Universidad de Cádiz
Escuela Superior de Ingeniería

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

SISTEMA DE DETECCIÓN AUTOMÁTICA DE NOTICIAS FALSAS EN ESPAÑOL BASADO EN DEEP LEARNING

AUTORA: REBECA VILLALBA CALVILLO

Cádiz, diciembre 2024

Universidad de Cádiz
Escuela Superior de Ingeniería

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

SISTEMA DE DETECCIÓN AUTOMÁTICA DE NOTICIAS FALSAS EN ESPAÑOL BASADO EN DEEP LEARNING

DIRECTORA: MARÍA DE LA PAZ GUERRERO LEBRERO
CODIRECTORA: ELISA GUERRERO VÁZQUEZ
AUTORA: REBECA VILLALBA CALVILLO

Cádiz, diciembre 2024

DECLARACIÓN PERSONAL DE AUTORIA

Rebeca Villalba Calvillo con DNI 32087579A estudiante en la Escuela Superior de Ingeniería Informática de la Universidad de Cádiz, como autor de este documento académico titulado "Sistema de Detección Automática de Noticias Falsas en Español basado en Deep Learning" y presentado como Trabajo Final de Grado

DECLARO QUE

Es un trabajo original, que no copio ni utilizo parte de obra alguna sin mencionar de forma clara y precisa su origen tanto en el cuerpo del texto como en su bibliografía y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Asimismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse.

En Puerto Real, a 1 de diciembre de 2024

Fdo: Rebeca Villalba Calvillo

Agradecimientos

En primer lugar, quería agradecer a mis padres, Atanasio Villalba y Francisca Calvillo, y mi hermano Guillermo la educación y los valores que me han transmitido a lo largo de los años. También el constante e innumerable apoyo que he recibido por parte de ellos, y por parte de mi pareja Gabriel, especialmente durante los años del grado.

En segundo lugar, me gustaría agradecer a María de la Paz y Elisa por todas las reuniones, la disponibilidad, paciencia y apoyo que han tenido conmigo. Sin su guía no hubiese podido hacer un trabajo tan completo y correcto como lo ha sido.

Por último, y no menos importante, a los amigos que he tenido durante el grado, y en especial a mis dos mejores amigas, María Vázquez y Marina Utrera. A las cuales, les estaré siempre agradecida por la convivencia que he tenido con ellas y por los tan buenos momentos que hemos pasado juntas.

Resumen

En este proyecto, se ha llevado a cabo el desarrollo de un sistema de detección automática de noticias falsas en español mediante el uso de técnicas avanzadas de procesamiento del lenguaje natural (PLN) y modelos de Deep Learning. A través de este enfoque, se ha buscado abordar uno de los principales retos actuales: la propagación de fake news, especialmente en el ámbito hispanohablante, donde la desinformación puede tener efectos considerables en la opinión pública y la estabilidad.

La creciente presencia de noticias falsas, difundidas rápidamente a través de redes sociales y medios digitales, subraya la necesidad de contar con herramientas que permitan identificar la veracidad de la información de manera automática.

Para lograr este objetivo, se ha implementado la técnica fine-tunning en modelos preentrenados basados en RoBERTa, utilizando diversos optimizadores como Adam, Adamax y AdamW. Tras los experimentos realizados y el análisis de los resultados, el mejor modelo combinado con AdamW demostró ser el más eficaz obteniendo un F1-Score del 85,3% lo que supone, un rendimiento superior a los demás optimizadores.

Se ha desarrollado una plataforma web interactiva basada en React, donde los usuarios pueden introducir el texto de la noticia para evaluar si es falsa o verdadera. Esta herramienta, con un diseño intuitivo y fácil de usar permite que cualquier usuario pueda interactuar con ella de manera cómoda y efectiva.

Índice

Índice de figuras

Índice de tablas

Índice de ecuaciones

Capítulo 1

Introducción

El procesamiento del lenguaje natural (PLN) es una rama de la inteligencia artificial dedicada a diseñar métodos y algoritmos que manejan lenguaje natural no estructurado [1]. El objetivo es que un ordenador sea capaz de generar y "entender" los contenidos de un texto para así poder interactuar con él. A continuación, se muestra en la figura 1.1 la posición del PLN dentro del campo de la inteligencia artificial (IA).

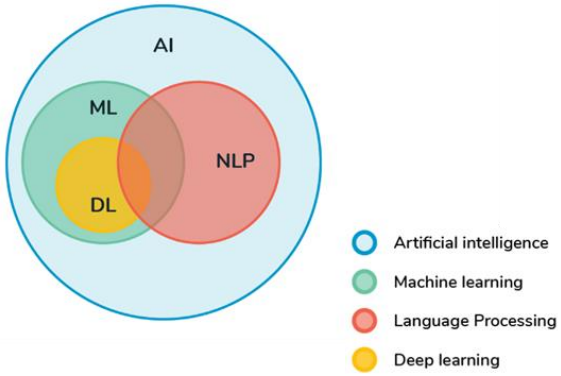


Figura 1.1. Diagrama PLN. Fuente: *[2]*

El PLN ha evolucionado significativamente desde sus inicios. En sus primeras etapas, los modelos como el bag-of-words (una técnica que representa el texto como una colección de palabras individuales sin tener en cuenta el orden ni el contexto en el que aparecen) y las técnicas basadas en n-grams (que dividen el texto en secuencias de palabras de tamaño fijo) ofrecían formas rudimentarias de representar el texto, pero eran limitados en su capacidad para comprender el contexto.

Con el tiempo, los modelos como Word2Vec introdujeron representaciones más sofisticadas de las palabras mediante embeddings, vectores que capturan el significado semántico de las palabras al representar relaciones como la similitud entre términos. Sin embargo, estos modelos seguían siendo limitados al no poder manejar contextos largos o complejos, ya que su enfoque se centraba principalmente en las palabras individuales y no en cómo estas interactúan en frases o textos completos.

El gran cambio llegó en 2017, cuando el artículo "Attention is All You Need" [3] introdujo los Transformers, un nuevo paradigma en el aprendizaje profundo. Estos modelos aprovecharon el mecanismo de atención, que permitía enfocarse en las partes más relevantes de una secuencia de texto, independientemente de la longitud.

Uno de los momentos más destacados en el campo del Procesamiento del Lenguaje Natural (PLN) en los últimos años ha sido el desarrollo del modelo GPT (Generative Pre-trained Transformer) por parte de OpenAI. Este avance demostró que, mediante el

preentrenamiento de un modelo en grandes volúmenes de datos textuales y su posterior ajuste para tareas específicas, es posible alcanzar un rendimiento comparable al nivel humano en una amplia variedad de aplicaciones de PLN. Entre estas aplicaciones destacan las siguientes:

• Reconocimiento de voz: Transformar una señal de audio que contiene el habla de una persona en texto escrito, permitiendo su análisis o procesamiento posterior.

• Text-to-speech: Realiza el proceso inverso al reconocimiento de voz, convirtiendo texto escrito en un audio sintetizado que imita el habla humana.

• Clasificación de texto: Asignar una o más categorías a un texto o documento en función de su contenido, facilitando tareas como la organización y la segmentación de información.

• Traducción automática: Traducir textos de un idioma a otro de manera autónoma, manteniendo la fidelidad semántica y el contexto del contenido original.

• Generación automática de texto: Crear texto coherente y relevante a partir de características o instrucciones específicas, como longitud, tono o contenido temático.

• Análisis de sentimientos: Una aplicación particular de la clasificación de texto en la que se identifica y etiqueta el sentimiento predominante en un texto, como positivo, negativo o neutral

Otra aplicación significativa del PLN reside en su capacidad para contribuir a la detección u diferenciado entre noticias falsas y verdaderas. Mediante el análisis avanzado de texto, los modelos PLN pueden identificar patrones lingüísticos, inconsistencias en los datos o fuentes, y características propias de desinformación, ofreciendo herramientas efectivas para combatir la propagación de información errónea.

1.1 Motivación

En plena era digital, el fenómeno de las noticias falsas, o "*fake news",* ha emergido como una poderosa fuerza que trasciende fronteras lingüísticas, influenciando de manera significativa la percepción pública, la toma de decisiones o la estabilidad democrática en todo el mundo. Este impacto se ve aumentado por la rapidez con la que la desinformación se propaga a través de las redes sociales y otros canales de información.

Uno de los principales objetivos de las *fake news* es la manipulación de la opinión de la sociedad para fines políticos o sociales. Actualmente, podemos apreciarlo con claridad con el conflicto entre Israel y Palestina, donde a diferencia de otros conflictos bélicos en el pasado las nuevas armas toman forma de 'post' o publicación cuya crudeza interpela directamente a los sentimientos de los usuarios [3].

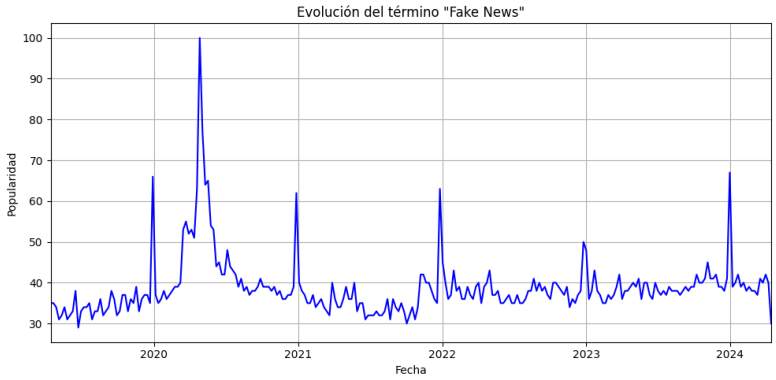


Figura 1.2. Evolución del término "Fake News" durante los años según Google Trends. Fuente: *[4]*

En la siguiente gráfica podemos observar el crecimiento de la popularidad del término *fake news,* destacando 3 escenarios históricos relevantes. El primero es en plena pandemia y se extiende hasta finales de 2021. El segundo en 2022, el conflicto entre Ucrania y Rusia. Por último, a finales de 2023, cuando el conflicto entre Israel y Palestina estaba en su punto álgido.

Aunque a día de hoy la mayoría de *fake news* que vemos provienen de usuarios anónimos con un bajo nivel cultural, detrás de la propagación de bulos y desinformación se encuentran partidos políticos o grandes empresas.

Un ejemplo claro fue el caso de *Cambridge Analytica*, una empresa que saltó a la fama por su supuesta participación en la campaña electoral presidencial de Estados Unidos en 2016 o con el Brexit en Europa. La propia empresa tiene en la portada de su página web el slogan "Data drives all we do. Cambridge Analytica uses data to change audience behaviour" [5].

En el siguiente artículo del periódico EL PAÍS se hace especial énfasis en la comunidad hispanohablante en Estados Unidos señalando que esta población es la más afectada por la desinformación en el país. Las traducciones al español pueden transformar un contexto en otro totalmente diferente, algo de lo que se aprovechan los medios de comunicación en la zona. Temas muy críticos como la salud pública o los derechos de inmigración son los que más preocupan a los habitantes hispanohablantes. [6]

Además, dentro del propio idioma español, existen variaciones regionales y culturales que pueden distorsionar el mensaje original, lo que incrementa la vulnerabilidad a la desinformación. Según este otro artículo del periódico EL PAÍS [7], un estudio realizado por el Instituto Reuters muestra que el desinterés por las noticias se ve agravado por la creciente dificultad para distinguir entre información confiable y manipulada, particularmente en plataformas digitales como TikTok y X.

La combinación de estas problemáticas incrementa la necesidad de sistemas automatizados capaces de detectar noticias falsas, no solo analizando el contenido textual, sino también el contexto cultural, regional y digital en el que se originan y difunden. Este enfoque es esencial para contrarrestar las distorsiones que afectan a una comunidad hispanohablante diversa global. Sin embargo, el desafío de distinguir entre noticias falsas y verdaderas se complica actualmente por la falta de tiempo. Cada día recibimos más información proveniente de diversos medios como redes sociales y medios de comunicación tradicionales entre otros.

Las redes sociales, en particular, han incrementado esta problemática al priorizar la viralidad y el impacto emocional por encima de la veracidad del contenido. Esto genera un entorno donde las noticias falsas o manipuladas se propagan con rapidez, y las personas no disponen del tiempo que se necesita para contrastar que la noticia que estamos leyendo es falsa o no.

Esta situación se agrava más si hablamos de las noticias en español. La comunidad hispanohablante se enfrenta a barreras adicionales en el acceso a información confiable en su propio idioma debido a que el idioma que predomina en estos casos es el inglés, lo que implica procesos de traducción y adaptación que, en ocasiones, tergiversan el mensaje original. Además, las propias diferencias culturales y regionales dentro del idioma español pueden generar confusión, dificultando aún más poder distinguir entre información veraz y desinformación.

La creación de un sistema automático de detección de *fake news* que se enfoque en el español se vuelve crucial. El objetivo es abordar el problema con el texto como el principal criterio para determinar la veracidad de la información. Esto se llevará a cabo desde la perspectiva del PLN, con el fin de analizar y evaluar la autenticidad del texto.

1.2 Objetivos

El objetivo principal de este trabajo fin de grado es el desarrollo de un sistema de detección de desinformación basado en un modelo de IA.

A continuación, se enumeran los objetivos específicos necesarios para la realización del objetivo principal:

1. **Analizar el estado del arte del problema**: Investigar el panorama actual en cuanto a la detección de desinformación, *Deep Learning* y PLN revisando las contribuciones más destacadas en la literatura.

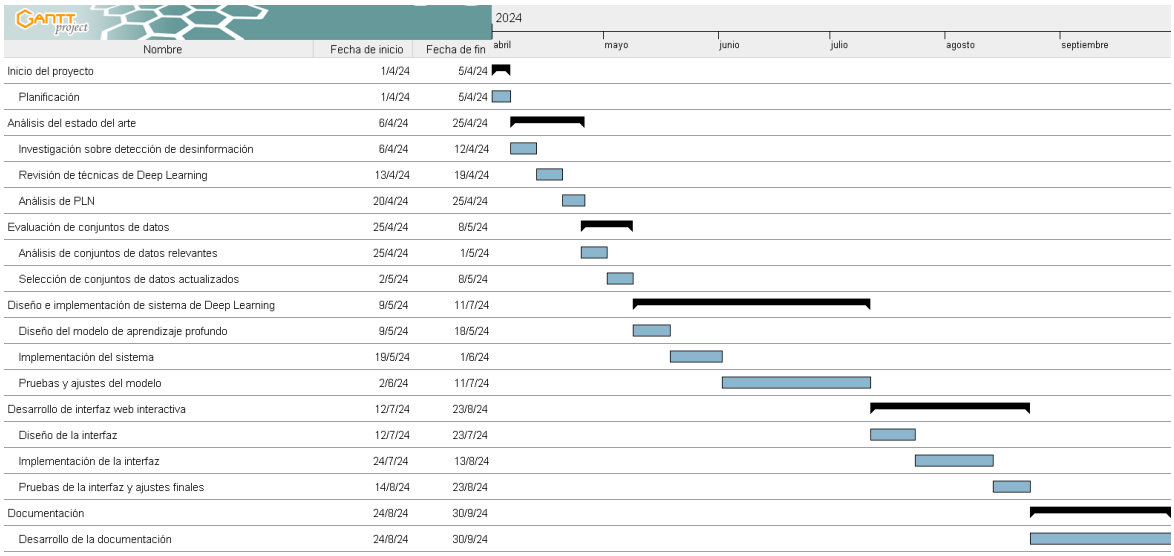
2. **Evaluar los conjuntos de datos más relevantes:** Analizar los conjuntos más relevantes y actualizados disponibles para abordar el problema de detección de noticias falsas.

3. **Diseñar e implementar un sistema de aprendizaje profundo:** Se desarrollará un sistema de *Deep Learning* que se enfoque en la detección de noticias falsas, utilizando exclusivamente el contenido textual de las noticias (título y cuerpo) sin depender de datos adicionales.

4. **Desarrollar una interfaz web interactiva:** Implementar una interfaz que permita visualizar y demostrar de manera efectiva el funcionamiento del sistema desarrollado.

1.3 Planificación

En la figura 1.3, se muestra la planificación temporal del proyecto, desarrollada con el software **GanttProject** [8]**.** La duración del proyecto comprende desde el 01/04/2024 hasta el 29/09/2024. El tiempo comprendido para realizar el proyecto será de 181 días. El proyecto estará dividido en 4 etapas bien diferenciadas: Inicio del proyecto, Análisis del estado del arte, Evaluación de conjuntos de datos, Diseño e implementación de sistema de *Deep Learning*, Desarrollo de interfaz web interactiva, Documentación.



Nombre	Fecha de inicio	Fecha de fin
Inicio del proyecto	1/4/24	5/4/24
Planificación	1/4/24	5/4/24
Análisis del estado del arte	6/4/24	25/4/24
Investigación sobre detección de desinformación	6/4/24	12/4/24
Revisión de técnicas de Deep Learning	13/4/24	19/4/24
Análisis de PLN	20/4/24	25/4/24
Evaluación de conjuntos de datos	25/4/24	8/5/24
Análisis de conjuntos de datos relevantes	25/4/24	1/5/24
Selección de conjuntos de datos actualizados	2/5/24	8/5/24
Diseño e implementación de sistema de Deep Learning	9/5/24	11/7/24
Diseño del modelo de aprendizaje profundo	9/5/24	18/5/24
Implementación del sistema	19/5/24	1/6/24
Pruebas y ajustes del modelo	2/6/24	11/7/24
Desarrollo de interfaz web interactiva	12/7/24	23/8/24
Diseño de la interfaz	12/7/24	23/7/24
Implementación de la interfaz	24/7/24	13/8/24
Pruebas de la interfaz y ajustes finales	14/8/24	23/8/24
Documentación	24/8/24	30/9/24
Desarrollo de la documentación	24/8/24	30/9/24

Figura 1.3. Diagrama de Gantt para la planificación del proyecto.

El proyecto se divide en diversas etapas que engloban un conjunto de tareas específicas. A continuación, se detalla qué actividades corresponden a cada etapa:

- **Inicio del proyecto:** Incluye la planificación inicial donde se define el cronograma y objetivos del trabajo.

- **Análisis del estado del arte:** Comprende actividades como la investigación sobre la detección de desinformación, la revisión de técnicas de *Deep Learning* y el análisis de PLB para establecer una base teórica sólida.

- **Evaluación de conjuntos de datos:** Abarca el análisis y selección de conjuntos de datos relevantes y actualizados que serán utilizados para entrenar el sistema de *Deep Learning*.

- **Diseño e implementación del sistema de *Deep Learning*:** Incluye el diseño del modelo, su desarrollo y los ajustes finales necesarios para optimizar su funcionamiento.
- **Desarrollo de la interfaz web interactiva:** Contempla el diseño, implementación y pruebas finales de una interfaz que permita a los usuarios interactuar con el sistema de manera intuitiva.

- **Documentación:** Consiste en registrar de todo el proceso, incluyendo informes técnicos y manuales de uso.

1.3 Alcance

El presente proyecto se enfoca en la creación de un sistema automático para la detección de noticias falsas en español, abordando el creciente desafío que representa a la desinformación en la comunidad hispanohablante. Se desarrollará una solución basada en PLN que analizará el contenido textual de las noticias, específicamente el título y el cuerpo, para evaluar su veracidad.

Este sistema no utilizará fuentes externas ni datos de redes sociales, centrando su análisis únicamente en el texto proporcionado. Además, el proyecto se limitará exclusivamente al tratamiento de noticias en español, asegurando que el enfoque esté adaptado a las particularidades lingüísticas y culturales de este idioma. Asimismo, se implementará una interfaz web interactiva para demostrar la funcionalidad del sistema y su efectividad en la detección y clasificación de noticias falsas.

1.4 Estructura

- Capítulo 1: Introducción. En este primer capítulo se presenta el tema del proyecto. Además, se detallará las ideas generales que harán posible el desarrollo del trabajo, como los objetivos a cumplir, su planificación y la metodología empleada.

- Capítulo 2: Desinformación y Procesamiento del Lenguaje Natural. Este capítulo aborda el concepto de desinformación y sus variantes, explorando cómo afecta al entorno digital. También se examina el campo del Procesamiento del lenguaje natural (PLN), cubriendo su aplicación en la detección de noticias falsas. Se incluyen definiciones de varios modelos PLN, como BERT y RoBERTa, fundamentales para el desarrollo del sistema propuesto.

- Capítulo 3: Detección de Fake News. Este capítulo se centra en el análisis de los conjuntos de datos disponibles para la detección de noticias falsas en español, identificado los más relevantes y actualizados. Se presentarán varios modelos base que servirán como referencia para el estudio. Finalmente, se introducirán los modelos propuestos y se discutirán los resultados obtenidos.

- Capítulo 4: Desarrollo de la aplicación web. Este capítulo describe el diseño de la aplicación web de detección de noticias falsas en español, incluyendo los requisitos, casos de uso y el diagrama de secuencia. También se detallarán las tecnologías empleadas en la implementación.

- Capítulo 5: Pruebas. Este capítulo se centrará en las pruebas estándar que se realizarán para asegurar el correcto funcionamiento de la aplicación web.

1.5 Presupuesto

Para estimar el presupuesto necesario para la realización de este proyecto, se han considerado los siguientes puntos presentados en la tabla 1.1.

Concepto	Cantidad	Total
Horas de trabajo	6 meses x 20 días x 8 horas	960 horas
Sueldo por hora	21.014,55€/año ÷ 12 meses ÷ 160 horas = 10,94€/h	10,94€/h x 960 horas = 10.402,40€
Suscripción Colab Pro (durante 3 meses)	11.19€/mes x 3 meses	33,57€
Coste portátil del alumno	Uso estimado de 6 meses sobre 4 años de vida útil (600€ de coste total)	75€
Total		10.610,97€
Total con IVA (21%)		12.839,26€

Tabla 1.1. Desglose presupuesto del proyecto

1. **Horas de trabajo**:

    - Cantidad: El proyecto se estima en 6 meses de trabajo, con una jornada de 20 días laborales al mes y 8 horas diarias, lo que da un total de 960 horas.

    - Sueldo por hora: Según el salario anual de un analista programador establecido en el BOE [9], el suelo anual es de 21.014,55 €. Dividiendo por 12 meses y 160 horas laborales mensuales, el coste por hora es de 10,94 €/h. Multiplicado por las 960 horas, el costo total es de 10.502,40 €.

2. **Suscripción a Colab Pro**:

    - Cantidad: Se requiere una suscripción a Google Colab Pro durante 3 meses, con un coste mensual de 11,19 €, lo que resulta en un gasto total de 33,57 €.

3. **Coste del portátil del alumno**:

    - Cantidad: El uso del portátil se amortiza considerando un coste inicial de 600 € con una vida útil de 4 años. Para 6 meses (1/8 del tiempo total de vida útil), el coste se calcula como 600 € ÷ 8 = 75 €.

Capítulo 2

Desinformación y Procesamiento del Lenguaje Natural

En este capítulo, se establece el marco teórico del trabajo de investigación, enfocado en el estudio y comprensión de las *fake news,* procesamiento del lenguaje natural, *Transformers,* BERT y RoBERTa.

2.1 Desinformación

La desinformación es un tipo de información falsa o parcialmente falsa, la cual se difunde de manera intencionada para manipular al lector. "Este fenómeno hace referencia tanto al contenido informativo fraudulento (*fake news),* al engañoso (*misleading content),* los discursos de odio (*hate speech),* los discursos falsos deliberados (*false speech)* o los errores informáticos no deliberados de medios o periodistas (*misinformation)"* [10].

- **Noticias Falsas (*Fake News*)**: "Se conocen como ***fake news*** al tipo de bulo que consiste en un contenido seudoperiodístico difundido a través de portales de noticias, prensa escrita, radio, televisión y redes sociales y que tiene como objetivo desinformar a un público en específico" [11].

- **Contenido Engañoso (*Misleading Content*):** Es el tipo de noticia más difícil de descubrir. El *misleading content* o contenido engañoso en español, puede aparecer en muchísimas noticias reales, y la razón por la que es tan difícil de descubrir es porque se requiere algún tipo de experiencia o conocimiento sobre un tema determinado para determinar si los hechos y los detales de cualquier artículo de noticias están siendo tergiversados [12].

- **Discurso de odio (*Hate Speech)*:** Discurso público que expresa odio o fomenta la violencia hacia una persona o grupo por motivos de raza, religión, sexo u orientación sexual [13].

- **Discurso Falso Deliberado (*False Speech)*:** Declaraciones o discursos intencionalmente falsos o engañosos pronunciados por individuos con la intención de manipular la opinión pública o alcanzar algún objetivo específico. Puede incluir declaraciones políticas falsas, promesas incumplidas o afirmaciones engañosas para obtener ventaja personal o política [10].

- **Desinformación (*Misinformation*):** Este término abarca todos los tipos de información incorrecta, ya sea intencional o no. Incluye errores involuntarios, información desactualizada o malinterpretada, así como contenido falso o engañoso

[14]. La desinformación puede propagarse fácilmente en línea y puede tener consecuencias significativas en la percepción pública y en la toma de decisiones.

2.2 Procesamiento del Lenguaje Natural

Como ya se comentó en la introducción del Capítulo 1, el PLN es uno de los campos de la inteligencia artificial que está teniendo mayor repercusión. En este apartado, se profundizará en los aspectos técnicos de este campo, desglosando los elementos claves a la hora de resolver un problema de este campo. Podemos distinguir dos aspectos esenciales:

- La parte "**lingüística**", que consiste en preprocesar y transformar la información de entrada en una serie de datos explotables. (contar algo de lingüística)
- La parte de "**aprendizaje automático**", que se basa en la aplicación de modelos de *Machine Learning* o *Deep Learning* a ese conjunto de datos.

La fase de preprocesamiento consiste en limpiar y preparar el texto para que un algoritmo PLN pueda analizarlo. Entre las principales etapas que se siguen, encontramos:

- **Limpieza del texto**: eliminar signos de puntuación, números, eliminar etiquetas HTML, símbolos, cambio a minúsculas.

- **Tokenización**: proceso de dividir la cadena de texto en unidades más pequeñas llamadas **tokens** usando el espacio como separador**.** Los tokens incluyen palabras, caracteres y subpalabras. Podemos observar un ejemplo de tokenización en la figura 2.1.

Figura 2.1. Ejemplo de tokenización de una frase. Fuente: Propia

- ***Stemming***: una misma palabra se puede encontrar de diferentes formas dependiendo del género, del número, la persona etc. El *stemming* designa el simple proceso heurístico de cortar el final de las palabras para mantener solo la raíz de la palabra. Para verlo más claro en la figura 2.2 tenemos un ejemplo de *stemming*

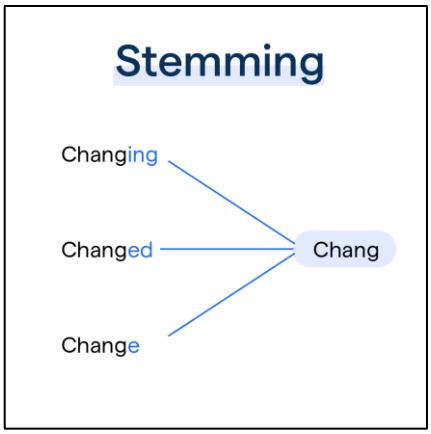


Figura 2.2. Ejemplo de *Stemming*. Fuente: *[15]*

- **Lematización**: consiste en realizar la misma tarea que el *stemming* pero utilizando un vocabulario y un análisis minucioso de la construcción de las palabras. La lematización permite eliminar únicamente las terminaciones inflexibles y de ese modo aislar la forma canónica de la palabra, es decir, el lema. Podemos ver un ejemplo de lematización en la figura 2.3.

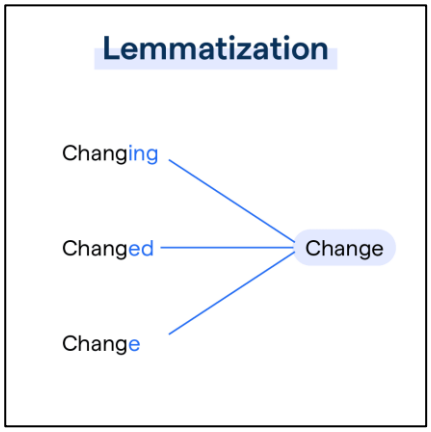


Figura 2.3. Ejemplo de lematización. Fuente: *[16]*

Para poder aplicar métodos de Machine Learning a los problemas relativos al lenguaje natural, es indispensable transformar los datos textuales en datos digitales.

Alguna de las técnicas clásicas de PLN utiliza el **tf-idf** (*Term Frequency-Inverse Document Frequency*) como entrada para modelos clásicos de *Machine Learning* como pueden ser las máquinas de vectores soporte y los árboles de decisión.

El TF-IDF [17] es una medida numérica que indica la relevancia de una palabra en un documento específico dentro de un conjunto de documentos. Este método se calcula contando las apariciones de cada palabra en un documento (*Term Frequency),* y luego ponderando esa frecuencia por la frecuencia inversa de documentos que contienen ese término (*Inverse Document Frequency).* La formulación matemática se ve presentada en la Ecuación 1:

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \quad (1)$$

Siendo ***tf (t, d)*** la frecuencia del término *t* en un documento *d* (ver Ecuación 2)*:*

$$\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (2)$$

Donde:

- $\boldsymbol{f_{t,d}}$ es el número de veces que aparece *t* en un documento *d.*
- $\sum_{t' \in d} f_{t',d}$ es el número total de términos en el documento *d*

Siendo $\boldsymbol{idf(t,D)}$ la frecuencia de documento inversa, es decir, una medida de cuanta información proporciona una palabra, sobre como de común o específica es en todos los documentos (ver Ecuación 3).

$$idf(t,D) = log\ \frac{N}{1+|d \in D : t \in d|} \quad (3)$$

Donde:

- *N* es el número total de documentos en el corpus, $N = |D|$
- $|d \in D : t \in d|$ es el número de documento en los que el término *t* aparece.

2.3 Aprendizaje Automático en PLN

Por tanto, el valor de *tf-idf* de las palabras de un texto, sirve como entrada para los modelos de *Machine Learning* tales como las Máquinas de Soporte Vectorial, Árboles de Decisión o Regresión Logística entre otros. Si bien estos modelos logran un buen resultado a la hora de resolver los problemas de PLN y más en concreto de clasificación de texto, el estado del arte viene marcado por distintos modelos basados en *Deep Learning*, entre los que destacan:

- Redes Neuronales Convolucionales, **CNN**
- Redes Neuronales Recurrentes, **RNN**
- Redes de memoria a corto-largo plazo, **LSTM**
- **Transformers y Modelos de lenguaje preentrenados.**

Las redes neuronales convolucionales son un modelo avanzado de aprendizaje profundo diseñado para analizar y procesar datos, especialmente imágenes 2D, utilizando operaciones de convolución para extraer automáticamente características relevantes y estructuradas a diferentes niveles de abstracción. Su arquitectura está inspirada en la organización del cerebro humano y utiliza capas convolucionales para identificar patrones y características jerárquicas [18].

En cuanto a las redes neuronales recurrentes, son modelos diseñados para procesar datos secuenciales aprovechando una memoria interna que almacena información del contexto previo. Aunque son útiles en tareas de PLN, presentan ciertas limitaciones al manejar dependencias a largo plazo debido a dos problemas, conocidos como gradientes de explosión y gradientes de desaparición. Durante el entrenamiento, los gradientes utilizados para actualizar los pesos de la red pueden volverse muy pequeños (desaparecen) o muy grandes (explotar), lo que dificulta la captura de relaciones entre palabras distantes en un texto. Además, su procesamiento secuencial las hace menos eficiente, lo que dificulta el procesamiento en paralelo [19].

Las redes de memoria a corto-largo plazo son un tipo de red neuronal recurrente diseñadas para resolver el problema los gradientes que "desaparecen" durante el entreno. Esto lo hace gracias a su arquitectura basada en celdas de memoria y puertas de control [20].

Con las celdas de memoria mantienen y actualizan información relevante durante largos periodos de tiempo. Al igual que las redes recurrentes, debido a su procesamiento secuencial no pueden capturar relaciones entre palabras distantes en un texto.

Los Transformers representan una arquitectura avanzada diseñada para superar las limitaciones de modelos como las CNN, RNN y LSTM. A través del mecanismo de atención, los Transformers son capaces de analizar relaciones globales entre palabras de forma eficiente, procesando todo el texto en paralelo y capturando dependencias a cualquier distancia dentro del texto [21].

Esto los hace ideales para tareas complejas como la detección de noticias falsas, donde es crucial comprender tanto el contexto local como el global del contenido. Dada su capacidad para manejar grandes volúmenes de datos y extraer patrones contextuales con mayor precisión, los Transformers son la elección más adecuada para este proyecto.

2.4 Métricas de Evaluación

Para poder medir el rendimiento de los modelos tras su entrenamiento, es de vital importancia hacer uso de ciertas métricas para evaluar si los modelos realizan un buen trabajo de predicción para nuevos datos.

Las métricas de evaluación dependen del tipo de aprendizaje que realiza el modelo, por ejemplo, en el caso de aprendizaje para la clasificación, el modelo se evalúa midiendo el nivel de concordancia entre la clase predicha por el modelo y la clase real.

2.4.1 Matriz de Confusión

La matriz de confusión es una herramienta que evalúa el desempeño de un modelo de clasificación, organizando los resultados en una tabla de cruza las clases predichas con las reales. Un eje representa las predicciones del modelo y el otro las etiquetas reales permitiendo identificar aciertos y errores, como verdaderos positivos, falsos positivos, verdaderos

negativos y falsos negativos [22]. En la figura 2.9 se muestra la estructura de una matriz de confusión.

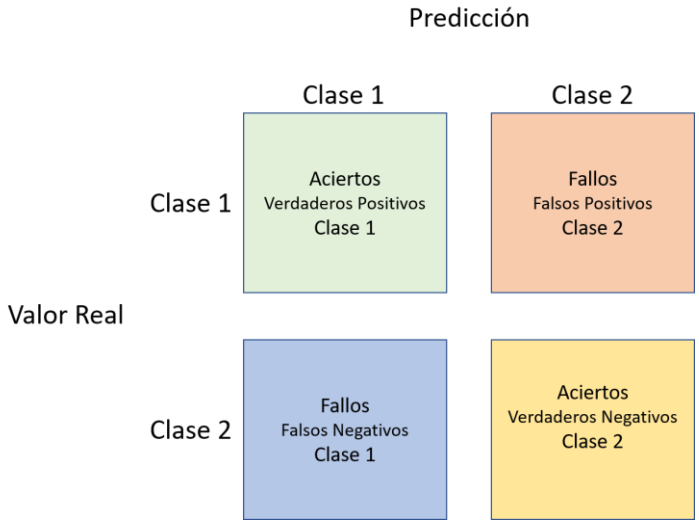


Figura 2.4. Ilustración matriz de confusión. Fuente: Propia

En cada predicción podemos encontrar uno de cuatro resultados, cada uno de ellos basándose en su coincidencia con el valor real:

- **Verdadero Positivo (TP):** El modelo predice correctamente un caso como verdadero, coincidiendo con su valor real.

- **Verdadero negativo (TN):** El modelo predice correctamente un caso como falso, coincidiendo con su valor real.

- **Falso Positivo (FP):** El modelo predice erróneamente un caso como verdadero cuando en realidad es negativo.

- **Falso negativo (FN):** El modelo predice erróneamente u caso como falso cuando en realidad es verdadero.

2.4.2 Precisión

Precisión es la relación entre las predicciones positivas correctas y el número total de predicciones positivas [22]. Se representa con la siguiente ecuación (ver Ecuación 8).

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

2.4.3 Recall

Es la relación entre las predicciones positivas correctas y el número total de ejemplos positivos en el conjunto de datos [22]. Se muestra en la Ecuación 6.

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

2.4.4 F1-Score

La puntuación F1 se conoce como la media armónica de la precisión y del recall. El principal objetivo de esta medida es combinar en un solo valor la precisión y el recall como se ve representado en la Ecuación 7 [22].

$$F1\ Score = \frac{2 \cdot (precision \cdot recall)}{precision + recall} = \frac{2 \cdot TP}{s \cdot TP + FP + FN} \tag{6}$$

2.5 Transformers

Un Transformer es una arquitectura de red neuronal desarrollada para tareas de PLN. Esta arquitectura se ha convertido en una de las más dominantes en este campo, superando en rendimiento a otros modelos neuronales alternativos como las redes neuronales convolucionales y recurrentes, en tareas de comprensión y generación de lenguaje natural [21].

Esta arquitectura soluciona los problemas de dependencias largas que presentan las redes neuronales recurrentes, las cuales procesan el texto palabra a palabra y de manera secuencial. En lugar de depender de la recurrencia para procesar secuencias de textos, los Transformers utilizan mecanismos de atención para permitir que cada token en la secuencia se relacione con todos los demás tokens de manera simultánea [23]. Esta capacidad de atención global les permite capturar relaciones complejas y de largo alcance en el texto, lo que ha demostrado ser fundamental para el éxito en tareas como clasificación, traducción automática y generación de texto.

Además, los Transformers eliminan esa dependencia que tenía las redes recurrentes. Esta arquitectura tiene la capacidad de procesar secuencias de entrada más rápidas y de forma efectiva, lo que los hace ideales para tareas que involucran grandes cantidades de datos.

La arquitectura de los Transformers tiene una estructura *encoder-decoder* como podemos observar en la figura 2.4*,* esta estructura se basa en codificar la secuencia de entrada para posteriormente descodificarla. La peculiaridad de esta arquitectura es el uso únicamente de mecanismos de atención sin la inclusión de redes recurrentes o redes convolucionales [24].

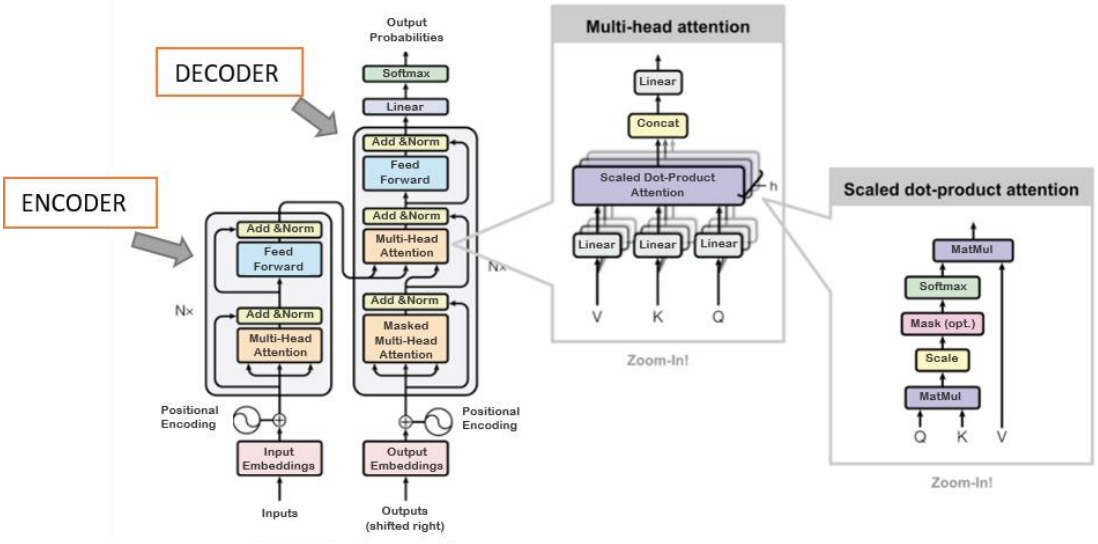


Figura 2.5. Arquitectura de Transformers con modificaciones para mostrar Endor y Decoder. Fuente: Adaptado de *[25]*

Esto hace que los *Input Embeddings* y los *Positional Encodings* sean componentes críticos para que el modelo pueda procesar y comprender contextos esenciales en el texto.

Primero, los *Input Embeddings* transforman las palabras o tokens en vectores numéricos de dimensión fija que capturan relaciones semánticas. Por ejemplo, en un espacio de *emebeddings* bien entrenado, palabras como "gato" y "felino" se representarían con vectores muy cercanos, reflejando su similitud en significado. La dimensión típica de los *embeddings,* denotada como *d,* puede variar entre 256 y 1024, dependiendo de la complejidad del modelo y la tarea. Una mayor dimensión permite al modelo capturar relaciones más complejas, aunque a costa de mayores recursos computacionales [26] [27].

Para complementar los *embeddings* y suplir la ausencia de estructuras secuenciales intrínsecas en los Transfomers, se introducen los *Positional Encodings.* Estos se suman directamente a los *Input Embeddings* y se calculan utilizando funciones seno y coseno de diferentes frecuencias. Este enfoque garantiza que cada posición en una secuencia tenga una representación única, permitiendo que el modelo entienda relaciones relativas como la dependencia entre sujeto y verbo, independientemente de la distancia entre ellos en la oración. Por ejemplo, los *Positional Encodings* permiten que el modelo distinga entre "el gato persigue al perro" y "el perro persigue al gato" al identificar cómo las palabras se relacionan en diferentes contextos [26].

La combinación de *Input Embeddings* y *Positional Encoding* es fundamental en tareas de PLN. Los *embeddings* representan el significado de las palabras, mientras que los *Positional Encoding* incorporan el orden en la secuencia. Esto resulta crucial en aplicaciones de traducción automática, donde la estructura gramatical entre idiomas puede variar significativamente, o en tareas de resumen de texto, donde la posición de ciertas palabras puede influir en su importancia [28] [29]. Además, herramientas modernas como "Sentence Transformers" o "HuggingFace" han facilitado el uso de *embeddings* para resolver tareas avanzadas de PLN, integrando representaciones contextuales y semánticas más robustas [27].

Por otro lado, tenemos el *encoder* que consta de 6 bloques idénticos. Cada bloque del *encoder* tiene una estructura específica que incluye diferentes subcomponentes:

- **Bloque Atencional (*Multi-head self attention mechanism)***
  El mecanismo de atención de un Transformer permite al modelo calcular la importancia relativa de cada token en la secuencia con respecto a un token de consulta dado, lo que ayuda al modelo a capturar relaciones y dependencias entre los tokens de manera más efectiva [23].

  Para expresar numéricamente las relaciones entre las partes según los grados de asociación entre palabras se utiliza el mecanismo de atención *Scaled Dot-Product Attention* (ver Ecuaión 4).

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (7)$$

  Este mecanismo recibe como entrada tres conjuntos de vectores: *query*(Q), *key*(K) y *value*(V) [24], que se obtienen mediante transformaciones lineales de los *embeddings* de entrada. Se calcula el producto escalar QK^T y posteriormente se divide entre $\sqrt{d_k}$ para controlar la escala de los vectores resultantes y evitar que se vuelvan demasiado grandes o demasiado pequeños a medida que aumenta la dimensión de los vectores.

- **Bloque Residual (*Add & Norm*)**
  El bloque residual combina tanto la entrada como la salida del bloque atencional. Esto es fundamental en redes profundas, ya que enviar únicamente la salida del bloque atencional podría provocar una degradación progresiva de la información, dificultando el entrenamiento y desempeño de la red [30].

  Este bloque suma los datos de la entrada y salida para después normalizarlos ya que estos datos han sido "deformados" tras pasar por las capas anteriores. Al sumar la entrada original a la salida transformada, se crea una ruta directa para el flujo de información, mejorando eficiencia del aprendizaje.

- **Bloque *Feedforward***
  Este bloque transforma la salida del bloque residual mediante dos capas lineales y una función de activación no lineal [24]. Esto permite al modelo aprender relaciones más complejas entre los datos de entrada.

  La entrada y la salida de este bloque son llevadas a otro bloque residual que preparará los datos para el siguiente *encoder*.

El *decoder* también consta de 6 bloques idénticos. Estos bloques tienen la misma estructura que los bloques del *encoder,* con la diferencia de que el *decoder* introduce al inicio una capa *multi-head self attention mechanism* enmascarada, que asegura que solo utilicen las palabras que ya ha procesadas. Esto se logra mediante un mecanismo de enmascaramiento que evita que cada posición pueda "ver" posiciones futuras en la secuencia. De esta manera, el *decoder* genera las

palabras de salida de manera autorregresiva, utilizando solo la información de las palabras anteriores en la secuencia.

Además, cada bloque del *decoder* incluye una segunda capa de atención, conocida como **atención cruzada codificador-decodificador** (a veces llamada *source-target-attention).* Esta subcapa calcula la atención entre las características generadas por el codificador (*source)* y las representaciones parciales producidas por el decodificador (*target).* En esta etapa, las salidas del codificador se utilizan como claves y valores, mientras que las consultas provienen de la salida de la capa de autoatención enmascarada del decodificador. Este diseño permite que el *decoder* identifique y enfoque su atención en las partes más relevantes de la secuencia de entrada para generar las predicciones de salida. Aunque no siempre se etiqueta explícitamente como *source-target attention,* esta capa es esencial para integrar la información de entrada procesada por el codificador con las predicciones parciales generadas por el decodificador.

Cada bloque del *decoder* también incluye un bloque *Feed Feedforward*, que procesa cada posición de manera independiente, refinando las representaciones internas. También cuentan con varios bloques residuales para mejorar la estabilidad del modelo y facilitar el flujo del gradiente durante el entrenamiento.

El proceso de decodificación termina con una capa lineal y una función *softmax* que convierten las representaciones internas en una distribución de probabilidad sobre el vocabulario. Este flujo operativo permite al *decoder* generar la secuencia de salida palabra por palabra hasta alcanzar un token especial que señala el final de la secuencia [31].

2.5.1 Modelos del Lenguaje

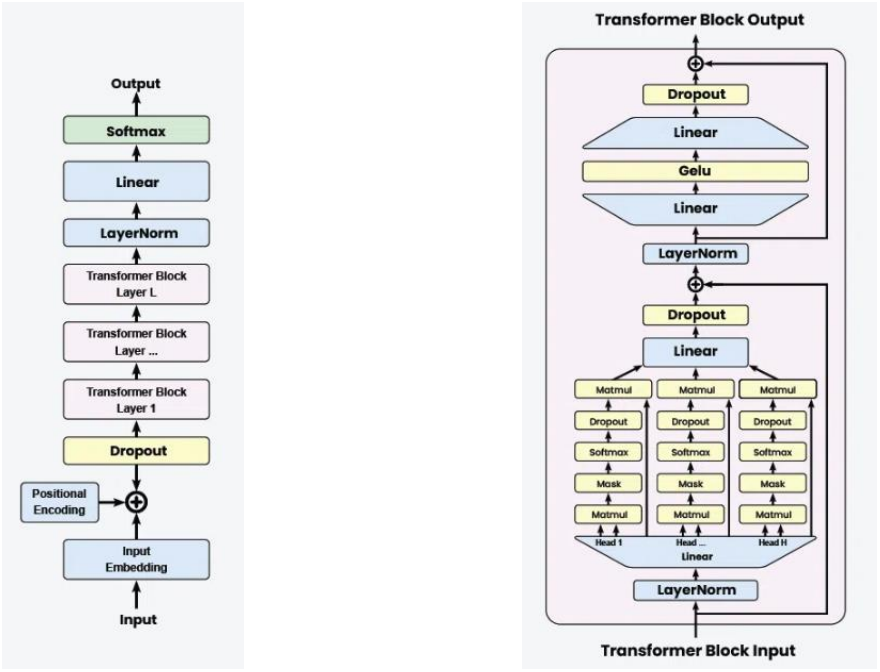
Los Modelos del Lenguaje (LM) son herramientas diseñadas para distinguir entre secuencias gramaticales y no gramaticales. Dada una frase, un modelo de lenguaje ha de identificar si esta frase es plausible o no dada la gramática del lenguaje [32]. En los últimos años se han publicado una gran cantidad de modelos basados en redes neuronales profundas, entre los más conocidos en la literatura se encuentran:

- GPT-3.5
- GPT-4
- BERT
- RoBERTa

2.5.2 GPT-3.5

GPT 3.5 es la tercera versión del modelo desarrollado por OpenAI GPT (*Generative Pre-trained Transformer*) [33]. Su diseño se basa en un enfoque generativo, lo que significa que está entrenado para predecir la siguiente palabra en una secuencia de texto. Esta capacidad le permite generar texto coherente y contextualizado de manera fluida, lo que lo hace especialmente eficaz en tareas que requieren la creación de contenido, como la redacción de artículos, generación de diálogos y la respuesta a preguntas.

Una de las características más destacadas de GPT es su habilidad para manejar dependencias a largo plazo en el texto, gracias a su arquitectura de Transformer unidireccional. Esto le permite capturar mejor la estructura y el significado de textos extensos, resultando en una comprensión más profunda del contenido. Además, GPT puede ser adaptado a una amplia variedad de tareas de NLP mediante un proceso de *fine-tunning*, lo que lo convierte en un modelo versátil y poderoso en el ámbito del PLN.



a) Flujo general arquitectura GPT b) Bloque Transformer en GPT

Figura 2.6. Arquitectura GPT. Fuente: *[34]*

Como se ve en la figura 2.5 a), el proceso comienza con la tokenización del texto en bruto y continúa pasando por las etapas de *input embedding*, codificaciones posicionales y bloques Transformers, hasta llegar a la capa de salida. A continuación, se describe cada una de las etapas del flujo.

- ***Input Embedding*:** El primer paso es la tokenización del texto en unidades discretas, como palabras o subpalabras. Esto convierte el texto en una secuencia de tokens numéricos que el modelo puede procesar.

- ***Positional Encoding*:** Dado que el Transformer no tiene una estructura secuencial, se añaden codificaciones posicionales a los *embeddings*. Estos vectores permiten al modelo considerar el orden de los tokens.

- ***Dropout Layer*:** Para evitar el sobreajuste, se aplica una capa de *dropout* a los *embeddings*. Este mecanismo desactiva aleatoriamente algunas neuronas durante el entrenamiento, ayudando al modelo a generalizar mejor en datos nuevos.

En cuanto al bloque Transformer ilustrado en la figura 2.5 b), está compuesto por varias capas que trabajan juntas para procesar el texto. A continuación, se detallan las funciones claves de cada capa.

- ***LayerNorm*:** Cada bloque Transformer comienza con la normalización de capas, ajustando la salida previa para tener una media de cero y varianza de uno. Esto estabiliza el entrenamiento, evitando problemas como el desvanecimiento del gradiente y acelerando la convergencia.

- ***Multi-Head Self-Attention*:** Como ya se explicó en el punto 2.2.1, este bloque evalúa diferentes partes de la secuencia para determinar la importancia de cada token.

- ***Feed-Forward Network*:** Se aplica una red *feed-forward*, que usa transformaciones lineales y activaciones no lineales para aprender representaciones más complejas. Esta red actúa de forma independiente en cada posición del token y mejora la comprensión del lenguaje.

- ***Dropout*:** Se aplica nuevamente *dropout* a la salida para evitar el sobreajuste.

- ***Layer Stack*:** Los bloques de Transformer se apilan, lo que permite al modelo capturar patrones más complejos y aprender representaciones cada vez más abstractas.

A lo largo de las versiones, el modelo GPT ha experimentado mejoras significativas desde su primera versión. La primera versión del modelo se centra en una arquitectura basada en el preentrenamiento generativo, con un número limitado de parámetros y capacidades, lo que lo hacía adecuado para tareas básicas de PLN.

Con el avance a GPT 3.5 [35], el modelo se ha vuelto más potente y capaz, con miles de millones de parámetros que permiten una comprensión más profunda del contexto, mayor fluidez en la generación de texto y la capacidad de realizar tareas más complejas. Una diferencia clave es la mejora en la *Multi-Head Self-Attention* y el tamaño de la red, lo que permite manejar dependencias a largo plazo de manera más eficiente.

2.5.3 GPT-4

GPT 4 es la versión más reciente del modelo GPT, mejorando en un 82% en respuestas seguras y un 40% en exactitud a su predecesor GPT-3.5. Además, añade capacidad de reconocimiento de imágenes, ofreciendo respuestas tanto a entradas de texto como visuales. Esto abre nuevas posibilidades para aplicaciones en áreas de la educación, atención médica y el diseño.

Las principales diferencias con GPT-3.5 incluyen mayor capacidad multilingüe (capaz de procesar más de 26 idiomas), más precisión en la personalización del estilo de escritura.

Puede procesar hasta 32.000 tokens, lo que lo hace ideal para textos largos y tareas más complejas que GPT-3.5 no podría manejar.

Otra diferencia clave es el enfoque en la alineación y la seguridad, con un proceso de entrenamiento que incluye retroalimentación humana para mejorar la adherencia a comportamientos deseados y reducción de sesgos. Esto se traduce en un modelo que no solo es más potente, sino también más consciente de las implicaciones éticas de su uso.

En este informe técnico [36], se realizaron diversas evaluaciones que arrojaron resultados significativos sobre el rendimiento y la seguridad del modelo. En las evaluaciones cualitativas, se observó que GPT-4 mostró una mejora notable en la coherencia y relevancia de las respuestas en comparación con versiones anteriores. Sin embargo, también se identificaron riesgos persistentes, como la generación de contenido sesgado y poco fiable. Por ejemplo, se encontró que el modelo tenía tendencia a "alucinar", produciendo contenido que era engañoso y no veraz, lo que puede ser particularmente perjudicial dado que el modelo puede parecer convincente y creíble para los usuarios.

En cuanto a las evaluaciones cuantitativas, se midió la probabilidad de que GPT-4 generara contenido no deseado, como discursos de odio o consejos autolesivos. Los resultados mostraron que, a pesar de las mitigaciones implementadas, el modelo aún presentaba limitaciones, con un 10% de las salidas generadas clasificadas como potencialmente dañinas en ciertas pruebas. Esto subraya la necesidad de continuar refinando las estrategias de mitigación y evaluación para abordar estos problemas de manera efectiva.

2.2.3 RoBERTa

RoBERTa (*Robustly Optimized BERT pre-training Approach)* es una variante de BERT (*Bidirectional Encoder Representations from Transformers*), que fue desarrollada por investigadores de Facebook AI [37]. Al igual que BERT, RoBERTa es un modelo de lenguaje basado en Transformers que utiliza la autoatención para procesar secuencias de entrada y generar representaciones contextualizadas de palabras en una frase.

A continuación, se explicará la arquitectura BERT, ya que, como se ha mencionado, RoBERTa es una variante de BERT.

El modelo BERT surgió de la idea de crear un modelo base que aprende a interpretar el lenguaje en general y una vez preentrenado el modelo de lenguaje base, añadir capas adicionales para especializarse en una tarea en concreto [38] . Una de las principales diferencias de este modelo con los modelos de lenguaje tradicionales es que BERT utiliza un enfoque bidireccional, considerando el contexto izquierdo y derecho de las palabras de una oración. En lugar de analizar el texto secuencialmente, BERT analiza todas las palabras de una oración simultáneamente.

BERT emplea una arquitectura de solo codificador. En la arquitectura *Transformer* original, hay módulos codificadores y decodificadores. La decisión de utilizar una arquitectura de solo codificador en BERT sugiere un enfoque principal en comprender secuencias de entrada en lugar de generar las secuencias de salida.

El nombre de la arquitectura en concreto que utiliza BERT es *multi-layer bidirectional Transformer encoder* [38]. Como se observa en la figura 2.6, se trata de una concatenación de varias capas, siendo cada una de estas capas un bloque *transformer* como los explicados anteriormente.

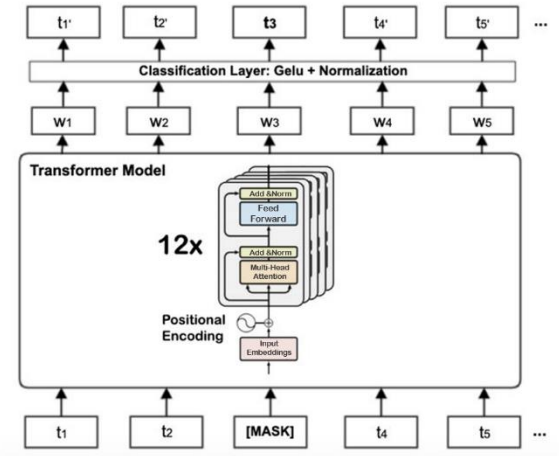


Figura 2.7. Arquitectura BERT. Fuente: *[39]*

El entrenamiento de BERT consta de 2 fases, un pre-entrenamiento en el que BERT "aprende" que es el lenguaje y su contexto. Para ello BERT es entrenado con 2.500 millones de palabras provenientes de la Wikipedia y 850 millones provenientes de *BookCorpus* [38]. En concreto BERT se entrena en las tareas de *Masked Language (MLM) y Next Sentence Prediction (NSP),* lo cual ayuda a BERT a entender el lenguaje. En la figura 2.7 se muestra una imagen del funcionamiento de la fase de entrenamiento de BERT.

En la tarea de *Masked Language Modeling* (Modelo de Lenguaje enmascarado en español), BERT enmascara algunas palabras de una secuencia de entrada y se entrena para predecir las palabras originales basándose en el contexto de las palabras circundantes [38]. Este enfoque ayuda a comprender mejor el significado y contexto de las palabras en un texto.

Por otro lado, en la tarea de *Next Sentence Prediction* (Predicción de la siguiente frase en español) [38], BERT predice si una segunda frase está conectada con la primera. Para ello, transforma la salida del token *[CLS]* en un vector y calcula la probabilidad de que la segunda oración siga a la primera usando una función *softmax*. Con esta tarea se entrena al modelo para entender la relación entre pares de frases.

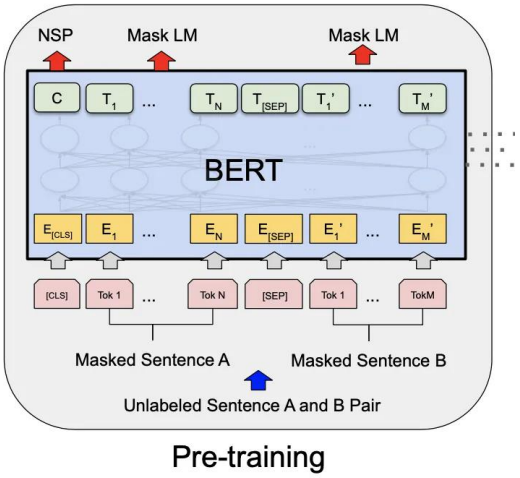


Figura 2.8. Representación de la fase de preentrenamiento de BERT. Fuente *[40]*

En la segunda fase, se realiza un *fine-tunning* donde BERT aprende a resolver el problema en concreto que se trata de resolver. El *fine-tunning* consiste en adaptar un modelo base previamente entrenado para una tarea concreta utilizando un conjunto de datos adicional y más específico [56]. Para ello se deben añadir las capas de salida que requiera la tarea en concreto a resolver. Se puede observar en la figura 2.8 una representación del *fine-tunning*.

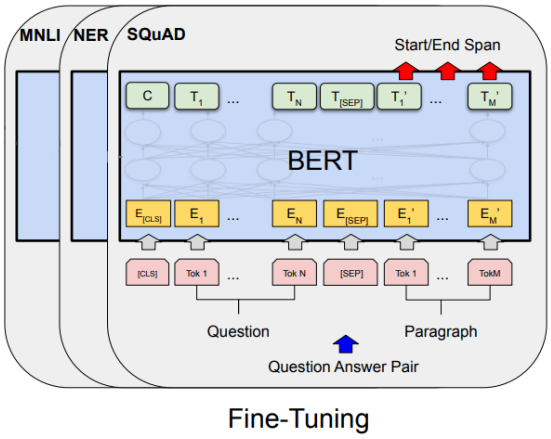


Figura 2.9. Representación de la fase de *fine-tunning* de BERT. Fuente *[40]*

La fase de *fine-tunning* en BERT adapta el modelo preentrenado para tareas específicas de PLN, ajustando sus parámetros junto con nuevas capas de salida diseñadas para cada tarea. Las entradas, que pueden ser una o dos oraciones separadas por el token *[SEP],* se procesan a través de las capas del Transformer de BERT, generando representaciones contextuales de los tokens.

Dependiendo de la tarea, se utilizan diferentes estrategias: en clasificación la representación del token *[CLS]* se pasa por una capa *softmax* para asignar etiquetas. En tareas a nivel de tokens, como NER (Reconocimiento de Entidades Nombradas, por sus siglas en inglés), cada token se clasifica individualmente. Para tareas de respuesta a preguntas, como SQuD (*Stanford Question Answering Dataset,* conjunto de datos utilizados para evaluar modelos de

respuesta a preguntas basados en fragmentos de texto), el modelo identifica las posiciones de inicio y fin de la respuesta dentro del párrafo dado [41].

El *fine-tunning* se realiza con datos etiquetados, entrenando todos los parámetros del modelo junto con las nuevas capas de salida. Esto permite que BERT se adapte eficientemente a una amplia variedad de tareas.

2.5.5 Comparativa entre BERT y RoBERTa

Tras haber detallado la arquitectura BERT, vamos a ver los principales puntos donde RoBERTa gana a su predecesor:

- **Mayor tamaño de datos de entrenamiento:** RoBERTa se entrenó con un conjunto de datos mucho más grande y utilizó un procedimiento de entrenamiento más efectivo. Este modelo fue entrenado con 160 Gb de texto, que es 10 veces más grande que el conjunto de datos con el que se entrenó BERT. Al incluir más datos, también es capaz de generalizar mejor en diferentes dominios y mejorar su rendimiento en una amplia gama de tareas de PLN. Además, este conjunto de datos incluye textos en varios idiomas, lo que hace que RoBERTa sea especialmente versátil para aplicaciones multilingües, algo que BERT no logra con la misma efectividad.

- **Enmascaramiento dinámico**: BERT utiliza un enfoque estático para enmascarar tokens en el que las mismas partes del texto se enmascaran en todas las épocas de entrenamiento. Esto significa que BERT puede sobrentrenarse en los mismos patrones de enmascaramiento, lo que reduce su capacidad para aprender de manera eficiente sobre el contexto del texto. RoBERTa soluciona este problema mediante el enmascaramiento dinámico, donde los tokens enmascarados varían con cada pasada del conjunto de datos. Este enfoque introduce más variabilidad y diversidad en los patrones de enmascaramiento, lo que permite que el modelo sea más robusto al aprender múltiples contextos. Así, reduce la duplicación de datos y aumenta la exposición a diferentes aspectos del texto, lo que mejora su capacidad de comprensión en tareas de clasificación y predicción.

- **Eliminación de NSP**: En el preentrenamiento de RoBERTa, solo se utiliza la tarea de MLM y se elimina la tarea NSP *(Next Sentence Prediction)*. Los autores experimentaron con la inclusión y exclusión de la pérdida de NSP en varias versiones y concluyeron que eliminar la tarea del NSP iguala o mejora ligeramente el rendimiento en tareas posteriores. Al enfocarse únicamente en el MLM, RoBERTa puede concentrar toda su capacidad en aprender representaciones más ricas de palabras y oraciones sin verse afectado por la complejidad adicional de NSP.

- **Mayor tamaño del *batch*:** El tamaño del lote durante el entrenamiento tiene un impacto directo en la estabilidad y la rapidez con la que un modelo converge a una solución óptima. En BERT, el tamaño del *batch* es de 256, pero RoBERTa lo aumenta a 8.000. Este aumento permite que el modelo procese una mayor cantidad de datos

simultáneamente, lo que acelera el proceso de entrenamiento y también mejora la estabilidad del modelo.

- **Codificación de texto en bytes:** RoBERTa mejora la tokenización utilizada en BERT. Mientras que BERT se basa en la tokenización a nivel de subpalabras con un vocabulario de 30.000 tokens, RoBERTa aumenta este número a 50.000 y utiliza un enfoque basado en bytes, lo que lo hace más flexible y efectivo al manejar palabras raras o no vistas previamente.

- **Preentrenamiento optimizado y mayor número de pasos:** Una de las principales diferencias entre BERT y RoBERTa es el número de pasos de entrenamiento. Mientras que BERT fue entrenado durante 100.000 pasos, RoBERTa extiende este proceso a 500.000 pasos, lo que le permite afinar su aprendizaje de manera más profunda. Este preentrenamiento extendido, combinado con un conjunto de datos más grande, asegura que RoBERTa pueda capturar mejora las complejidades del lenguaje y las relaciones entre las palabras en diversos contextos. Esta mayor cantidad de pasos de entrenamiento le da a RoBERTa una ventaja competitiva en tareas de comprensión del lenguaje, donde es necesario un análisis detallado de las relaciones semánticas en grandes volúmenes de texto

En [42] se realizó un estudio sobre el reconocimiento de emociones en textos, se han utilizado varios modelos de lenguaje basados en la arquitectura Transformers, destacando BERT, RoBERTa, DistilBERT y XLNet. Estos modelos han sido evaluados en función de su capacidad para clasificar emociones utilizando el conjunto de datos ISEAR, que contiene anotaciones para diversas emociones.

Los experimentos realizados mostraron que cada modelo tenía un rendimiento diferente en términos de precisión, recall y F1-Score. En particular, RoBERTa se destacó como el modelo más eficaz, alcanzando un F1-Score de 74,2%. Por otro lado, XLNet y BERT también mostraron resultados competitivos, con F1-Score de 73,1% y 70,2% respectivamente. DistillBERT, aunque fue el modelo más rápido en términos de tiempo de entrenamiento, obtuvo el rendimiento más bajo con un F1-Score del 69,3%. En la tabla 2.1 se ven reflejados en forma de resumen los resultados de los distintos modelos evaluados en el experimento.

Modelo	F1-Score
BERT	70,2%
RoBERTa	74,2%
XLNet	73,31%
DistillBERT	69,3%

Tabla 2.1. Resultados modelos en reconocimiento de emociones según [33].

Los resultados sugieren que, aunque todos los modelos basados en Transformers son efectivos para la tarea de reconocimiento de emociones, RoBERTa se posiciona como el mejor debido a su combinación de precisión y eficiencia.

En [43] se aborda la detección y clasificación de contenido sexista en redes sociales utilizando técnicas avanzadas de PLN para el desafío EXIST 2024. Para llevar a cabo esta investigación, se emplearon dos modelos principales XLM-RoBERTa y GPT-3.5. El primer modelo es una versión avanzada de RoBERTa que fue ajustado en un conjunto de datos diseñado para reconocer y clasificar contenido sexista. Este modelo se benefició de su entrenamiento previo en un amplio conjunto de datos multilingüe, lo que le permitió captar patrones lingüísticos complejos de manera más efectiva.

En contraste, se utilizó GPT-3.5 a través de un enfoque de aprendizaje de pocos ejemplos (few-shot learning), donde se proporcionaron ejemplos de tweets en inglés y español para ayudar al modelo a adaptarse a tareas específicas. Aunque GPT-3.5 mostró una buena capacidad de adaptación, su rendimiento no alcanzó el nivel de precisión que logró XLM-RoBERTa.

En la tarea 1, que se enfoca en la identificación de contenido sexista, XLM-RoBERTa alcanzó un F1-Score de 78% en el conjunto de datos general, mientras que GPT-3.5 obtuvo un 71%. En los tweets en inglés, XLM-RoBERTa logró un 75% de F1-Score, superando a GPT-3.5, que alcanzó un 70% de F1-Score. En los tweets españoles, XLM-RoBERTa también destacó con un F1-Score de 79%, en comparación con el 72% de GPT-3.5.

Por otro lado, en la tarea 2, que se centra en la identificación de la intención detrás de los comentarios sexistas en tweets, el modelo XLM-RoBERTa obtuvo una puntuación de 48% en el conjunto de datos generales, superando a GPT-3.5, que alcanzó un 43% de F1-Score. En el análisis de tweets en inglés, XLM-RoBERTa logró un 43% de F1-Score, mientras que GPT-3.5 obtuvo un 40% de F1-Score. En el caso de los tweets en español, XLM-RoBERTa destacó con una puntuación de F1-Score del 52%, en comparación con al 44% de F1-Score de GPT-3.5.

Para facilitar la comparación de los resultados entre XLM-RoBERTa y GPT-3.5 en este estudio, se presenta en la tabla 2.1 un resumen del desempeño de ambos modelos en términos de F1-Score.

Tarea	**Tipo de datos de evaluación**	**XLM-RoBERTa**	**GPT-3.5**
Tarea 1(Identificación contenido sexista)	Datos de carácter general	48%	43%
	Tweets en inglés	43%	40%
	Tweets en español	52%	44%
Tarea 2 (Detección de intención de comentarios sexistas)	Datos de carácter general	78%	71%
	Tweets en inglés	75%	70%
	Tweets en español	79%	72%

Tabla 2.2**.** Resultados modelos en detección de contenido sexista según [34].

Estos resultados evidencian el rendimiento superior de XLM-RoBERTa en ambas tareas en comparación con GPT-3.5.

En [44] para el caso de detección de noticias falsas, este informe investiga la eficacia de varios modelos de aprendizaje profundo, centrándose en el *fine-tunning* de RoBERTa para la clasificación de la veracidad de las afirmaciones en el conjunto de datos LIAR. Este conjunto de datos está diseñado específicamente para la detección de noticias falsas en inglés.

La investigación destaca como RoBERTa, con su arquitectura y *embeddings* contextuales preentrenados, supera significativamente a los modelos MegatronBERT y LSTM. Se observa que, aunque los modelos tradicionales como LSTM y MegatronBERT han mostrado resultados aceptables, con Accuracy del 58,3% y 64% respectivamente, los modelos como RoBERTa logran un Accuracy del 66%. En la tabla 2.3 se ven representados los resultados del informe.

Modelo	Accuracy
MegatronBERT	58,3%
RoBERTa	66%
LSTM	64%

Tabla 2.3. Resultados modelos detección noticias falsas según [35].

Según los estudios presentados, se ha decidido escoger el modelo RoBERTa para la detección de noticias falsas en español, dado su rendimiento superior en comparación a otros modelos. RoBERTa ha mostrado resultados sólidos en tareas similares, sugiriendo que su aplicación en español podría resultar igualmente exitosa.

Capítulo 3

Detección de Fake News

Para realizar un estudio científico sobre la detección de noticias falsas en español, primero se examinarán los conjuntos de datos disponibles en la literatura. Se identificarán aquellos conjuntos de datos que sean más relevantes y actualizados, con un enfoque particular en aquellos que contengan noticias en español.

Después de este análisis, se presentarán unos modelos base que servirán como referencia. Finalmente, se presentarán los modelos propuestos para el estudio y sus resultados correspondientes.

3.1 Datasets disponibles

Para resolver el problema de la detección de *fake news,* se ha realizado un estudio de los conjuntos de datos más reconocidos y utilizados en este ámbito. A continuación, se muestra un listado de los conjuntos de datos analizados y sus características.

- **Liar** [45]**:** Este conjunto contiene un total de 12.836 declaraciones ciertas recogidas de Politifact. Cada declaración se mide en una escala de seis opciones de veracidad y también incluye información sobre la temática, partido político, contexto y orador. Los datos están en inglés.

- **FakesNewsNet** [46]**:** Consiste en la cabecera y cuerpo del texto de artículos de noticias falsas obtenidas en BuzzFeed y Politifact. Los datos están en inglés.

- **CONSTRAINT@AAI 2021** [47]**:** Consiste en 10.700 publicaciones en inglés, con 5.100 noticias falsas y 5.600 verdaderas. Las noticias verdaderas se recogen en X (antiguo Twitter), mientras que las falsas provienen de varias fuentes, incluyendo X, Facebook y sitios de fact-checking como Politifact, NewsChecker y Boomlive.

- **Spanish Fake News Corpus** [48]**:** Este conjunto contiene 971 noticias en español sobre 9 dominios distintos, clasificadas en verdadero y falso. El corpus contiene noticias de 9 tópicos diferentes: ciencia, deporte, economía, educación, entretenimiento, política, salud, seguridad y sociedad.

- **FNC-1 dataset** [49]**:** *Fake News Challenge* es un conjunto de datos en inglés que contiene 49.972 artículos provenientes del projecto EMERGENT sobre política, sociedad y tecnología. El conjunto de datos está preparado para detectar si la noticia es falsa o no según si está relacionado el título de la noticia con el cuerpo.

- **Fact checking dataset** [50]**:** Es el primer conjunto de datos públicos para la detección de fake news. Los datos provienen de Politifact y Channel4. El conjunto de datos en inglés contiene 221 declaraciones, con la fecha en la que fueron hechas, el orador y la URL. La veracidad se mide en una escala entre 5 opciones: verdadero, mayoritariamente verdadero, medio verdad, mayoritariamente falso y falso.

Para este estudio se ha escogido el conjunto de datos de **Spanish Fake News Corpus** ya que ha participado en IberLEF [51], el cual se celebra anualmente en el congreso internacional de la Sociedad Española de Procesamiento del Lenguaje Natural. IberLEF se enfoca en la evaluación de sistemas PLN en español y otros idiomas ibéricos.

En particular, el corpus ha sido empleado en la edición de FakeDeS 2021 [52], una tarea específica de IberLEF que se centra en la detección de noticias falsas en español. En dicha competición, se probaron distintos enfoques que abarcaron desde técnicas más tradicionales como *Bag of Words* y *n-grams,* hasta modelos más avanzados basados en Transformers, incluyendo el uso de BERT preentrenado. El modelo que obtuvo el mejor rendimiento estuvo basado en el modelo BERT, alcanzando un F1 Score de 70%. Este resultado prueba el buen desempeño de los modelos basados en Transformers para trabajar con corpus más complejos.

Por otro lado, [53] presenta un estudio que aborda la detección de noticias falsas en español usando técnicas de *Deep Learning*. Se implementaron modelos basados en arquitecturas como LSTM (*Long-Short-Term-Memory*) que combinan el procesamiento secuencial con *embeddings* preentrenados, entre ellos el modelo BETO. Aunque el mejor resultado obtenido fue un accuracy del 80%, el estudio destacó que el modelo tendía a confundir noticias falsas como verdaderas.

Adicionalmente, los creadores del corpus realizaron un análisis experimental en el que combinaron técnicas de *Bag of Words* y *Part-of-Speech* con el clasificador *Random Forest*.Este enfoque tradicional alcanzó un F1-Score de 76,94% [48]. Este resultado demuestra la efectividad de este corpus para evaluar y optimizar algoritmos de clasificación.

En el contexto de [54], se presenta un artículo en la 40º Conferencia de la Sociedad Española de Procesamiento del Lenguaje Natural (SEPLN-2024), donde se lleva a cabo la tarea de detección de noticias falsas en español. Se adaptó el modelo FakeFlow que está diseñado para aprender el flujo de información semántica y afectiva en los artículos de noticia. A pesar de que el modelo se utiliza, se observa que su rendimiento es inferior al de un clasificador basado en RoBERTa, que logra un F1-Score de 76% sin ninguna mejor adicional. Además, se comparan versiones simplificadas del modelo FakeFlow, pero estas no logran superar el rendimiento del clasificador RoBERTa.

3.1 Análisis y Estudio del conjunto de datos

Antes de profundizar en el análisis detallado del corpus, primero se examinarán algunas medidas básicas para establecer una comprensión inicial del conjunto de datos. Estas medidas incluyen número total de palabras del corpus, el balance de clases, tamaño total del conjunto de datos y la división del conjunto de datos en subconjuntos de entrenamiento y prueba.

Medida	Spanish Fake News Corpus
Tamaño total del conjunto de datos	**1.543**
Total de palabras del corpus	**84.835**
Tamaño del conjunto de entrenamiento	**998**
Tamaño del conjunto de validación	**250**
Tamaño del conjunto de prueba	**295**
Número de noticias verdaderas	**777**
Número de noticias falsas	**766**

Tabla 3.1. Medidas básicas del Spanish Fake News Corpus

El corpus presenta un conjunto de datos bien equilibrado para el análisis de noticias en español. Aunque no sea un conjunto muy grande, ofrece una muestra significativa para el desarrollo de la investigación. En total, el corpus incluye 1.543 noticias, de las cuales 777 son verdaderas y 766 son falsas, logrando un balance notable entre ambas clases.

El corpus se divide en distintos archivos, incluyendo "train.xlsx", "test.xlsx" y "development.xlsx". Este último archivo, destinado al ajuste y validación de los modelos, complementan los conjuntos de entrenamiento y prueba al proporcionar un subconjunto adicional de noticias.

En el análisis que se va a llevar a cabo, se examinarán 3 tipos de N-gramas: los unigramas, bigramas y trigramas. Un N-grama es una secuencia de n elementos consecutivos en un texto [55]. Por ejemplo, si estamos analizando los unigramas, serán las secuencias compuestas por una sola palabra. Los bigramas son secuencias de dos palabras consecutivas, y los trigramas son secuencias de tres palabras consecutivas. Esto se hará con el fin de analizar alguna peculiaridad del corpus.

Para concluir, se mostrará una nube de palabras del conjunto de datos. Estas nubes de palabras representan gráficamente las palabras más repetidas en todo el conjunto de datos, ajustando su tamaño a la frecuencia con la que aparecen. Las palabras más repetidas ocuparán más tamaño que las que no se repiten tanto. Además, se analizarán las nubes de palabras del

conjunto de noticias falsas y verdaderas, con el objetivo de identificar diferencias entre ambos grupos.

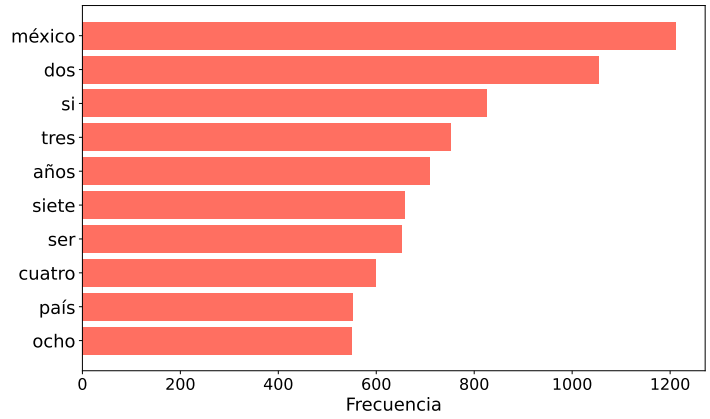


Figura 3.1. Unigramas más repetidos

Los unigramas más repetidos en el corpus destacan términos como "México" con la mayor frecuencia, lo que indica que el país es un tema central en los textos analizados. Otros unigramas como "país" refuerzan la idea de que el tema del corpus está muy ligado a asuntos nacionales, probablemente con un enfoque significativo en México.

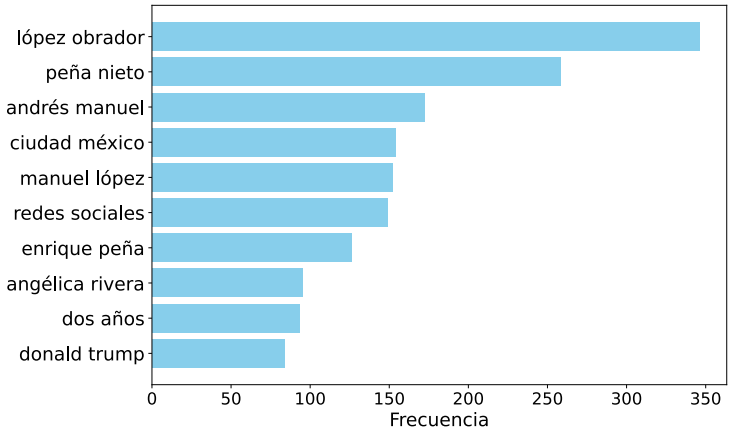


Figura 3.2. Bigramas más repetidos

En los bigramas, los términos más repetidos incluyen nombres de figuras políticas clave, como "López Obrador", que es el más mencionad, seguido de "Peña Nieto" y "Andrés Manuel". Esto sugiere que el corpus tiene un claro enfoque en la política mexicana, con una gran atención en estos personajes. Además, la aparición de "ciudad de México" refleja la importancia de la capital en las discusiones, y términos como "redes sociales", "Ángelica Rivera" y "Donald Trump" indican una mezcla de referencias a plataformas digitales, personalidades políticas y figuras internacionales.

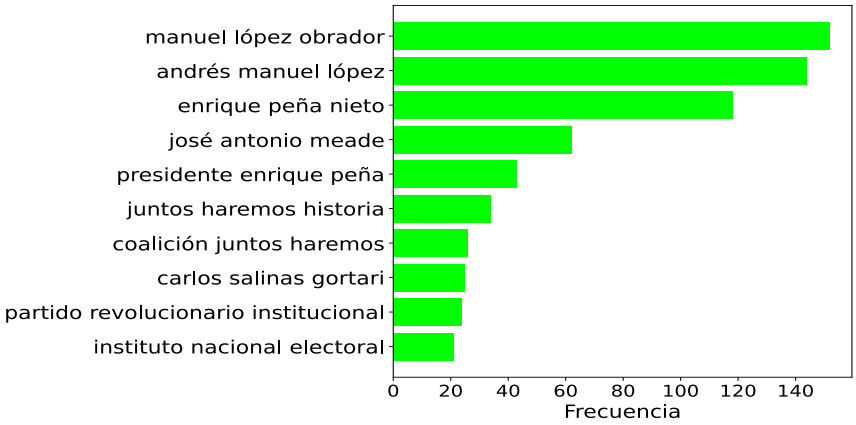


Figura 3.3. Trigramas más repetidos

Finalmente, en los trigramas, se observa que los nombres completos de políticos son los más repetidos, con "Manuel López Obrador", "Andrés Manuel López" y "Enrique Peña Nieto" liderando la lista. Esto refuerza la idea de que la política mexicana es un tema central del corpus. Además, aparecen otros términos políticos como "José Antonio Meade" y "instituto nacional electoral", lo que evidencia que el corpus no solo menciona a los principales actores políticos, sino también a instituciones de gran importancia en el contexto electoral del país.

En conjunto, la repetición de estos N-gramas subraya el enfoque principal del corpus es la política mexicana, haciendo hincapié en nombres, lugares y entidades relacionadas con ella.

Si observamos la nube de palabras del corpus, como se muestra en la figura 3.4 a), podemos identificar los términos más repetidos por su tamaño. Muchas de estas palabras ya han sido destacadas como los N-gramas más frecuentes en las figuras 3.1, 3.2 y 3.3. Al igual que en las gráficas anteriores, notamos que los términos más repetidos incluyen "México", "país" o "López obrador" entre otros.

a) Nube de palabras del corpus

b)Verdaderas

c)Falsas

Figura 3.4. Nube de palabras del corpus separando los conjuntos de noticias verdaderas y falsas

Al comparar las nubes de palabras de las noticias verdaderas y falsas, observamos que en ambas destacan términos como "México" y "país". Sin embargo, las noticias verdaderas incluyen palabras más específicas como "año" y "López Obrador", enfocándose en eventos y figuras concretas. En contraste, las noticias falsas tienen a usar términos más generales como "persona", "ser" y "así", reflejando un estilo menos preciso, lo que sugiere una menor claridad en los detalles frente a las noticias veraces.

3.2 Fine-Tunning del modelo RoBERTa

Para realizar la resolución de la tarea de clasificación de noticias falsas, se han propuesto tres modelos base basados en RoBERTa. A estos tres modelos se le aplicará un *fine-tunning*.

Además, se usarán tres optimizadores diferentes. Los optimizadores se emplean para ajustar los parámetros del modelo durante el entrenamiento, minimizando la función de pérdida y mejorando la precisión del modelo en la tarea de clasificación. Los diferentes optimizadores pueden influir en la rapidez y efectividad del entrenamiento.

Los 3 modelos base basados en RoBERTa son los siguientes:

- **PlanTL-GOB-ES/roberta-large-bne** [56]**:** Es el modelo más grande del conjunto de modelos del lenguaje del Plan de Gobierno de España de Tecnologías del Lenguaje. Ha sido entrenado en el corpus más grande conocido en español, incluye 570GB de texto limpio, recopilado por la Biblioteca Nacional de España entre 2009 y 2019. Su dominio es independiente, es decir, abarca datos desde el ámbito de la literatura, prensa hasta ciencia tecnología y otros ámbitos de conocimiento general.

- **FacebookAI/xlm-roberta-large** [57]**:** Es una versión multilingüe de RoBERTa, preenetrenada con 2,5 TB de datos de CommonCrawl filtrados, que abarcan 100 idiomas. Este modelo abarca una amplia variedad de dominios, como artículos de noticias, blogs, foros, literatura y sitios web generales.

- **PlanTL-GOB-ES/RoBERTalex** [58]**:** Este modelo también del Plan de Gobierno de España de Tecnologías del Lenguaje está entrenado específicamente en el dominio legal español. Utiliza un corpus de 8,9 GB de datos legales, incluyendo textos como legislación, sentencias judiciales y documentos administrativos.

En cuanto a los optimizadores, se usarán los siguientes:

- **Adam:** Este optimizador es ampliamente usado en clasificación binaria debido a su capacidad para adaptar dinámicamente las tasas de aprendizaje según la magnitud de los gradientes de los parámetros. Combina las ventajas del método del gradiente descendente estocástico con un término de *momentun* (método que acelera la convergencia del entrenamiento acumulando el gradiente de iteraciones anteriores para moverse en la dirección adecuada con mayor consistencia), acelerando la convergencia del entrenamiento y mejorando la estabilidad. Adam inicializa y actualiza los momentos de primer y segundo orden durante el proceso de

optimización. Este optimizador y todas sus variantes tienen un hiperparámetro llamado épsilon, se trata de un término muy pequeño que se añade al denominador durante la actualización de los parámetros para evitar divisiones por cero y garantizar la estabilidad numérica. Esto es especialmente importante en las primeras iteraciones o cuando los gradientes son muy pequeños, mejorando la robustez del proceso de optimización.

Una característica importante de Adam es su compatibilidad con la regularización L2. Este método añade una penalización a la función de pérdida para evitar que los pesos del modelo crezcan demasiado. La regularización L2, también conocida como **decaimineto de peso** (*weight decay* en inglés), se expresa de la siguiente manera como podemos ver en la Ecuación 8.

$$f_{reg}(\Theta) = f(\Theta) + \frac{\lambda}{2} \| \Theta \|^2 \tag{8}$$

Donde:

- $f(\Theta)$: Función de pérdida original.
- λ: Hiperparámetro que controla la intensidad de la penalización.
- $\| \Theta \|^2$: Norma L2, que es la suma de los cuadrados de los pesos.

Sin embargo, en Adam, el término de regularización L2 se suma al gradiente durante la actualización de los parámetros (ver Ecuación 9).

$$\Theta_{t+1} = \Theta_t + \eta \cdot (\nabla f(\Theta_t) + \lambda \cdot \Theta_t) \tag{9}$$

Donde:

- $\nabla f(\Theta_t)$: Gradiente de la pérdida respecto a los pesos.
- $\lambda \cdot \Theta_t$: Término de regularización L2.
- η: Tasa de aprendizaje.

Esto puede interferir con las propiedades adaptativas del optimizador, lo que penaliza de manera inconsistente los pesos grandes y afecta la eficacia de la regularización [59].

- **AdamW:** Es una variante de Adam diseñada para mejorar la regularización durante el entrenamiento. A diferencia de Adam, AdamW desacopla la penalización por **decaimiento de peso** del cálculo de los gradientes. En lugar de integrarlo en el gradiente, AdamW aplica directamente la penalización sobre los pesos en la actualización de los parámetros. Esto evita que el cálculo del gradiente sea afectado por la regularización, garantizando una penalización uniforme [60].

  La fórmula de actualización en AdamW se ve representada en la Ecuación 10.

$$\Theta_{t+1} = \Theta_t - \eta \cdot \frac{m_t}{\sqrt{v_t + \epsilon}} - \eta \cdot \lambda \cdot \Theta_t \tag{10}$$

Donde:

- $\frac{m_t}{\sqrt{v_t+\epsilon}}$: Gradiente adaptativo basado en los momentos de primer y segundo orden (m_t y v_t).
- $\lambda \cdot \Theta_t$: Penalización por decaimiento de peso aplicada directamente sobre los pesos.
- η: Tasa de aprendizaje.
- ϵ: Pequeño valor para garantizar la estabilidad numérica.

La regularización en AdamW es más efectiva porque el decaimiento de peso se aplica de manera independiente, evitando las inconsistencias que pueden surgir al integrarlo directamente en los gradientes adaptativos. Esto no solo mejora la estabilidad del entrenamiento, sino que también permite una mayor capacidad de generalización al reducir el sobreajuste, controlando de manera más precisa los presos grandes. Además, AdamW ofrece mayor flexibilidad al permitir un ajuste separado de la tasa de aprendizaje y el término de penalización, lo que facilita la optimización y mejora el rendimiento [61].

- **Adamax:** Es una variante de Adam que simplifica el cálculo al usar solo el momento de primer orden y adapta las tasas de aprendizaje según la norma infinito de los gradientes. A diferencia de Adam, que utiliza momentos de primer y segundo orden, Adamax mejora la eficiencia computacional al eliminar el cálculo del segundo momento. En comparación con AdamW, que incorpora regularización para mejorar la generalización del modelo, Adamax se enfoca exclusivamente en la adaptación dinámica de las tasas de aprendizaje sin agregar términos de regularización adicionales [59].

Para los 3 modelos se han utilizado los siguientes hiperparámetros: el mismo *learning rate* de $2e^{-5}$ recomendado [62] [63], un tamaño del lote de entrenamiento de 8 y un tamaño de lote de validación de 16. Se han seleccionado estos tamaños debido a la disponibilidad de una GPU lo suficientemente potente, como es una L4 de Nvidia proporcionada por los servidores de Google Colab Pro.

Se ha usado un 70% de los datos para el entrenamiento, un 10% para validación y 20% para pruebas. En [64] donde se presenta un estudio el cual analiza la clasificación de tweets informativos relacionados con COVID-29 utilizando modelos como BERT y RoBERTa, esta división permitió obtener los mejores resultados en términos de rendimiento. En concreto, el modelo RoBERTa fue el mejor con un F1-Score de 91,31%.

Por otro lado, en [65] se han explorado técnicas para detectar rumores y noticias falas usando los modelos BERT, RoBERTa y DistilBERT. Se probaron diferentes divisiones del conjunto de datos, incluyendo una división del 70% para entrenamiento, 10% para validación y 20% para pruebas, que demostró ser la más efectiva.

Se ha probado la validación cruzada como técnica de evaluación. Sin embargo, los tiempos de entrenamiento fueron demasiado largos, lo que consumió más recursos de los que se disponían en Google Colab Pro, haciendo inviable completar este enfoque. Los resultados

obtenidos con la validación cruzada fueron similares a los obtenidos con la división de datos planteada. Por esta razón, se ha decidido continuar con la división de datos inicialmente planteada.

Durante el proceso, se estudiarán los valores de pérdida (*loss)* tanto en el conjunto de entrenamiento como el de validación. La pérdida en el conjunto de entrenamiento indicará como el modelo se ajusta a los datos utilizados para el entrenamiento, mientras que la pérdida en el conjunto de validación proporcionará una medida de la capacidad del modelo para generalizar a datos nuevos. Además, se analizará F1-Score en ambos conjuntos para evaluar cuán de bien el modelo está clasificando ambos conjuntos de datos.

Para los 3 optimizadores se usarán como parámetros una épsilon de $1e^{-8}$ y un decaimiento del peso de 0,01. Estos parámetros han demostrado ser efectivos en varios estudios sobre el rendimiento de los optimizadores Adam y sus variantes [59].

Como función de pérdida se usará la función de pérdida de entropía cruzada binaria (en inglés *Binary Cross Entropy Loss)*. Esta función es comúnmente utilizada en problemas de clasificación binaria, donde las etiquetas reales son 0 o 1, y mide la discrepancia entre las probabilidades predichas por el modelo y las etiquetas verdaderas [66].

La entropía cruzada binaria se calcula mediante la fórmula presenta en la Ecuación 11:

$$Loss = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{C}[y_i \cdot \log(p_i) + (1 - y_i)\log(1 - p_i)] \quad (11)$$

Donde:

- N: Número de muestras.
- y_i: Etiqueta verdadera de la muestra i (0 o 1).
- C: Número de clases.
- p_i: Probabilidad predicha para la muestra i.

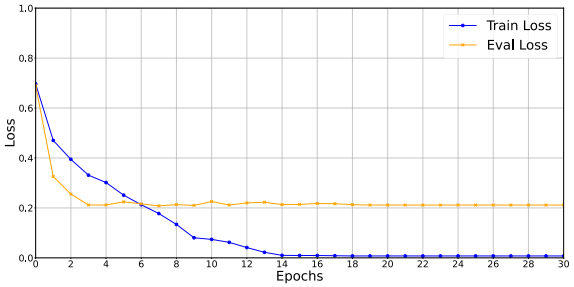
3.2.1 RoBERTa-large-bne

En el análisis de rendimiento de los modelos en el conjunto de test, se observa que el modelo RoBERTa_large_bne + Adam presenta un F1-Score del 78,05%. Por otro lado, el modelo RoBERTa_large_bne + Adamax muestra un rendimiento inferior con un F1-Score del 64,3%. Finalmente, el modelo RoBERTa_large_bne + AdamW destaca con un F1-Score de 83,71%.

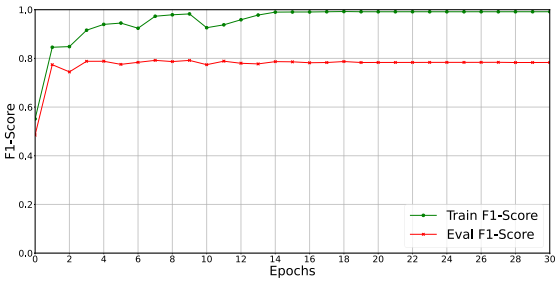
Modelo	**F1-Score**
RoBERTa_large_bne + Adam	78,05 %
RoBERTa_large_bne + Adamax	64,3 %
RoBERTa_large_bne + AdamW	83,71 %

Tabla 3.2. F1-Score RoBERTa-large-bne

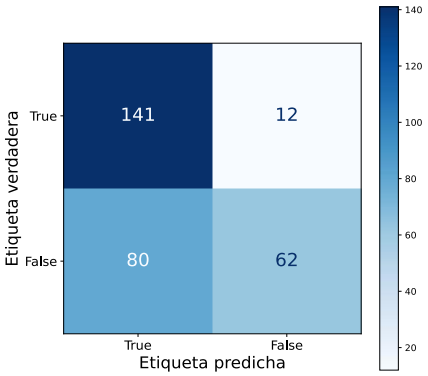
A continuación, se presentan las gráficas que muestran el desempeño de estos modelos, incluyendo la curva de pérdida (loss) y la curva de F1-Score a lo largo de las épocas, así como la matriz de confusión, que se realizará con la mejor época de cada modelo sobre el conjunto de test, que refleja su rendimiento en la clasificación.



a) Curva de pérdida



b) Curva de F1-Score



c) Matriz de Confusión del conjunto de test (Época 7)

Figura 3.5. Resultados RoBERTa-large-bne + Adam

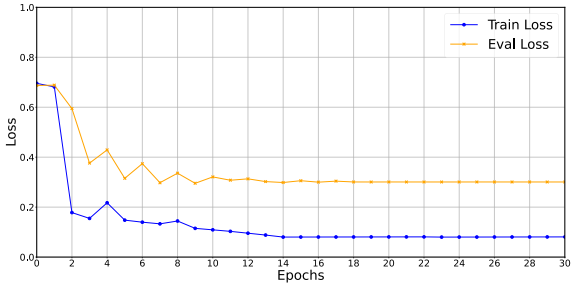
Se puede observar en las gráficas anteriores que el modelo se estabiliza alrededor de la época 9 o 10. En la figura 3.5 a), que representa la curva de pérdida tanto en entrenamiento como en validación, la pérdida en entrenamiento desciende rápidamente y se estabiliza en un valor muy bajo cerca de cero alrededor de la época 10. La pérdida en validación muestra un descenso inicial rápido, pero se estabiliza alrededor de la época 9 en un valor cercano al 0,2.

En la figura 3.5 b), que muestra la curva de F1-Score en el conjunto de entrenamiento y validación, el F1-Score en el conjunto de entreno alcanza una meseta cerca del 100% alrededor de la época 7, y se mantiene estable a partir de ahí. En cuanto al F1-Score en el conjunto de validación, se estabiliza alrededor de la época 10, y no muestra mejoras significativas después de ese punto, manteniéndose constante hasta la época 30. El mejor resultado se alcanza en la época 7 con un 79,4%.

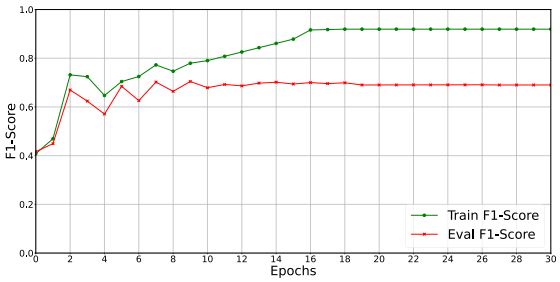
Al observar el F1-Score en el conjunto de test en comparación con los resultados obtenidos en validación, se aprecia una ligera disminución en el rendimiento del modelo. En validación consiguió un 79,4%, mientras que en el conjunto de test este valor bajó a 78,05%. Esta diferencia, aunque pequeña, refleja que el modelo generaliza razonablemente bien a nuevos datos, ya que la caída en rendimiento entre ambos es mínima.

Finalmente, en la matriz de confusión de la figura 3.5 c) se observa un alto número de verdaderos positivos (141), pero también una cantidad considerable de falsos negativos (80), lo que indica que el modelo tiende a clasificar las noticias verdaderas como falsas.

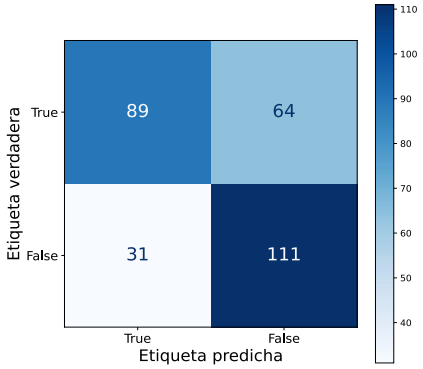
A continuación, se presentarán las gráficas que ilustran el rendimiento del modelo utilizando el optimizador Adamax.



a) Curva de pérdida



b) Curva de F1-Score



c) Matriz de confusión del conjunto de test (Época 7)

Figura 3.6. Resultados RoBERTa-large-bne + Adamax

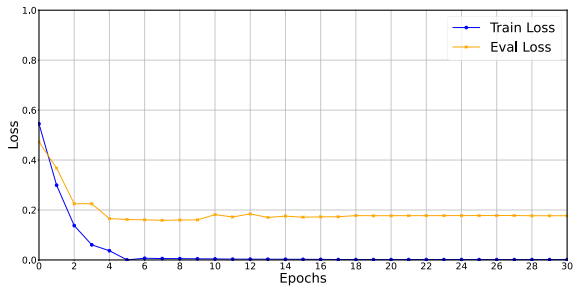
En la curva de pérdida de la figura 3.6 a), el entreno desciende rápidamente hasta estabilizarse en valores cercanos al cero alrededor de la época 14. Sin embargo, la pérdida de validación desciende inicialmente, pero se estabiliza alrededor de la época 16 en un valor aproximado al 0,2.

En cuanto a la curva de F1-Score en la figura 3.6 b), el F1-Score en el conjunto de entreno aumenta de manera consistente y alcanza un valor cercano al 0.9 alrededor de la época 16, donde se estabiliza. Por otro lado, en validación muestra un crecimiento inicial y se empieza a estabilizar en torno a la época 10, sin mostrar mejoras significativas posteriores. A partir de la época 10 se observa que el conjunto de entrenamiento sigue mejorando al contario que el de validación, lo que provoca un sobreajuste del modelo a los datos. El mejor resultado obtenido en el conjunto de validación ha sido en la época 7 con un valor del 70,8%.

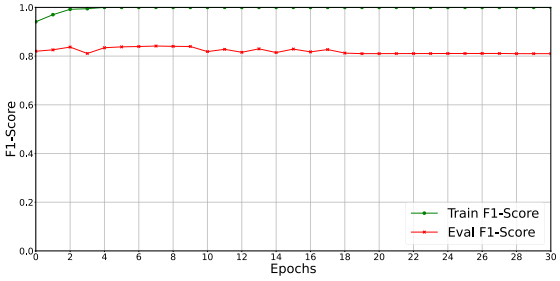
Si miramos el F1-Score en el conjunto de test en comparación con los resultados obtenidos en validación, se aprecia una caída más pronunciada en su rendimiento. En validación se consiguió un 70,8%, mientras que en el conjunto de test este valor bajó a 64,3%. Esta diferencia refleja que el modelo tiene dificultades para generalizar adecuadamente a nuevos datos, evidenciando el problema de sobreajuste.

Por último, la matriz de confusión de la figura 3.6 c) muestra un patrón de clasificación con 89 verdaderas positivos y 64 falsos negativos, además de 111 verdaderas negativos y 31 falsos positivos. Se puede observar con estos resultados que el modelo con Adamax tiende a predecir como falsas noticias veraces al igual que pasaba con el optimizador Adam.

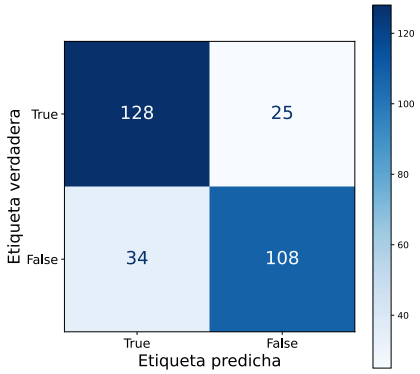
Para concluir las pruebas con este modelo, se mostrarán las gráficas correspondientes al rendimiento del modelo con el optimizador AdamW.



a) Curva de pérdida



b) Curva de F1-Score



c)Matriz de Confusión del conjunto de test (Época 4)

Figura 3.7. Resultados RoBERTa-large-bne + AdamW

Si observamos la gráfica de la curva de pérdida para el conjunto de entrenamiento de la figura 3.7 a), podemos observar que desciende rápidamente y se estabiliza cerca del cero alrededor de la época 5. Sin embargo, la pérdida de validación se estabiliza en torno a la época 4 en un valor de aproximadamente 0.2, lo que sugiere que el modelo ha alcanzado su límite en términos de generalización y ya no mejora en los datos de validación.

Por otro lado, la curva del F1-Score del conjunto de entrenamiento que se muestra en la figura 3.7 b) se estabiliza cerca del 100% a partir de la época 4, lo que muestra un gran rendimiento en el conjunto de entrenamiento. Sin embargo, en el conjunto de validación parece estabilizarse en la época 4, aunque después experimenta varios cambios. No es hasta la época 18 cuando ya no sufre ningún cambio más. El mejor resultado de validación se ha obtenido en la época 7 con un 84,13%.

Al observar el F1-Score en el conjunto de test en comparación con el de validación, se puede apreciar un rendimiento estable y coherente entre ambos conjuntos. En validación, el modelo alcanzó un máximo del 84,13%, mientras que en el conjunto de test este valor se mantuvo

similar, logrando un F1-Score de 83,71%. Esta pequeña diferencia indica que el modelo AdamW generaliza bien a nuevos datos.

La matriz de confusión de la figura 3.7 c) muestra un buen número de verdaderos positivos (128) y verdaderos negativos (108). No obstante, la presencia de 34 falsos positivos y 25 falsos negativos reflejan que aún hay margen de mejora. Estos resultados sugieren que este modelo se ve beneficiado más por el optimizador AdamW que por Adam o Adamax.

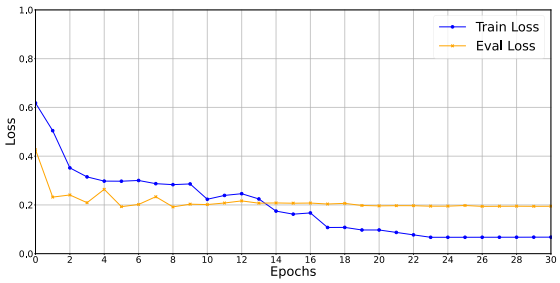
3.2.2 Xlm-RoBERTa-large

En el caso de Xlm-RoBERTa-large observamos que los 3 optimizadores mejoran sus respectivos F1-Score respecto a RoBERTa-large-bne en el conjunto de test, siendo AdamW otra vez el que mejor se adapta a este modelo obteniendo un F1-Score de 85,3%. Al utilizar el optimizador Adam, el modelo obtiene un F1-Score del 79,8%. Sin embargo, al aplicar Adamax, el F1-Score disminuye a 69,8% indicando una menor eficiencia que los otros modelos.

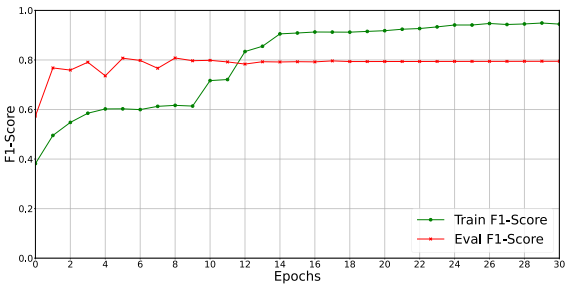
Modelo	**F1-Score**
xlm_RoBERTa_large + Adam	79,1 %
xlm_RoBERTa_large + Adamax	69,8 %
xlm_RoBERTa_large + AdamW	85,3 %

Tabla 3.3. F1-Score Xlm-RoBERTa-large

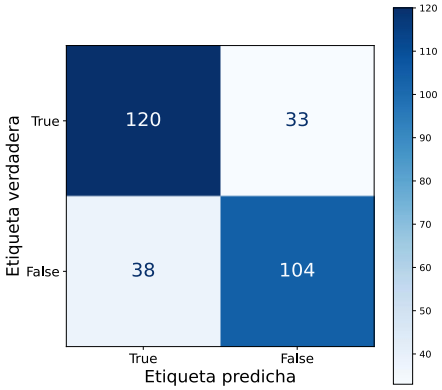
A continuación, se presentarán las gráficas correspondientes al rendimiento del modelo xlm_RoBERTa_large con cada uno de los tres optimizadores, para analizar en detalle cómo evolucionan las métricas de pérdida, F1-Score y matriz de confusión, obtenida con la versión del modelo en la época que mejor resultado ha dado, a lo largo del entrenamiento y validación.



a) Curva de pérdida



b) Curva de F1-Score



c)Matriz de confusión del conjunto de test (Época 10)

Figura 3.8. Resultados xlm_RoBERTa_large + Adam

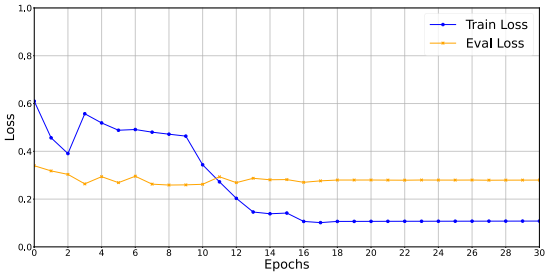
Como se muestra en la curva de pérdida del modelo con Adam en la figura 3.8 a), vemos que la pérdida de entrenamiento desciende rápidamente durante las primeras épocas, estabilizándose alrededor de la época 23 en valores cercanos a cero. En cambio, la pérdida de validación también muestra una reducción inicial, pero se estabiliza en un valor constante a 0,2 a partir de la época 13, sin una mejora notable en las épocas posteriores.

En la curva de F1-Score de la figura 3.8 b), el conjunto de entrenamiento aumenta rápidamente, estabilizándose cerca del 100% alrededor de la época 16. Sin embargo, la precisión de validación presenta una estabilización desde la época 13. A partir de esa época en validación, tanto en pérdida como en F1-Score, este no mejora al contario que el conjunto de entrenamiento, mostrando así un pequeño sobreajuste en los datos.

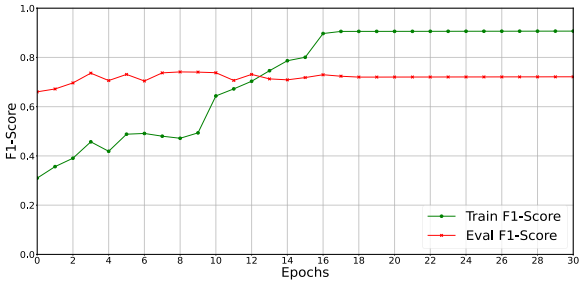
El mejor resultado del conjunto de validación ha sido en la época 10 con un F1-Score de 80,8%. Al compararlo con el conjunto de test, se observa una leve caída en el rendimiento, con un F1-Score de 79,1%, lo que indica que el modelo presenta una buena capacidad de generalización a pesar de ese pequeño sobreajuste que puede haber.

La matriz de confusión de la figura 3.8 c) refleja un buen desempeño general, con 120 verdaderos positivos y 104 verdaderos negativos. Por el contrario, la presencia de 33 falsos negativos y 38 falsos positivos indica que hay errores de clasificación en ambas clases.

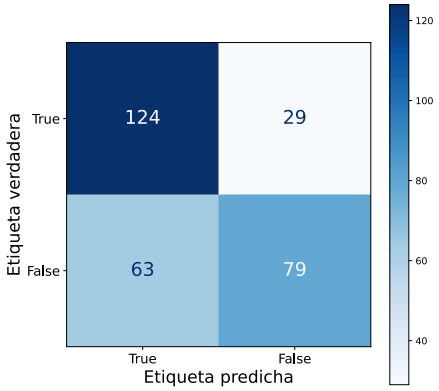
A continuación, se presentarán las gráficas correspondientes al modelo con el optimizador Adamax.



a) Curva de pérdida



b) Curva de F1-Score



c)Matriz de confusión del conjunto de test (Época 8)

Figura 3.9. Resultados xlm_RoBERTa_large + Adamax

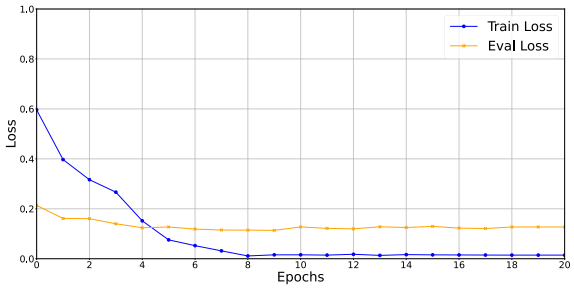
En comparación con los resultados de Adam, la curva de pérdida con Adamax, mostrada en la figura 3.9 a), sigue un patrón similar en el conjunto de entrenamiento, con un descenso continuo a largo de las épocas que se estabiliza en la época 16 en valores cercanos a cero. Por otro lado, la pérdida de validación muestra una reducción inicial, pero se intenta estabilizar en la época 6 pero sufre pequeñas variaciones en las siguientes épocas. No es hasta la época 14 que se estabiliza con un valor cercano a 0,2. Esto sugiere que, aunque el modelo mejora en los datos de entrenamiento, no muestra una mejora significativa en los datos de validación, lo que podría estar indicando un pequeño sobreajuste.

De manera similar, la figura 3.9 b) muestra que la curva de F1-Score para el conjunto de entrenamiento con Adamax crece considerablemente hasta estabilizarse cerca del 90% a partir de la época 16. En cuanto al conjunto de validación, no llega a estabilizarse hasta la época 16, mostrando ligeras variaciones desde la época 7. El mejor resultado obtenido en el conjunto de validación sucede en la época 8 con un F1-Score de 74.1 %.

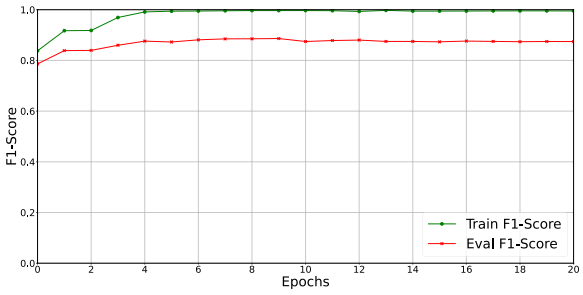
Al comparar el rendimiento del modelo entre el conjunto de validación y el conjunto de test, se observa una caída significativa en el F1-Score, pasando de 74,1% en validación a un 69,8% en test. Este descenso sugiere que, a pesar de que el modelo logra aprender durante el entrenamiento, tiene dificultades para generalizar correctamente a nuevos datos.

La matriz de confusión de la figura 3.9 c) muestra que el modelo presenta 124 verdaderas positivos y 79 verdaderos negativos. No obstante, el modelo ha clasificado 63 falsos positivos y 29 falsos negativos. Esto indica que al igual que pasa con el optimizador Adam, este modelo tiende a predecir noticias falsas como verdaderas.

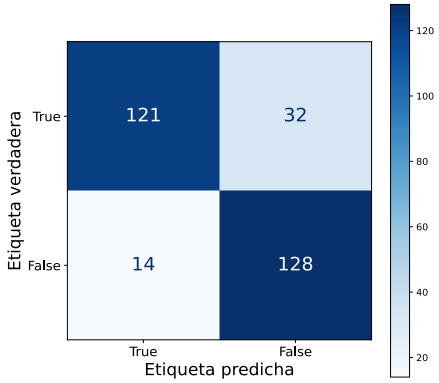
A continuación, se presentan las gráficas correspondientes al modelo con el optimizador AdamW.



a) Curva de pérdida



b) Curva de precisión



c)Matriz de confusión del conjunto de test (Época 9)

Figura 3.10. Resultados xlm_RoBERTa_large + AdamW

La gráfica de pérdida del modelo entrenado con AdamW en el conjunto de entrenamiento de la figura 3.10 a) muestra como desciende rápidamente durante las primeras épocas, estabilizándose en valores cercanos a cero en la época 8. En contraste, en el conjunto de validación también se muestra una reducción inicial, pero se estabiliza en un valor cercano a 0,2 e la época 4, sin experimentar mejoras adicionales. Este comportamiento sugiere que, aunque el modelo mejora considerablemente en el conjunto de entrenamiento, en el conjunto de validación se mantiene sin mejora, lo que puede llevar a un pequeño sobreajuste de los datos. Comparado con Adam y Adamax, donde la estabilización en validación ocurre más tarde, AdamW logra una convergencia más rápida y eficiente.

El F1-Score del modelo con AdamW de la figura 3.10 b) aumenta rápidamente y alcanza valores cercanos al 100% a partir de la época 4 en el conjunto de entrenamiento. Por otro lado, en el conjunto de validación se estabiliza en torno al 87% a partir de la época 4, sin presentar mejoras significativas en las épocas posteriores. El mejor resultado obtenido en el conjunto de validación es en la época 9 con un 88,6% de F1-Score. Si miramos los resultados de los otros optimizadores con este modelo, Adam y Adamax muestran más variaciones y se estabilizan más tarde. Esto muestra que AdamW tiene una mayor consistencia y alcanza un rendimiento superior en menos épocas.

Por otro lado, en el rendimiento del modelo se puede notar un comportamiento bastante estable entre los resultados de validación y test. En validación, con un F1-Score de 88,6%,

mientras que en el conjunto de test se observa una ligera caída, obteniendo un 85,3%. Esta diferencia mínima refleja que el modelo generaliza bien a nuevos datos, manteniendo un rendimiento sólido y coherente entre validación y test.

La matriz de confusión de la figura 3.10 c) muestra que el modelo tiene 121 verdaderos positivos y 128 verdaderos negativos, lo que indica un buen rendimiento en ambas clases. Sin embargo, también presenta 32 falsos negativo y 14 falsos positivos, lo que refleja algunos errores en la clasificación, especialmente en la clase de noticias verdaderas. A pesar de estos errores, el modelo muestra un desempeño sólido.

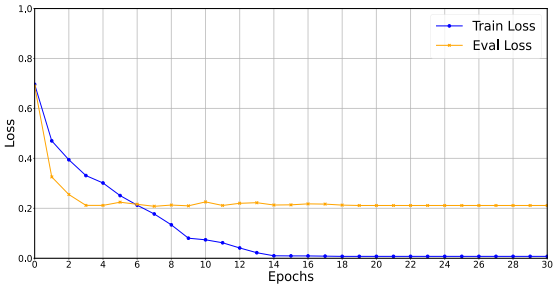
3.2.3 RoBERTalex

En el caso del modelo RoBERTalex, los resultados de F1-Score muestran diferencias significativas en función del optimizador utilizado en el conjunto de test. Con el optimizador Adam, el modelo alcanza un F1-Score de 80,23%. Al aplicar el optimizador Adamax, se observa una caída notable en el rendimiento, con un F1-Score del 61,6%, lo que sugiere que este optimizador es menos efectivo para este modelo en particular. Por otro lado, el mejor rendimiento se obtiene utilizando el optimizador AdamW, con un F1-Score de 81,9%, lo que refleja una ligera mejora con respecto a Adam y una mejor adaptación del modelo a los datos.

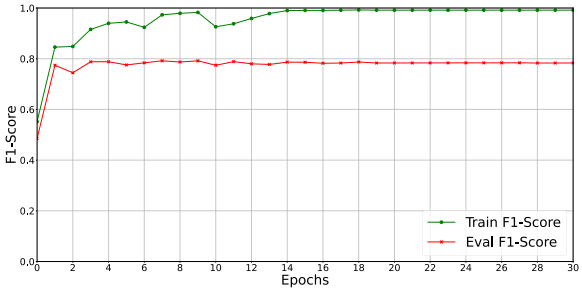
Modelo	**F1-Score**
RoBERTalex + Adam	80,23 %
RoBERTalex + Adamax	61,6 %
RoBERTalex + AdamW	81,9 %

Tabla 3.4. F1-Score de RoBERTalex

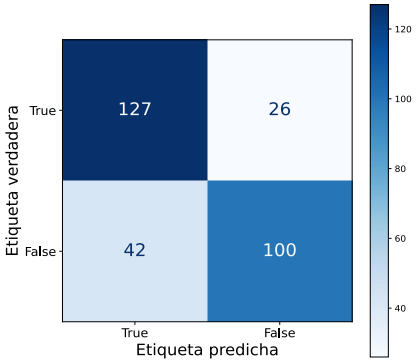
A continuación, se presentarán las gráficas correspondientes al rendimiento de los modelos RoBERTalex con los tres optimizadores.



a) Curva de pérdida



b) Curva de F1-Score



c)Matriz de Confusión del conjunto de test (Época 1)

Figura 3.11. Resultados RoBERTalex + Adam

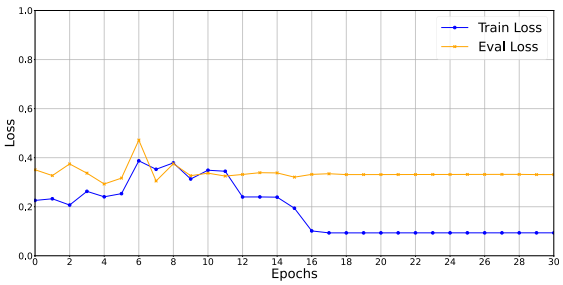
En la gráfica de pérdida de la figura 3.11 a) podemos ver que en el conjunto de entrenamiento se estabiliza a partir de la época 13, alcanzando una meseta cercana a cero. En cambio, en el conjunto de validación se mantiene alrededor de 0,2 desde las primeras épocas, estabilizándose a partir de la época 8, y no experimenta variaciones importantes a lo largo del entrenamiento. Al seguir el modelo mejorando en el conjunto de entrenamiento cuando el modelo se ha estabilizado en el de validación quiere decir que hay sobreajuste del modelo a los datos.

Por otro lado, en la figura 3.11 b) que presenta la curva de F1-Score, se observa que en el conjunto de entrenamiento sube rápidamente y se estabiliza cerca del 100%. Sin embargo, en el conjunto de validación, se estabiliza desde la época 12. Al igual que con la gráfica de pérdida, el modelo sigue mejorando en el conjunto de entrenamiento cuando el de validación ya está estable, por lo que puede haber un pequeño sobreajuste del modelo. El mejor resultado en el conjunto de validación es en la época 1 con un F1-Score de 82,3%.

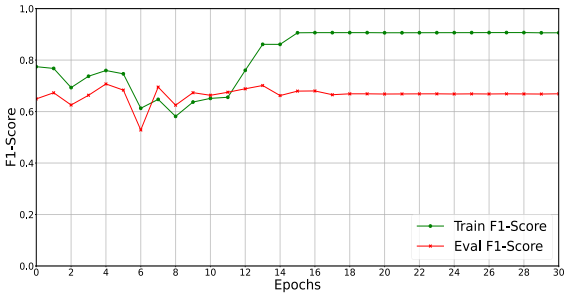
Podemos apreciar una pequeña diferencia entre el F1-Score obtenido en el conjunto de validación y en el conjunto de test. En validación, el modelo logró un 82,3%, mientras que, en el conjunto de test, el rendimiento fue ligeramente inferior, alcanzando un F1-Score de 80,23%.

La matriz de confusión de la figura 3.11 c) muestra que el modelo ha clasificado 127 verdaderos positivos y 100 verdaderos negativos, lo que indica un buen rendimiento en ambas clases. No obstante, también hay 26 falsos negativos y 42 falsos positivos, lo que indica errores de clasificación, especialmente en la clase negativa.

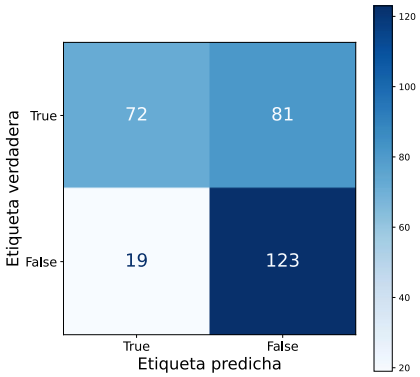
A continuación, se presentan las gráficas correspondientes al modelo con el optimizador Adamax.



a) Curva de pérdida



b) Curva de F1-Score



c)Matriz de confusión del conjunto de test (Época 4)

Figura 3.12. Resultados RoBERTalex + Adamax

Con Adamax la pérdida del conjunto de entrenamiento de la figura 3.12 a) muestra una variación notable durante las primeras épocas y se estabiliza a partir de la época 16. En cambio, la pérdida en el conjunto de validación, aunque comparte un patrón similar al observado con Adam, presenta un comportamiento bastante irregular en las primeras épocas y se estabiliza alrededor de 0,35 a partir de la época 8, sin mostrar grandes mejoras adicionales. Esto también refuerza la posibilidad de un sobreajuste, ya que, al igual que con Adam, el modelo continúa mejorando en el conjunto de entrenamiento sin avances significativos en validación, aunque con una mayor variabilidad inicial en las primeras épocas.

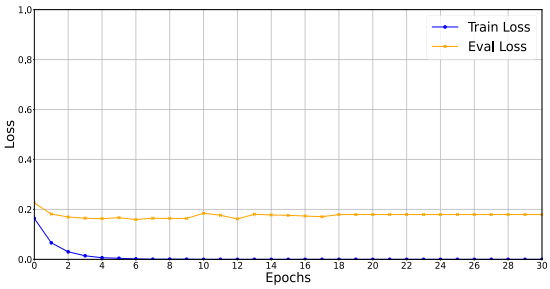
En cuanto a la curva de F1-Score de la figura 3.12 b) muestra que, en el conjunto de entrenamiento, el rendimiento varía durante las primeras épocas y se estabiliza en torno al 90% a partir de la época 15. Sin embargo, el F1-Score del conjunto de validación muestra variaciones importantes en las primeras épocas y se estabiliza a partir de la época 17. Comparado con Adam, el F1-Score en validación de Adamax tiene una estabilización más tardía y con un rendimiento máximo menos, alcanzando su mejor resultado en la época 4 con un F1-Score del 70,7%.

Con respecto al F1-Score, se observa una notable diferencia entre el obtenido en el conjunto de validación y en el conjunto de test. En validación, el mejor resultado fue de 70,7%. Sin embargo, cuando se evalúa el rendimiento en el conjunto de test, el F1-Score disminuye

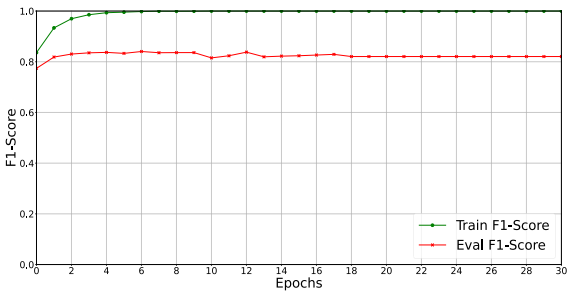
considerablemente a 61,6%, lo que refleja una clara dificultad del modelo para generalizar correctamente nuevos datos.

La matriz de confusión de la figura 3.12 c) refleja este rendimiento inestable, con 72 verdaderos positivos y 123 verdaderos negativos, pero también una cantidad significativa de errores de clasificación, con 81 falsos negativos y 19 falsos positivos. Esto muestra que el modelo tiende a clasificar las noticias verdaderas como falsas.

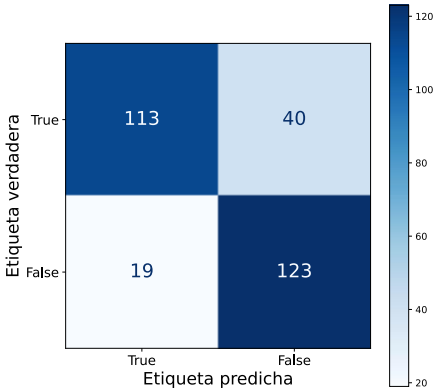
A continuación, se presentarán las gráficas correspondientes al modelo con el optimizador AdamW.



a) Curva de pérdida



b) Curva de precisión



c)Matriz de confusión de conjunto de test (Época 6)

Figura 3.13. Resultados RoBERTalex + AdamW

Teniendo en cuenta las pérdidas de los optimizadores Adam y Adamax, en el caso de AdamW, la curva de pérdida mostrada en la figura 3.13 a), desciende rápidamente en el conjunto de entrenamiento durante las primeras épocas y se estabiliza a partir de la época 4. En contraste, la pérdida en el conjunto de validación muestra una ligera disminución al principio, pero se estabiliza y se mantiene a lo largo de las épocas restantes. A diferencia de Adam y Adamax, AdamW logra una estabilización más temprana en entrenamiento como en validación, lo que indica una mayor eficiencia y menor variabilidad en las etapas iniciales del aprendizaje

En cuanto a la curva de F1 de la figura 3.13 b), se observa que en el conjunto de entrenamiento aumenta rápidamente y alcanza valores cercanos al 100% desde la época 5. Sin embargo, en el conjunto de validación se estabiliza alrededor del 80% desde las primeras épocas, siendo el mejor resultado obtenido en la época 6 con un F1-Score de 83,6%. Si

miramos los resultados de los 3 optimizadores en cuanto al F1-Score, AdamW presenta una estabilización más rápida y una mayor consistencia en el conjunto de validación, esto muestra su capacidad para optimizar de manera eficiente y reducir la variabilidad en el rendimiento entre épocas.

Respecto al F1-Score obtenido en el conjunto de validación frente al conjunto de test, e observa una diferencia leve entre los resultados. En validación, el modelo logró su mejor desempeño con un F1-Score de 83,6%, mientras que en el conjunto de test alcanzó un F1-Score de 81,9%, lo que refleja un rendimiento bastante consistente.

Por otro lado, la matriz de confusión de la figura 3.13 c) revela que el modelo tiene 113 verdaderos positivos y 123 verdaderos negativos, lo que indica un buen rendimiento en ambas clases. No obstante, también se observa una cantidad considerable de errores de clasificación, con 40 falsos negativos y 19 falsos positivos, lo que sugiere que el modelo tiende a clasificar las noticias verdaderas como falsas al igual que pasaba con el modelo con el optimizador Adamax.

3.2.4 Conclusiones

Tras entrenar todos los modelos y evaluar su rendimiento en el conjunto de test, obtenemos la siguiente clasificación representado en la tabla 3.5 donde la mejor combinación modelo + optimizador es la compuesta por Xlm-RoBERTa-large + AdamW con un F1-Score de 85,3%. Este resultado no solo subraya la eficacia de esta combinación en términos de precisión, sino también resalta la capacidad de Xlm-RoBERTa-large para manejar variaciones y complejidades en los datos.

Modelo	**F1-Score**
Xlm-RoBERTa-large + AdamW	85,3 %
RoBERTa-large-bne + AdamW	83,71 %
RoBERTalex + AdamW	81,9%
RoBERTalex + Adam	80,23%
Xlm-RoBERTa-large + Adam	79,1%
RoBERTa-large-bne + Adam	78,05%
Xlm-RoBERTa-large + Adamax	69,8%
RoBERTa-large-bne + Adamax	64,3%
RoBERTalex + Adamax	61,6%

Tabla 3.5. Resultados generales de los modelos

AdamW se destaca como el mejor optimizador, mostrando un rendimiento superior en varios modelos, gracias a su capacidad para evitar el sobreajuste mediante el decaimiento de peso. Además, su consistencia a través de los diferentes modelos subraya su eficiencia general en la mejora del F1-Score. En el conjunto de test, esto se ve reflejado en los resultados obtenidos, donde AdamW proporciona los mejores resultados, manteniendo un buen equilibrio entre la precisión en validación y test.

Por otro lado, los modelos preentrenados en dominios específicos, como RoBERTalex, no alcanzan el rendimiento de los entrenados en corpus más amplios y diversos como Xlm-RoBERTa-large. Esto puede atribuirse a la capacidad de los modelos grandes de generalizar mejor a distintos contextos, lo que los hace más efectivos en tareas de clasificación más complejas.

En primer lugar, en el modelo RoBERTa-large-bne, los tres optimizadores mostraron un comportamiento sólido, pero AdamW resultó ser el más eficiente. La pérdida de entrenamiento mostrada en la figura 3.7a con AdamW refleja como se estabilizó rápidamente alrededor de la época 5, mientras que la pérdida de validación se estabilizó en la época 4. El mejor F1-Score obtenido con este optimizador en el conjunto de validación fue de 84,3% en la época 7 y un 83,71% en el conjunto de test. Estos resultados son indicativos de una buena capacidad de generalización, ya que el modelo no solo se ajustó a los datos de entrenamiento, sino que también mantuvo un rendimiento robusto en el conjunto de validación.

Con Adam, el modelo mostró un buen rendimiento con un F1-Score de 79,4% en validación conseguido en la época 9 como se presenta en la figura 3.5b y un 78,05% en test, mostrando una leve caída en el rendimiento. Adamax, sin embargo, no fue capaz de superar a los otros optimizadores con F1-Score del 70,7% en validación y un 64,3% en test, lo que refleja mayores dificultades para generalizar correctamente los datos nuevos.

Este comportamiento también sugiere que los optimizadores no solo influyen en la velocidad de convergencia, sino que pueden impactar directamente en la capacidad del modelo para aprender patrones relevantes. La elección del optimizador, por lo tanto, se convierte en un aspecto crítico en el diseño del modelo.

Podemos observar que, aunque RoBERTa-large-bne es más pequeño en escala y complejidad en comparación con Xlm-RoBERTa-large, ofrece una alternativa viable para escenarios de restricciones de recursos. Su buen desempeño, especialmente con AdamW, hace que sea una opción atractiva para aquellos que buscan un balance entre rendimiento y costo computacional.

Xlm-RoBERTa-large fue el modelo con mejor rendimiento general, destacándose por su capacidad de capturar patrones diversos gracias a su entrenamiento en un corpus extenso y variado. De los tres optimizadores, AdamW fue el que mejor se adaptó a este modelo, logrando un F1-Score de 88,6% en la época 9 en validación como se ve en la figura 3.10b, con una pérdida de validación que se estabilizó temprano en la época 4, y un 85,3% en test.

Adam también ofreció buenos resultados, alcanzando un F1-Score de 80,8% en el conjunto de validación en la época 10 y un 79,1% en test, mientras que Adamax se quedó con un 74,1% en la época 8 en validación y 69,8% en test, lo que evidencia una menor capacidad para mejorar la generalización del modelo. Estos resultados refuerzan la ventaja de utilizar modelos grandes y generalistas para tareas complejas, especialmente cuando se combinan optimizadores como AdamW.

Dado que este modelo requiere más recursos, es importante considerar los costos computacionales, pero sin duda, Xlm-RoBERTa-large con AdamW proporciona los mejores resultados en términos de rendimiento y robustez.

En cuanto al modelo RoBERTalex, diseñado específicamente para el ámbito jurídico, mostró un rendimiento competitivo, aunque no alcanzó los niveles de Xlm-RoBERTa-large. Aquí,

AdamW volvió a ser el mejor optimizador, con un F1-Score de 83,6% obtenido en la época 6 en el conjunto de validación que se presenta en la figura 3.13b y 81,9% en test, lo que demuestra un buen equilibrio entre entrenamiento y test. La pérdida en validación se estabilizó rápidamente en 0,2, lo que confirma la capacidad de generalización del modelo dentro de su dominio específico.

Con Adam, el F1-Score fue de 82,3% alcanzado en la época 1 en el conjunto de validación como se muestra en la figura 3.11b, mientras que en el conjunto de test se logró un 80,23%. Esto refleja un rendimiento sólido, mientras que Adamax presentó más inestabilidad, con un 70,7% como mejor resultado en la época 4 del conjunto de validación y un 61,6% en test, lo que evidencia una mayor caída en la capacidad de generalización. A pesar de la diferencia en puntuación con los otros modelos, RoBERTalex sigue siendo una opción a considerar para problemas específicos dentro del ámbito jurídico, gracias a su preentrenamiento en datos del dominio.

En resumen, aunque no alcanza el mismo nivel de desempeño que modelos más generalistas como Xlm-RoBERTa-large, sigue siendo una opción valiosa cuando el problema está dentro de su dominio especializado. Sin embargo, para tareas generales, RoBERTa-large-bne ofrece un mejor equilibrio entre rendimiento y costo computacional, siendo una alternativa viable cuando los recursos son limitados.

Por último, en cuanto a las matrices de confusión, calculadas con la versión del modelo en su mejor época en cada caso, se observa de forma general un buen número de verdaderos positivos y verdaderos negativos en la mayoría de los modelos, lo que indica que los modelos son efectivos al clasificar correctamente tanto noticias verdaderas como falsas. Sin embargo, también se presentan errores de clasificación en forma de falsos positivos y falsos negativos, lo que significa que algunos modelos tienden a clasificar noticias falsas como verdaderas y viceversa.

En particular, en los modelos con Adamax, se observa una tendencia más marcada a clasificar incorrectamente, con más falsos negativos en comparación con Adam o AdamW. Esto sugiere una debilidad en su capacidad de reconocer correctamente las noticias verdaderas.

Por otro lado, los modelos con Adam y AdamW presentan un mejor equilibrio, con un menor número de errores de clasificación. El modelo RoBERTalex con Adam muestra 26 falsos negativos y 42 falsos positivos, lo que indica que, aunque presenta algunos errores, su rendimiento es relativamente sólido en comparación con Adamax. Asimismo, el modelo Xlm-RoBERTa-large con AdamW presentado en la figura 3.10c muestra 40 falsos negativos, lo que, aunque no es ideal, sigue siendo un resultado aceptable y refleja una mejor capacidad para generalizar correctamente a nuevos datos.

Tras revisar los resultados obtenidos en los distintos experimentos, se ha decidido que el modelo a utilizar será Xlm-RoBERTa-large combinado con el optimizador AdamW. Esta elección se fundamenta en que esta combinación ha mostrado el mejor rendimiento, alcanzando un F1-Score de 85,3% en el conjunto de test. A pesar de los mayores costos computacionales asociados a esta opción, el excelente desempeño del modelo lo justifica, asegurando que se maximice la efectividad en la clasificación de noticias.

Esta combinación de modelo y optimizador repercute directamente en la mejora de la clasificación de noticias falsas en español, al reducir significativamente los errores de falsos positivos y falsos negativos observados en otros modelos y optimizadores. Esto asegura que

más noticias falsas sean detectadas correctamente, minimizando la posibilidad de que información errónea sea clasificada como verdadera, un aspecto crítico en este proyecto. Además, su capacidad para generalizar eficazmente mejora la robustez del sistema, haciéndolo más confiable frente a nuevos datos, lo cual es fundamental en escenarios donde las noticias falsas pueden adoptar múltiples formatos y estilos lingüísticos.

Capítulo 4

Desarrollo de la aplicación web

En este capítulo se describirá el diseño del sistema de la aplicación web de detección de noticias falsas en español. Se incluirán los requisitos funcionales y no funcionales, el diagrama de casos de uso junto a la descripción de los actores y el diagrama de secuencia del sistema. Además, se detallarán las tecnologías usadas en el desarrollo, tanto en el front-end como en el back-end.

4.1 Tecnologías empleadas

4.1.1 React

React es una biblioteca de JavaScript desarrollada por Facebook para la construcción de interfaces de usuario, especialmente aquellas que requieren una actualización dinámica de datos sin necesidad de recargar toda la página. Su principal característica es el uso de componentes, que son bloques reutilizables de código que representan partes de la interfaz.

Una de las características más potentes de React es su **Virtual DOM (Document Object Model)**. El DOM es la representación estructurada de los elementos de una página web, y actualizarlo directamente puede ser costoso en términos de rendimiento. React optimiza este proceso mediante el uso de Virtual DOM, una copia ligera del DOM real. Cuando un cambio ocurre en la interfaz, React actualiza el Virtual DOM primero, compara las diferencias con el DOM real y solo aplica los cambios necesarios. Este enfoque minimiza las actualizaciones innecesarias, mejorando la velocidad y eficiencia de la aplicación [67].

4.1.2 Node.js

Node.js es un entorno de ejecución para JavaScript que permite ejecutar código JavaScript en el servidor, basado en el motor V8 de Google Chrome. Su arquitectura basada en eventos y su modelo de entrada/salida no bloqueante permiten manejar múltiples operaciones simultáneamente de manera eficiente, lo que es ideal para aplicaciones de alta escalabilidad y rendimiento, como servidores web y servicios de red.

Esta tecnología también es destacada por su ecosistema de paquetes a través de **npm (Node Package Manager)**, que proporciona acceso a una amplia colección de bibliotecas y módulos reutilizables. Esta funcionalidad permite a los desarrolladores integrar fácilmente herramientas y paquetes en sus proyectos, acelerando el desarrollo y asegurando que el software se mantenga actualizado con las últimas mejoras y correcciones. Además, muchas

de las herramientas utilizadas en el desarrollo con React, se ejecutan en el entorno de Node.js, demostrando una gran sintonía entre ambos [68].

4.1.3 CSS

CSS (Cascading Style Sheets) es un lenguaje de estilos utilizados para describir la presentación de los elementos en documentos HTML o XML. Es comúnmente usado para desarrollar el diseño visual e interfaces de los documentos web.

Este lenguaje fue desarrollado por W3C (World Wide Web Consortium) para permitir la separación de los contenidos de los documentos HTML o XML de la propia presentación del documento. Se le denomina estilos en cascada porque las reglas se aplican con un sistema llamada "de cascada", es decir, de arriba abajo y, en el caso de existir ambigüedad, se siguen una serie de normas para resolverla.

CSS tiene una sintaxis bastante sencilla, con gran legibilidad. Éste contiene un conjunto de palabras clave en inglés para especificar los nombres de las propiedades del estilo. Con él es posible definir elementos como bordes, colores, fondos, tipografías, etc [69].

4.1.4 Flask

Flask es un microframework para Python que facilita el desarrollo de aplicaciones web de manera rápida y flexible. Diseñado con una filosofía minimalista, Flask proporciona los componentes básicos necesarios para construir aplicaciones web sin imponer una estructura rígida. Esto permite a los desarrolladores elegir las herramientas y bibliotecas que mejor se adapten a sus necesidades, ofreciendo una gran libertad en el diseño y la implementación de la aplicación.

Una de las características distintivas de Flask es su sistema de rutas sencillo pero potente, que permite definir cómo se deben manejar las solicitudes HTTP y cómo se deben renderizar las respuestas. Además. Flask es compatible con diversas extensiones que agregan funcionalidades como autenticación, manejo de formularios y conexión con bases de datos, lo que facilita la expansión de la aplicación según los requisitos del proyecto [70].

4.2 Requisitos del Sistema

4.2.1 Objetivos de Negocio

El objetivo de negocio para esta aplicación es el siguiente:

- **Predecir veracidad de la noticia**: El objetivo principal del sistema es ayudar a los usuarios a verificar la autenticidad de las noticias ingresadas a través del chat. El sistema permitirá a los usuarios introducir el título y el cuerpo de la noticia, procesará esta información utilizando un modelo de inteligencia artificial, y devolverá si la noticia es verdadera o falsa. Se espera que la aplicación ayude a los usuarios a tomar decisiones más informadas respecto a la veracidad del contenido que consumen.

4.2.2 Objetivos del Sistema

OBJ-0001	**Predecir Noticia**
Versión	1.0 (20/09/2024)
Autor	Rebeca Villalba Calvillo
Descripción	El sistema permitirá a los usuarios verificar si una noticia es verdadera o falsa mediante un modelo de inteligencia artificial a través de un chat.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Hay presión
Estado	Terminado
Estabilidad	Media
Comentario	Ninguno

Tabla 4.1. Objetivo de Sistema 0001 Predecir Noticia

4.2.3 Requisitos Funcionales

Los requisitos funcionales del sistema se presentan en forma de casos de uso, los cuales describen las interacciones entre los usuarios y el sistema para logar ciertos objetivos. Estos casos de uso se detallarán en el apartado correspondiente más adelante en este documento.

4.2.4 Requisitos No Funcionales

NFR-0001	**Interfaz intuitiva**
Versión	1.0 (20/09/2024)
Autor	Rebeca Villalba Calvillo
Descripción	El sistema deberá contar con una interfaz fácil de usar y comprender, dirigida a usuarios de todas las edades y niveles de habilidad.
Importancia	Vital
Urgencia	Hay presión
Estado	Terminado
Estabilidad	Media
Comentario	Ninguno

Tabla 4.2. NFR Interfaz intuitiva

NFR-0002	**Rendimiento óptimo**
Versión	1.0 (20/09/2024)
Autor	Rebeca Villalba Calvillo
Descripción	El sistema deberá tener un rendimiento fluido, asegurando que las respuestas no pueden tardar más de 15 segundos en ser procesadas.
Importancia	Vital
Urgencia	Hay presión
Estado	Terminado
Estabilidad	Media
Comentario	Ninguno

Tabla 4.3. NFR Rendimiento óptimo

NFR-0003	**Compatibilidad con los navegadores**
Versión	1.0 (20/09/2024)
Autor	Rebeca Villalba Calvillo
Descripción	El sistema deberá ser compatible con las versiones más recientes de los navegadores web más utilizados.
Importancia	Vital
Urgencia	Hay presión
Estado	Terminado
Estabilidad	Media
Comentario	Ninguno

Tabla 4.4. Compatibilidad con los navegadores

4.3 Análisis del Sistema

4.3.1 Diagrama de casos de uso

En primer lugar, se presenta una descripción de los potenciales actores que interactuarían con la interfaz web. Estos son "Usuario" y un "Sistema" los cuales se describen a continuación.

ACT-0001	**Usuario**
Versión	1.0 (20/09/2024)
Autor	Rebeca Villalba Calvillo
Fuente	Rebeca Villalba Calvillo
Descripción	Este actor representa al *usuario* que hace uso de la web.
Comentario	Ninguno

Tabla 4.5. Descripción del actor "Usuario"

ACT-0002	**Sistema**
Versión	1.0 (20/09/2024)
Autor	Rebeca Villalba Calvillo
Fuente	Rebeca Villalba Calvillo
Descripción	Este actor representa al *sistema* que analiza la veracidad de las noticias integradas.
Comentario	Ninguno

Tabla 4.6. Descripción del actor "Sistema"

Una vez explicado cada uno de los actores involucrados, se presenta el diagrama de casos de uso, mostrando la funcionalidad que tendrá la interfaz web.

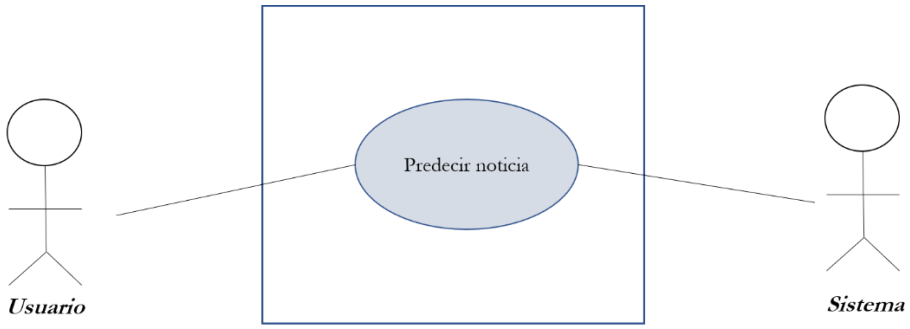


Figura 4.1. Diagrama caso de uso Predecir Noticia.

Como se puede observar en la figura 4.1, hay 2 actores, un "Usuario" y un "Sistema". El usuario interactúa con la web proporcionando una noticia para analizar. El "Sistema", por su parte, procesa la información y se la devuelve al "Usuario".

A continuación, se muestra la narrativa del caso de uso del sistema. Con esto se presenta una descripción básica del caso de uso como la identificación de los pasos que constan el curso normal del caso de uso.

Caso de Uso:	Predecir noticia
Identificador:	UC-0001
Actor:	Usuario
Personal involucrado e intereses:	1. Usuario: El usuario desea predecir si una noticia es verdadera o falsa. 2. Sistema: La Api procesa la noticia y devuelve el resultado.
Precondición:	Ninguna.
Postcondición	El usuario obtiene el resultado de la veracidad de la noticia
Escenario Principal (éxito)	El usuario introduce el título y cuerpo de la noticia, el sistema procesa la información, la clasifica y muestra si la noticia es verdadera o falsa.

Escenario Alternativo (error)	El usuario no inserta ninguna noticia para clasificar.
Escenario Principal:	1. El sistema muestra por pantalla la página inicial con un mensaje de bienvenida y una breve guía sobre cómo introducir las noticias a clasificar. 2. El usuario introduce el texto de la noticia. 3. El usuario confirma el envío de la noticia para su clasificación. 4. El sistema verifica que los datos de la noticia han sido ingresados correctamente. 5. El sistema envía los datos a la API que contiene el modelo IA. 6. La API procesa los datos y clasifica la noticia como verdadera o falsa. 7. El sistema recibe la predicción de la API. 8. El sistema muestra por pantalla la predicción en la interfaz del chat.
Extensiones:	4a. No se introduce ninguna noticia. 4a.1. El sistema detecta que no se ha introducido ninguna información y muestra un error por la interfaz del chat. 4a.2. Vuelta al paso 1.

Tabla 4.7. Caso de uso Predecir noticia

4.3.2 Diagrama de secuencia del sistema

Para mostrar mayor detalle sobre el caso de uso, se muestra a continuación el diagrama de secuencia del caso de uso "Predecir noticia". En este diagrama de secuencia se muestran las líneas de vida de los procesos y los mensajes intercambiados entre ellos.

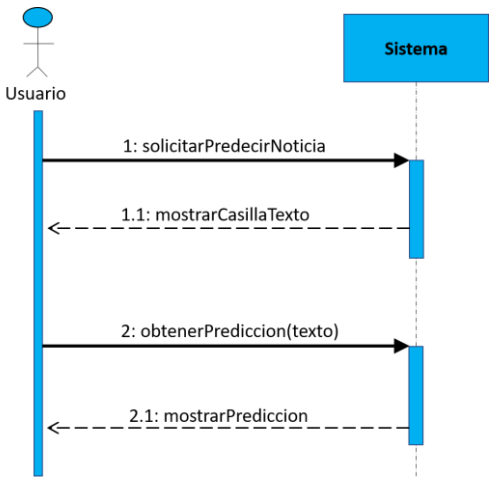


Figura 4.2. Diagrama de secuencia del sistema-Predecir Noticia

4.4 Diseño del sistema

El objetivo principal de esta etapa es crear un conjunto de especificaciones y componentes del sistema que cumplan con los requisitos definidos previamente.

Para lograrlo, presentaremos el diseño de la aplicación, incluyendo su modelo arquitectónico y como se han implementado.

4.4.1 Arquitectura

La arquitectura de un sistema se refiere al diseño y planificación de los diferentes componentes, tanto visuales como funcionales y la interacción entre ellos. En este caso, se empleará la arquitectura clásica cliente-servidor, comúnmente utilizada en la mayoría de aplicaciones web.

Esta arquitectura divide la aplicación en dos módulos principales:

- Front-End: Es la parte encargada de la interfaz de usuario, gestionando elementos visuales como colores, animaciones y estilos. Corresponde al lado del cliente, ejecutándose en el navegador del usuario, permitiéndole interactuar con la aplicación.

- Back-End: Se ocupa de la gestión de datos y la ejecución de funciones del sistema. Procesa las solicitudes, accede a la información y la combina antes de enviarla de vuelta al cliente. Esta parte corresponde al servidor y no es visible para el usuario, ya que no contiene elementos gráficos. Un ejemplo de esta funcionalidad es la conexión con un servidor para recuperar datos de una base de datos.

Para entender cómo se complementan estos módulos representados en la figura 4.3, procederemos a explicar paso a paso el comportamiento de ellos en la aplicación.

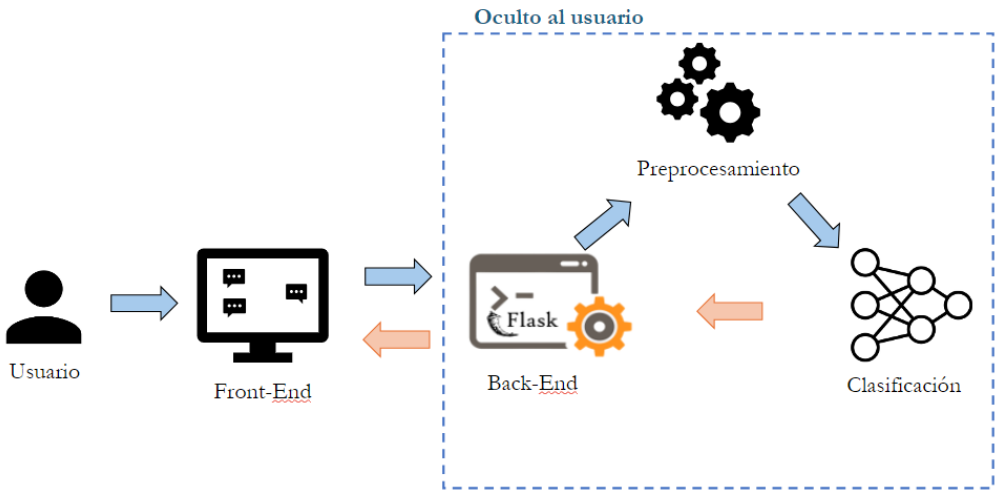


Figura 4.3. Ilustración de la arquitectura del sistema.

1. Cuando el usuario accede a la web, donde el Front-End desarrollado en React se carga e inicia.
2. Al abrir la web, el usuario ve un mensaje de bienvenida en el chat junto con una pequeña guía sobre cómo usar la aplicación.
3. Después de leer la guía, el usuario puede comenzar a interactuar con la interfaz del chat, enviando el texto de la noticia que desea verificar.
4. Una vez suministrada la información, se enviará al Back-End la información para ser analizada.
5. El Back-End recibe el texto de la noticia y valida la información para asegurarse que no esté vacío.
6. Si la información es válida, el Back-End preprocesa el contenido (limpieza, normalización y tokenización) para preparar los datos para el análisis.
7. El contenido preprocesado se envía al modelo de IA RoBERTa, para que clasifique la noticia, prediciendo su veracidad.
8. El Back-End envía la predicción al Front-End.
9. Finalmente, el usuario verá el resultado de la clasificación en la interfaz del chat, indicando si la noticia es verdadera o falsa.

4.5 Interfaz de usuario

La interfaz estará compuesta por varios elementos que facilitan la interacción del usuario con la aplicación. En la parte superior izquierda, se encuentra el logo de la aplicación, representado por un icono de un robot. Junto a este logo, aparece el título "Spanish Fake News Detector".

Debajo del título principal, se encuentra un encabezado grande con el texto "Detecta Noticias Falsas en Español", que capta la atención del usuario y refuerza el objetivo de la aplicación. A continuación, hay una descripción breve, que explica al usuario cómo funciona el sistema. Específicamente, menciona que se debe ingresar el título y el cuerpo de la noticia para que la aplicación pueda analizar si es verdadera o falsa, utilizando un modelo de inteligencia artificial entrenado.

En la parte de la derecha de la interfaz se encuentra el chatbot, el cual será la vía de comunicación entre el usuario y el sistema de predicción. Este chatbot muestra un mensaje de bienvenida donde se enseña al usuario sobre cómo interactuar con el chat. El usuario debe escribir el título y el cuerpo de la noticia para que el sistema determine su veracidad.

En la parte inferior del chat, se encuentra un campo de texto con el marcador de posición "Escribe tu noticia…". Este espacio es donde el usuario puede escribir la noticia que desea verificar. Justo a la derecha de este campo de texto, se halla el botón de envío, representado por un icono de un avión de papel. Al hacer clic en este botón o presionar la tecla "intro", el mensaje se envía al sistema para ser procesado y analizado por el modelo de inteligencia artificial.

Cuando el usuario envía la noticia, el Front-End toma el contenido que el usuario ha escrito y lo envía al Back-End mediante una solicitud HTTP POST. Esta solicitud contiene la noticia, que es recibida por la API.

Una vez el Back-End recibe la noticia, el primer paso es comprobar que no está vacía. En caso de estarlo se envía un mensaje de error de vuelta al Front-End para mostrarlo en el chat y avisar al usuario que no ha enviado ninguna noticia.

Después de haberse comprobado que la noticia no está vacía, pasa a la fase de preprocesamiento. Durante esta fase, se limpia el texto eliminando tildes, acentos y caracteres especiales que pueden interferir con el análisis posterior. Este proceso asegura que el texto esté normalizado, lo que facilita el trabajo del modelo de IA. Tras la limpieza, el texto es tokenizado, lo que significa que se convierte en una secuencia numérica comprensible para el modelo RoBERTa.

Con el texto ya procesado y tokenizado, el modelo RoBERTa se encarga de analizar el contenido. Si la noticia es clasificada como verdadera, se elige una frase del conjunto de frases verdaderas predefinidas que tiene el sistema que indica su autenticidad, mientras que si se clasifica como falsa, se selecciona una frase del conjunto de frases falsas que indica que la noticia no es confiable.

El Back-End, después de procesar y clasificar la noticia, devuelve esta respuesta al Front-End, específicamente a la ventana del chat. El Front-End se encarga de mostrar la frase elegida, ya sea indicando que la noticia es verdadera o falsa, lo que permite que el usuario vea instantáneamente la evaluación de la veracidad de la noticia que introdujo.

La interfaz debe utilizar una combinación de colores y fuentes de letra que no solo capten la atención del usuario, sino también aseguren un contraste adecuado para facilitar la navegación y la legibilidad. En la figura 4.4 se observa el diseño de la interfaz descrita.

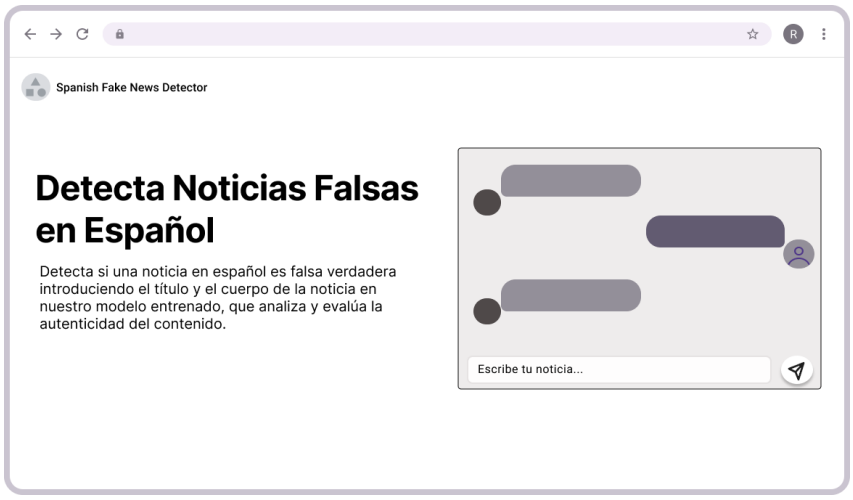


Figura 4.4. Diseño de la interfaz de usuario

Para el fondo general de la página se ha usado un gradiente en colores pastel con una transición suave entre colores cálidos y fríos, lo que crea una atmósfera amigable y atractiva sin distraer. Este gradiente incluye colores como el rosa, amarillo, azul y lila que se mezclan gradualmente, lo cual mantiene la atención del usuario sin resultar abrumador.

Para el chat se ha usado un cian muy claro (#e9f3ff) que proporciona una sensación de suavidad y claridad visual, mientras que los mensajes del usuario y del Bot tienen colores

diferenciados. En el caso del Bot se ha empleado una sombra clara de verde-cian (#b6e9de) y una sombra clara azul-magenta para el usuario. Esto ayuda al usuario a distinguir fácilmente entre las respuestas del sistema y sus propias interacciones. Finalmente, el resultado tras la implementación es el que se observa en la figura 4.5.

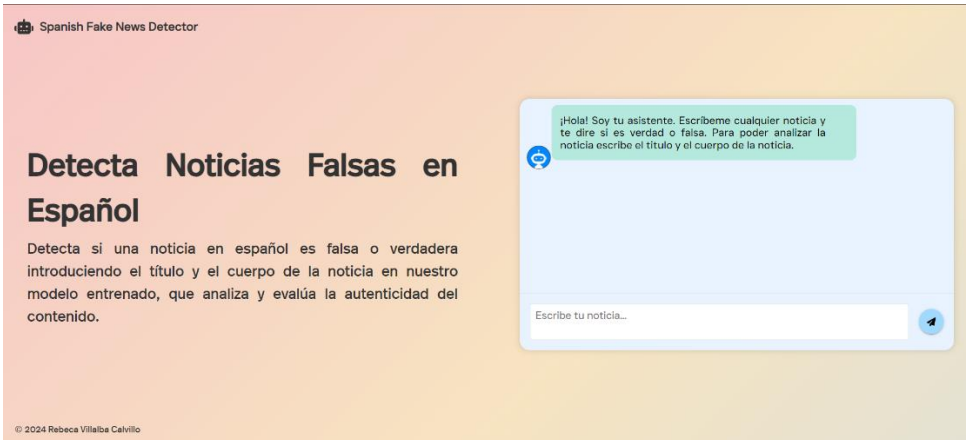


Figura 4.5. Resultado final de la web

Capítulo 5

Pruebas

Para asegurar el correcto funcionamiento del sistema de detección de *fake news* en español, se realizarán diversas pruebas estándar utilizadas en procesos de ingeniería del software. Con estas pruebas, se completará todo el trabajo relacionado con el desarrollo del sistema de detección de noticias falsas en español.

5.1 Pruebas Funcionales

5.1.1 Prueba de RF01 Introducir Noticia.

Se realizarán tres pruebas de caja negra para el requisito funcional "Predecir noticia".

Prueba 1. Introducción de entrada de texto vacía.

La primera prueba de caja negra para este requisito funcional consistirá en introducir un texto en blanco para comprobar si el sistema intenta predecir la noticia o bien avisa al usuario de que ha de introducir al menos una palabra.

Figura 5.1. Prueba del requisito "Predecir noticia" - Prueba 1

Como se puede observar en la figura 5.1, el sistema avisa al usuario de que ha introducido una cadena de texto vacía y debe introducir al menos una palabra.

Prueba 2. Introducción de una noticia verdadera.

La segunda prueba de caja negra para este requisito consistirá en introducir una noticia verdadera para comprobar si el sistema predice correctamente la noticia y notifica al usuario correctamente.

Figura 5.2. Prueba del requisito "Predecir noticia" - Prueba 2

Como se puede observar en la figura 5.2, el sistema notifica correctamente al usuario que la noticia es verdadera mediante una de las frases que hay predefinidas para el caso de noticias verdaderas.

Prueba 3. Introducción de una noticia falsa.

La tercera prueba de caja negra consistirá en introducir una noticia falsa para comprobar que el sistema la clasifica correctamente y notifica al usuario.

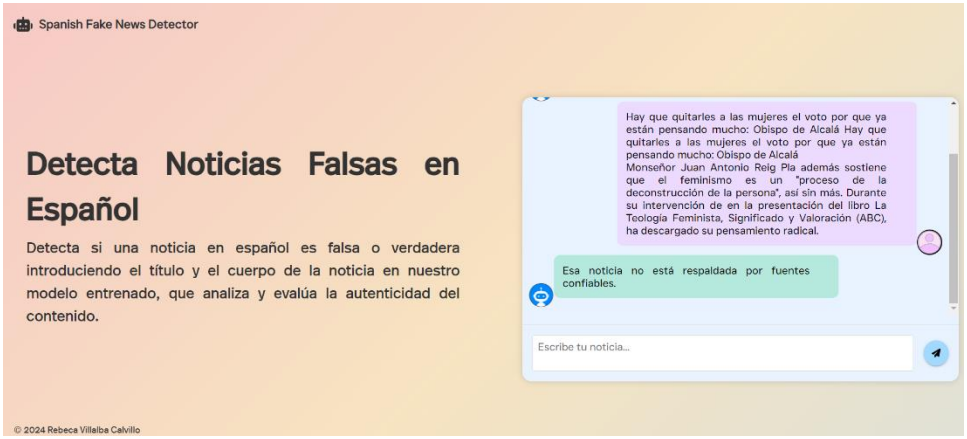


Figura 5.3. Prueba del requisito "Predecir noticia" - Prueba 3

Como se puede observar en la figura 5.3, el sistema notifica correctamente al usuario que la noticia es falsa mediante una de las frases que hay predefinidas para el caso de noticias falsas.

5.2 Pruebas No Funcionales

5.1.1 Prueba de NFR-0001

Objetivo: Asegurar que al interfaz de usuario de la plataforma sea intuitiva y fácil de usar para todos los usuarios. La interfaz debe ofrecer navegación clara y accesibilidad a todas las funcionalidades clave del sistema, facilitando una experiencia de usuario óptima.

Resultados: Se realizaron pruebas de usabilidad con 10 personas de diferentes rangos de edad, comprendidos entre 15 y 83 años, con el objetivo de asegurar que al interfaz de usuario de la plataforma sea intuitiva y fácil para todos los usuarios. Durante la prueba los usuarios interactuaron con el sistema introduciendo varias noticias en el chat de la web. Los resultados fueron positivos en todos los casos, los participantes no tuvieron problemas para acceder ni para usar la plataforma, completando las tareas propuestas sin dificultades.

5.1.2 Prueba de NFR-0002

Objetivo: Asegurar que el sistema deberá tener un rendimiento fluido, asegurando que las respuestas no tardarán más de 15 segundos en ser procesadas.

Resultados: Se realizaron pruebas de rendimiento para asegurar que el sistema responde de manera fluida y que las respuestas no tardan más de 15 segundos en ser procesadas. Se probaron 100 noticias de distintas longitudes, incluyendo textos cortos y largos, con resultados positivos en todos los casos. La mayoría de las noticias fueron procesadas en un tiempo promedio de 4 a 5 segundos, mientras que las más largas alcanzaron un tiempo máximo de 10 segundos. Esto demuestras que el sistema cumple con el objetivo planteado.

5.1.3 Prueba de NFR-0003

Objetivo: El sistema deberá ser compatible con las versiones más recientes de los navegadores web más utilizados, para garantizar una experiencia de usuario óptima. Los navegadores específicos a probar son Google Chrome, Mozilla Firefox, Microsoft Edge, Opera y Brave.

Procedimiento: La aplicación se ejecutó en modo local, pero fue probada en múltiples navegadores. Esto simula cómo los navegadores interpretarían la aplicación en un entorno de producción.

Resultados: Las pruebas de compatibilidad con navegadores se llevaron a cabo satisfactoriamente. La plataforma demostró ser completamente funcional en los cinco navegadores en los que ha sido probada.

5.3 Conclusiones

En los capítulos 4 y 5, se ha desarrollado una aplicación web destinada a comprobar el correcto funcionamiento del modelo y ofrecer al usuario una forma cómoda de interactuar con él. Esta aplicación web se ha llevado a cabo siguiendo un proceso típico de ingeniería del software.

Primero, se definieron los requisitos del sistema en la fase de análisis. Luego, se abordó el diseño de la arquitectura del sistema y de sus diferentes componentes software. También se diseñó la interfaz web. Finalmente, en este capítulo se detallan las pruebas de caja negra que validan y verifican su correcto funcionamiento.

De este modo, se presenta una solución web para el usuario para la detección de noticias falsas en español.

Capítulo 6

Conclusiones y trabajo futuro

6.1 Conclusiones

Los objetivos planteados para este proyecto se centraron en abordar desafíos relacionados con la detección de noticias falsas en español, desde el análisis inicial del problema hasta la implementación de un sistema funcional.

Se ha diseñado e implementado un sistema basado en inteligencia artificial y PLN para la identificación automática de *fake news*. Este sistema, utilizando modelos avanzados de aprendizaje profundo, se entrenó exclusivamente con datos textuales (título y cuerpo de las noticias), descartando la necesidad de información adicional.

El primer reto que encontré era analizar el estado del arte del problema, investigando el panorama actual en cuanto a la detección de desinformación, el campo del PLN y *DeepLearning*. Se llevó acabo una investigación exhaustiva de la literatura, revisando los diferentes enfoques que se han ido tomando a largo de los años. Enfoques tradicionales como *Bag of Words* hasta el uso de modelos de lenguaje avanzados como los que vemos a día de hoy y con los cuales estamos mucho más relacionados.

Seguido de esto, se debía evaluar y seleccionar el conjunto de datos más óptimo para el desarrollo del proyecto. Se analizaron varios corpus de datos, evaluando su adaptabilidad al problema. Finalmente, se seleccionaron corpus que contenían tanto títulos como cuerpos de las noticias, ya que son la base de la resolución del problema.

Uno de los mayores retos alcanzados ha sido la correcta selección y preprocesamiento de los datos en español, dada la variedad de estilos y registros lingüísticos presentes en los medios hispanohablantes. Sin embargo, este desafío no fue el único. Un aspecto igualmente complejo fue aprender y comprender las arquitecturas de los modelos de lenguaje avanzados como RoBERTa. Esto implicó estudiar en profundidad cómo funcionan estos modelos, cómo están diseñados para procesar texto y cómo adaptarlos eficazmente a la tarea específica de detección de noticias falsas. Este aprendizaje incluyó aspectos como el *fine-tunning* y la configuración de los parámetros necesarios para obtener dichas predicciones.

Además, fue necesario adquirir conocimientos sobre los optimizadores, como Adam y sus variantes, para entender su papel en el entrenamiento de los modelos. Esto incluyó explorar cómo funcionan, cuando utilizar cada tipo y cómo seleccionar los hiperparámetros más adecuados. En este proceso, se evaluaron diferentes configuraciones hasta identificar que la mejor combinación resultó ser Xlm-RoBERTa-large con el optimizador AdamW, la cual alcanzó un F1-Score de 85,3%. Este resultado refleja la capacidad de los modelos preentrenados en corpus amplios y diversos para generalizar mejor y abordar tareas de clasificación complejas, como la detección de noticias falsas.

El uso de optimizadores como AdamW ha mostrado ser crucial para mejorar el rendimiento del sistema y evitar grandes problemas de sobreajuste. A pesar de que modelos como

RoBERTalex, preentrenados en dominios específicos, han demostrado ser competitivos, no logran superar a los modelos generalistas en términos de precisión y capacidad de generalización.

Finalmente, con el objetivo de hacer accesible el sistema desarrollado a un público más amplio, se ha implementado una plataforma web interactiva. Esta web permite a los usuarios analizar noticias en español y descubrir su veracidad de manera automatizada. A través de esta interfaz, cualquier usuario puede introducir el texto de la noticia, y el sistema, basado en Xlm-RoBERTa-large con AdamW, determinará si se trata de una noticia verdadera o falsas.

En conclusión, el proyecto ha logrado cumplir todos los objetivos definidos. Desde el análisis del problema hasta la implementación de una solución accesible, se ha desarrollado un sistema completo y efectivo para la detección de noticias falsas en español, que combina una sólida base teórica con herramientas avanzadas de *Deep Learning* y una interfaz diseñada para usuarios finales.

6.2 Valoración Personal

Este Trabo de Fin de Grado ha sido un reto significativo, ya que me he adentrado en el campo del PLN, un área que está en constante evolución en la actualidad. Aprender a utilizar modelos de inteligencia artificial para detectar noticias falsas ha requerido esfuerzo y dedicación. Hasta este proyecto, mi experiencia se limitaba a trabajar con algoritmos clásicos, por lo que esta ha sido mi primer proyecto con modelos avanzados de lenguaje. El proceso de *fine-tunning,* que nunca había realizado antes, fue especialmente desafiante, ya que implicó adaptar un modelo preentrenado a una tarea específica.

La necesidad de comprender cómo funcionan estos modelos, así como los conceptos detrás del PLN, ha sido un proceso desafiante, pero gratificante. Además, el hecho de poder llegar a entender cómo funcionan los modelos de lenguaje más recientes como GPT hacen que este proyecto sea aún más significativo.

Me he enfrentado a la tarea de analizar y seleccionar los mejores modelos y su correcta optimización para lograr los mejores resultados, lo que ha mejorado mi capacidad para tomar decisiones técnicas.

Además, el desarrollo de una interfaz web completamente funcional ha sido otro gran desafío. Hasta ahora, nunca había completado una aplicación de principio a fin, lo que ha sido una gran experiencia. Esto me ha permitido aplicar mis conocimientos técnicos de una manera práctica y entender la importancia de la usabilidad para los usuarios finales. La creación de esta interfaz no solo implicó programación, sino también considerar cómo los usuarios interactúan con el sistema.

Considero que esta experiencia ha ampliado mis aptitudes no solo en mi campo, sino también en otras ramas relacionadas, algo que considero vital para cualquier ingeniero en la actualidad. La capacidad de abordar un proyecto integral, desde el desarrollo del sistema hasta su implementación en una plataforma accesible, me ha proporcionado una visión más completa y multidisciplinaria que será de gran ayuda en mi futuro profesional.

En conjunto, este proyecto me ha preparado para enfrentar proyectos futuros en el ámbito de la inteligencia artificial y el desarrollo de software de manera más completa y eficiente.

6.3 Trabajo futuro

En esta sección se abordan los posibles trabajos futuros relacionados tanto con la investigación e implementación de modelos de *Deep Learning* para la tarea de detección de *fake news* como para posibles mejoras de la interfaz web implementada para visualizar el funcionamiento de estos modelos. A continuación, se describen estos posibles trabajos futuros.

- **Análisis de veracidad mediante URLs:** Actualmente, el sistema solo admite texto ingresado manualmente. Una mejora importante sería permitir el análisis automático del contenido de una noticia a partir de su URL, lo que facilitaría la detección de *fake news* en sitios web.

- **Detección en redes sociales:** Ampliar el sistema para detectar notificas falsas en redes sociales permitiría analizar el contenido y las interacciones en plataformas como Twitter y Facebook, lo cual es crucial dada la rápida propagación de desinformación en estos entornos.

- **Empleo de modelos LLM:** Modelos más avanzados con Llama 3 de Meta ofrecen un gran potencial para mejorar la precisión, pero debido a su alta demanda de recursos, no han sido implementados en este proyecto.

- **Implementación de sistemas de retroalimentación:** La retroalimentación de los usuarios sería de una mejora importante para ajustar y mejorar continuamente el sistema. Permitir a los usuarios marcar noticias como incorrectamente clasificadas o proporcionar información adicional sobre noticias sospechosas ayudaría a refinar el modelo con el tiempo.

Referencias

[1] J. Holdsworth, «IBM.com,» 6 Junio 2024. [En línea]. Available: https://www.ibm.com/topics/natural-language-processing. [Último acceso: 8 Junio 2024].

[2] A. Bhattacharya, «NLP and text mining: A natural fit for business growth,» sentisum.com, [En línea]. Available: https://www.sentisum.com/library/nlp-and-text-mining. [Último acceso: 15 Abril 2024].

[3] J. Figueras, P. Enériz y A. Solà, «¿Por qué surgen 'fake news' en los conflictos armados?,» *elPeriódico,* 26 Octubre 2023.

[4] «fake news - Explorar- Google Trends,» [En línea]. Available: https://trends.google.es/trends/explore?date=2019-04-14%202024-04-14&q=fake%20news. [Último acceso: 14 Abril 2024].

[5] «Cambridge Analytica – Data drives all that we do,» [En línea]. Available: https://web.archive.org/web/20171113180809/https://cambridgeanalytica.org. [Último acceso: 15 Abril 2024].

[6] A. J. ARRATIBEL, «El peligroso impacto de los bulos en las vidas de los hispanohablantes de Estados Unidos,» *EL PAÍS,* 26 Diciembre 2023.

[7] Q. Petit, «El desinterés por las noticias alcanza un récord histórico global, según el Instituto Reuters,» *El País,* p. 2024, 17 Junio 2024.

[8] «GanttProject,» [En línea]. Available: https://www.ganttproject.biz/. [Último acceso: Marzo 29 2024].

[9] G. d. España, Resolución de 16 de enero de 2024 por la que se publica el salario anual de los analistas programadores., Boletín Oficial del Estado, 2024.

[10] C. Rodríguez Pérez, «No diga fake news, di desinformación: una revisión sobre el fenómeno de las noticias falsas y sus implicaciones,» *Dialnet,* 2019.

[11] «Fundéu BBVA: "noticias falsas" o "falseadas", mejor que "fake news",» *LA VANGUARDIA,* 28 Septiembre 2017.

[12] C. Wardle, «Fake news. It's complicated.,» *Medium,* 2017.

[13] «Cambridge Dictionary,» [En línea]. Available: https://dictionary.cambridge.org/us/dictionary/english/hate-speech. [Último acceso: 20 Abril 2024].

[14] «Misinformation and disinformation,» Julio 2022. [En línea]. Available: https://www.apa.org/topics/journalism-facts/misinformation-disinformation. [Último acceso: 18 Mayo 2024].

[15] «Stemming,» botpenguin.com, [En línea]. Available: https://botpenguin.com/glossary/stemming. [Último acceso: 15 Abril 2024].

[16] «Lemmatization,» botpenguin.vom, [En línea]. Available: https://botpenguin.com/glossary/lemmatization. [Último acceso: 15 Abril 2014].

[17] J. Barnard, «IBM.com,» 23 Enero 2024. [En línea]. Available: https://www.ibm.com/topics/word-embeddings. [Último acceso: 8 Junio 2024].

[18] «IBM.com,» [En línea]. Available: https://www.ibm.com/es-es/topics/convolutional-neural-networks. [Último acceso: 29 Junio 2024].

[19] «IBM.com,» [En línea]. Available: https://www.ibm.com/es-es/topics/recurrent-neural-networks#:~:text=%C2%BFQu%C3%A9%20es%20una%20RNN%3F,conclusiones%20basadas%20en%20entradas%20secuenciales.. [Último acceso: 29 Junio 2024].

[20] S. Hochreiter y J. Schmidhuber, «Long Short-term Memory,» doi: 10.1162/neco.1997.9.8.1735, 1997.

[21] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cisac y T. Rault, «Transformers: State-of-the-Art Natural Language Processing,» arXiv preprint arXiv:1910.03771v5, 2019.

[22] A. Lee, «DataSource.ai,» 20 Mayo 2020. [En línea]. Available: https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python. [Último acceso: 2 Julio 2024].

[23] M. Oren, M. Hassid, Y. Adi y R. Schwartz, «Transformers are Multi-State RNNs,» arXiv preprint arXiv:2401.06104v2, 2024.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones y A. N. Gomez, «Attention Is All You Need,» arXiv preprint arXiv:1706.03762v7, 2017.

[25] J. Espigule-Pons, «The Encoder-Decoder Transformer Neural Network Architecture,» [En línea]. Available: https://community.wolfram.com/groups/-/m/t/2913235. [Último acceso: 16 Abril 2024].

[26] D. Ibáñez, «Baeldung.com,» 18 Marzxo 2024. [En línea]. Available: https://www.baeldung.com/cs/transformer-text-embeddings. [Último acceso: 29 Junio 2024].

[27] O. Espejel, «Huggingface.com,» 23 Junio 2022. [En línea]. Available: https://huggingface.co/blog/getting-started-with-embeddings?sc_channel=el&sc_content=fullstack-llm-langchain-chatbot-on-aws&sc_country=mult&sc_geo=mult&sc_outcome=acq. [Último acceso: 29 Junio 2024].

[28] «Sbert.net,» [En línea]. Available: https://sbert.net. [Último acceso: 30 Junio 2024].

[29] P. Huet, «OpenWebinars.net,» 12 Julio 2024. [En línea]. Available: https://openwebinars.net/blog/embeddings/. [Último acceso: 18 Julio 2024].

[30] K. He, X. Zhang, S. Ren y J. Sun , «Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778,» arXiv preprint arXiv:1512.03385v1, 2016.

[31] J. Ferrer, «Datacamp.com,» 9 Enero 2024. [En línea]. Available: https://www.datacamp.com/tutorial/how-transformers-work. [Último acceso: 29 Junio 2024].

[32] J. Camacho-Collados y M. Taher , Embeddings in natural language processing: Theory and advnaces in vector representation of meaning. Synthesis Lectures on Human Language Technologies, 13(4):1–175, Springer, 2021.

[33] A. Radford, K. Narasimhan, T. Salimans y I. Sutskever, «Improving Language Understanding,» 2018.

[34] «Introduction to Generative Pre-trained Transformer (GPT),» GeeksforGeeks, 12 Julio 2024. [En línea]. Available: https://www.geeksforgeeks.org/introduction-to-generative-pre-trained-transformer-gpt/. [Último acceso: 20 Julio 2024].

[35] J. Ye, X. Chen, N. Xu, Z. Shao, S. Liu, Y. Cui, Z. Zhou, C. Gong y Y. Shen, «A Comprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models,» arXiv preprint arXiv:2303.10420v2, 2023.

[36] O. J. Achiam, S. Adler, L. Ahmad, F. Leoni y D. Almeida, «GPT-4 Technical Report,» 2024.

[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis y L. Zettlemoyer, «Roberta: A robustly optimized BERT pretraining approach.,» arXiv preprint arXiv:1907.11692v1, 2019.

[38] J. Devlin, M.-W. Chang, K. Lee y K. Toutanova, «Bert: Pre-training of deep bidirectional transformers for language understanding.,» arXiv preprint arXiv:1810.04805, 2018.

[39] U. Khalid, M. Beg y M. Umair Arshad, «RUBERT: A Bilingual Roman Urdu BERT Using Cross Lingual Transfer Learning,» researchgate, 2021.

[40] J. Gatto, «An Overview of the Various BERT Pre-Training Methods,» [En línea]. Available: https://medium.com/analytics-vidhya/an-overview-of-the-various-bert-pre-training-methods-c365512342d8. [Último acceso: 17 Abril 2024].

[41] J. Devlin, M.-W. Chang, K. Lee y K. Toutanova, «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,» arXiv:preprint arXiv:1810.04805v2 , 2019.

[42] D. Cortiz, «Exploring Transformers in Emotion Recognition: a comparison of BERT, DistillBERT, RoBERTa, XLNet and ELECTRA,» arXiv preprint arXiv:2104.02041v1, 2021.

[43] A. Azadi, B. Ansari y S. Zamani, «Bilingual Sexism Classification: Fine-Tuned XLM-RoBERTa,» arXiv preprint arXiv:2406.07287v1, 2024.

[44] B. Bhatt, «Fine-Tuning RoBERTa for Truthfulness Classification: A Case Study on the LIAR Dataset,» *Research Square,* 2024.

[45] W. Yang Wang, «"liar, liar pants on fire": A new benchmark dataset for fake news detection.,» arXiv preprint arXiv:1705.00648, 2017.

[46] K. Shu, D. Mahudeswaran, D. Lee y H. Liu, «Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media,» Big data, 8(3):171–188, 2020.

[47] P. Patwa, M. Bhardwaj, V. Guptha, G. Kumari, S. Sharma, S. PYKL, A. Das, A. Ekbal, S. Akhtar y T. Chakraborty, «Overview of constraint 2021 shared tasks: Detecting english covid-19 fake news and hindi hostile posts. In Proceedings of the First Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situation,» Springer, 2021.

[48] J.-P. Posadas-Durán, H. Gómez-Adorno, G. Sidorov y J. J. Moreno Esobar, «Detection of fake news in a new corpus for the spanish language.,» Journal of Intelligent & Fuzzy Systems,36(5):4869–4876, 2019.

[49] B. Riedel, I. Augenstein, G. P Spithourakis y S. Riedel, «A simple but tough-to-beat baseline for the fake news challenge stance detection task.,» arXiv preprint arXiv:1707.03264, 2017.

[50] A. Vlachos y S. Riedel, «Fact checking: Task definition and dataset construction. In Proceedings of the ACL 2014 workshop on language technologies and computational social science, pages 18–22,» Association for Computational Linguistics, Baltimore (USA), 2014.

[51] «IberLEF2022,» 2022. [En línea]. Available: https://sites.google.com/view/iberlef2022. [Último acceso: 15 Junio 2024].

[52] J. P. Posadas Durán, H. Gómez Adorno, G. Bel Enguix y C. Porto Capetillo, «Overview of FakeDeS at IberLEF 2021: Fake News Detection in Spanish Shared Task,» *Procesamiento del Lenguaje Natural,* 2021.

[53] K. Martínez-Gallego, A. M. Álvarez Ortiz y J. D. Arias Londoño, «Fake News Detection in Spanish Using Deep Learning Techniques,» arXiv preprint arXiv:2110.06461, 2021.

[54] B. Togni, M. Coll Ardanuy y P. Rosso, «Emotions and News Structure: An Analysis of the Language of Fake News in Spanish,» de *40ª Conferencia de la Sociedad Española de Procesamiento del Lenguaje Natural*, 2024.

[55] «IBM.com,» 29 Septiembre 2023. [En línea]. Available: https://www.ibm.com/docs/en/imdm/14.0?topic=functions-ngram. [Último acceso: 14 Junio 2024].

[56] PlanTL, «HuggingFace.com,» 2021. [En línea]. Available: https://huggingface.co/PlanTL-GOB-ES/roberta-large-bne. [Último acceso: 27 Mayo 2024].

[57] FacebookAI, «HuggingFace,» 2019. [En línea]. Available: https://huggingface.co/FacebookAI/xlm-roberta-large. [Último acceso: 24 Mayo 2024].

[58] PlanTL, «HuggingFace.com,» 2021. [En línea]. Available: https://huggingface.co/PlanTL-GOB-ES/RoBERTalex. [Último acceso: 24 Mayo 2024].

[59] D. P.Kingma y J. Ba, «Adam: A Method for Stochastic Optimization,» arXiv preprint arXiv:1412.6980v9, 2014.

[60] I. Loshchilov y F. Hutter, «Decoupled Weight Decay Regularization,» arXiv preprint arXiv:1711.05101v3, 2017.

[61] «Geeksforgeeks.org,» 27 Agosto 2024. [En línea]. Available: https://www.geeksforgeeks.org/why-is-adamw-often-superior-to-adam-with-l2-regularization-in-practice/. [Último acceso: 29 Agosto 2024].

[62] C. Sun, X. Qiu y X. Huang, «How to Fine-Tune BERT for Text Classification?,» arXiv:preprint arXiv:1905.05583v3 , 2019.

[63] R. Gu y X. Meng, «AISPACE at SemEval-2024 task 8: A Class-balanced Soft-voting System for Detecting Multi-generator Machine-generated Text,» arXiv:preprint arXiv:2404.00950v1, 2024.

[64] C. Perrio y T. Madabushi, «Informative COVID-19 Tweets -- RoBERTa Ensembles and The Continued Relevance of Handcrafted Features,» rXiv: preprint arXiv:2010.07988v1, 2020.

[65] R. Anggrainingsih, G. Mubashar Hassan y A. Datta, «Evaluating BERT-based Pre-training Language Models for Detecting Misinformation,» arXiv: preprint arXiv:2203.07731v1, 2022.

[66] A. Rubiales, «Freedium.cfd,» 5 Enero 2022. [En línea]. Available: https://freedium.cfd/https://rubialesalberto.medium.com/funciones-de-error-con-entropia-cross-entropy-y-binary-cross-entropy-8df8442cdf35. [Último acceso: 29 Junio 2024].

[67] D. Abramov y R. Nabors, «Introducing react.dev,» 16 Marzo 2023. [En línea]. Available: https://react.dev/blog/2023/03/16/introducing-react-dev. [Último acceso: 10 Septiembre 2024].

[68] «Introduction to Node.js,» [En línea]. Available: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs. [Último acceso: 10 Septiembre 2024].

[69] R. G. Schulz, Diseño web con CSS, Marcombo, 2008.

[70] «Flask,» [En línea]. Available: https://flask.palletsprojects.com/en/3.0.x/#. [Último acceso: 10 Septiembre 2024].

[71] D. Bergmann, «IBM.com,» 12 Enero 2023. [En línea]. Available: https://www.ibm.com/es-es/topics/fine-tuning. [Último acceso: 10 Junio 2024].

[72] M. Awad y R. Khanna, Efficient Learning Machines. Theories, Concepts, and Applications for Engineers and System Designers, Apress, 2015.

[73] M. Bhardwaj, V. Guptha, G. Kumari, S. Sharma, S. PYKL, A. Das, A. Ekbal, S. Akhtar y T. Chakraborty, «Overview of constraint 2021 shared tasks: Detecting english covid-19 fake news and hindi hostile posts. In Proceedings of the First Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situation,» Springer, 2021.

[74] «Introduction to Natural Language Processing (NLP),» thelead.io, [En línea]. Available: https://thelead.io/data-science/what-is-natural-language-processing/. [Último acceso: 15 Abril 2024].