámetros(Fuente: Elaboración propia)

BERT	Yes_precision	Yes_Recall	Yes_F-score
Época 1	0.704	0.761	0.731
Época 2	0.736	0.840	0.785
Época 3	0.803	0.725	0.762
Época 4	0.816	0.787	0.801
Época 5	0.814	0.814	0.814

Tabla 4.6: Resultados Métricas clase Yes BERT con búsqueda de hiper parámetros (Fuente: Elaboración propia)

BERT	macro_precision	macro_recall	macro_F-score
Época 1	0.713	0.710	0.710
Época 2	0.768	0.762	0.759
Época 3	0.769	0.768	0.767
Época 4	0.799	0.798	0.799
Época 5	0.808	0.807	0.808

Tabla 4.7: Resultados Métricas macro BERT con búsqueda de hiper parámetros (Fuente: Elaboración propia)

Una vez más esta técnica de mejora es capaz de mejorar nuestros resultados iniciales con el modelo, por lo que podemos asegurar que la búsqueda de hiper parámetros es una técnica que presenta grandes resultados, sobre todo en tareas similares a nuestro estudio, es decir, técnicas de clasificación binaria (ver Figura 4.4).

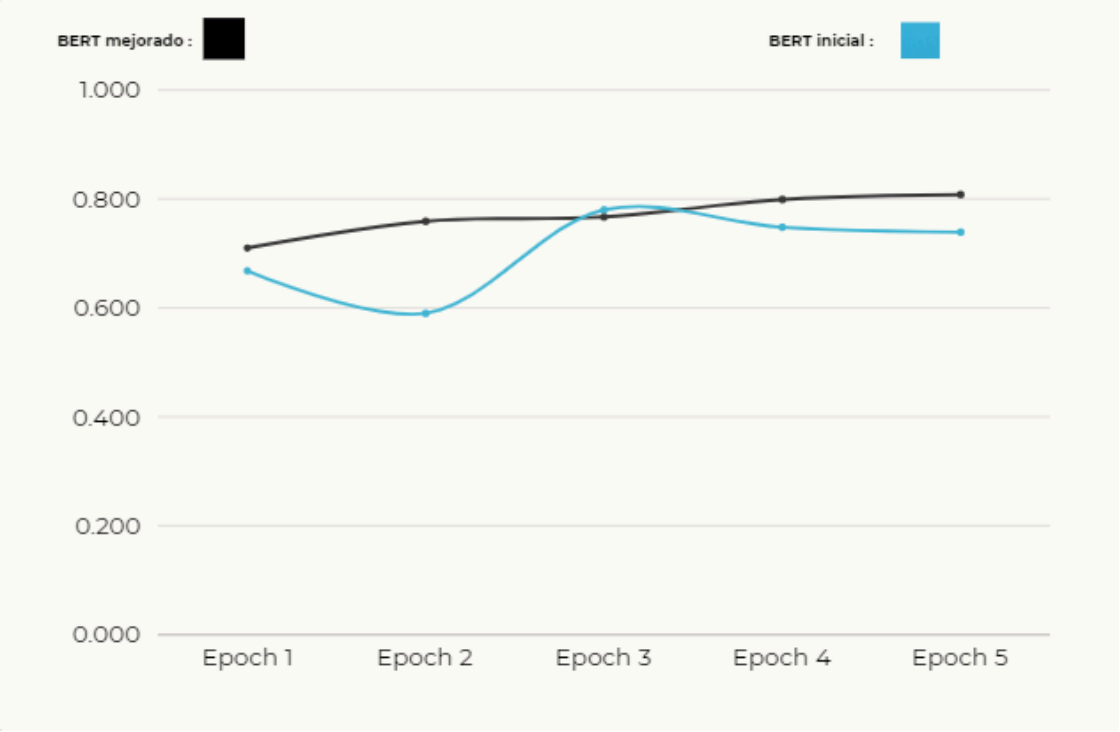

Figura 4.4: Gráfica de comparación de modelo BERT con estrategia de búsqueda hiper parámetros tarea 2(Fuente: Elaboración propia)

Mejores parámetros Tarea 2

Seguidamente pasamos a comentar la búsqueda de parámetros realizada para la tarea 2 del proyecto, y para ella los que nos proporcionaron una mayor eficiencia y mejores resultados fueron:

- **Batch size ideal**: El tamaño del batch size ideal para nuestro modelo en la tarea 2, es también 8, esto al igual que en la tarea anterior consigue crear un equilibrio entre el exceso en memoria y la velocidad de entrenamiento y ejecución del modelo entrenado.
- **Learning rate idea**l: En cuanto al learning rate es un parámetro que a parte de mejorar mucho la eficiencia de nuestro modelo también es capaz de mejorar la velocidad de entrenamiento del mismo obteniendo un modelo mucho más eficaz en ambos aspectos, para esta tarea obtuvimos el mejor valor del mismo en 3.220e-06.
- Por último el **número de épocas ideal** para no llegar al sobreentrenamiento fue 9, como en la tarea anterior se aplicó el código para para la búsqueda cuando se haya alcanzado el tope de mejora de nuestro modelo.

Al igual que en la tarea anterior ejecutamos el código previamente definido para ejecutar el número de épocas necesarias para encontrar el mejor ajuste de los parámetros a nuestro modelo.

4.3. Resultados Check That!

Como bien hemos nombrado al inicio de este proyecto, se iban a desarrollar modelos los cuales serán evaluados por la competición CLEF 2024 la cual tiene un gran prestigio a nivel mundial .

Pese a haber investigado y desarrollado buenos modelos para ambas competiciones, por temas ajenos a nosotros, los organizadores decidieron de última hora cancelar la su tarea en español, es decir nuestra tarea 1, por lo que nuestro modelo desarrollado no podría participar en esta tarea. Por ello, en pocos días viendo nuestros experimentos, observando los mejores modelos obtenidos y las mejoras que mejores resultados nos han aportado, desarrollamos otro modelo basado en el transformer BERT y le aplicamos una búsqueda de hiperparametros.

Después de este contratiempo conseguimos presentar los dos modelos a tiempo, además ambos con unos resultados bastante decentes.

Una vez terminados estos modelos para la fecha establecida, procedimos a enviar dichos proyectos hacia el comité de evaluación de la competición.

Task 1 English

	Team	F1
1	FactFinders	0.802
2	teamopenfact	0.796
3	innavogel	0.780
4	mjmanas54	0.778
5	ZHAW_Students	0.771
6	SemanticCUETSync	0.763
7	SINAI	0.761
8	DSHacker	0.760
9	visty	0.753
10	Fired_from_NLP	0.745
11	TurQUaz	0.718
12	hybrinfox	0.711
13	SSN-NLP	0.706
14	sz06571	0.696
15	NapierNLP	0.675
16	Mirela	0.658
17	Kushal_Chandani	0.658
18	DataBees	0.619
19	Trio_Titans	0.600
20	Madussree	0.583
21	pandas	0.579
22	JUNLP	0.541
23	mariuxi	0.517
24	grig95	0.497
25	CLaC-2	0.494
26	Aqua_Wave	0.339
27	**Baseline**	0.307

Figura 5.1: Gráfica de comparación de modelo BERT con estrategia de búsqueda hiper parámetros tarea 2 (Fuente: Clasificación de la subtarea 1 | Checkthat!)

Una vez evaluados nuestros modelos con detenimiento los resultados obtenidos en la competición a nivel mundial fueron bastante optimistas:

En la primera tarea en la que participamos obtuvimos una clasificación en 7° lugar entre 27 participantes con un resultado de predicción del modelo de 76%, mientras que en en la segunda tarea nos encontramos en la 6° posición de entre 15 desarrolladores, en el cual el porcentaje de predicción de nuestro modelo rondaba el 70%.

Estos resultados fueron bastante buenos debido a que mi experiencia en esta rama de la informática era muy corta, al igual que la investigación y desarrollo en cualquier ámbito informático, sobre todo dentro del mundo del procesamiento del lenguaje natural, por lo que el desarrollo de nuestros modelos fue todo un éxito.

English

	Team	Macro F1	SUBJ F1
1	Hybrinfox	0.7442	0.6
2	ToniRodriguez	0.7372	0.58
3	SSN-NLP	0.7120	0.54
4	Checker Hacker	0.7081	0.54
5	JK_PCIC_UNAM	0.7079	0.55
6	SINAI	0.7035	0.53
7	FactFinders	0.6955	0.51
8	Vigilantes	0.6955	0.52
8	eevvgg	0.6955	0.52
9	nullpointer	0.6893	0.54
10	Indigo	0.6388	0.47
11	(baseline)	0.6346	0.45
12	SemanticCuetSync	0.6265	0.43
13	JUNLP	0.5598	0.36
14	CLaC-2	0.4500	0.37
15	IAI Group	0.4491	0.39

Figura 5.2: Gráfica de comparación de modelo BERT con estrategia de búsqueda hiper parámetros tarea 2 (Fuente: CheckThat! subtarea 2 | Lab at CLEF 2024)

En cuanto a la tarea 2, como observamos en la imagen anterior los resultados fueron muy semejantes, con un resultado final en torno al 0.7 de la métrica de macro_f1 lo que hace indicar que ambos modelos son consistentes.

Finalmente después de las evaluaciones obtenidas en nuestra investigación las cuales eran valores mayormente superiores, los valores bajaron un poco en el conjunto final de testeo de la competición, pero fueron buenos resultados, ya que nos quedamos en ambas tareas a tan solo una mejora de 0.04 puntos de clasificar como 1º.

4.3.1. Artículo sobre la investigación

Como fruto del proceso de desarrollo de nuestro trabajo, se ha redactado un artículo de investigación con toda la información acerca del mismo, y este ha sido aceptado para su publicación por la propia campaña del CLEF 2024, el día 1 de julio de 2024. Dicho artículo será accesible para cualquier persona que quiera realizar un aprendizaje sobre nuestro proyecto en la propia web del CLEF 2024.

Debido a que en el momento de escribir este trabajo, aún no está disponible, se adjunta como anexo I el artículo aceptado.

5. Conclusiones

Como hemos podido ver a lo largo de este proyecto el hecho de la existencia de las falsas informaciones existe prácticamente desde el inicio de la comunicación, lo cual lo hace algo inevitable dentro de la tecnología actual gracias a la gran cantidad de medios y facilidades para el intercambio de información entre usuarios.

Pero gracias al desarrollo de las nuevas tecnologías, somos capaces de tratar de resolver este tipo de problemas, además día a día se consiguen mejorar o desarrollar nuevas herramientas para obtener resultados cada vez más cercanos a la perfección.

Gracias a este trabajo de fin de grado hemos podido observar en primera persona cómo es el desarrollo de estos sistemas computacionales capaces de simplificar tareas o incluso realizarlas favoreciendo al ser humano.

Con el desarrollo de todos los modelos de este trabajo hemos podido analizar muchos elementos dentro del mismo. Primeramente con el desarrollo de los

algoritmos de aprendizaje automático vimos como es imprescindible un buen preprocesamiento del conjunto de datos de entrada para poder obtener unos buenos resultados dentro de nuestro modelo y a partir de ellos seleccionar cuales de ellos presentaban los mejores resultados, como es el caso de SVM para la primera tarea o SGD para la segunda, de los cuales analizando posteriormente su eficacia en conjunto y su fiabilidad podemos determinar a SGD como el algoritmo más completo de los examinados.

Pero no fue hasta el desarrollo de aquellos modelos del lenguaje que surgieron como sucesores de estos anteriores algoritmos de aprendizaje automático en los que observamos como los resultados no solo mejoran en todas las métricas evaluadas, si no que al contrario que los anteriores, estos mantienen una fiabilidad y una continuidad en sus resultados, lo cual es cada vez más requerido por los nuevos problemas del mundo actual. En cuanto a este tipo de aprendizaje, el aprendizaje profundo, después de las pruebas con varios de ellos y su posterior análisis de resultados, conseguimos decantarnos por dos de ellos los cuales eran mayormente superiores al resto estableciendo a “RoBERTa base BNE” como mejor modelo para la tarea 1 en español, y al modelo original “BERT” como mejor modelo para la tarea 2, ambos con resultados muy buenos con el entrenamiento inicial.

Seguidamente del descubrimiento y desarrollo de estos nuevos modelos, tratamos de conseguir unos modelos con mejores resultados aún, con la aplicación de distintas estrategias de mejora como la aumentación de datos con traducción automática o la búsqueda de la polaridad en las secuencias de texto las cuales por distintos motivos no fueron efectivas. Al contrario que estas si encontramos una técnica capaz de aumentar la eficiencia como fue la búsqueda de hiper parámetros la cual consiguió mejorar ambos modelos en un porcentaje alto con respecto al modelo por defecto.

Después de muchos contratiempos durante el desarrollo de nuestro trabajo, entre ellos la anulación de una de las tareas de la competición en la que participabamos **[3]**, con la consecuencia del desarrollo de un 3º modelo del lenguaje en un periodo de tiempo bastante corto, el cual conseguimos desarrollar sobresalientemente pese a las circunstancias, finalmente hemos desarrollado un proyecto de fin de grado muy interesante e innovador del cual estamos orgullosos y pese a que la perfección de los modelos de aprendizaje, incluyendo los nuestros, todavía está muy lejos y con mucha investigación y desarrollo por delante, podemos

encontrar diferentes tareas en las que podemos mejorar la seguridad y velar por gran porcentaje de toda la información con intenciones dañinas que pueden causar dentro del mundo actual, como en nuestro caso, ayudar a la prevención de una sociedad de la desinformación.

6. Trabajos Futuros

Una vez terminado este trabajo de fin de grado el cual hemos conseguido resolver con unos resultados gratamente buenos, podemos pensar en cómo podemos mejorar aún más estos modelos o cómo podemos conseguir que los resultados lleguen a su máximo posible.

Para ello podemos pensar primeramente en nuevas estrategias de mejora de nuestros modelos que como sabemos después de nuestros análisis que los algoritmos de aprendizaje profundo destacan por encima del resto, con diferentes técnicas como inyecciones de ruido para aumentar su robustez o las llamadas técnicas de regularización como “Dropout” o la “Regularización L2” que pueden prevenir el sobreajuste y haciendo una red más dispersa, ambas serían claros candidatos de nuestros modelos, ya que encajan perfectamente dentro de las debilidades de nuestros modelos **[27]**.

También podemos pensar en un enfoque diferente utilizando otros modelos de aprendizaje profundo como pueden ser los derivados de Llama o los nacidos a partir de la rama de GPT que pese a centrarse en otras tareas podrían presentar buenos resultados en tareas de clasificación como en nuestro proyecto.

Como ya sabemos en nuestro trabajo no hemos podido aplicar todas las técnicas posibles, debido a la infinidad de ellas, pero en un posible desarrollo futuro y una continuación de este trabajo, seguramente llegaríamos a obtener incluso unos modelos más preparados con mejores resultados, que como indicamos anteriormente, esta rama de investigación está en constante desarrollo y cada día nos invita a mejorar un poco más dentro de ella.

7. Bibliografía

[1] Pennycook, G., Epstein, Z., Mosleh, M., Arechar, A. A., Eckles, D., & Rand, D. G. (2021). Shifting attention to accuracy can reduce misinformation online. Nature, 592(7855), 590-595. https://doi.org/10.1038/s41586-021-03344-2

[2] BBC News Mundo. Qué fue el Gran Engaño de la Luna y por qué tantos creyeron una fantasía tan extravagante. (2022). https://www.bbc.com/mundo/noticias-63519952

[3] CLEF 2024. Conference and Labs of the Evaluation Forum. (consultado el 06/06/2024). https://clef2024.imag.fr/

[4]El tutorial de Python. (consultado el 07/06/2024). Python Documentation. https://docs.python.org/es/3/tutorial/

[5] BBC News Mundo. Six Degrees: cómo fue y quién creó la primera red social de internet, inspirada por la teoría de los «seis grados». (2019). https://www.bbc.com/mundo/noticias-48558989

[6] Organización de Consumidores y Usuarios (OCU), Quiénes somos: conoce OCU, (consultado el 02/06/2024), https://www.ocu.org/info/quienes-somos

[7] Llisterri, J. (consultado el 10/06/2024). Tecnologías lingüísticas: los recursos lingüísticos. https://joaquimllisterri.cat/language_technology/HLT/tecnol_ling_recursos.html

[8] INCIBE. ¿Bulos y noticias falsas? No te creas todo lo que lees (2024). https://www.incibe.es/ciudadania/blog/bulos-y-noticias-falsas-no-te-creas-todo-lo-que-lees

[9] Lope, P. Cómo funciona el test de Turing y cuándo se ha superado (2024). https://www.mundodeportivo.com/urbantecno/ciencia/como-funciona-el-test-de-turing-y-cuando-se-ha-superado

[10] Master en Big data de la Universidad de Málaga. Los primeros programas de IA capaces de aprender. (2024). https://www.bigdata.uma.es/los-primeros-programas-de-ia-capaces-de-aprender/#:~:text=En%20el%20a%C3%B1o%201959%2C%20Arthur,parecido%20a%20las%20damas%20espa%C3%B1olas

[11] Arrabales, R. Deep Learning: qué es y por qué va a ser una tecnología clave en el futuro de la inteligencia artificial. Xataka. (2016). https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial

[12] Tania Lorenzo. Fake News a lo largo de la historia | Espacio Fundación Telefónica. Espacio Fundación Telefónica | Fundación Telefónica. (2023). https://espacio.fundaciontelefonica.com/noticia/fake-news-a-lo-largo-de-la-historia/

[13] Fernández, E. C. Conoce la historia del machine learning: ¡desde sus inicios! Tokio School. (2024). https://www.tokioschool.com/noticias/historia-machine-learning/

[14] Milo, A. Qué es la teoría del valle inquietante y cómo explica el miedo a los robots. National Geographic En Español. (2023). https://www.ngenespanol.com/ciencia/que-es-la-teoria-del-valle-inquietante-y-como-explica-el-miedo-a-los-robots/

[15] NLTK. NLTK ::nltk.tokenize.TreebankWordTokenizer. (2023). https://www.nltk.org/api/nltk.tokenize.TreebankWordTokenizer.html

[16] Bescos, C. Todo sobre BERT, el modelo lingüístico más avanzado para NLP a día de hoy. IBIDEM EDU. (2024). https://www.ibidemgroup.com/edu/bert-nlp-machine-translation/

[17] Pérez, A. El algoritmo SVM y sus aplicaciones empresariales. OBS Business School. (2024). https://www.obsbusiness.school/blog/el-algoritmo-svm-y-sus-aplicaciones-empresariales

[18] IBM, ¿Qué es el descenso de gradiente? . Consultado el 18/06/2024). https://www.ibm.com/es-es/topics/gradient-descent

[19] Daniel. Random Forest: Bosque aleatorio. Definición y funcionamiento. Formación En Ciencia de Datos | DataScientest.com. (2023). https://datascientest.com/es/random-forest-bosque-aleatorio-definicion-y-funcionamiento

[20] Buitrago,B, Regresión logística en machine learning, (2021), https://medium.com/iwannabedatadriven/regresi%C3%B3n-log%C3%ADstica-i-machine-learning-84ffe9d6be15

[21] AprendeMachineLearning.com. Qué es overfitting y underfitting y cómo solucionarlo | Aprende Machine Learning. Aprende Machine Learning. (2017). https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/

[22] Equipo editorial de IONOS. ¿Qué es el test de Turing? Explicación de su definición y función. IONOS Digital Guide. (2022). https://www.ionos.es/digitalguide/online-marketing/analisis-web/test-de-turing/?ac=OM.WE.WEo56K430308T7073a&gad_source=1&gclid=CjwKCAjw-O6zBhASEiwAOHeGxaew0mJBOlmayCrTiR1A7yrXQ8N5vYdLendfXeQJlMnSIVxDmrvAxhoC5FsQAvD_BwE&itc=76D5IUBG-TH53Y8-H7NV6TK&utm_campaign=DIS-ES-BRA-BRAX-PMA-COP-TVC_5---&utm_content=SMB_Mix_CM_VA&utm_medium=display&utm_source=brand&utm_term=TVC_5_MYW_Mix

[23] Papers with Code - Fake News Detection. Consultado el 19/06/2024). https://paperswithcode.com/task/fake-news-detection

[24] RTVE, Las noticias falsas en Twitter corren un 70% más rápido que las verídicas, según un estudio, (2018), https://www.rtve.es/noticias/20180309/noticias-falsas-twitter-corren-70-mas-rapido-veridicas-segun-estudio/1692780.shtml

[25] De Ingeniería del Conocimiento, I. Investigación sobre la generación de noticias falsas de la COVID-19. Instituto de Ingeniería del Conocimiento. (2024). https://www.iic.uam.es/innovacion/investigacion-generacion-noticias-falsas-covid19/

[26] Garcia-Mila, P. Detectando fake news con IA: herramientas y técnicas para la verificación de información. Founderz Blog | Últimas novedades en Innovación y Tecnología. (2024). https://founderz.com/blog/es/fake-news-deteccion-ia/

[27] Gutiérrez, J. P. Comprendiendo la Robustez en el Aprendizaje Profundo. (2024). https://es.linkedin.com/pulse/comprendiendo-la-robustez-en-el-aprendizaje-profundo-jordi-qza7f

[28] Gutiérrez-Fandiño, A., Armengol-Estapé, J., Pàmies, M., Llop-Palao, J., Silveira-Ocampo, J., Carrino, C. P., ... & Villegas, M. (2021). Maria: Spanish language models. arXiv preprint arXiv:2107.07253. http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6405

Anexo I. Artículo científico

SINAI at CheckThat! 2024: Transformer-based approaches for Check-Worthiness Classification

Notebook for the CheckThat! Lab at CLEF 2024

Sergiu Stoia[1,*], Jaime Collado-Montañez[1], Cristian Ibáñez-Bautista[1], Arturo Montejo-Ráez[1], María Teresa Martín-Valdivia[1] and Manuel Carlos Díaz-Galiano[1]

[1]Department of Computer Science (University of Jaén), Campus Las Lagunillas, s/n, Jaén, 23071, Spain

Abstract

This paper discusses the participation of the SINAI team in the CLEF-2024 CheckThat! lab Task 1 for English. The task involves assessing whether claims extracted from transcribed texts should be fact-checked. We explored two approaches to address this challenge: adjusting a Transformers-based model and using prompting-based techniques. In order to address imbalances within the data provided by the organizers, the method of class weighting is employed. Our best-performing system achieved an F1 score of 0.761 for the positive class and was ranked seventh among all twenty-six submissions in the competition.

Keywords

Fact-checking, Transformers, LLM, Fine-tuning, Prompting, Data augmentation

1. Introduction

In the era of information overload, the ability to efficiently identify and verify potentially misleading or false claims is paramount. Fact-checking has become a critical task to ensure the integrity of information disseminated across various platforms. The CheckThat! Task 1 [1] aims to advance research in this domain by focusing on the initial step of the fact-checking pipeline: determining which claims in a text are worth fact-checking.

Our participation in the lab was focused on the English dataset, where the objective was to develop methods to automatically assess the check-worthiness of claims within transcriptions of debates and speeches. We explored two distinct approaches to tackle this challenge: fine-tuning a transformer-based model [2] and leveraging the capabilities of large language models (LLMs) through prompting.

This paper presents the contribution of the SINAI team to the CheckThat! Lab. We analyze the strengths and limitations of each method, providing insights into their potential for enhancing automated fact-checking systems. Our findings contribute to the ongoing efforts [3] to refine the process of identifying claims that merit verification, ultimately supporting the broader goal of combating misinformation.

The remainder of the paper is organized as follows. Section 2 analyses the data provided by the organizers. Section 3 presents a description of the two different approaches developed for this lab. Section 4 shows the results obtained with such approaches and Section 5 summarizes our conclusion and future research directions.

2. The CheckThat! task

The CheckThat! task proposed in the CLEF forum [4, 5] is to determine whether a claim in a tweet or transcription is worth fact-checking. Traditionally, this decision involves judgments from professional

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France

*Corresponding author.

sstoia@ujaen.es (S. Stoia); jcollado@ujaen.es (J. Collado-Montañez); cib00005@red.ujaen.es (C. Ibáñez-Bautista); amontejo@ujaen.es (A. Montejo-Ráez); maite@ujaen.es (M. T. Martín-Valdivia); mcdiaz@ujaen.es (M. C. Díaz-Galiano)

0009-0009-2599-2820 (S. Stoia); 0000-0002-9672-6740 (J. Collado-Montañez); 0000-0002-8643-2714 (A. Montejo-Ráez); 0000-0002-2874-0401 (M. T. Martín-Valdivia); 0000-0001-9298-1376 (M. C. Díaz-Galiano)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Table 1
Class imbalance for each split

Split	No	Yes
english_train	75.94 %	24.06 %
english_dev	76.94 %	23,06 %
english_dev-test	66.04 %	33.96 %
english_test	74.19 %	25.81 %

fact-checkers or human annotators answering auxiliary questions like "does it contain a verifiable factual claim?" and "is it harmful?" before assigning a check-worthiness label. This year, the task uses multi-genre data and requires judgments based solely on the text. It is available in Arabic, English, and Spanish.

3. Data

For the task, the organisers provide multi-genre textual data in Arabic, Dutch, Spanish and English taken from tweets and transcriptions [6, 1]. In this section we present a brief analysis regarding the selected datasets used for the development of Task 1.

In order to feed the data to the systems described in Section 3, an analysis of the data obtained for each language was necessary, showing notable distinctions among the different languages. Whilst the Arabic and Dutch sets only contain texts sourced from tweets, those in English are based on transcriptions, whereas the Spanish collection comprises texts from both data sources. In order to use data that accurately reflects the desired outcome, the sets in Arabic and Dutch have been discarded, and only the texts from the Spanish set based on transcriptions have been chosen to increase the amount of available data.

In supervised learning tasks, analyzing the class proportion within the dataset is crucial, since the distribution of classes can significantly impact the learning algorithm. An analysis of this ratio has been conducted for each data split, revealing a clear imbalance among classes 1.

Sequence length is an important feature to take into account since most transformer-based models have a limited amount of tokens per sequence to be trained on. Thus, we used the Tiktoken[1] library from OpenAI to calculate the average length of the provided texts. Figure 1 shows an histogram that reveals that most sequences contain less than a hundred tokens, which is short enough for most state-of-the-art models. The average tokens in the dataset is 21.06 and the standard deviation 14.38.

4. System description

In this section, we describe the different approaches presented for the official evaluation of the first task.

4.1. Transformer fine-tuning

Leveraging the power of models based on Transformers like RoBERTa-base [7] has become a prevalent approach for achieving robust and accurate results for classification tasks. When applied to binary classification tasks, RoBERTa-base demonstrates remarkable performance by capturing intricate patterns and semantic nuances within the text data.

As previously mentioned, the imbalance among classes in the data might negatively impact the performance of this fine-tuning. To address the class proportion disparities, the method of class weighting is employed. This technique rectifies imbalances within datasets by assigning greater importance to underrepresented classes, leading to more equitable and accurate predictions. This value is calculated as the ratio of negative samples to positive samples.

[1] https://github.com/openai/tiktoken

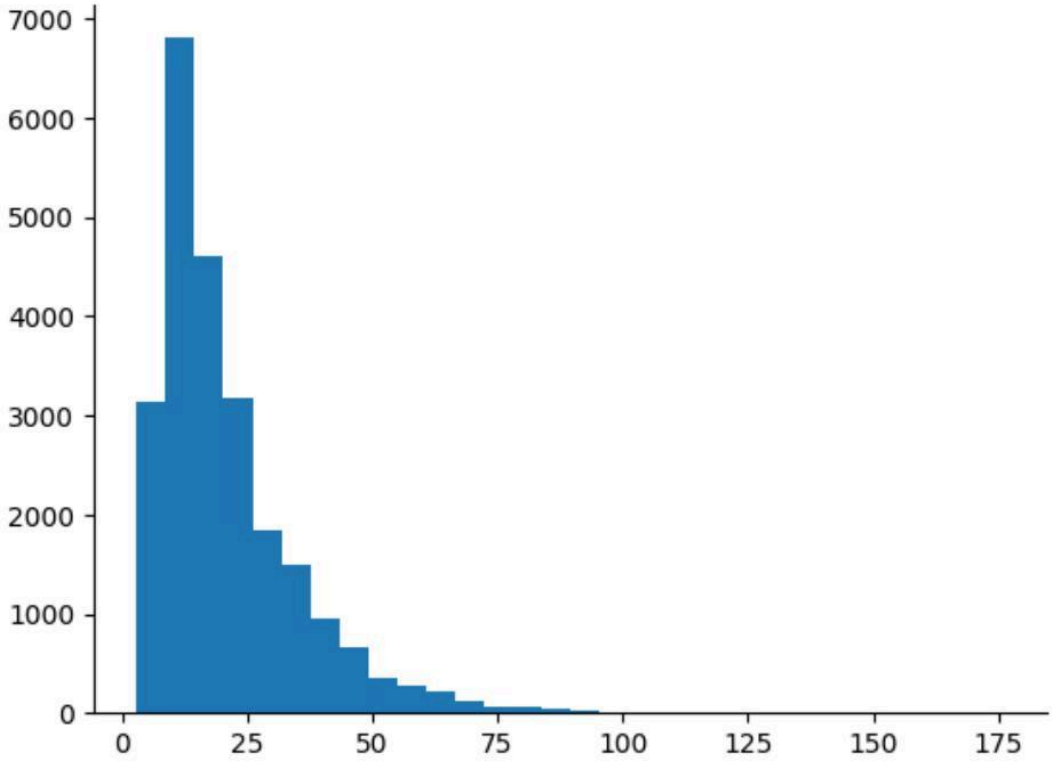

Figure 1: Token count histogram.

Table 2
Finetuning evaluation with dev-test set

Experiment	F1 positive class	macro precision	macro recall	macro F1
RoBERTa fine-tuning with original set	0.8965	0.9408	0.9117	0.9240
RoBERTa fine-tuning with augmented set	0.8899	0.9416	0.9048	0.9197

Two finetunings were conducted using the same base model. The first finetuning was performed exclusively with the original English dataset, whereas the second expanded the original training data by incorporating translated texts from the Spanish set. These experiments were performed with a learning rate of 1e-06 and a batch size of 8, using both train and dev sets combined as training data and dev-test for evaluating the checkpoints obtained during the execution. To safeguard against overfitting and unnecessary training cycles, the strategy of early-stopping was implemented. This technique stopped the training process if the model shows no improvement over three consecutive epochs. The results from the best checkpoints of both experiments showed minimal differences 2, obtaining F1 score of 0.896 for the positive class by using the original dataset.

4.2. LLM prompting

In addition to text and class labels, the organizers provided a sentence_id field that can be sorted to get consecutive sentences in a longer paragraph transcription. An example of this is shown in Table 3 where all the sentences belong to the same intervention from a presidential debate between candidates George Bush and Michael Dukakis in September 1988[2]:

[2]https://www.presidency.ucsb.edu/documents/presidential-debate-winston-salem-north-carolina

Table 3
Consecutive sentences

Sentence_id	Text	class_label
16	I think we've seen a deterioration of values.	No
17	I think for a while as a nation we condoned those things we should have condemned.	No
18	For a while, as I recall, it even seems to me that there was talk of legalizing or decriminalizing marijuana and other drugs, and I think that's all wrong.	No
19	So we've seen a deterioration in values, and one of the things that I think we should do about it in terms of cause is to instill values into the young people in our schools.	No
...	...	...

Table 4
Duplicated texts with different class_label

Sentence_id	Text	class_label
40	We've been dealing with him; he's been dealing drugs to our kids.	Yes
64	We've been dealing with him; he's been dealing drugs to our kids.	No
19096	We are better off than we were four years ago.	Yes
20136	We are better off than we were four years ago.	No
27908	That will not help us compete with China.	No
27910	That will not help us compete with China.	Yes
29254	I do not say that.	Yes
29256	I do not say that.	No
34258	We don't know who the rebels are.	Yes
34260	We don't know who the rebels are.	No

[BUSH:] *"I think we've seen a deterioration of values. I think for a while as a nation we condoned those things we should have condemned. For a while, as I recall, it even seems to me that there was talk of legalizing or decriminalizing marijuana and other drugs, and I think that's all wrong. So we've seen a deterioration in values, and one of the things that I think we should do about it in terms of cause is to instill values into the young people in our schools. We got away, we got into this feeling that value- free education was the thing."*

Further analysis highlights the need of utilizing context to determine the model prediction as there are similar sentences labeled differently as shown in Table 4.

With this in mind, we took two different prompting approaches, both with GPT-3.5-turbo [8]: a baseline prompt where no context is provided and another one where we concatenate all messages previous to the one being predicted:

1. The no-context prompt utilized is: *Check-worthiness definition: The process of finding whether a given TEXT contains verifiable factual claims prone to be fact-checked. You are an expert in check-worthiness. Determine if the TEXT contains a verifiable factual claim that is subject to fact-checking. Before responding, systematically consider these auxiliary questions: "Does it contain a verifiable factual claim?" and "Is it harmful?" Respond only with Yes if the TEXT contains such a claim, and No if it does not.\nTEXT:<Text>*
2. The context prompt utilized is: *Check-worthiness definition: The process of finding whether a given TEXT contains verifiable factual claims prone to be fact-checked. You are an expert in check-worthiness. You will be provided with several sentences but only the last one is relevant to the task; the rest of the text is only provided as extra context if needed. Before responding, systematically consider these auxiliary questions: "Does it contain a verifiable factual claim?" and "Is it harmful?"*

Table 5
Prompting evaluation with dev-test set for the positive class.

Experiment	Accuracy	Precision	Recall	F1
Prompting without context	0.8270	0.8354	0.6111	0.7059
Prompting with context	0.7390	0.7778	0.3241	0.4575

Table 6
Evaluation with test set

Experiment	F1 positive class
* RoBERTa fine-tuning with original set	0.7613
RoBERTa fine-tuning with augmented set	0.7305
Prompting without context	0.6233
Prompting with context	0.4647

Respond with either Yes or No based solely on your analysis of the last sentence. Yes means the last sentence should be fact-checked, No means it shouldn't.\nTEXT:<Text>

The results obtained with both approaches in the dev-test dataset for the positive class are shown in Table 5. Prompting with context underperformed significantly with respect to the no-context version, which scored an F1-score of 0.7059.

5. Results

In this section, we report the results obtained during the evaluation cycle. As only the last submission was selected, the best result from all the experiments we performed was submitted. This result corresponds to the RoBERTa-base fine-tuning approach using the original dataset 2.

Surprisingly, the metrics obtained using the gold labels provided by the organizers 6 show a considerable drop in the F1 score of the positive class compared to the results we obtained with the dev-test set.

The fine-tuned models showcased superior efficacy in determining which texts are worth fact-checking, as it is reflected in the F1 score of the positive class. This is due to the fundamental difference in training processes: while fine-tuning adjusts the model parameters to a specific task through iterative learning, the prompting approach relies solely on inference without any training process. This crucial distinction underpins the disparities in performance observed between the two methods. The fine-tuning process allows the model to adapt and specialize, leveraging task-specific information and fine-grained adjustments.

6. Conclusions and future work

In this paper we presented the systems developed by the SINAI team at the CheckThat! Lab to tackle Task 1: Check-worthiness detection. We compare two different approaches: A RoBERTa-based finetuning with and without data augmentation from the Spanish dataset provided by the organizers, and GPT-3.5-turbo prompting approaches including a baseline and a context-aware prompt. Results show transformer finetuning as a promising technique to tackle this task as we ranked 7th out of 26 participants scoring 0.761363 F1-score for the positive class.

Regarding future work, we aim to use balancing techniques such as downsampling of the majority class or data augmentation with external resources. We would also like to further explore the context-aware prompting approach as we believe extra context is important given the duplicated examples with different labels, even though it did not achieve good results the way we applied it. Other prompting

techniques such as few-shot learning [9] and chain of thought [10] could be useful to tackle this task too.

Acknowledgments

This work has been partially supported by projects CONSENSO (PID2021-122263OB-C21), MODERATES (TED2021-130145B-I00), SocialTOX (PDC2022-133146-C21) funded by Plan Nacional I+D+i from the Spanish Government.

References

[1] M. Hasanain, R. Suwaileh, S. Weering, C. Li, T. Caselli, W. Zaghouani, A. Barrón-Cedeño, P. Nakov, F. Alam, Overview of the CLEF-2024 CheckThat! lab task 1 on check-worthiness estimation of multigenre content, in: G. Faggioli, N. Ferro, P. Galuščáková, A. García Seco de Herrera (Eds.), Working Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum, CLEF 2024, Grenoble, France, 2024.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[3] E. Lazarski, M. Al-Khassaweneh, C. Howard, Using nlp for fact checking: A survey, Designs 5 (2021) 42.

[4] A. Barrón-Cedeño, F. Alam, J. M. Struß, P. Nakov, T. Chakraborty, T. Elsayed, P. Przybyła, T. Caselli, G. Da San Martino, F. Haouari, C. Li, J. Piskorski, F. Ruggeri, X. Song, R. Suwaileh, Overview of the CLEF-2024 CheckThat! Lab: Check-worthiness, subjectivity, persuasion, roles, authorities and adversarial robustness, in: L. Goeuriot, P. Mulhem, G. Quénot, D. Schwab, L. Soulier, G. M. Di Nunzio, P. Galuščáková, A. García Seco de Herrera, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fifteenth International Conference of the CLEF Association (CLEF 2024), 2024.

[5] A. Barrón-Cedeño, F. Alam, T. Chakraborty, T. Elsayed, P. Nakov, P. Przybyła, J. M. Struß, F. Haouari, M. Hasanain, F. Ruggeri, X. Song, R. Suwaileh, The clef-2024 checkthat! lab: Check-worthiness, subjectivity, persuasion, roles, authorities, and adversarial robustness, in: N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, I. Ounis (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2024, pp. 449–458.

[6] F. Alam, S. Shaar, F. Dalvi, H. Sajjad, A. Nikolov, H. Mubarak, G. Da San Martino, A. Abdelali, N. Durrani, K. Darwish, et al., Fighting the covid-19 infodemic: Modeling the perspective of journalists, fact-checkers, social media platforms, policy makers, and the society, in: Findings of the Association for Computational Linguistics: EMNLP 2021, 2021, pp. 611–649.

[7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized BERT pretraining approach, CoRR abs/1907.11692 (2019). URL: http://arxiv.org/abs/1907.11692. arXiv:1907.11692.

[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[9] A. Parnami, M. Lee, Learning from few examples: A summary of approaches to few-shot learning, 2022. arXiv:2203.04291.

[10] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: S. Koyejo, S. Mohamed,

A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, volume 35, Curran Associates, Inc., 2022, pp. 24824–24837. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.