UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

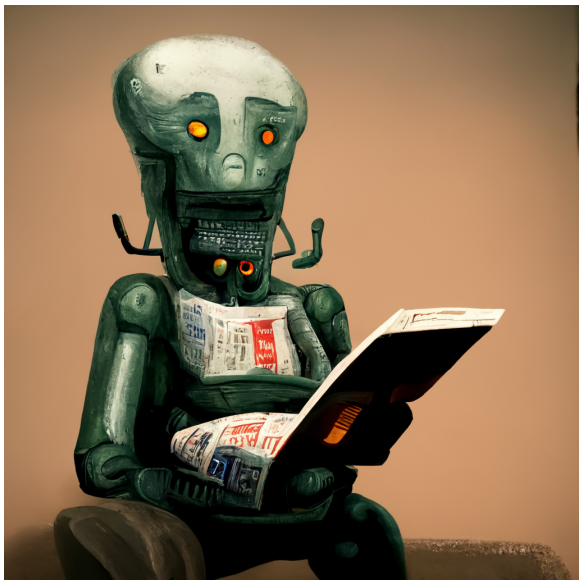
fIlfA: Modelado computacional de la desinformación

Autor
Santiago González Silot

Directores
Eugenio Martínez Cámara
María Victoria Luzón García

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, 6 de julio 2022



fIlfA : Modelado computacional de la desinformación.

Autor
Santiago González Silot

Directores
Eugenio Martínez Cámara
María Victoria Luzón García

fIlfA: Modelado computacional de la desinformación

Santiago González Silot

Palabras clave: Noticias falsas, Deep Learning, PLN, BERT, Transformers, Fake News

Resumen

La propagación *fake news* es cada vez un mayor problema en la sociedad actual. Cada vez el lector promedio tiene menos tiempo para poder verificar con certeza la veracidad de una noticia, lo cual hace necesario la creación de un sistema para la detección de *fake news*. Por tanto en este trabajo se realiza un estudio sobre la detección de *fake news* y se presentan 2 sistemas para la detección de fake-news, uno en español y otro en inglés. Para ello se utilizarán técnicas de Procesamiento del Lenguaje Natural, más en concreto se aplicará la técnica de *fine-tuning* sobre distintos modelos de BERT y RoBERTa al ser los modelos de lenguaje basados en *transformers* los modelos de *deep learning* que representan el estado del arte para tareas de clasificación de texto.

fIlfA: Computational modeling of disinformation

Santiago González Silot

Keywords: Deep Learning, PLN, BERT, Transformers, Fake News

Abstract

The spread of fake news is a growing problem in today's society. The average reader has less time to be able to verify with certainty the veracity of a news item, which makes necessary the creation of a system for the detection of fake news. Therefore, in this work a study on fake news detection is carried out and two systems for fake news detection are presented, one in Spanish and the other in English. For this purpose, Natural Language Processing techniques will be used, more specifically, the fine-tuning technique will be applied on different BERT and RoBERTa models, since transformer-based language models are the deep learning models that represent the state of the art for text classification tasks.

Yo, **Santiago González Silot**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77034478A, autorizo la publicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Santiago González Silot

Granada a 6 de julio de 2022.

D. **Eugenio Martínez Cámara**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

D. **María Victoria Luzón García**, Profesora del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***fIlfA, Modelado computacional de la desinformación***, ha sido realizado bajo su supervisión por **Santiago González Silot**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 6 de julio de 2022.

Los directores:

Eugenio Martínez Cámara **María Victoria Luzón García**

Agradecimientos

En primer lugar me gustaría agradecer a Eugenio y a María Victoria por todas las reuniones, la disponibilidad, paciencia y apoyo absoluto que han tenido conmigo. Sin su guía y mentoría no hubiese podido hacer un trabajo tan completo y correcto como lo ha sido. Gracias por los conocimientos aportados y la oportunidad. De ellos me llevo el no ser tan científico de datos y sí un poco más ingeniero informático.

También quiero agradecer a mis padres y pareja y en general a toda mi familia y amigos por haberme apoyado durante estos 4 años. Sin un hombro al que apoyarse este camino hubiese sido mucho más duro aún.

Finalmente quisiera agradecer a todos los profesores de la facultad por todo el conocimiento que me han transmitido estos años y su dedicación.

Índice general

1. Introducción **19**
1.1. Motivación 20
1.2. Objetivos 21
1.3. Presupuesto 21
1.4. Planificación 22
1.5. Estructura de la memoria 24

I Estudio Científico del Problema **27**

2. Contexto **29**
2.1. Desinformación 29
2.1.1. *Fake news* 30
2.1.2. Rumores 30
2.1.3. Otros tipos de desinformación 31
2.2. Procesamiento del Lenguaje Natural 32
2.2.1. Transformers 35
2.2.2. Mecanismos de atención 36
2.2.3. Modelos del lenguaje 37

3. Detección de *Fake News* **41**
3.1. Conjuntos de datos disponibles 41
3.2. Análisis Exploratorio de los datos 44
3.2.1. CONSTRAINT AAAI 46
3.2.2. IberLEF 49
3.3. Estudio y resolución del problema 52
3.3.1. Casos Base 52
3.3.2. Fine-Tuning 54
3.4. Conclusiones 82

II Desarrollo de aplicación web demostrativa **85**

4. Análisis **87**

4.1. Requisitos funcionales 87
4.2. Diagrama de casos de uso 88
4.2.1. Descripción de los actores 88
4.2.2. Descripción de los casos de uso 90
4.2.3. Diagramas de secuencia del sistema 92

5. Diseño **93**
5.1. Diseño de la arquitectura 93
5.1.1. Diseño de las componentes software 94
5.1.2. Diagramas de secuencia del sistema 95
5.2. Diseño de la interfaz 96

6. Implementación y Pruebas **101**
6.1. Implementación 101
6.2. Pruebas 102
6.2.1. Predecir noticia 102
6.2.2. Descargar modelo 106
6.3. Conclusiones 106

III Conclusiones finales **109**

7. Conclusiones y trabajo futuro **111**
7.1. Conclusiones 111
7.2. Trabajo futuro 112

Bibliografía **113**

Índice de figuras

1.1. Evolución del término "Fake News" a lo largo del tiempo. Fuente: Google Trends. 20
1.2. Diagrama de Gantt estimado. 23
1.3. Diagrama de Gantt final. 24

2.1. Diagrama NLP. Fuente: Pinterest. 32
2.2. Arquitectura de *Transformers*, *Multi-Head-Attention* y *Attention*. Fuente: Medium. 36
2.3. Arquitectura de BERT junto con la arquitectura *Transformers*. Fuente: ResearchGate. 38
2.4. Representación de la fase de entrenamiento de BERT. Fuente: Medium. 38

3.1. Distribución del número de palabras del conjunto de entrenamiento (CONSTRAINT AAAI). 46
3.2. 10 N-gramas más repetidos (CONSTRAINT AAAI). 47
3.3. Nube de palabras más repetidas (CONSTRAINT AAAI). . . 48
3.4. Nube de palabras más repetidas separando los conjuntos de noticias falsas y verdaderas (CONSTRAINT AAAI). 49
3.5. Distribución del número de palabras del conjunto de entrenamiento (IberLEF). 49
3.6. 10 N-gramas más repetidos (IberLEF) 50
3.7. Nube de palabras más repetidas (IberLEF). 51
3.8. Nube de palabras más repetidas separando los conjuntos de noticias falsas y verdaderas (IberLEF). 52
3.9. Matriz de confusión de *Linear SVC* para el conjunto de datos de CONSTRAINT AAAI. 53
3.10. Matriz de confusión de *Linear SVC* para el conjunto de datos de IberLEF. 54
3.11. Curvas de perdida RobertaGob. 58
3.12. Curvas de precisión RobertaGob. 59
3.13. Matrices de confusión RobertaGob. 60
3.14. Curvas de perdida RobertaBIO. 61

3.15. Curvas de precisión RobertaBIO. 62
3.16. Matrices de confusión RobertaBIO. 63
3.17. Curvas de perdida BETO. 64
3.18. Curvas de precisión BETO. 65
3.19. Matrices de confusión BETO. 66
3.20. Curvas de perdida Vinai-Bertweet. 69
3.21. Curvas de precisión Vinai-Bertweet. 70
3.22. Matrices de confusión Vinai-Bertweet. 71
3.23. Curvas de perdida Cardiff. 72
3.24. Curvas de precisión Cardiff. 73
3.25. Matrices de confusión Cardiff. 74
3.26. Curvas de perdida Digital-Epidemology-V2. 75
3.27. Curvas de precisión Digital-Epidemology-V2. 76
3.28. Matrices de confusión Digital-Epidemology-V2. 77
3.29. Curvas de perdida Sci-Bert. 78
3.30. Curvas de precisión Sci-Bert. 79
3.31. Matrices de confusión Sci-Bert. 80

4.1. Diagrama de casos de uso. 88
4.2. DSS-Predecir noticia. 92
4.3. DSS-Descargar Modelo. 92

5.1. Diagrama de clases del sistema. 94
5.2. DSS de diseño - Predecir noticia. 96
5.3. DSS de diseño - Descargar modelo. 96
5.4. Boceto de la página principal. 97
5.5. Resultado final de página principal. 97
5.6. Resultado final de página inicial. 98
5.7. Captura de las distintas activaciones de los enlaces de la página inicial. 99

6.1. Estructura de ficheros. 102
6.2. Prueba del requisito “Predecir noticia” - Prueba 1. 103
6.3. Prueba del requisito “Predecir noticia” - Prueba 2. 104
6.4. Prueba del requisito “Predecir noticia” - Prueba 3. 105
6.5. Prueba del requisito “Descargar modelo” - Prueba 4. 106

Índice de tablas

3.1. Características de los conjuntos de datos que han sido considerados. 44
3.2. Analisís Básico de los conjuntos de datos. 45
3.3. F1-score para los modelos básicos. 53
3.4. Modelos de BERT considerados para el fine-tuning del conjunto de datos de IberLEF. 57
3.5. Modelos propuestos RobertaGob. 58
3.6. Modelos propuestos RobertaBIO. 62
3.7. Modelos propuestos BETO. 65
3.8. Resultado tras los distintos *Fine-Tuning* de los modelos de IberLEF. 67
3.9. Mejores 10 resultados de la competición de IberLEF. 68
3.10. Modelos de BERT considerados para el fine-tuning del conjunto de datos de CONSTRAINT AAAI. 68
3.11. Modelos propuestos Vinai-Bertweet. 70
3.12. Modelos propuestos Cardiff. 73
3.13. Modelos propuestos Digital-Epidemology-V2. 76
3.14. Modelos propuestos Sci-Bert. 79
3.15. Resultado tras los distintos *Fine-Tuning* de los modelos de CONSTRAINT AAAI. 81
3.16. Mejores 10 resultados de la competición de CONSTRAINT AAAI. 82

4.1. Descripción del actor “Usuario común”. 89
4.2. Descripción del actor “Científico de datos”. 89
4.3. Descripción del CU-1 Descargar modelo. 90
4.4. Descripción del CU-2 Predecir noticia. 91

Capítulo 1

Introducción

En pleno siglo XXI cada vez es más habitual informarse de la actualidad a través de medios digitales o redes sociales. Estos nos permiten estar actualizados con todo lo que ocurre en el mundo con el único requisito de disponer de un dispositivo móvil y conexión a Internet, lo cual es un gran avance como sociedad y nos ayuda a progresar, pero tiene sus peligros y uno de ellos es la desinformación. Existen diversos tipos de desinformación, en concreto durante este trabajo se hará referencia a las notificas falsas, o como son popularmente conocidas, *fake news*.

Uno de los objetivos de la divulgación de *fake news* es la manipulación de la opinión de la sociedad para fines políticos o sociales. Uno de los casos más conocidos de divulgación de *fake news* con fines políticos y sociales es el de la campaña a la presidencia de los Estados Unidos en 2016 en la cual *Donald Trump* se aprovechó de múltiples movimientos de desinformación para lograr su victoria en las elecciones [3]. Como se puede observar en la figura 1.1 el crecimiento de la popularidad del término "*fake news*" coincide con este suceso. Pero este no es el único caso conocido del uso de las *fake news* para fines políticos, también se conoce la influencia de las *fake news* durante el referéndum del Brexit[1].

En ocasiones se tiende a pensar que las noticias falsas suelen ser divulgadas por *trolls* o personas con un bajo nivel cultural, pero esto no es así. Tras algunos de estos bulos o noticias falsas en ocasiones se encuentran partidos políticos o grandes empresas, como el conocido caso de *Cambridge Analytica*, empresa la cual está supuestamente relacionada con los casos anteriormente mencionados de la elección a la presidencia de Estados Unidos del 2016 y la salida del Reino Unido de la Comunidad Económica Europea, entre otros tantos casos. La propia empresa en la portada de su página web[2]

[1] https://en.wikipedia.org/wiki/Brexit

[2] https://web.archive.org/web/20171113180809/https://cambridgeanalytica.org

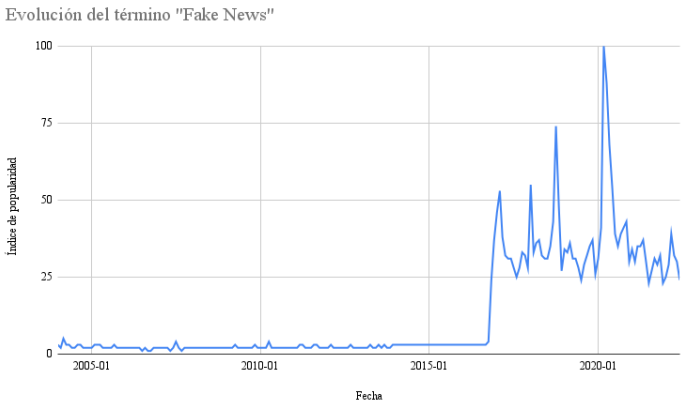


Figura 1.1: Evolución del término “Fake News” a lo largo del tiempo. Fuente: Google Trends.

decía textualmente: “Data drives all we do. Cambridge Analytica uses data to change audience behaviour”.

Junto a estos casos sociales mundialmente conocidos en los cuales ha habido una influencia de las *fake news*, se encuentra la situación actual de pandemia mundial debido a la COVID-19 la cual ha sido un gran foco de desinformación y más en concreto de *fake news*, siendo en ocasiones el gran flujo de *fake news* el culpable de decisiones socio-sanitarias cuestionables de distintos sectores de la población.

Con estos ejemplos de actualidad social, se esclarece la importancia para todos los ciudadanos de saber diferenciar entre información veraz y falsa, ya que puede tener un gran impacto tanto en la sociedad como en el individuo.

Por tanto, en la Sección 1.1 se justificará la necesidad de la realización de este trabajo. En la Sección 1.2 se detallan los objetivos de este trabajo. En la Sección 1.3 se analizará el presupuesto necesario para desarrollar el trabajo. En la Sección 1.4 se hará una planificación del tiempo necesario para el desarrollo del trabajo. Finalmente en la Sección 1.5 se detallará brevemente la estructura que tendrá la memoria de este trabajo.

1.1. Motivación

Si bien es de gran importancia aprender a diferenciar entre noticias falsas y verdaderas, cada vez el lector tiene menos tiempo para contrastar la información en varios medios o en fuentes oficiales, lo cual lo hace más vulnerable a las *fake news*.

Además, no solo el lector cada vez tiene menos tiempo a contrastar la

información, sino que cada vez recibe más información debido a las redes sociales y al gran flujo de información que estas conllevan, lo cual hace prácticamente imposible la comprobación del lector de cada noticia que lee. Esto suscita a la creación de un sistema capaz de detectar la veracidad de la información, por tanto este trabajo tiene como objetivo la creación de un sistema automático de detección de *fake news*.

El objetivo es abordar este problema con el texto como eje principal para la decisión de la veracidad de la información. Esto es, desde un enfoque del Procesamiento del Lenguaje Natural (PLN) y no utilizando otros métodos como las fuentes o la red subyacente de comunicación entre usuarios que comparten la noticia falsa.

1.2. Objetivos

El objetivo principal de este trabajo fin de grado es el **desarrollo de un sistema automático de detección de *fake news***.

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

1. **Analizar el estado del arte del problema**: Se estudiará el estado del arte del problema de detección de *fake news*, *Deep Learning* y PLN.

2. **Analizar los conjuntos de datos disponibles**: Se analizarán los conjuntos de datos más populares y recientes en la literatura y se seleccionarán 2 para su resolución, uno de ellos en inglés y otro en español.

3. **Desarrollar un sistema de *Deep Learning* para resolver el problema**: Se desarrollará un sistema de *Deep Learning* para resolver el problema utilizando únicamente el texto de la noticia (titulo y cuerpo), sin ningún tipo de información adicional.

4. **Implementar una interfaz web básica para el problema**: Se implementará una interfaz web básica parar mostrar el funcionamiento del sistema de forma interactiva.

1.3. Presupuesto

A continuación se detallan las distintas variables que se han tenido en cuenta para calcular el presupuesto necesario para llevar a cabo este trabajo.

Detalles	Cantidad
Horas trabajadas:	5 meses x 20 días x 8 horas = 800 horas
Sueldo por hora:	20€/h
Precio total mano de obra:	20€/h x 800€= 16.000€
Precio suscripción Colab Pro durante 5 meses (10€/mes)	50€
Precio portátil del alumno (suponiendo 5 meses de trabajo, con una vida útil de 4 años y un precio total de 600€)	62,5€
Total	16.112,5€
Total (IVA incluido)	16.112,5€+ 21 % de IVA = 19.496,125€

1.4. Planificación

Para el correcto desarrollo del trabajo y su finalización en la fecha límite correspondiente se han identificado las distintas tareas a realizar y el tiempo estimado para la realización de cada una de ellas. Estas tareas son las siguientes:

- **Aprendizaje sobre PLN, *Transformers* y BERT**: Estudio de las técnicas de PLN, fine-tuning y modelos de lenguaje pre-entrenados (como BERT y RoBERTa) esenciales para el desarrollo del trabajo.
- **Búsqueda de conjuntos de datos**: Para resolver el problema de detección *fake news* se analizarán los distintos conjuntos de datos de la literatura y se seleccionarán los más adecuados.
- **Análisis Exploratorio de los Datos**: En primer lugar, para la solución del problema se hará un análisis de los datos, lo cual supone el punto de partida de cualquier problema de *Machine Learning*.
- **Implementación y prueba de los modelos**: Esta tarea será la principal del trabajo y la cual llevará una gran parte de este. Esto es debido a la complejidad que tiene encontrar modelos correctos, entenderlos, ajustarlos y su largo periodo de entrenamiento.
- **Creación del prototipo para las pruebas**: Se creará un prototipo tanto para la ejecución de las pruebas como para la posterior revisión del trabajo realizado.

- **Implementación de la Interfaz web**: Una vez llegado a un punto avanzado de la implementación de los modelos, se comenzará a implementar la interfaz web siguiendo un proceso típico de ingeniería del software, es decir, definiendo los requisitos, casos de uso, actores, diagramas UML, etc. Posteriormente se realizará la implementación de la Interfaz web en sí.
- **Redacción de la memoria**: Aproximadamente en un punto medio del desarrollo del trabajo se comenzará con el desarrollo de la memoria.

En el diagrama de Gantt de la figura 1.2 se puede observar el orden y la duración en el que se deben realizar las tareas para realizar correctamente este trabajo. Como se puede observar en la leyenda del diagrama, las tareas han sido clasificadas por colores, las rojas son tareas de investigación, las amarillas de implementación y las azules de redacción de la memoria.

En la figura 1.3 se puede observar el diagrama de Gantt de la planificación finalmente realizada. Como se puede observar hay varias diferencias entre la estimación y la planificación final las cuales se explican a continuación:

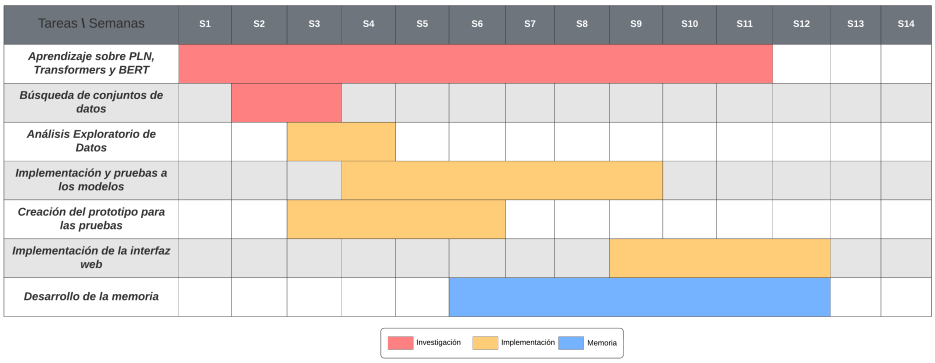


Figura 1.2: Diagrama de Gantt estimado.

- Se necesitaron 2 semanas más para aprendizaje sobre PLN, *Transformers* y BERT debido a que son temáticas muy avanzadas y se han tenido que aprender desde cero hasta un alto nivel para poder trabajar con los modelos que suponen el estado del arte como lo es BERT.
- Se necesitaron 3 semanas más para la creación de los modelos propuestos ya que estos necesitaban mucho tiempo durante el entrenamiento y se realizaron una gran cantidad de pruebas con distintos parámetros lo cual conllevó a este retraso.
- Se necesitaron 2 semanas más para la implementación de la interfaz web ya que se mejoraron al máximo sus funcionalidades y estética.

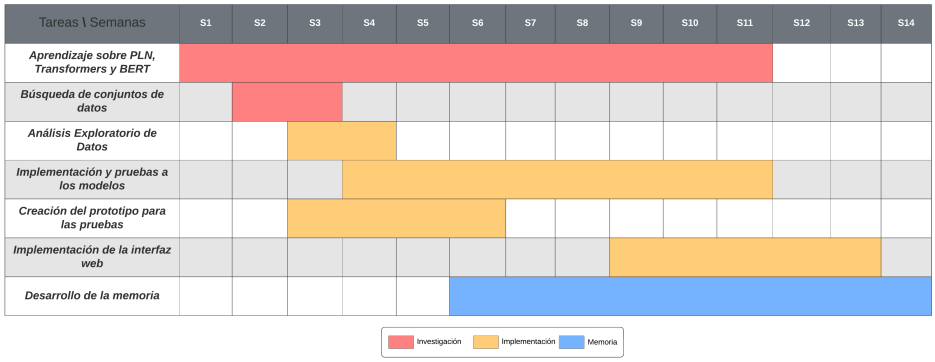


Figura 1.3: Diagrama de Gantt final.

- Se necesitaron 2 semanas más para la redacción de esta memoria para mejorar su presentación y contenido.

1.5. Estructura de la memoria

Para facilitar el seguimiento de la memoria al lector, esta se ha estructurado en tres partes. En la primera parte se introduce al lector en el contexto de las *fake news* y PLN, para posteriormente explicar la resolución del problema. En la segunda parte se explica todo el proceso de ingeniería del software llevado a cabo durante el desarrollo de la aplicación web que sirve como prueba del correcto funcionamiento de los modelos propuestos en la primera parte. Finalmente, la tercera parte constará de las conclusiones de todo este trabajo, tanto las relativas a la primera parte, como a la segunda. A continuación se detalle más en profundidad el contenido de cada una de estas partes.

1. **Parte 1, Estudio Científico del Problema**: La primera parte de la memoria abordará todo lo relacionado con el núcleo principal del trabajo, es decir, el análisis científico del problema y su resolución.
   1.1. **Contexto**: En este capítulo se hará una breve introducción a los distintos conceptos necesarios para la correcta comprensión del trabajo realizado. Esto es, tanto la definición de *fake news* y relacionados como conceptos sobre PLN y *Deep Learning* tales como *Transformers* y BERT.
   1.2. **Implementación**: En este capítulo se analizan los conjuntos de datos disponibles y los seleccionados. A continuación se procede a resolver el problema en primer lugar con una serie de modelos base y posteriormente con los modelos propuestos.

2. **Parte 2, Desarrollo de aplicación web demostrativa**: Como prueba del correcto funcionamiento de los modelos propuestos durante la primera parte, en la segunda parte de la memoria se ha desarrollado una aplicación web.

    2.1. **Análisis**: Este capítulo abordará el análisis del sistema, desde la declaración de los casos de uso hasta el desarrollo de los diagramas UML necesarios los cuales describan el sistema.

    2.2. **Diseño**: En este capítulo se explicarán las distintas decisiones tomadas durante el diseño del sistema junto al diseño de las distintas componentes software y la interfaz web.

    2.3. **Implementación y pruebas**: Durante este capítulo se explicarán los detalles de la implementación del sistema así como un conjunto de pruebas que validen su correcto funcionamiento.

3. **Parte 3, Conclusiones**: Como recapitulación de todo el trabajo realizado, en esta tercera y última parte se harán las conclusiones de todo este trabajo.

    3.1. **Conclusiones y trabajo futuro**: En el capítulo final se analizarán las conclusiones obtenidas de este trabajo además de mencionar posibles mejoras y trabajo futuro para seguir avanzando en la detección de *fake news*.

Parte I

Estudio Científico del Problema

Capítulo 2

Contexto

Para que el lector pueda entender correctamente el trabajo de investigación e implementación realizado, en este capítulo se presenta el contexto teórico sobre las distintas temáticas específicas de las cuales tratará este trabajo. Estas temáticas son las de *fake news*, Procesamiento del Lenguaje Natural, *Transformers* y BERT.

Además, es de gran importancia para la realización de un trabajo de ingeniería, y más en concreto de *Machine Learning* tener un alto nivel de conocimiento no solo de los modelos a utilizar si no del contexto para el que van a ser empleados y sus peculiaridades.

Por tanto en la Sección 2.1 se realizará una introducción a las *fake news* y conceptos relacionados, y en la Sección 2.2 sobre Procesamiento del Lenguaje Natural.

2.1. Desinformación

La desinformación es un tipo de información falsa o parcialmente falsa, la cual se difunde generalmente de manera intencionada para manipular al lector. En inglés se hace la diferencia entre *disinformation* para cuando la desinformación es de manera intencionada, y se utiliza *misinformation* para cuando se trata de falta de información la cual se comparte de forma no maliciosa aparentemente.

Las difusión de la desinformación se produce en diferentes medios tanto convencionales como digitales. El término desinformación es muy amplio y por tanto se puede clasificar según la fuente, tipo de información e intencionalidad[50].

Si bien algunos de estos términos no tienen una definición solida por la comunidad científica ni periodística, a continuación se presenta una clasifi-

cación de los distintos tipos de desinformación [50].

2.1.1. *Fake news*

Fake news se ha convertido en el término en la práctica para identificar información falsa en los medios de comunicación, especialmente en ámbitos relacionados con Internet y las redes sociales.

Como primera definición de *fake news*, [3] propone : "*fake news* es un artículo de prensa que es intencionalmente y de manera verificable falso". Esta definición depende de la intencionalidad y la verificabilidad de la noticia. Según esta definición, las *fake news* son noticias que se escriben intencionalmente para engañar o desinformar a los lectores y se puede verificar la veracidad de la noticia. Dentro del concepto de *fake news* se diferencian los siguientes términos:

- **Fabricaciones serias**: Son la forma prototípica de *fake news*, tales como artículos con intención maliciosa que en ocasiones se vuelven virales en redes sociales. Un ejemplo de fabricación seria es la suspensión de François Bugingo de los medios de comunicación debido a las noticias falsas sobre su manipulación de ciertos artículos antiguos [39].
- **Engaños a gran escala**: Son relatos de información falsa disfrazados como noticias verdaderas. Suelen estar organizados en mayor escala que un simple artículo y en ocasiones están dirigidos a ciertas figuras públicas o ideas. Un ejemplo de engaño a gran escala es la creencia de que el grupo terrorista ISIS fue el responsable de una explosión en 2014 en Columbia [39].
- **Sátira**: Son noticias falsas humorísticas escritas con la intencionalidad de entretener y divertir al lector. Se considera a los lectores como conscientes del tono humorístico de la noticias y en ocasiones las revistas de este tipo se denominan como tales en su página web para dejar clara su intencionalidad. Un ejemplo de periódico digital de sátira es "El Mundo Today"[1].

2.1.2. Rumores

En la literatura científica, los rumores son probablemente el mayor caso de estudio de información falsa propagada por Internet [50]. Los rumores hacen referencia a información la cual no ha sido confirmada por medios oficiales todavía y se propaga comúnmente por los usuarios en redes sociales. Si bien los rumores no son producto de las redes sociales, ya que algunos

[1] https://www.elmundotoday.com

estudios datan al final de la Segunda Guerra Mundial como el inicio de los rumores tal y como se conocen hoy en día, estos se propagan con mayor facilidad en la actualidad gracias a Internet, las redes sociales y el gran flujo de información [8].

Es complicado proporcionar una definición formal de rumor, en cambio, distintos investigadores han propuesto distintas interpretaciones. Una de las definiciones más ampliamente aceptadas proviene de [14], los autores en su investigación identifican los rumores como: "declaraciones informativas sin verificar y potencialmente relevantes en circulación". En cambio [52] define los rumores más específicamente como: "historia en circulación de veracidad cuestionable, la cual es aparentemente creíble pero difícil de verificar y produce suficiente escepticismo y/o ansiedad", por lo que según esta definición un rumor ha de producir una reacción impactante en su audiencia.

Estas definiciones giran en torno a la característica "sin verificar" de la información. Esta información sin verificar puede ser cierta, parcialmente cierta, completamente falsa o permanecer sin verificación. Dentro de los "rumores" se ha tratado de clasificar dependiendo del tipo, ámbito y características. Por ejemplo, [51] separa los rumores en dos categorías principales:

- **Rumores a gran escala**: Representa información que circula durante un largo periodo de tiempo y puede que nunca se verifique como cierta o falsa. Las leyendas urbanas y teorías de la conspiración se pueden consideran como tales. Un ejemplo de rumor a gran escala estudiado por la literatura es el rumor de que Barack Obama es musulmán [8].

- **Rumores de noticias de última hora**: Son los más comunes y aparecen junto a *breaking news stories*. Estos pueden ser producto de falta de información sin intención maliciosa o de naturaleza engañosa. Este tipo de rumores ha cobrado una mayor importancia en la literatura ya que pueden ser más peligrosos en un corto periodo de tiempo respecto a los rumores a gran escala. Es necesario identificarlos pronto para parar su propagación, especialmente si la intención es maliciosa. Durante el día a día se pueden leer muchos rumores de este tipo en la prensa, pero uno de los ejemplos más comunes es el del terremoto en Fukushima en 2011 el cual afectó a 3 reactores, a raíz de este suceso surgieron rápidamente muchos rumores sobre lo ocurrido, el número de afectados y las consecuencias [2].

2.1.3. Otros tipos de desinformación

Si bien las *fake news* y los rumores representan el tópico principal de interés en el ámbito de la desinformación, existen otros tipos de desinformación en la web. A continuación se muestra una breve descripción de algunos

de estos conceptos:

- ***Clickbait***: Hace referencia a titulos de artículos o publicaciones en redes sociales cuyo objetivo es atraer a los lectores a seguir un enlace a la página web del artículo. Además se ha identificado que los titulares con clickbait son una de las mayores contribuciones a la propagación de fake news [42].

- ***Social spammers***: Son usuarios en redes sociales, los cuales de forma coordinada lanzan distintos tipos de ataques tales como virus, anuncios y *phising*.

- **Opiniones falsas**: Este término hace referencia a reseñas falsas típicamente encontradas en sitios web de comercio electrónico y de reseñas como Rotten Tomates, IMDB o Metacritic. El objetivo de estas reseñas falsas suele ser mejorar o empeorar la popularidad de un determinado producto.

2.2. Procesamiento del Lenguaje Natural

Como se puede observar en la figura 2.1, el Procesamiento del Lenguaje Natural (PLN), es un subcampo de la lingüística, ciencia de computadores e inteligencia artificial dedicado a diseñar métodos y algoritmos que toman como entrada o salida lenguaje natural no estructurado. El objetivo es que un ordenador sea capaz de generar y "entender" los contenidos de un texto para así poder interactuar con él.

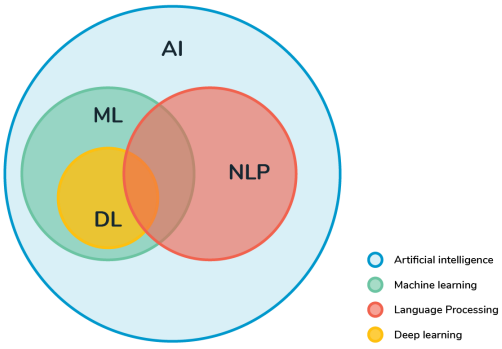


Figura 2.1: Diagrama NLP. Fuente: Pinterest.

Algunas de las tareas más típicas que aborda el procesamiento del lenguaje natural son:

- **Reconocimiento de voz**. Dado un audio de una persona hablando, determinar la representación textual del audio.
- ***Text-to-speech***. Justo lo contrario al reconocimiento de voz, es decir, dado un texto, transformarlo y reproducir su representación sonora.
- **Recuperación de información**. Encontrar documentos en un conjunto de documentos los cuales satisfagan ciertas características.
- **Clasificación de texto**. Asignar a un texto o documento una o más categorías.
- **Traducción automática**. Traducción automática de texto de un idioma a otro.
- **Autocompletado de texto (Texto predictivo)**. Dado un texto, automáticamente completarlo con una o más palabras.
- **Generación automática de texto**. Dadas unas características generar un texto que las cumpla.
- ***Question-answering***. Dada una pregunta, identificar una posible respuesta y viceversa.
- **Análisis de opiniones**. Caso específico de clasificación de texto en el que se le asigna un sentimiento o intencionalidad al texto.
- ***Chatbots* y agentes conversacionales**. Aplicación software que simulan mantener conversaciones con una persona.

Algunas de las técnicas clásicas de PLN utilizan el *tf-idf* como entrada para modelos clásicos de *Machine Learning* como pueden ser las Máquinas de Vectores Soporte y los Árboles de Decisión.

Por un lado el *Machine Learning* o Aprendizaje Automático es un subcampo de la Inteligencia Artificial el cual tiene como objetivo otorgar a las máquinas la habilidad de aprender sin ser explícitamente programadas (Arthur Samuel, 1959) [4]. O desde un punto más moderno (Tom Mitchell, 1997) [4]: " Un programa de computador se dice que aprende de la experiencia E con respecto a alguna clase de tareas T y una medida de rendimiento P si su rendimiento en las tareas de T, medido por P, mejora con la experiencia E. "

Dentro del *Machine Learning* existen distintos tipos de aprendizaje, entre ellos los más comunes son el aprendizaje supervisado, en el cual se aprende una función a través de las etiquetas y de los datos, y el no supervisado el cual se aprende una función a través de los datos.

Por otro lado el Tf-idf (Term frequency - Inverse document frequency), es una medida numérica que expresa cuán relevante es una palabra para un documento en concreto dentro de un conjunto de documentos. Es una medida típica en el ámbito de la recuperación de información y minería de datos. Este valor aumenta proporcionalmente al número de veces que aparece una palabra en un documento pero se ve compensado por la frecuencia de esa palabra en el resto de documentos. Su formulación matemática es la siguiente:

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

Siendo $tf(t, d)$ la frecuencia del término t en un documento d:

$$\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Donde:

- $f_{t,d}$ 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 $idf(t, D)$ 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.

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

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.

Por tanto el valor del *tf-idf* de las palabras de un texto, sirve como entrada para los modelos de *Machine Learning* tales como las Máquinas de Soporte Vectorial [11], Árboles de Decisión [36] o Regresión Logística [21] entre otros. Si bien estos modelos logran un buen resultado a la hora de resolver los problemas de PLN [27, 12] y más en concreto de clasificación de texto, el estado del arte viene marcado por distintos modelos basados en *Deep Learning* [49, 1], entre los que destacan los siguientes:

- **RNN**, Redes Neuronales Recurrentes.
- **CNN**, Redes Neuronales Convoluciones.
- **Modelos con mecanismos de atención**.
- **Redes de memoria aumentada**.
- **Transformers** y **Modelos de lenguaje pre-entrenados**.

En concreto, los siguientes apartados se centrarán en el mecanismo de atención, *transformers* y modelos de lenguaje pre-entrenados ya que suponen el estado del arte para problemas de PLN y serán los modelos utilizados.

2.2.1. Transformers

En primer lugar, un *transformer* es una arquitectura de red neuronal considera como estado del arte [49] en los modelos secuenciales. En el ámbito del PLN soluciona los problemas de dependencias largas que presentan las redes neuronales recurrentes [46] las cuales procesan el texto palabra a palabra y debido a su propia arquitectura empeoran con frases largas las cuales necesitan de contexto.

La arquitectura de *transformers* fue propuesta por primera vez en el artículo "Attention is All You Need" [49]. Esta arquitectura tiene una estructura *encoder-decoder*, esta estructura se basa en codificar la secuencia de entrada para posteriormente descodificarla, es una estructura típica en la tarea de traducción automática [10]. La peculiaridad de esta arquitectura es el uso únicamente de mecanismos de atención sin la inclusión de RNN o CNN [49]. En la figura 2.2 se puede observar con detalle la arquitectura *transformers*.

El *encoder* se compone de 6 bloques idénticos, cada uno de ellos se compone de una capa de *multi-head self attention mechanism* (la cual se explica en profundidad en la sección 2.2.2) seguido de una capa *Fully Connected*, es decir una capa de neuronas totalmente conectadas. Además después de tanto el mecanismo de atención, como de la capa *Fully Connected* se encuentra una capa residual [22] y una capa de normalización [5]. La primera de estas se utiliza para que no se pierda ni se degrade la información mientras que la capa de normalización se utiliza para normalizar los datos los cuales han sido "deformados" tras el resto de capas, ayudando así al proceso de aprendizaje.

El *decoder* también se compone 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 del bloque una capa de *multi-head self attention mechanism* enmascarada para que tan solo utilice las palabras que ya ha procesado.

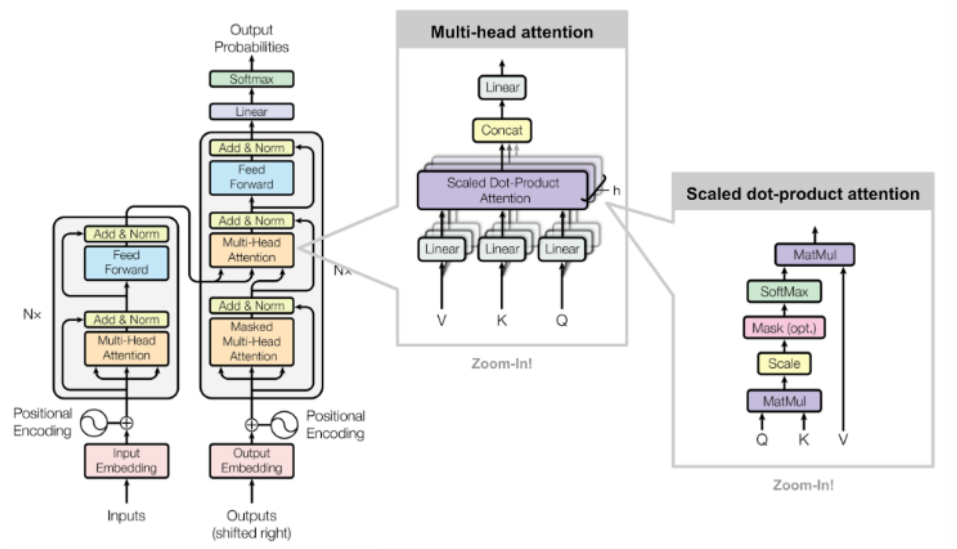


Figura 2.2: Arquitectura de *Transformers*, *Multi-Head-Attention* y *Attention*. Fuente: Medium.

2.2.2. Mecanismos de atención

Los mecanismos de atención [6] permiten a la red centrarse en una parte de la entrada y darle menos atención a otras. Se encarga de analizar la totalidad de la secuencia de entrada y encontrar relaciones entre las distintas partes de la secuencia. Para ello expresa numéricamente las relaciones entre las partes según los grados de asociación entre palabras.

Existen distintos tipos de mecanismos de atención, entre ellos el que se utiliza en la arquitectura *transformers* es llamado *Scaled Dot-Product Attention* el cual tiene la siguiente formulación:

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

Además, en vez de utilizar una única función de atención se utilizan varias y posteriormente se concatenan, lo cual lo hace un modelo de *Multi-Head-Attention*. Este mecanismo de atención recibe como entrada *queries* y *keys* [6] de dimensión d_k y valores de dimensión d_v. Se calcula el producto escalar QK^T y posteriormente se divide entre $\sqrt{d_k}$. La única diferencia entre *Dot-Product Attention* y *Scaled Dot-Product Attention* es el factor de escala $\frac{1}{\sqrt{d_k}}$ lo cual hace que este tipo de atención sea más eficiente en tiempo y en espacio [46].

2.2.3. Modelos del lenguaje

Los Modelos del Lenguaje (LM) son *word embeddings* contextuales, es decir, los modelos del lenguaje son distribuciones de probabilidad sobre las palabras o secuencias de palabras. Dada una frase, un modelo del lenguaje ha de identificar si esta frase es plausible o no dada la gramática del lenguaje [33]. En los últimos años se han publicado una gran cantidad de modelos basado en redes neuronales profundas, entre los más populares en la literatura destacan los siguientes:

- BERT
- RoBERTa
- GPT2
- GPT3
- ELECTRA

Para este trabajo se ha decidido utilizar BERT y RoBERTa ya que son modelos del lenguaje muy potentes desarrollados y respaldados por grandes instituciones como lo son Google y Facebook respectivamente. Además estos modelos a diferencia de otros tienen acceso libre desde diversas APIs como la de HuggingFace y Tensorflow, lo cual facilita su uso.

BERT

BERT (Bidirectional Encoder Representations from Transformers) [13] surgió con la idea de crear un modelo base que aprende a interpretar el lenguaje en general y una vez pre-entrenado el modelo de lenguaje base, añadir capa/s adicionales para especializarse en una tarea en concreto. Si bien la arquitectura *Transformers* se basa en una fase de codificación y otra de decodificación, en BERT tan solo hay codificación.

El nombre de la arquitectura en concreto que utiliza BERT es *multi-layer bidirectional Transformer encoder* [13]. Como se puede observar en la figura 2.3, se trata de una concatenación de varias capas, siendo cada una de una estas capas un bloque *transformers* como los explicados anteriormente en el apartado 2.2.1.

El entrenamiento de BERT se basa en 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 *BooksCorpus* [13]. En concreto BERT se entrena en las tareas de *Masked Language* y *Next Sentence Prediction*, lo cual ayuda BERT a entender el lenguaje. En la figura 2.4 se puede observar de forma gráfica este proceso de entrenamiento.

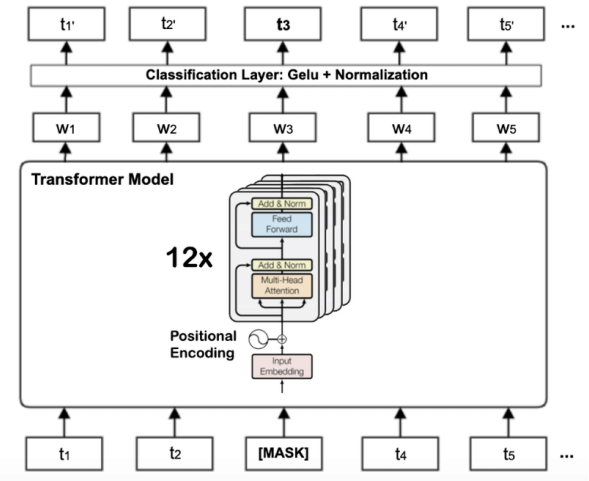


Figura 2.3: Arquitectura de BERT junto con la arquitectura *Transformers*. Fuente: ResearchGate.

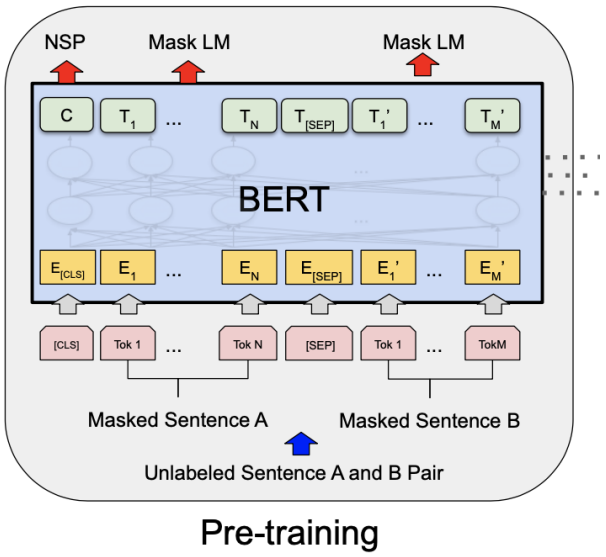


Figura 2.4: Representación de la fase de entrenamiento de BERT. Fuente: Medium.

Posteriormente en la segunda fase, se realiza un *fine-tuning* donde BERT aprende a resolver el problema en concreto que se trata de resolver. Para ello se deben añadir las capas de salida que requiera la tarea en concreto a resolver.

RoBERTa

RoBERTa: Robustly Optimized BERT pre-training Approach es una variante de BERT introducida por *Facebook AI*[2] en 2019 [24]. Algunas de las principales diferencias con BERT son las siguientes:

- **Enmascaramiento dinámico:** Mientras que BERT utiliza un enmascaramiento estático, es decir, la misma parte del texto está enmascarada en cada época, en RoBERTa se enmascaran distintas partes del texto en cada época, lo cual hace que el modelo sea mucho más robusto.
- **Eliminación de NSP:** A diferencia de BERT, RoBERTa durante el pre-entrenamiento tan solo se entrena para la tarea de MLM (*Masked Language*) y se elimina la tarea de NSP (*Next Sentence Prediction*) ya que se observó que no era muy útil.
- **Entrenado con más datos:** Mientras que la cantidad total de datos de BERT es de 16GB, RoBERTa también es entrenado con otros conjuntos de datos como *CC-NEWS*, *OPENWEBTEXT* y *STOREIS*, lo cual eleva el tamaño de los datos a 160GB.
- **Mayor tamaño del *batch*:** Mientras que BERT utiliza un tamaño del *batch* de 256 con 1 millón de pasos, RoBERTa utiliza un tamaño de *batch* de 8.000 con 300.000 pasos, lo cual tiene como consecuencia una mejora en la velocidad y rendimiento del modelo.

[2]https://ai.facebook.com/

Capítulo 3

Detección de *Fake News*

Para realizar un análisis científico del problema de la detección de *fake news* en primer lugar se han de analizar los conjuntos de datos disponibles en la literatura. Para ello se buscarán los conjuntos de datos que tengan un mayor impacto y los más recientes. Una vez escogidos los conjuntos de datos a utilizar para el estudio se realizará un análisis de los datos y sus características principales, lo cual supone el primer paso en la creación de modelos robustos de *Machine Learning*. Posteriormente se presentarán unos modelos base para finalmente presentar los modelos propuestos para el estudio y sus resultados correspondientes.

En la Sección 3.1 se mostrarán los conjuntos de datos más populares y recientes de la literatura. En la Sección 3.2 se realizará un análisis exploratorio de los datos. Finalmente en la Sección 3.3 se tratará la resolución del problema en sí, analizando una serie de modelos base para luego realizar una batería de *fine-tunings* con distintas configuraciones.

3.1. Conjuntos de datos disponibles

A la hora de resolver el problema de detección de *fake news* se ha de elegir un conjunto de datos. Para ello se ha realizado un estudio e investigación entre los conjuntos de datos más populares de la literatura y los más recientes. A continuación se muestra un listado de los conjuntos de datos analizados y sus características principales. Posteriormente en la tabla 3.1 se muestra un breve resumen de los conjuntos de datos analizados.

- ***Fact checking dataset*[47]**. Es el primer conjunto de datos público para la detección de *fake news*. Los datos provienen de *Politifact*[1]

[1]https://www.politifact.com/

y Channel4[2]. El conjunto de datos 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.

- ***EMERGENT*** **[16]**. Al igual que el anterior conjunto de datos, es de los primeros trabajos para la verificación de *fake news*. En total contiene 2.595 artículos de webs de rumores como *Snopes*[3] y Twitter[4]. El conjunto de datos es para la clasificación de postura, incluye afirmaciones con algunos documentos a favor o en contra.

- ***LIAR*** **[48]**. Este conjunto de datos incluye 12.836 declaraciones cortas recogidas de Politifact, cada una de ellas medida en una escala entre 6 opciones de veracidad. También incluye información sobre la temática, partido político, contexto y orador.

- ***Fever*** **[45]**. Este conjunto de datos contiene 185.445 afirmaciones generadas a partir de Wikipedia[5]. Cada declaración está etiquetada como: soportada, refutada, o sin suficiente información. Además las declaraciones tienen señalado que frases de Wikipedia utilizan como evidencia de la veracidad.

- ***FAKESNEWSNET*** **[41]**. Consiste en la cabecera y cuerpo del texto de artículos de *fake news* obtenidos en BuzzFeed[6] y PolitiFact. También recolecta información de Twitter sobre el compromiso social de estos artículos.

- ***BUZZFEEDNEWS*** **[23]**. Contiene 2.282 publicaciones de 9 agencias de Facebook. Cada uno de estos post ha sido verificado por 5 periodistas de BuzFeed. Una de las ventajas del conjunto de datos es que los artículos están recogidos de periódicos de distintas ideologías políticas.

- ***SOME-LIKE-IT-HOAX*** **[44]**. Este conjunto de datos contiene 15.500 publicaciones de 32 páginas de Facebook distintas. Entre estas páginas, 14 son de conspiración y 18 de organizaciones científicas.

- ***PHEME*** **[53]** Es un conjunto de datos de Twitter el cual contiene 330 hilos de Twitter (una serie de 4.842 tweets conectados de una persona) sobre 9 eventos de actualidad. Cada uno de estos está etiquetado como verdadero o falso.

[2]https://www.channel4.com/
[3]https://www.snopes.com/
[4]https://twitter.com
[5]https://es.wikipedia.org/
[6]https://www.buzzfeed.com/

- ***CREDBANK*** **[28]** Contiene 60 milliones de *tweets* agrupados en 1.049 eventos. Cada evento ha sido clasificado en una escala de 5 de veracidad por 30 anotadores humanos.

- ***FNC-1 dataset*** **[37]**. *Fake News Challenge* es un conjunto de datos 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.

- ***Spanish Fake News Corpus*** **[34]**. Este conjunto de datos es en español y contiene 971 noticias sobre 9 dominios distintos. La clasificación de las noticias es de 2 clases, verdadero o falso. Existe una versión más reciente del conjunto de datos la cual añade 572 noticias nuevas, las cuales al ser más actuales añaden la temática del COVID-19 y utiliza estas nuevas noticias como conjunto de test y las anteriores como conjunto de entrenamiento.

- ***CONSTRAINT@AAAI 2021*** **[31, 32]**. Este conjunto de datos consiste en 10.700 publicaciones en inglés de las cuales 5.100 son falsas y 5.600 verdaderas. Las noticias verdaderas están recogidas de Twitter y aportan información útil sobre el COVID-19. Por otro lado las noticias falsas provienen de Twitter, Facebook y Whatsapp además de distintos sitios de fact-cheking como Politifact, NewsChecker, Boomlive, etc [32].

Estos 2 últimos conjuntos de datos han sido los seleccionados para la realización del trabajo. Respecto al conjunto de datos en inglés se ha escogido este al ser un conjunto de datos el cual tiene una competición en un congreso muy importante como es CONSTRAINT@AAAI[7]. Además se trata de tweets extraídos recientemente y sobre un tema de actualidad y que atrae tantas *fake news* como es el COVID-19, el confinamiento y la pandemia [38]. También es de gran interés que sea sobre Twitter ya que gran cantidad de las *fake news* actuales se difunden por redes sociales [19]. Finalmente tiene una considerable cantidad de datos lo cual lo hace más atractivo para el proceso de aprendizaje.

Respecto al conjunto de datos en español se ha seleccionado este en concreto ya que también se proviene de una competición importante y sobre un taller muy relevante en español sobre PLN como lo es IberLEF[8], el cual se celebra anualmente en el congreso internacional de la Sociedad Española de Procesamiento del Lenguaje Natural. Además se ha de tener en cuenta que

[7]https://lcs2.iiitd.edu.in/CONSTRAINT-2021/
[8]https://sites.google.com/view/iberlef2022

Conjunto de datos	N° Items	Idioma	Tipo de desinformación	Dominio	N° Clases
Fact checking dataset	221	Inglés	Fake News	Política y sociedad	5
EMERGENT	2.595	Inglés	Rumores	Sociedad y tecnología	3
LIAR	12.800	Inglés	Fake News	Política	6
FEVER	185.445	Inglés	Fake News	Sociedad	3
FakesNewsNet	422	Inglés	Fake News	Sociedad y política	2
BuzzFeed News	2.282	Inglés	Fake News	Política	4
Some like it hoax	15.500	Inglés	Fake News	Ciencia	2
PHEME	330	Inglés y alemán	Rumor detection	Sociedad y política	3
CREDBANK	60 millones	Inglés	Rumores	Sociedad	5
FNC-1	49.972	Inglés	Fake News	Política, sociedad y tecnología	4
Spanish Fake News Corpus	1.543	Español	Fake News	Ciencia, política, economía, educación, entretenimiento, política, salud, seguridad y sociedad	2
CONSTRAINT AAAI 2021	10.700	Inglés	Fake News	COVID-19	2

Tabla 3.1: Características de los conjuntos de datos que han sido considerados.

en la literatura el número de conjuntos de datos en español es considerablemente bajo en comparación a los conjuntos de datos en inglés por tanto el abanico de posibilidades es bastante menor.

3.2. Análisis Exploratorio de los datos

En primer lugar se va realizar un análisis básico de ambos conjuntos de datos. Este análisis sirve como toma de contacto con los datos, aportando información relevante para su futuro estudio y resolución. Para ello se medirá el tamaño de los conjuntos de datos, como están distribuidos los datos entre los subconjuntos de entrenamiento y de test, cual es el número de palabras del conjunto de entrenamiento y finalmente la media, mediana y máxima del número de palabras en el conjunto de entrenamiento. En la tabla 3.2 se pueden observar estas medidas las cuales nos sirven para conocer de mejor forma los datos y sus peculiaridades.

Como se puede observar el conjunto de datos de CONSTRAINT AAAI tiene una mayor cantidad de datos desde el punto de vista de noticias únicas pero por otro lado el conjunto de datos de IberLEF tiene mayor número de palabras por cada noticia al tratarse cada una de estas noticias de textos más largos que el tamaño máximo que permite Twitter por cada uno de sus

Medida	**CONSTRAINT AAAI**	**IberLEF**
Tamaño del conjunto de datos	10.700	1.248
N° de palabras del conjunto de entrenamiento	227.724	68.616
Tamaño del conjunto de entrenamiento	8.560	676
Tamaño del conjunto de test	2.140	572
Media del número de palabras en el conjunto de entrenamiento	26,61	210,47
Mediana del número de palabras en el conjunto de entrenamiento	25	221,5
Máximo número de palabras en el conjunto de entrenamiento	130	303

Tabla 3.2: Analisís Básico de los conjuntos de datos.

tweets.

Respecto al tamaño de los conjuntos de test y de entrenamiento, podemos observar que para el conjunto de datos de CONSTRAINT AAAI el conjunto de entrenamiento representa en torno al 80 % de los datos, mientras que en el caso del conjunto de datos de IberLEF representa en torno al 55 % de los datos. Con esto observamos que el conjunto de datos de CONSTRAINT AAAI sigue una distribución de los conjuntos de *test* y de entrenamiento más típica de problemas de *Machine Learning* como puede ser una distribución 80-20.

Por otro lado se puede observar que tanto la media, mediana y máximo del número de palabras en el conjunto de CONSTRAINT AAAI es considerablemente menor que el conjunto de datos de IberLEF. Esto sucede de nuevo debido a que al tratarse de un conjunto de datos de Twitter, el número de caracteres máximo está limitado y esto de forma directa afecta el número de palabras que puede tener un tweet y hace que tanto la media, como mediana y máximo sean menores.

Finalmente cabe destacar que es de gran importancia conocer el máximo número de palabras ya que tanto BERT como RoBERTa necesitan como parámetro el tamaño del texto y por tanto se utilizará el máximo del número de palabras como parámetro.

A continuación se va a realizar un estudio para cada conjunto de datos en el que se analizarán distintas medidas con el objetivo de intentar esclarecer algunas características que diferencien las noticias falsas de las verdaderas. Para ello en primer lugar se mostrará para cada conjunto de datos un histograma del número de palabras de las noticias con el fin de intentar observar si se distribuyen homogéneamente o existe una clara diferencia entre el número de palabras de las distintas noticias.

Posteriormente se procederán a analizar los 10 N-gramas[9] más repetidos de ambos cojuntos de datos, en concreto unigramas, bigramas y trigramas. Un N-grama es un conjunto de n elementos consecutivos en un texto, por tanto, con los unigramas, bigramas y trigramas, se analizarán las 10 combinaciones de 1, 2 y 3 palabras seguidas más repetidas en todo el conjunto de datos, con la intención de intentar analizar alguna particularidad del conjunto de datos que pueda ayudar en el análisis.

Finalmente se mostrarán nubes de palabras de ambos conjuntos de datos. Las nubes de palabras muestran las palabras que más se repiten en todo el conjunto de datos de forma gráfica, mostrando su tamaño según el número de veces que se repiten. Esta gráfica nos aporta información similar a los unigramas pero de forma más intuitiva. Además para finalizar se analizarán las nubes de palabras separando el conjunto de datos entre noticias falsas y verdaderas para intentar encontrar diferencias entre ambos conjuntos.

3.2.1. CONSTRAINT AAAI

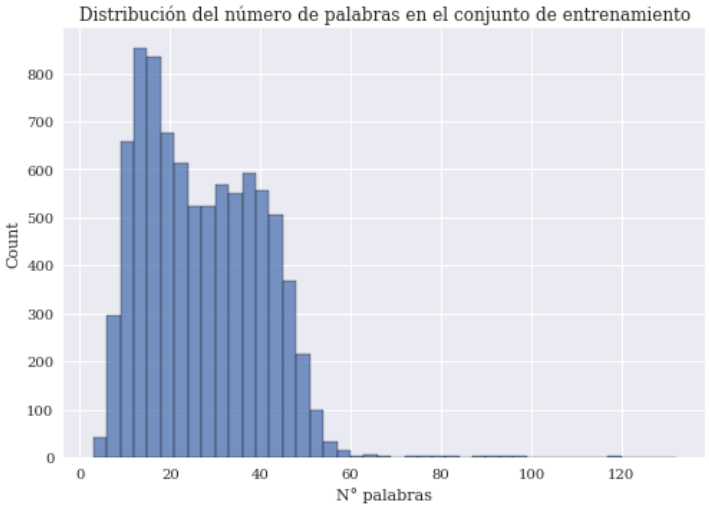


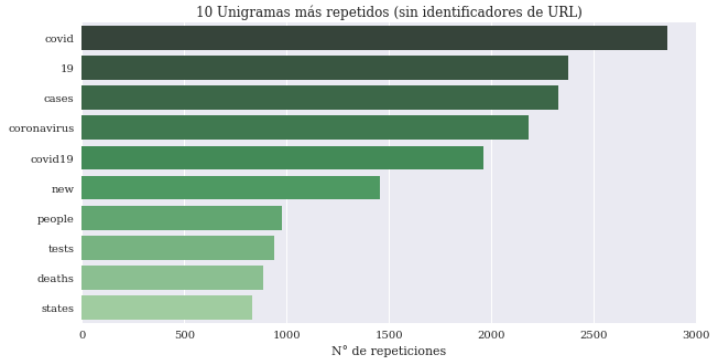
Figura 3.1: Distribución del número de palabras del conjunto de entrenamiento (CONSTRAINT AAAI).

Como se puede observar en la figura 3.1, el histograma del número de palabras está distribuido homogéneamente entre las 10 y 50 palabras aproximadamente por noticia, en este caso por *tweet*.

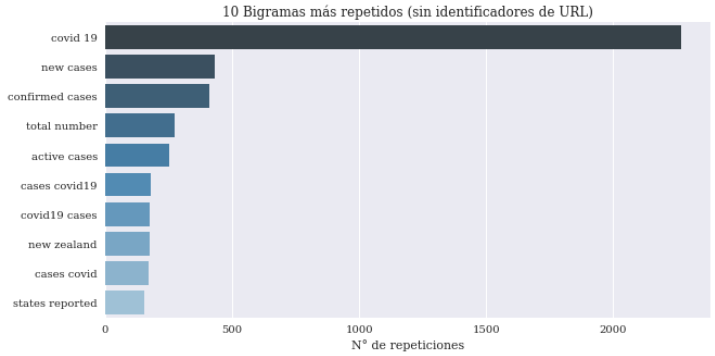
En la figura 3.2 se pueden observar los 10 N-gramas más repetidos junto al número de veces que se repite cada uno de ellos en el conjunto de datos. Al tratarse de un conjunto de datos centrado en el COVID-19, como era de

[9] https://en.wikipedia.org/wiki/N-gram

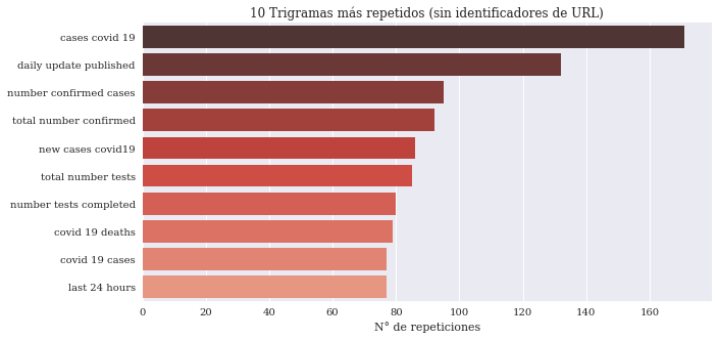
esperar, entre los N-gramas más significativos se encuentran términos como: "covid", "cases", "tests", "confirmed cases", "new cases", "daily update published" y "last 24 hours". Lo cual muestra la importancia del conocimiento específico del tema y una codificación adecuada de estas palabras para el correcto funcionamiento del modelo.



(a) Unigramas



(b) Bigramas



(c) Trigramas

Figura 3.2: 10 N-gramas más repetidos (CONSTRAINT AAAI).

En las nubes de palabras de la figura 3.3 se pueden observar por tamaño las palabras que más se repiten. Algunas de estas palabras ya han sido

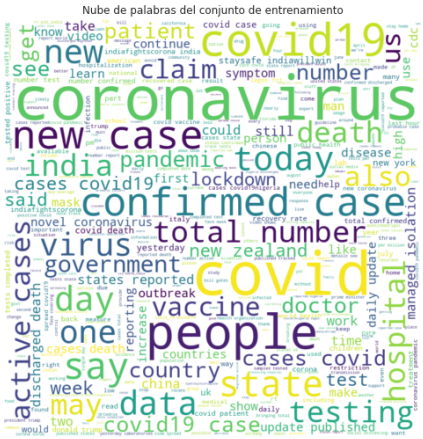
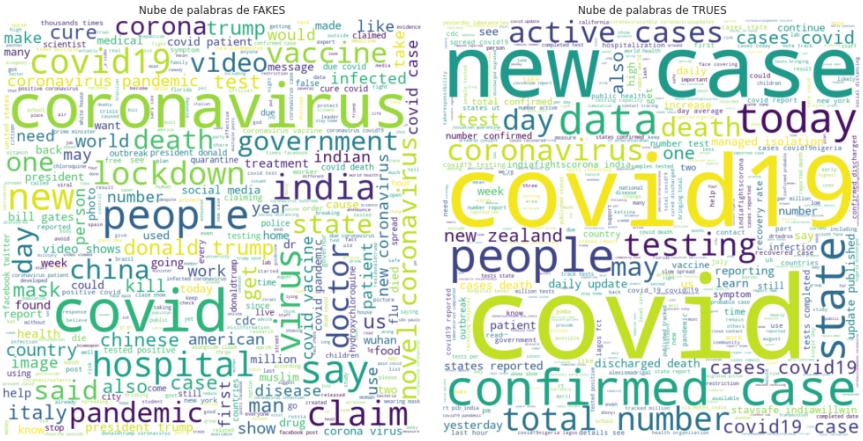


Figura 3.3: Nube de palabras más repetidas (CONSTRAINT AAAI).

mencionadas en los N-gramas de la figura 3.2. De nuevo se puede observar que entre las palabras que más se repiten se encuentran términos muy relacionados con el COVID-19 y la pandemia como pueden ser "coronavirus", "covid19" o "hospital" entre otros.

Finalmente, prestando atención en las nubes de palabras segregadas por noticias falsas y verdaderas de la figura 3.4 podemos observar algunas diferencias entre estos 2 conjuntos. En primer lugar se observa que en el conjunto de noticias verdaderas destacan palabras más "neutrales" (desde un punto de vista de posicionamiento político y social) como pueden ser: "covid19", "new case", "confirmed case" y "total number". En cambio en el conjunto de noticias falsas estas palabras no se encuentran tan presentes mientras que empiezan a destacar otras como: "doctor", "video", "hospital", "pandemic", "donald trump" y "vaccines".



(a) Falsas (b) Verdaderas

Figura 3.4: Nube de palabras más repetidas separando los conjuntos de noticias falsas y verdaderas (CONSTRAINT AAAI).

3.2.2. IberLEF

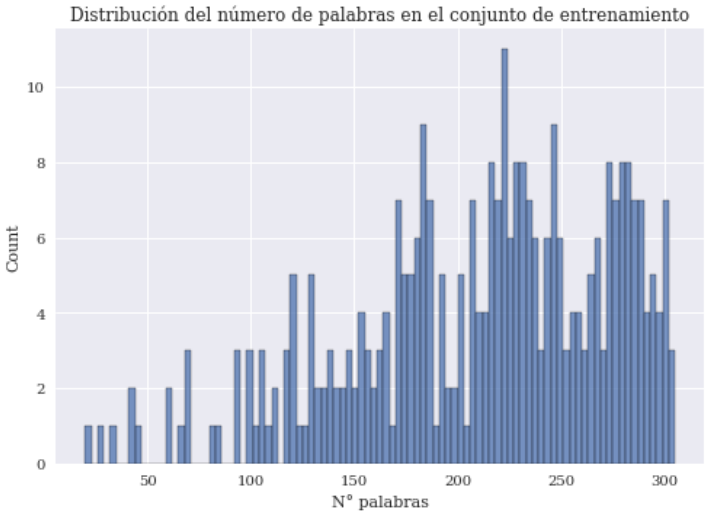
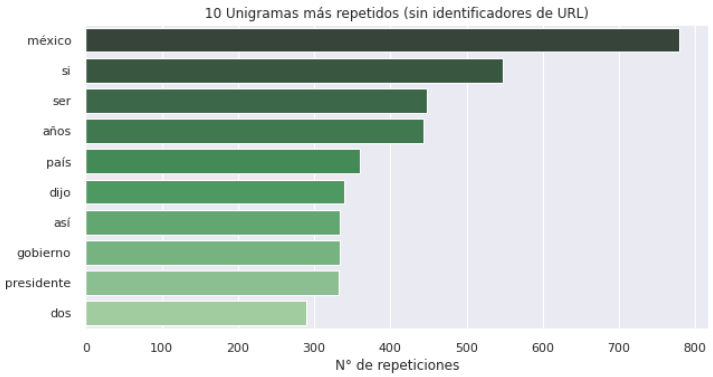


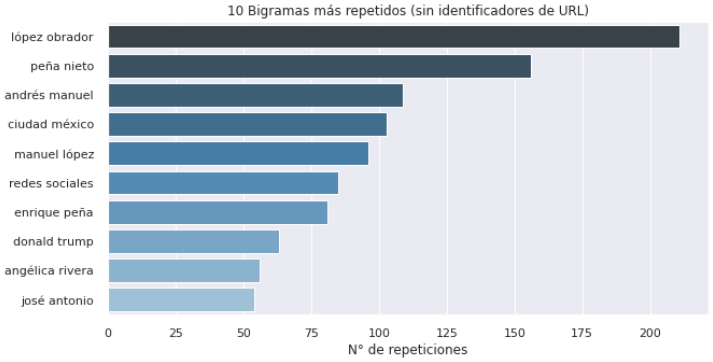
Figura 3.5: Distribución del número de palabras del conjunto de entrenamiento (IberLEF).

Como se puede observar en el histograma de la figura 3.5, para este conjunto de datos, la distribución del número de palabras tiene una mayor variabilidad respecto a la del conjunto de datos de CONSTRAINT AAAI, es decir, el tamaño del texto de cada noticia varía en gran medida entre sí. Esto es debido a que al tratarse de artículos de prensa y noticias en periódicos el

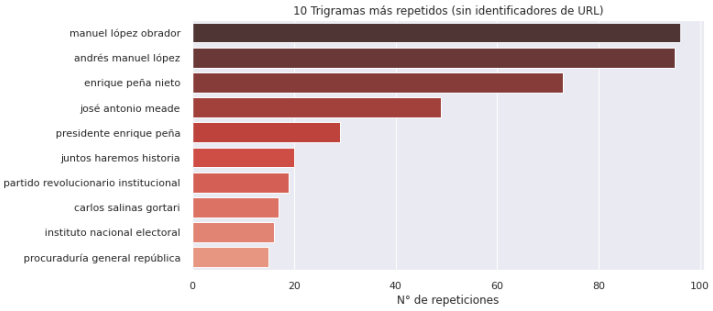
tamaño de los textos no está limitado como en Twitter y hace que el tamaño de la noticia pueda ser totalmente variable.



(a) Unigramas con preprocesado



(b) Bigramas con preprocesado



(c) Trigramas con preprocesado

Figura 3.6: 10 N-gramas más repetidos (IberLEF)

En la figura 3.6 se pueden observar los 10 N-gramas más repetidos junto al número de veces que se repite cada uno de ellos en el conjunto de datos. A diferencia del conjunto de datos de CONSTRAINT AAAI, al no tratarse de un conjunto de datos de un dominio específico como lo es el COVID-19, entre los N-gramas más repetidos aparecen otro tipo de pala-

bras más generales. Entre los términos más significativos están: “méxico”, “gobierno”, “presidente”, “lopez obrador”, “peña nieto”, “partido revolucionario institucional”, “donald trump”. Es decir, entre los 10 N-gramas más representativos se encuentran términos bastante relacionados con la política y actualidad de México.

En las nubes de palabras de la figura 3.7 se pueden observar por tamaño las palabras que más se repiten. Algunas de estas palabras ellas ya han sido mencionadas en los N-gramas de la figura 3.6. De nuevo se puede observar que entre las palabras que más se repiten se encuentran términos muy relacionados con la actualidad política y social de México como pueden ser “México”, “gobierno” o “presidente” entre otros.

Figura 3.7: Nube de palabras más repetidas (IberLEF).

Analizando las las nubes de palabras segregadas por noticias verdaderas o falsas de la figura 3.8, observamos que a diferencia del conjunto de datos de CONSTRAINT AAI, no existen diferencias evidentes entre las palabras que se repiten en las noticias falsas como en las verdaderas, por lo que puede que esto se traduzca en una mayor dificultad del modelo para diferenciar entre noticias falsas y verdaderas.

(a) Falsas (b) Verdaderas

Figura 3.8: Nube de palabras más repetidas separando los conjuntos de noticias falsas y verdaderas (IberLEF).

3.3. Estudio y resolución del problema

Para aportar una solución robusta al problema de detección de *fake news*, en primer lugar se presentarán unos casos base basados en modelos clásicos de *Machine Learning* los cuales servirán como una primera aproximación y comparativa para el estudio. Posteriormente, se procederá a aplicar la técnica de fine-tuning utilizando distintos modelos de BERT y RoBERTa.

3.3.1. Casos Base

Durante esta subsección se van a presentar y comparar distintos modelos básicos de *Machine Learning* como modelos base o *baselines*. Como característica de entrada se va a utilizar el *Tf-idf*, en concreto se utilizará la implementación de la biblioteca *scikit-learn* de los métodos y algoritmos para todo este apartado. Los modelos que se van a utilizar en la comparativa son: *Linear SVC*, *Logistic Regression*, *Decision Tree Classifier* y *Gradient Boost Classifier*.

Para la comparación entre los distintos modelos presentados en este trabajo se trabajará con la medida F1-score. Es una medida típica a la hora de evaluar modelos de *Machine Learning* y el motivo del uso de esta medida es que hace una combinación entre la precisión del modelo y la exhaustividad (comúnmente conocido como "recall"). Su formulación es la siguiente:

$$F_1 = \frac{precision \cdot recall}{precision + recall}$$

Modelo	Constraint AAAI	IberLEF
Linear SVC	**0.93645**	**0.68706**
Logistic Regression	0.92336	0.65210
Decision Tree Classifier	0.87430	0.53846
Gradient Boost Classifier	0.87196	0.63287

Tabla 3.3: F1-score para los modelos básicos.

A continuación, en la tabla 3.3 se muestran los resultados obtenidos por los modelos base para los distintos conjuntos de datos. Se ha de tener en cuenta que no se ha realizado una técnica de validación cruzada durante el entrenamiento, ya que, al pertenecer ambos conjuntos de datos a competiciones, tienen sus respectivos conjuntos de entrenamiento y test definidos. Por tanto, durante todo este trabajo se ha entrenado con el conjunto de entrenamiento y se ha evaluado con el de test, con el objetivo de que los resultados sean comparables con el estado del arte.

Respecto al conjunto de datos CONSTRAINT AAAI, se obtienen en general buenos resultados para todos los modelos básicos, con la principal diferencia de que los modelos basados en árboles [36], como lo son *Decision Tree Classifier* y *Gradient Boost Classifier* obtienen en torno a un 5 puntos menos de rendimiento comparados a *Linear SVC* y *Logistic Regression*.

En concreto el mejor modelo sería *Linear SVC* ya que no solo proporciona el mejor F1-score si no que además tiene en buen equilibrio en los errores de la matriz de confusión como se puede observar en la figura 3.9. Es decir, el número de falsos positivos y falsos negativos está muy parejo por lo tanto el modelo no está predominando una clase en concreto.

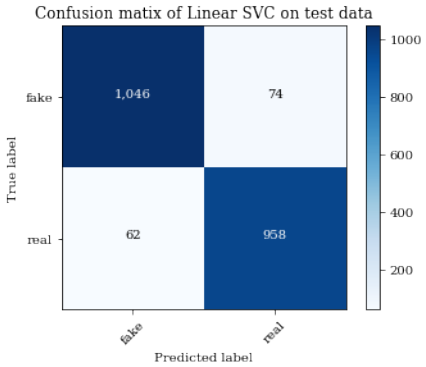


Figura 3.9: Matriz de confusión de *Linear SVC* para el conjunto de datos de CONSTRAINT AAAI.

Por otro lado, en el conjunto de datos en IberLEF los resultados no son tan buenos, estos rondan en torno al 60 % de F1-score. Al igual que en el conjunto de datos de CONSTRAINT AAAI los modelos que aportan peores

resultados son los basados en árboles mientras que el mejor modelo de nuevo es *Linear SVC*. Aunque como se puede observar en la figura 3.10 en este caso los errores no tienen tan buen equilibrio como para el conjunto de datos de IberLEF, existiendo una predominancia a predecir como verdaderas noticias falsas.

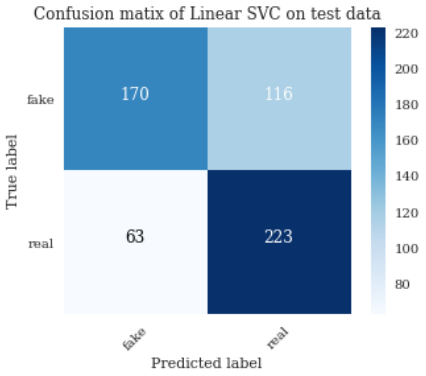


Figura 3.10: Matriz de confusión de *Linear SVC* para el conjunto de datos de IberLEF.

Por tanto se han establecido unos *baselines* los cuales son de gran ayuda para analizar el problema en mayor profundidad y establecer unos mínimos a la hora de realizar aproximaciones más complejas basadas en BERT.

3.3.2. Fine-Tuning

A continuación se procede a mostrar los distintos modelos basados en BERT con los que se ha experimentado durante la resolución del problema. Para todos los modelos se ha aplicado un proceso de *Fine-Tuning*, en el que al modelo previamente entrenado se le añade una capa de salida adecuada al problema, en concreto una capa de salida de tamaño 1 con una función de activación sigmoide. Posteriormente se re-entrena el modelo utilizando un *learning rate* pequeño utilizando los datos del problema a resolver.

Junto a esto se ha decidido experimentar el uso de diversos optimizadores más acordes al tipo de problema y más en concreto al uso de *Embeddings* (como es el caso de BERT y RoBERTa) [25]. Más en concreto para cada modelo se ha realizado la comparativa utilizando un optimizador clásico para problemas de *Deep Learning* como lo es Adam y se ha comparado con un optimizador orientado a *Embeddings* como lo es Adamax[10].

Además en los diversos modelos se ha experimentado a añadir una regularización de *dropout* de 0.5 en las capas de atención y de 0.2 en las capas

[10]https://keras.io/api/optimizers/adamax/

ocultas, tal y como se recomienda para el uso general en [15].

Por tanto para cada modelo a estudiar se van a realizar 4 pruebas.

1. Modelo + Adam
2. Modelo + Adam + Regularización *Dropout*
3. Modelo + Adamax
4. Modelo + Adamax + Regularización *Dropout*

Finalmente para cada uno de ellos se ha aplicado el mismo *learning rate* recomendado de $2e^{-5}$ [43][11] y un ϵ de $1e^{-8}$ junto al preprocesado adecuado para cada modelo el cual se especificará posteriormente para cada modelo. Todos los modelos han sido entrenados durante 20 épocas, utilizando un 80 % como conjunto de entrenamiento y un 20 % como conjunto de validación, con una política de *early stopping* de 5 épocas de paciencia, monitorizando el error de la función de perdida en el conjunto de validación. Se utiliza el error en vez de la precisión ya que el error cuantifica la certeza del modelo (valor continuo), mientras que la precisión tan solo cuantifica el número de predicciones correctas (valor discreto).

A la hora de seleccionar los modelos de lenguaje para resolver el problema para estos conjunto de datos se ha de tener tener en cuenta que al tratarse de unos conjunto de datos de un dominio concreto o multidominio se pueden separar los modelos seleccionados de BERT en 2 grandes tipos: dependientes e independientes del dominio.

Por un lado los modelos independientes del dominio son modelos generales de BERT los cuales han recibido datos de distintos dominios, autores, etc. Aun así estos modelos pueden recibir todos los datos de distintas fuentes o de la misma, como puede ser Twitter para alguno de estos modelos que analizaremos posteriormente. La idea principal de los modelos independientes del dominio es que no están especializados en un dominio en concreto.

Por otro lado los modelos dependientes del dominio reciben datos relacionados con el dominio en cuestión, en este caso el COVID-19 y temas relacionados. Esto hace que BERT pueda "especializarse" en una temática en concreto lo cual puede suponer una diferencia sustancial de rendimiento a la hora de resolver la tarea de detección de *fake news*.

A continuación se explicarán los modelos que se han tenido en cuenta para el conjunto de datos de IberLEF y se realizarán las pruebas que se acaban de mencionar para cada uno de estos modelos. Posteriormente se hará lo mismo para el conjunto de datos de CONSTRAINT AAAI. Ha de tenerse en cuenta que cuando se explique cada uno de estos modelos solo se hablará de los datos utilizados durante el proceso de entrenamiento, ya que

[11] https://github.com/google-research/bert

estas "versiones" de BERT y RoBERTa solo se diferencian de las originales en los conjuntos de datos de entrenamiento y el proceso de entrenamiento y no en la arquitectura del modelo.

IberLEF

A diferencia del conjunto de datos de CONSTRAINT AAAI, el conjunto de datos de IberLEF es multidominio, pero con la peculiaridad de que en el conjunto de entrenamiento no se encuentra la temática COVID-19 mientras que en el de test sí.

Esto supone un reto a la hora de resolver este problema ya que los modelos *Machine Learning* necesitan que los datos del entrenamiento y del test sean uniformemente idénticamente distribuidos[12]. Y este no es el caso, ya que no es solo no están distribuidos uniformemente si no que en el conjunto de entrenamiento no aparece la temática COVID-19 en ningún momento, por lo tanto el modelo no podrá aprender correctamente sobre las palabras específicas de la temática ni sus relaciones con el resto.

A continuación se procede a explicar brevemente los 3 modelos seleccionados y sus características principales en la tabla 3.4. Cabe destacar que no se detallará el preprocesado utilizado para los siguientes modelos ya que el texto proporcionado por el conjunto de datos ya viene preprocesado, así que únicamente se pasará a minúscula el texto cuando el modelo lo requiera. A continuación se procede a mostrar para cada uno de los modelos las pruebas correspondientes y sus resultados.

[12] https://en.wikipedia.org/wiki/Independent_and_identically_distributed_random_variables

Modelo base	Dominio	Tamaño del modelo	Observaciones
PlanTL-GOB-ES Roberta-large-bne	Independiente	570GB	Versión más grande del conjunto de modelos del lenguaje del Plan de Gobierno de España de Tecnologías del Lenguaje. Presentado junto al resto de modelos del plan en [20]. Supera a BETO en varios conjuntos de datos de evaluación.
PlanTL-GOB-ES BSC-BIO-ES	Dependiente	1B tokens	Versión para documentos biomédicos y clínicos del subconjunto de modelos de médicos del Plan de Gobierno de España de Tecnologías del Lenguaje. Ha sido entrenado con 91 millones de tokens de documentos clínicos y 963 millones de documentos médicos. Más detalles aquí. Supera a los modelos independientes del dominio en conjuntos de datos médicos.
BETO	Independiente	No especificado	Modelo de lenguaje de BERT para español de la universidad de Chile. Utiliza datos tanto de Wikipedia como de las fuentes del proyecto OPUS. Además agrega algunas características durante el entrenamiento típicas de RoBERTa [9]. Respecto a la cantidad concreta de datos no se especifíca pero se compara con la cantidad usada para BERT original, es decir, casi 3.000 millones de palabras.

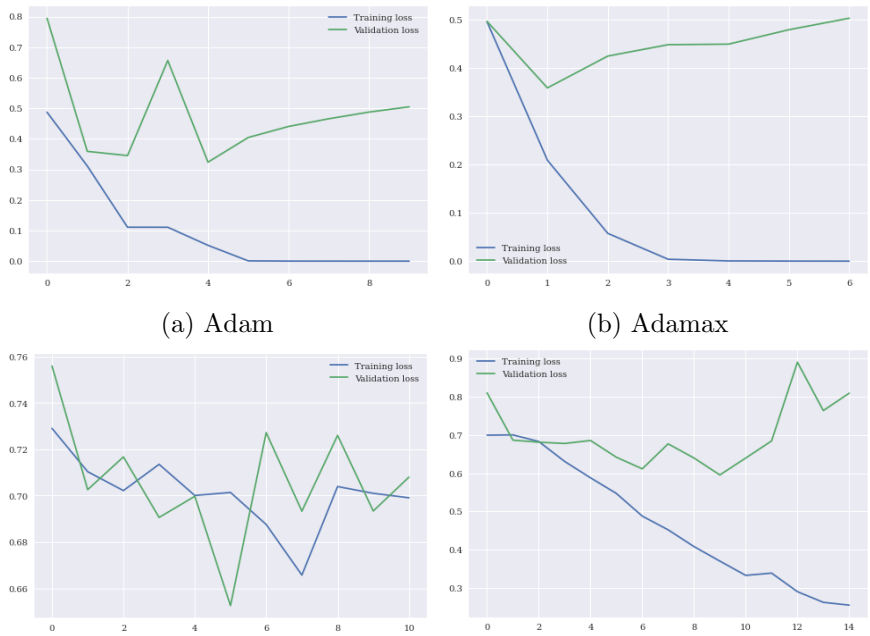
Tabla 3.4: Modelos de BERT considerados para el fine-tuning del conjunto de datos de IberLEF.

PlanTL-GOB-ES Roberta-large-bne

Como se puede observar en las curvas de aprendizaje y de pérdida de las figuras 3.11, 3.12 durante el proceso de aprendizaje de los 4 modelos se puede observar una cota claramente entorno al 70 % en el conjunto de validación. A partir de ese punto se ve mejorado el conjunto de entrenamiento pero no el de validación lo que provoca un sobreajuste del modelo a los datos. Nótese que tanto para este modelo como para el resto del trabajo, las curva de pérdida y aprendizaje están ampliadas por lo que pequeñas variaciones en los valores numéricos se pueden ver representados como grandes "picos" y fluctuaciones en las gráficas.

En la tabla 3.5 observamos que para el modelo RobertaGob, la mejor configuración es la que utiliza el optimizador Adamax con un 71.85 % de F1-score. Aunque observando las matrices de confusión de la figura 3.13 se puede observar que todos los modelos predominan a falsos negativos o a

falsos positivos.



(a) Adam (b) Adamax

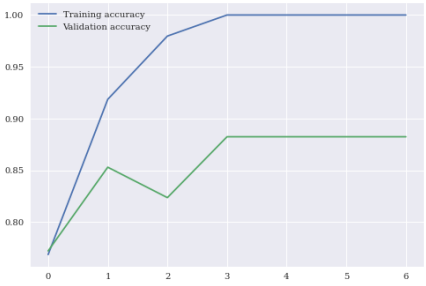
(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.11: Curvas de perdida RobertaGob.

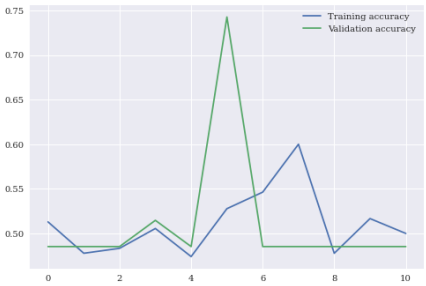
Modelo	**F1-Score**
RobertaGob Adam	71.51
RobertaGob Adam Regularization	63.46
RobertaGob Adamax	**71.85**
RobertaGob Adamax Regularization	68.88

Tabla 3.5: Modelos propuestos RobertaGob.

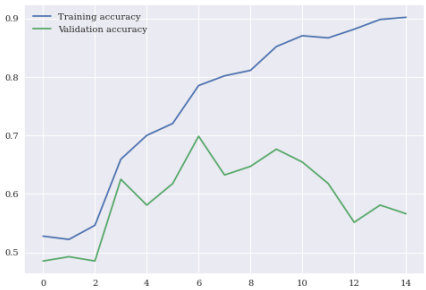
(a) Adam



(b) Adamax

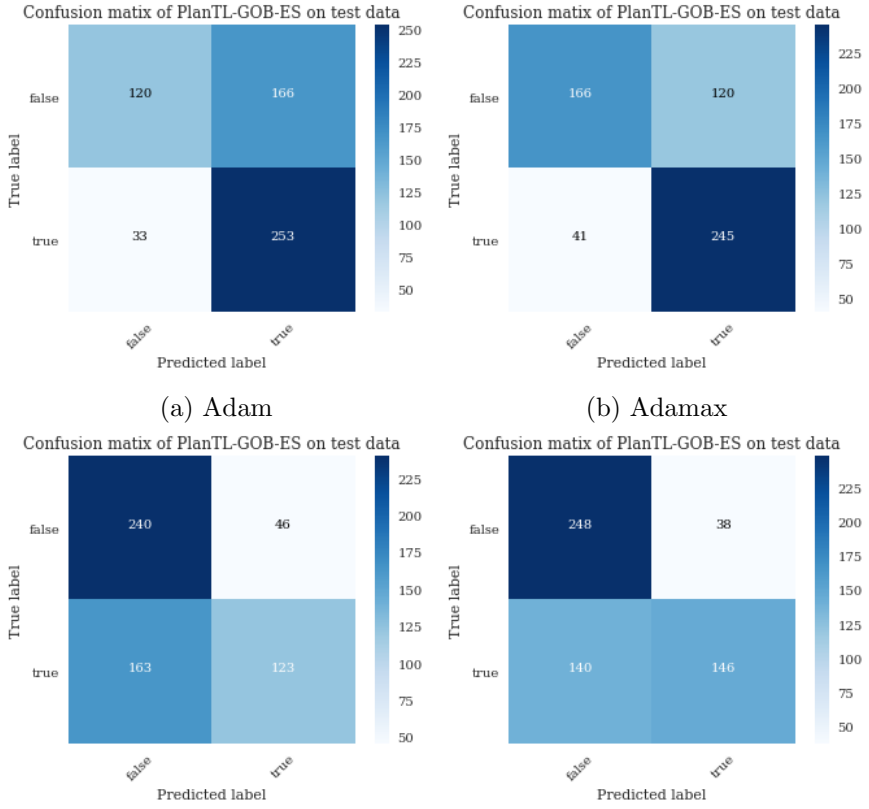


(c) Adam + Regularización



(d) Adamax + Regularización

Figura 3.12: Curvas de precisión RobertaGob.



(a) Adam (b) Adamax

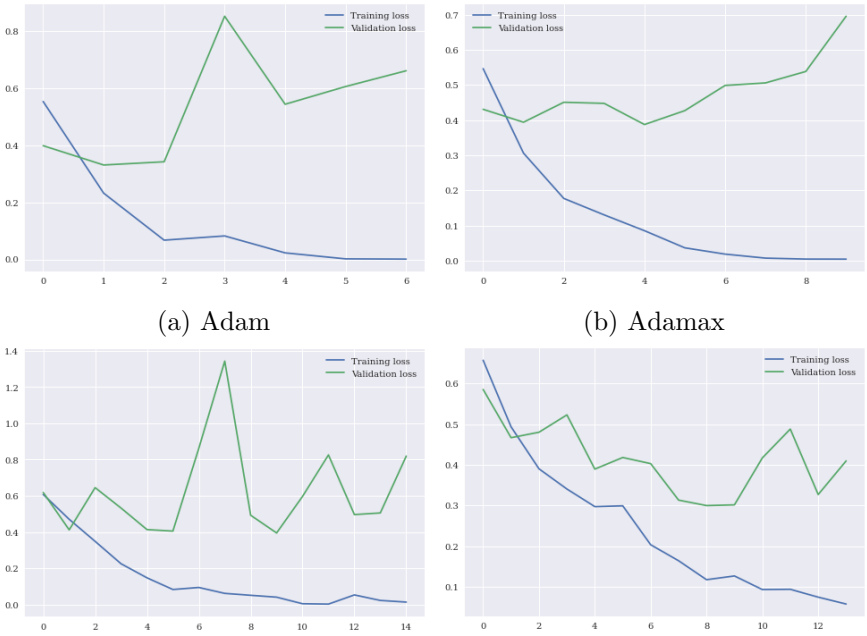
(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.13: Matrices de confusión RobertaGob.

PlanTL-GOB-ES Roberta-BIO

Como se puede observar en las curvas de aprendizaje y de pérdida de las figuras 3.14, 3.15 durante el proceso de aprendizaje de los 4 modelos se puede observar una cota claramente entorno al 80/85 % en el conjunto de validación, es decir, en torno a un 10/15 % superior que en el modelo anterior. A partir de ese punto se ve mejorado el conjunto de entrenamiento pero no el de validación lo que provoca un sobreajuste del modelo a los datos.

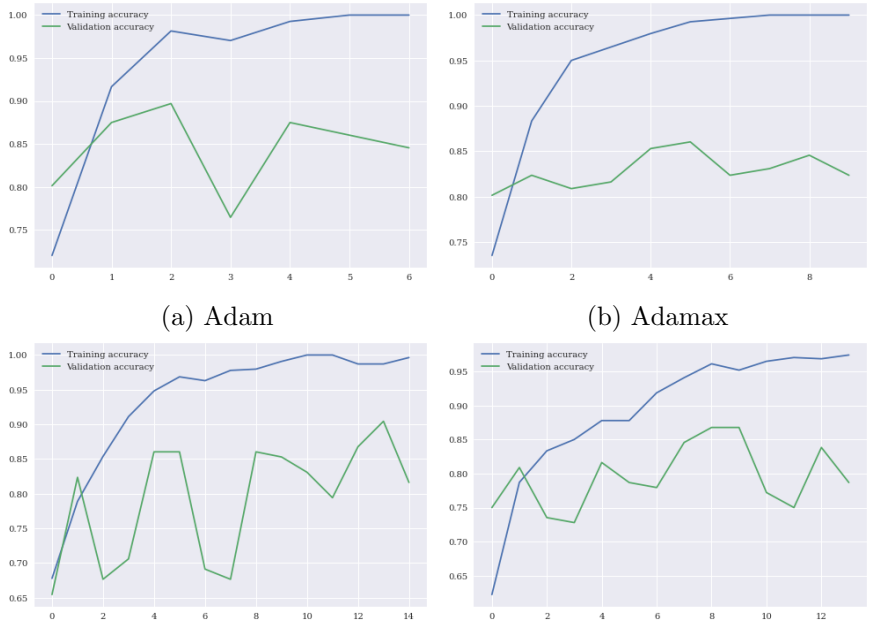
En la tabla 3.6 observamos que para el modelo RobertaBIO, la mejor configuración es la que utiliza el optimizador Adamax, con un 73.77 % de F1-score. Por lo que aunque el conjunto de validación obtenga mejores resultados que el modelo anterior, a la hora de evaluar el conjunto de test y por tanto introducir la temática COVID-19, los resultados empeoran. Además observando las matrices de confusión de la figura 3.16 se puede observar que todos los modelos predominan a falsos negativos o a falsos positivos.



(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.14: Curvas de perdida RobertaBIO.



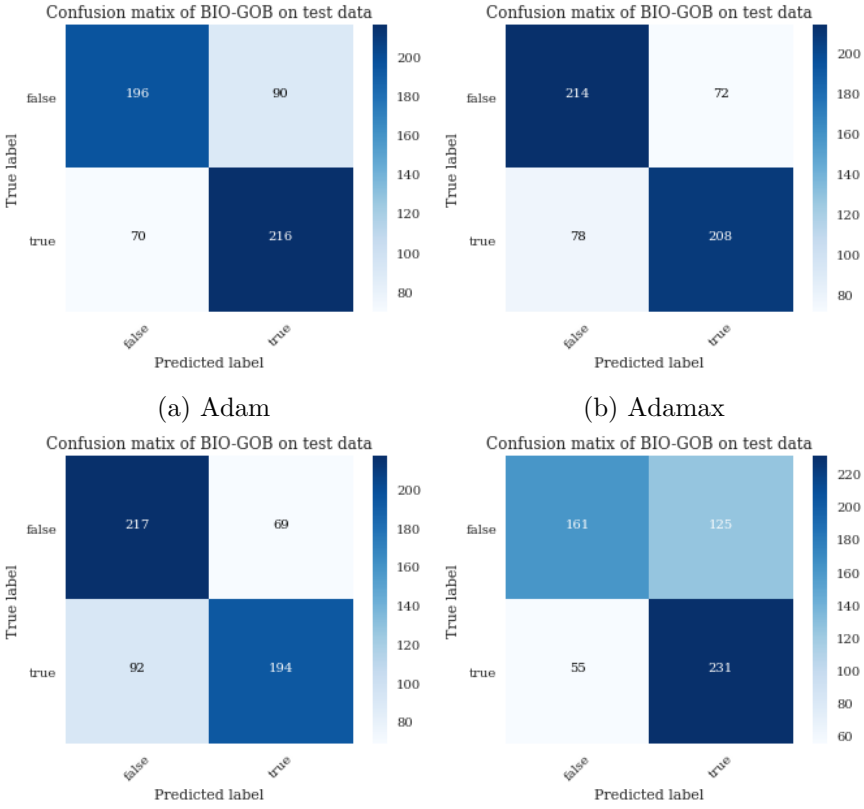
(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.15: Curvas de precisión RobertaBIO.

Modelo	**F1-Score**
RobertaBIO Adam	72.02
RobertaBIO Adam Regularization	71.85
RobertaBIO Adamax	**73.77**
RobertaBIO Adamax Regularization	68.53

Tabla 3.6: Modelos propuestos RobertaBIO.



(a) Adam (b) Adamax

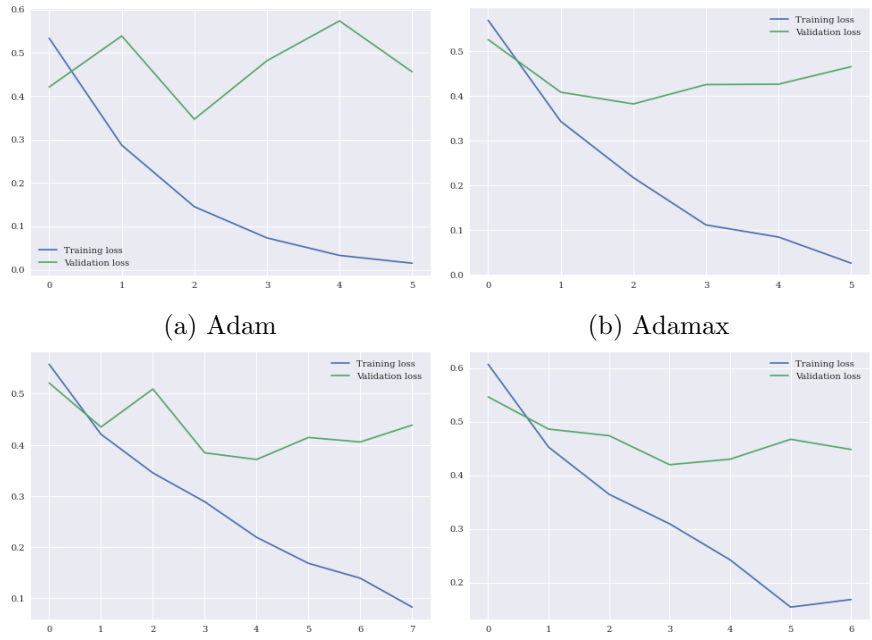
(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.16: Matrices de confusión RobertaBIO.

BETO

Como se puede observar en las curvas de aprendizaje y de pérdida de las figuras 3.17, 3.18 durante el proceso de aprendizaje de los 4 modelos se puede observar una cota claramente entorno al 80/85 % en el conjunto de validación, es decir, en torno a un 10/15 % superior que en el primer modelo. A partir de ese punto se ve mejorado el conjunto de entrenamiento pero no el de validación lo que provoca un sobreajuste del modelo a los datos.

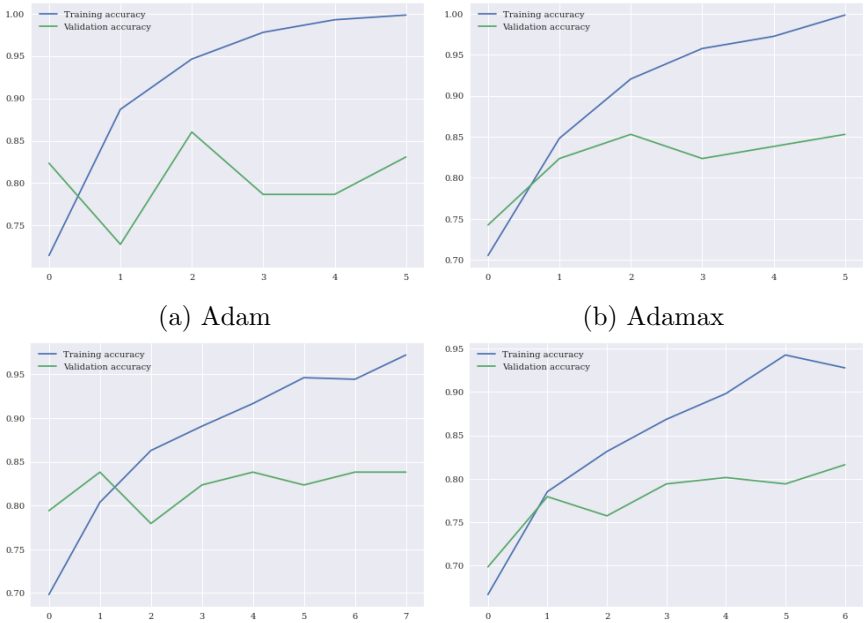
En la tabla 3.7 observamos que para el modelo BETO, la mejor configuración es la que utiliza el optimizador Adam igualado con la que utiliza Adamax y Adamax con regularización en las capas con un 67.83 % de F1-score. Por lo que aunque el conjunto de validación obtenga mejores resultados que el modelo anterior, a la hora de evaluar el conjunto de test y por tanto introducir la temática COVID-19, los resultados empeoran. Además observando las matrices de confusión de la figura 3.19 se puede observar que ningún modelo tiene una buena matriz de confusión ya que o bien predomina a falsos negativos o a falsos positivos, aunque en menor medida que el modelo RobertaBIO. Al ser la configuración que utiliza Adam con regularización la que tiene un mejor equilibrio de falsos negativos y falsos positivos, será considerada la mejor.



(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.17: Curvas de perdida BETO.



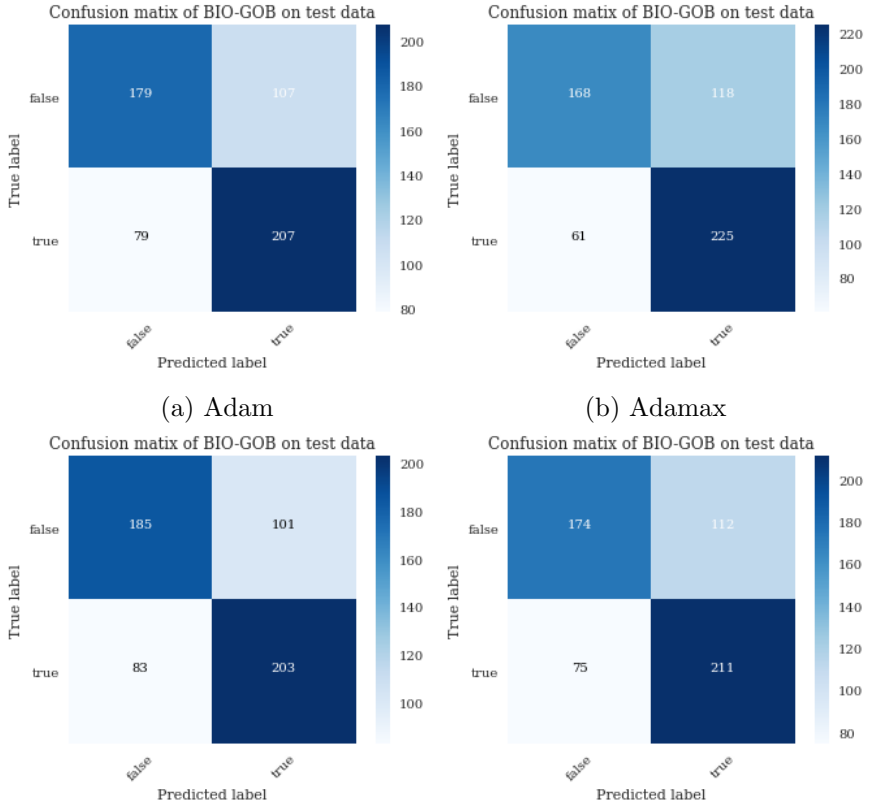
(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.18: Curvas de precisión BETO.

Modelo	**F1-Score**
BETO Adam	67.48
BETO Adam Regularization	**67.83**
BETO Adamax	67.83
BETO Adamax Regularization	67.83

Tabla 3.7: Modelos propuestos BETO.



(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.19: Matrices de confusión BETO.

Mejores modelos

Modelo base	Versión	Resultado
Roberta Gob Large	Adamax	71.85
Roberta Gob BIO	**Adamax**	**73.77**
BETO	Adam + Regularización	67.83

Tabla 3.8: Resultado tras los distintos *Fine-Tuning* de los modelos de IberLEF.

Finalmente, tras entrenar todos los modelos se observa los mejores resultados en la tabla 3.8. Como se puede observar, no hay una diferencia substancial entre los 4 modelos estudiados, el principal motivo se debe a que la gran mayoría del error que tienen estos modelos se debe a que a la hora de evaluar el conjunto de test fallan en las noticias relacionados con la temática COVID-19, independientemente de que el modelo sea dependiente o independiente del dominio. Esto es debido a que aunque el modelo dependiente del dominio "entienda" las palabras específicas de la temática, durante el *fine-tuning* no ha podido asociarlas a noticias falsas o verdaderas.

Aun así, el mejor modelo es el de RobertaBIO, el cual es dependiente del dominio y en su configuración con el optimizador Adamax obtiene un 73,77 de F1-score frente al 76,66 del modelo ganador de la competición. Esto es un 2,89 puntos de diferencia entre ambos modelos, aunque se ha de tener en cuenta que esta diferencia mínima se ha logrado usando un modelo considerablemente más sencillo. Atendiendo a la descripción del modelo ganador de la competición, este utiliza BERT junto "Sample memory" y mecanismos de atención. En concreto obtiene 2 *Embeddings* con el inicio y el final de la noticia lo cual introduce a su sistema de BERT. Posteriormente obtiene la llamada "Sample memory" la cual es una matriz obtenida a raíz de los inicios y finales de los *Embeddings* actuando como mecanismo de atención [18].

Por tanto los resultados son considerablemente buenos considerando que tan solo se ha logrado un modelo 2,89 puntos de F1-score peor que el mejor de la competición y utilizando un modelo más simple. En la tabla 3.9 se pueden observar los 10 mejores modelos de la competición entre los que se encuentra el propuesto en este trabajo en quinta posición.

Puesto	Usuario/Nombre del equipo	F1-score
1	**GDUFS_DM**	**0.7666**
2	haha	0.7548
3	ChaTs_	0.7514
4	SINAI	0.7385
5	**fIlfA**	**0.7377**
6	Lcad/UFES	0.7102
7	CiTIUS-NLP	0.7098
8	zk15120170770	0.7053
9	ForceNLP	0.6925
10	GRX	0.6915

Tabla 3.9: Mejores 10 resultados de la competición de IberLEF.

CONSTRAINT AAAI

A diferencia del conjunto de datos de IberLEF, el cual es multidominio, este conjunto de datos es de un dominio en concreto como es el COVID-19, la pandemia y todo los relacionado a estos conceptos. Por tanto como es de esperar, puede ser un factor diferencial el utilizar modelos dependientes del dominio los cuales proporcionen mejores resultados. En la tabla 3.10 se explican brevemente los 4 modelos seleccionados y sus características principales. A continuación se procede a mostrar para cada uno de los modelos el preprocesado que se ha utilizado junto a las pruebas correspondientes y sus resultados.

Modelo base	Dominio	Millones de Tweets	Observaciones
Sci-Bert	Dependiente	Mirar[13]	Modelo de lenguaje para texto científico desarrollado por el instituto de Allen de IA de Seattle [7].
Roberta-Twitter-Cardiffnlp - Marzo 2022	Independiente	128.06	Versión más reciente y entrenada con más datos del grupo de NLP de la universidad de Cardiff [26].
Vinai-Bertweet-covid-cased	Dependiente	850	Modelo creado en colaboración por VinAI, Oracle y NVIDIA. Versión *cased* para covid.[30]
DigitalEpidemology-V2	Dependiente	97	Modelo creado por DigitalEpidemologyLab en colaboración con FISABIO (Valencia) [29]. Utilizado para un *Ensemble* por el ganador de la competición [17].

Tabla 3.10: Modelos de BERT considerados para el fine-tuning del conjunto de datos de CONSTRAINT AAAI.

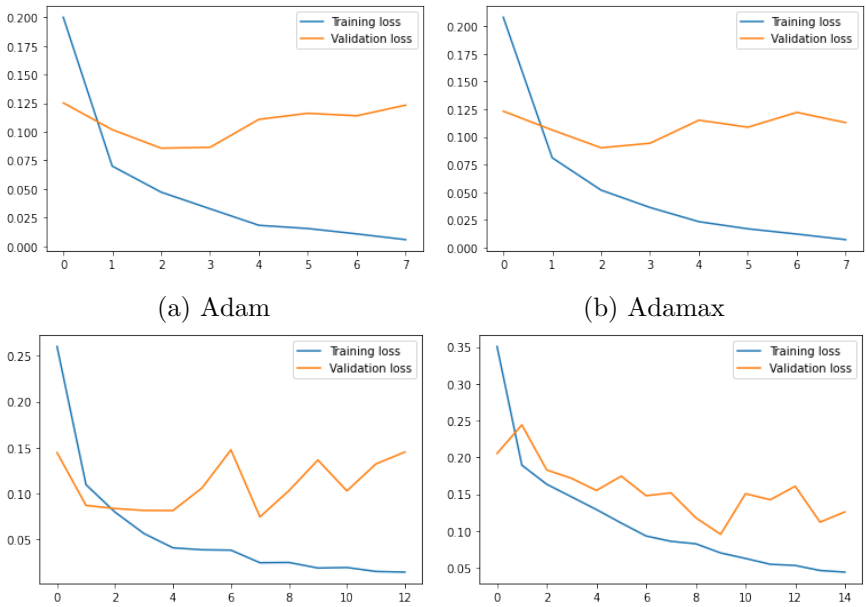
Vinai-Bertweet

Para este modelo, se ha utilizado como preprocesado el recomendado por los propios autores[14] [30]. Por tanto no se ha tenido que tomar ninguna decisión al respecto sobre como preprocesar el texto.

Como se puede observar en las curvas de aprendizaje y de pérdida de las figuras 3.20, 3.21 el proceso de aprendizaje de los 4 modelos es muy bueno y no conlleva a un sobreajuste significativo.

En la tabla 3.11 observamos que para el modelo Vinai-Bertweet, la mejor configuración es la que utiliza el optimizador Adam junto a la regularización en las capas.

Si bien todos los modelos obtienen buenos resultados, la versión que utiliza Adam junto a la regularización en las capas logra 0,33 puntos más de F1-score con respecto al segundo mejor modelo (Adamax), por lo que si bien no es una gran diferencia, es suficientemente apreciable. Esta diferencia se traduce en que el mejor modelo falla en 39 noticias de 2140 mientras que el segundo mejor falla en 46, tal y como se puede observar en las matrices de confusión de la figura 3.22.



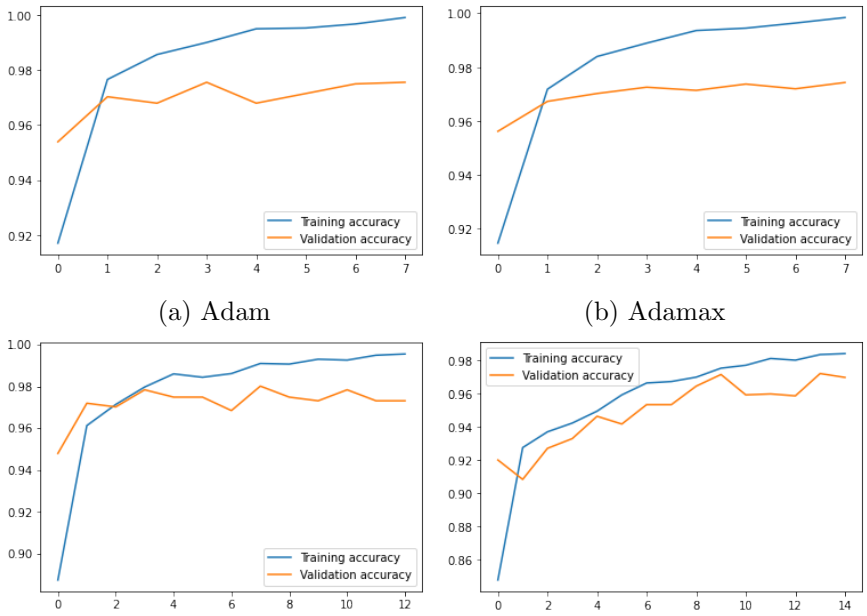
(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.20: Curvas de perdida Vinai-Bertweet.

[13]Este conjunto de datos no ha sido entrenado con *tweets* si no con artículos científicos. En concreto 1.14 millones de artículos de Semantic Scholar de carácter científico y más de 200.000 relacionados con la temática COVID-19.

[14]https://huggingface.co/vinai/bertweet-base



(a) Adam

(b) Adamax

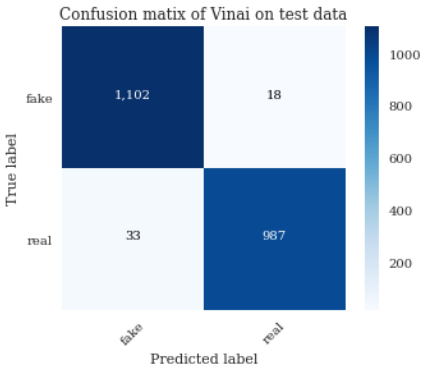
(c) Adam + Regularización

(d) Adamax + Regularización

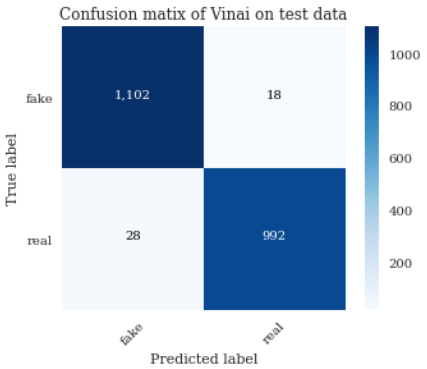
Figura 3.21: Curvas de precisión Vinai-Bertweet.

Modelo	**F1-Score**
Vinai-Bertweet Adam	97.62
Vinai-Bertweet Adam + Regularizacion	**98.18**
Vinai-Bertweet Adamax	97.85
Vinai-Bertweet Adamax + Regularizacion	97.05

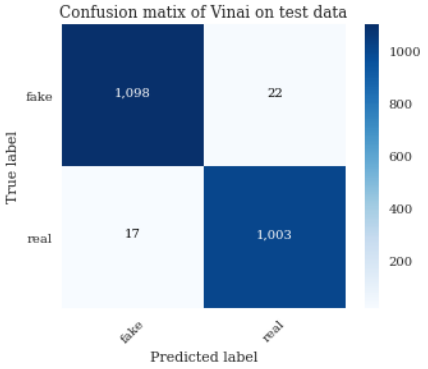
Tabla 3.11: Modelos propuestos Vinai-Bertweet.



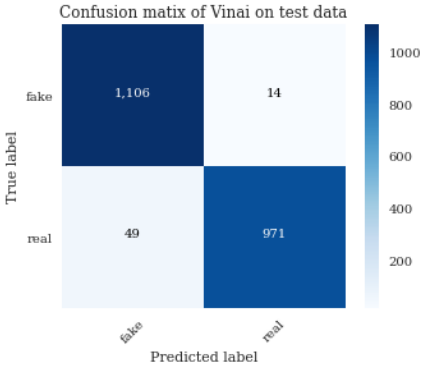
(a) Adam



(b) Adamax



(c) Adam + Regularización



(d) Adamax + Regularización

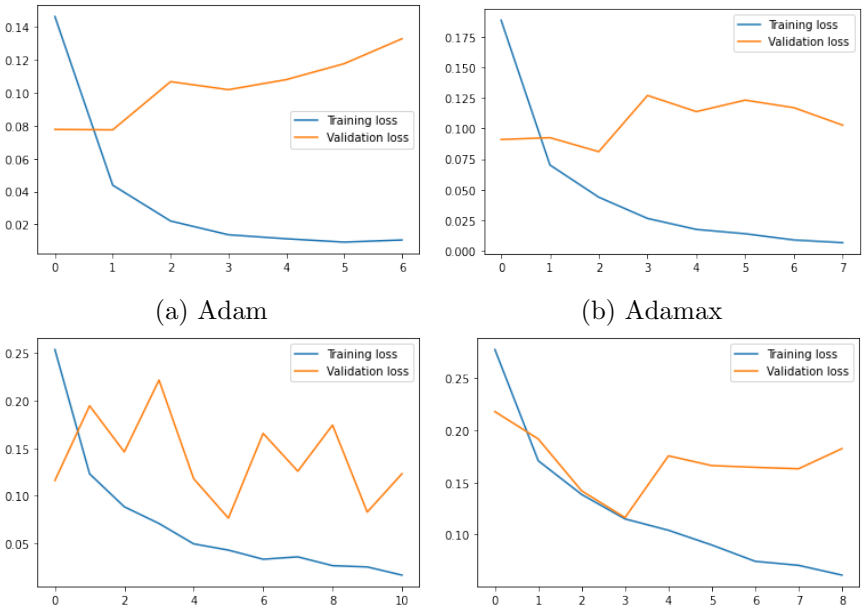
Figura 3.22: Matrices de confusión Vinai-Bertweet.

Roberta-Twitter-Cardiffnlp

Para este modelo se ha utilizado únicamente el preprocesado recomendado por los propios autores en su espacio de HuggingFace. Este preprocesado únicamente sustituye los nombres de usuario de Twitter por '@user' y las URL por 'http'.

Como se puede observar en las curvas de aprendizaje y de pérdida de las figuras 3.23, 3.24 el proceso de aprendizaje de los 4 modelos es bueno y no conlleva a un sobreajuste significativo.

En la tabla 3.12 observamos que para el modelo de Cardiff, la mejor configuración es la que utiliza el optimizador Adam únicamente. Si bien todos los modelos obtienen buenos resultados, la versión que utiliza Adam únicamente logra un 0,1 puntos más de F1-score con respecto al segundo mejor modelo (Adam junto a regularización en las capas), por lo que si bien la diferencia es mínima, es suficiente para observar que la configuración con Adam es mejor. Esta diferencia se traduce en que el mejor modelo falla en 51 noticias de 2140 mientras que el segundo mejor falla en 53, tal y como se puede observar en las matrices de confusión de la figura 3.25. Además cabe destacar, que tal y como se puede observar en las matrices de confusión, en este modelo hay una clara predominancia a falsos positivos.



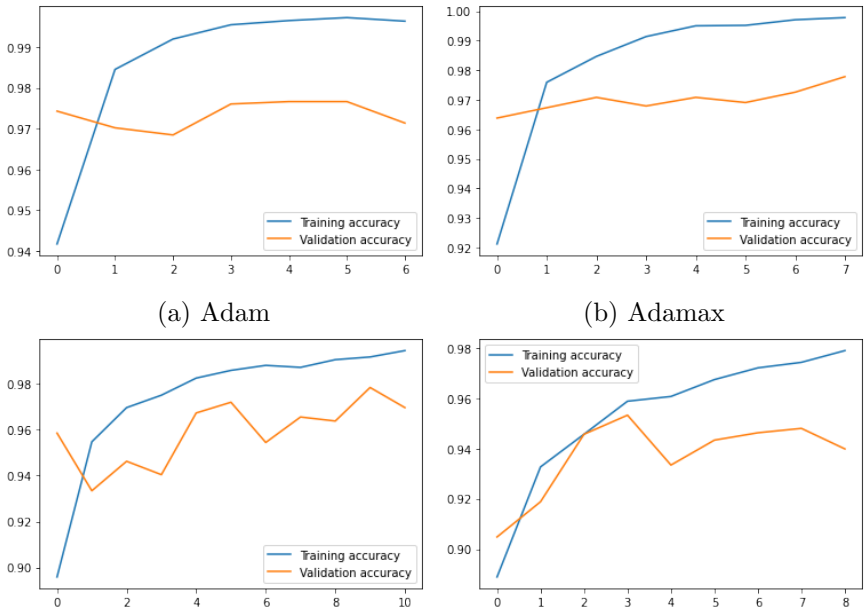
(a) Adam

(b) Adamax

(c) Adam + Regularización

(d) Adamax + Regularización

Figura 3.23: Curvas de perdida Cardiff.



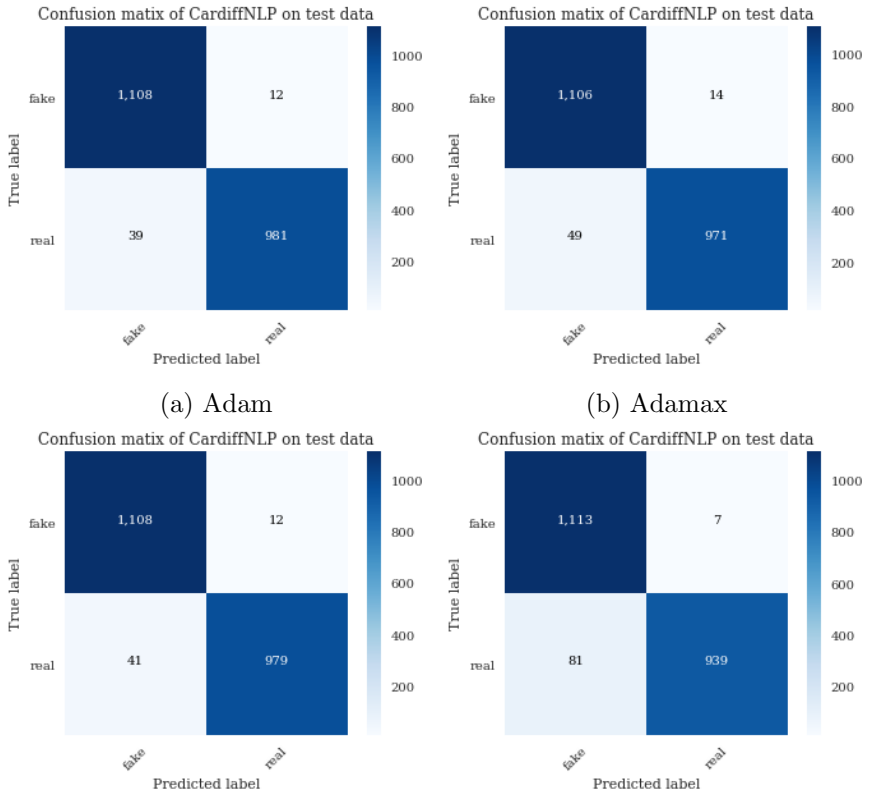
(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.24: Curvas de precisión Cardiff.

Modelo	**F1-Score**
Cardiff Adam	**97.62**
Cardiff Adam Regularización	97.52
Cardiff Adamax	97.05
Cardiff Adamax Regularización	95.89

Tabla 3.12: Modelos propuestos Cardiff.



(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

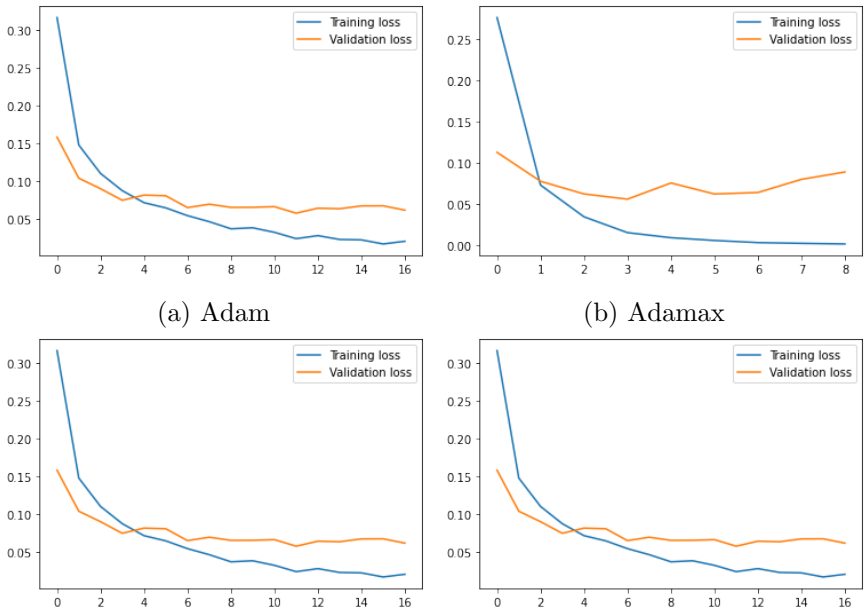
Figura 3.25: Matrices de confusión Cardiff.

Digital-Epidemology-V2

Como preprocesado para este modelo, se han pasado todas las palabras a minúsculas, se ha sustituido los URL por el token URL y los hashtag por el token $HASHTAG$.

Como se puede observar en las curvas de aprendizaje y de pérdida de las figuras 3.26, 3.27 el proceso de aprendizaje de las 4 configuraciones es bueno y no conlleva a un sobreajuste significativo.

En la tabla 3.13 observamos que para el modelo de Digital-Epidemology-V2, la mejor configuración es la que utiliza el optimizador Adamax únicamente. Si bien todos los modelos obtienen buenos resultados, la versión que utiliza Adamax únicamente logra un 0,09 puntos más de F1-score con respecto al segundo mejor modelo (Adam), por lo que si bien no es una gran diferencia, es suficientemente apreciable. Esta diferencia se traduce en que el mejor modelo falla en 37 noticias de 2140 mientras que el segundo mejor falla en 53, tal y como se puede observar en las matrices de confusión de la figura 3.28.



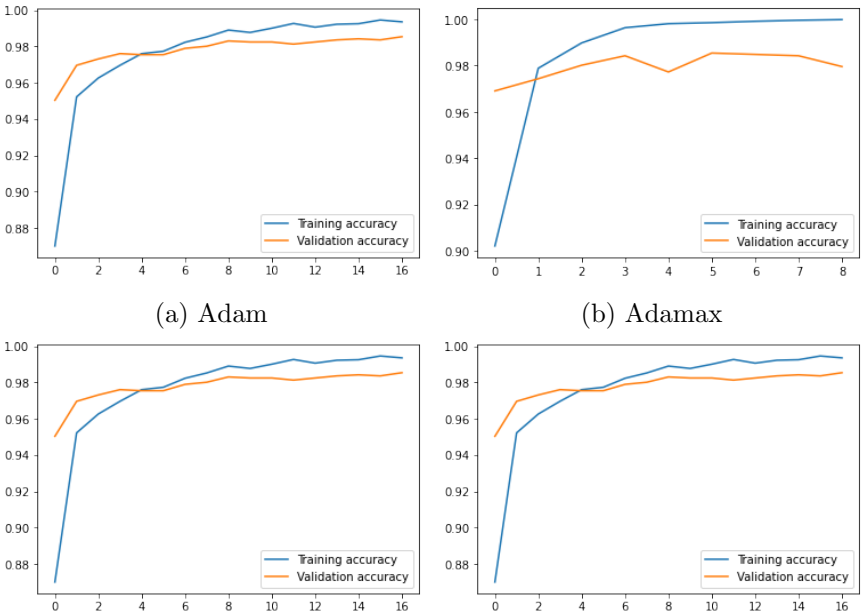
(a) Adam

(b) Adamax

(c) Adam + Regularización

(d) Adamax + Regularización

Figura 3.26: Curvas de perdida Digital-Epidemology-V2.



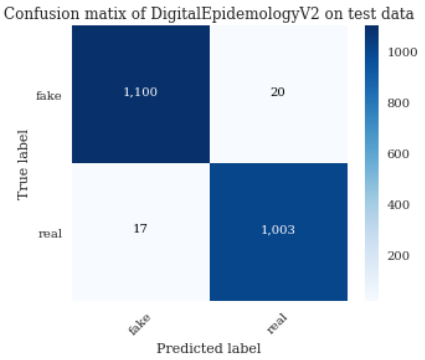
(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

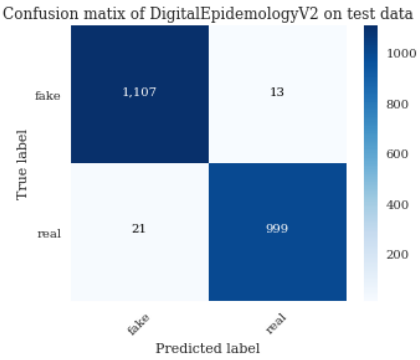
Figura 3.27: Curvas de precisión Digital-Epidemology-V2.

Modelo	**F1-Score**
V2 covid Adam	98.32
V2 covid Adam Regularización	97.52
V2 covid Adamax	**98.41**
V2 covid Adamax Regularización	98.27

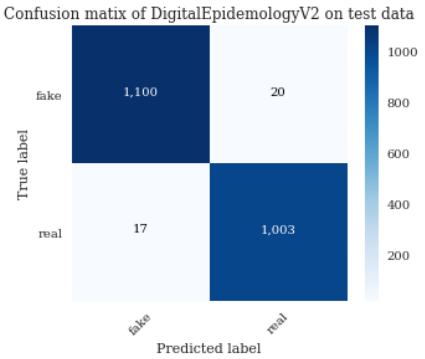
Tabla 3.13: Modelos propuestos Digital-Epidemology-V2.



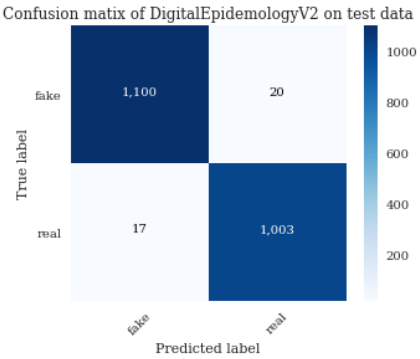
(a) Adam



(b) Adamax



(c) Adam + Regularización



(d) Adamax + Regularización

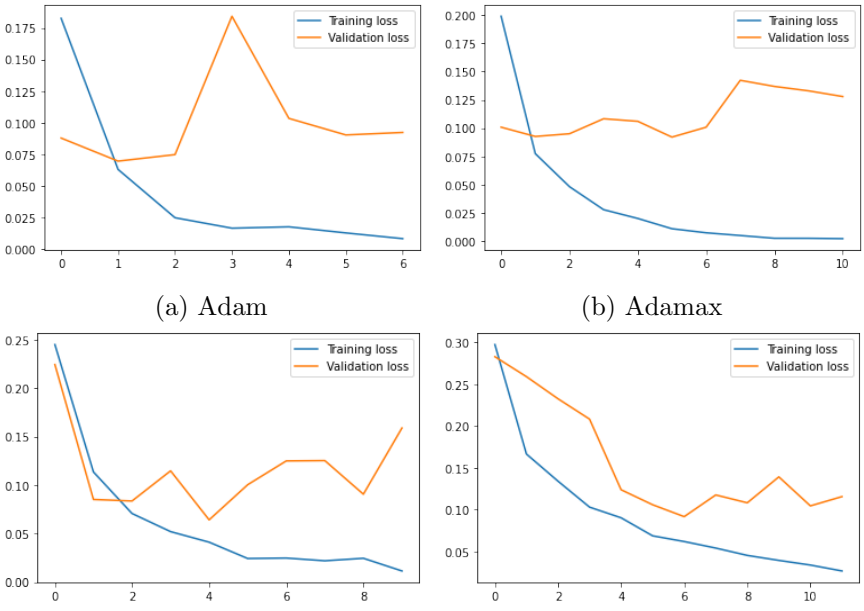
Figura 3.28: Matrices de confusión Digital-Epidemology-V2.

COVID-SciBert

Como preprocesado para este modelo, se han pasado todas las palabras a minúsculas ya que el modelo lo requiere al ser un modelo que trata todas las palabras como minúsculas y además se ha sustituido los URL por el token URL.

Como se puede observar en las curvas de aprendizaje y de pérdida de las figuras 3.29, 3.30 el proceso de aprendizaje de los 4 modelos es bueno y no conlleva a un sobreajuste significativo. En la tabla 3.14 observamos que para el modelo Vinai-Bertweet, la mejor configuración es la que utiliza el optimizador Adam junto al que utiliza el optimizador Adamax obteniendo ambas configuraciones 97.80 de F1-score.

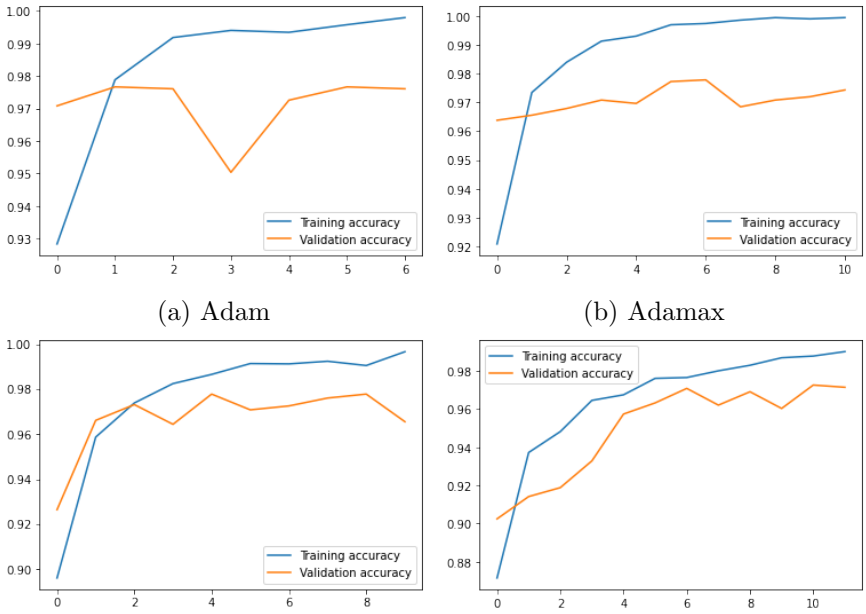
Si bien ambas configuraciones obtienen el mismo valor de F1-score como se puede observar en las matrices de confusión de la figura 3.31, el modelo que utiliza Adam tiene una mejor matriz de confusión que el que utiliza Adamax, es decir, no predomina entre falsos positivos o falsos negativos, por lo que se escoge la configuración que utiliza Adam como la mejor al ser más robusta.



(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.29: Curvas de perdida Sci-Bert.



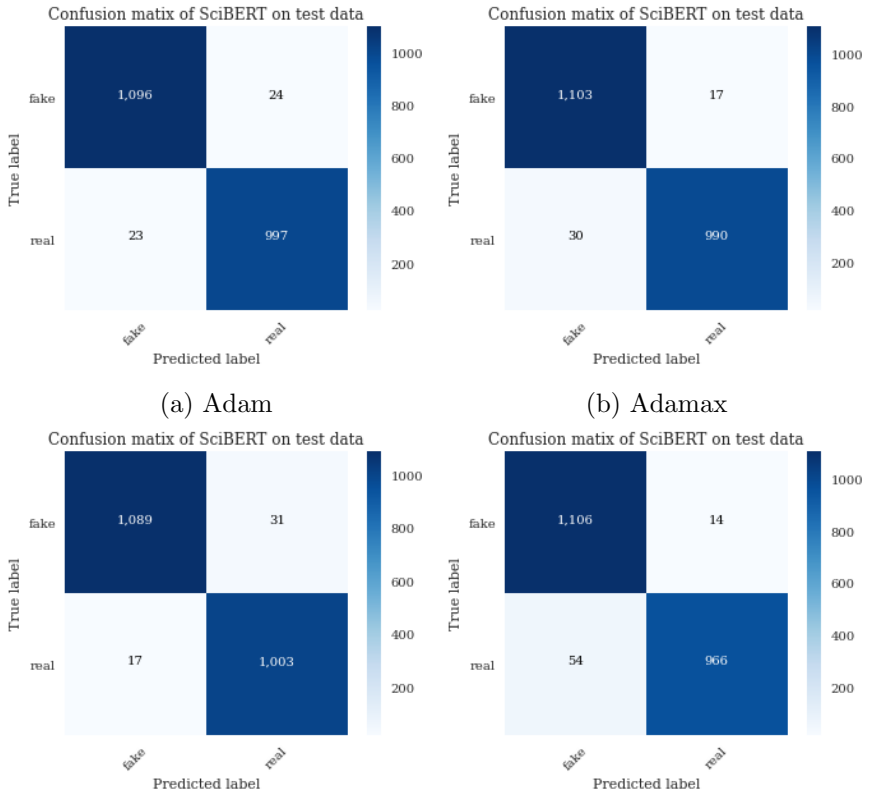
(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.30: Curvas de precisión Sci-Bert.

Modelo	**F1-Score**
Sci-Bert covid Adam	**97.80**
Sci-Bert covid Adam + Regularizaciónn	97.75
Sci-Bert covid Adamax	97.80
Sci-Bert covid Adamax + Regularización	96.82

Tabla 3.14: Modelos propuestos Sci-Bert.



(a) Adam (b) Adamax

(c) Adam + Regularización (d) Adamax + Regularización

Figura 3.31: Matrices de confusión Sci-Bert.

Mejores modelos

Modelo base	Versión	Resultado
Roberta-Twitter-Cardiffnlp	Adam	97.62
Sci-Bert	Adam	97.80
Vinai-BERTweet-covid-cased	Adam + Regularización	98.18
DigitalEpidemology-V2	**Adamax**	**98.41**

Tabla 3.15: Resultado tras los distintos *Fine-Tuning* de los modelos de CONSTRAINT AAAI.

En la tabla 3.15 se pueden observar las mejores versiones de los *fine-tuning* realizados. El modelo que mejor resultados nos aporta es el de *Digital Epidemology V2*, es decir el mismo modelo que utiliza el equipo ganador de la competición donde se presenta el conjunto de datos[32, 31].

Aun así, como se puede observar tanto en los resultados como en las matrices de confusión, todos los modelos logran una muy buena resolución del problema, fallando únicamente en unas 40-60 noticias (de 2140) cada uno de estos modelos como máximo. Además estos modelos por lo general tienen un buen equilibrio entre falsos negativos y falsos positivos.

Por otro lado estos resultados presentados se logran con un modelo más sencillo que el presentado por el equipo ganador de la competación: "g2tmn"[17]. El equipo presenta un modelo basado en un *Ensemble* de 3 *Fine-Tuning* de este modelo de *DigitalEpidemology-V2*, logrando un F1-score de $98,69$, lo que supone menos de un $0,28$ puntos de mejora con el mejor modelado presentado en este trabajo. Si se compara el número de noticias clasificadas incorrectamente, el modelo ganador de la competición falla en 28 noticias, mientras que el mejor modelo presentado en este trabajo falla en 47 noticias, es decir, una diferencia de 19 noticias teniendo en cuenta que se evalúan 2140 noticias en total.

Por tanto los resultados son muy buenos considerando que tan solo se ha logrado un modelo 2,89 puntos de F1-score peor que el mejor de la competición y utilizando un modelo más simple. En la tabla 3.16 se pueden observar los 10 mejores modelos de la competición entre los que se encuentra el propuesto en este trabajo en sexta posición (nótese que hay 2 empates por encima de este modelo).

Puesto	Usuario/Nombre del equipo	F1-score
1	**g2tmn**	**98.69**
2	saradhix	98.65
3	xiangyangli	98.60
4	Ferryman	98.55
4	gundapusunil	98.55
5	DarrenPeng	98.46
5	maxaforest	98.46
6	**fIlfA**	**98.41**
7	dyh930610	98.36
8	abhishek17276	98.32

Tabla 3.16: Mejores 10 resultados de la competición de CONSTRAINT AAAI.

3.4. Conclusiones

A lo largo de esta parte del trabajo se han trabajado con distintas versiones de BERT y RoBERTa, modelos de lenguaje los cuales suponen el estado del arte para el problema de clasificación de texto con PLN. Para ambos conjuntos de datos se ha realizado un batería de 4 experimentos por cada modelo probando distintas configuraciones de optimizadores y regularización en capas de atención y *dropout*.

Para el conjunto de datos en Español, en total se han realizado 12 experimentos, es decir, los 4 experimentos para los 3 modelos analizados, RobertaGOB, RobertaBIO y BETO. Entre estos 12 experimentos se ha obtenido el mejor resultado con RobertaBIO utilizando el optimizador Adamax únicamente. Este modelo dependiente del dominio con esta configuración obtiene un F1-score de 73,77, lo que supone 2,89 puntos de diferencia con el mejor modelo ganador de la competición. Aun así como ya se ha explicado anteriormente el mejor modelo de la competición logra este resultado con un modelo considerablemente más complejo. Además comparando con el resto de competidores, el modelo presentado en este trabajo obtiene una quinta posición en la competición.

Por otro lado para el conjunto de datos en Español, en total se han realizado 16 experimentos, es decir, los 4 experimentos para los 4 modelos analizados, Sci-Bert, Roberta-Twitter-Cardiffnlp-Marzo2022, Vinai-Bertweet y DigitalEpidemology-V2. Entre estos 16 experimentos se ha obtenido el mejor resultado con DigitalEpidemology-V2 utilizando el optimizador Adamax únicamente. Este modelo dependiente del dominio con esta configuración obtiene un F1-score de 98,41, lo que supone 0,28 puntos de diferencia con el mejor modelo ganador de la competición. Aun así como ya se ha explicado anteriormente el mejor modelo de la competición logra este resultado con un

modelo considerablemente más complejo, en concreto, utiliza un triple *Ensemble* de 3 modelos de DigitalEpidemology-V2. Además comparando con el resto de competidores, el modelo presentado en este trabajo obtiene una sexta posición en la competición.

Parte II

Desarrollo de aplicación web demostrativa

Capítulo 4

Análisis

Si bien el principal bloque de trabajo de este TFG es la resolución del problema de detección de *fake news* utilizando modelos que suponen el estado del arte para el PLN, la creación de una interfaz cómoda para que el usuario pueda comprobar la veracidad de las noticias s otro bloque importante de este trabajo.

Por tanto para mostrar el uso y funcionamiento de los modelos, y además presentar una aproximación a un sistema de detección de *fake news*, se ha decidido desarrollar una aplicación web para la detección de *fake news*.

A lo largo de este capítulo realizará el análisis del sistema, desde la declaración de los requisitos funcionales, los diagramas de casos de uso junto a la descripción de los actores y casos de uso involucrados y finalmente los diagramas de secuencia.

En primer lugar en la Sección 4.1 se presentan los requisitos funcionales que ha de tener la interfaz web. Posteriormente en la Sección 4.2 se mostrarán los casos de uso junto a sus descripciones y las de los actores y los correspondientes diagramas de secuencia de estos casos de uso.

4.1. Requisitos funcionales

Con la descripción de los siguientes requisitos funcionales se define el funcionamiento del sistema, es decir, que debe de ser capaz de realizar el sistema. Por tanto los requisitos funcionales que debe contar el sistema una vez terminado son los siguientes:

1. Predecir si una noticia es *fake news* o no dada una entrada de texto.

2. Descargar modelo de lenguaje utilizado.

4.2. Diagrama de casos de uso

En primer lugar, en la figura 4.1 se presenta un diagrama de casos de uso, con los distintas funcionalidades que tendrá la interfaz web. Como se puede observar hay 2 actores, un "Usuario común" y un "Científico de datos" el cual como se puede observar por la flecha discontinua, es un caso específico de "Usuario común". Por otro lado tan solo hay 2 casos de uso los cuales realizan los 2 requisitos funcionales del sistema, "Predecir noticia" y "Descargar modelo".

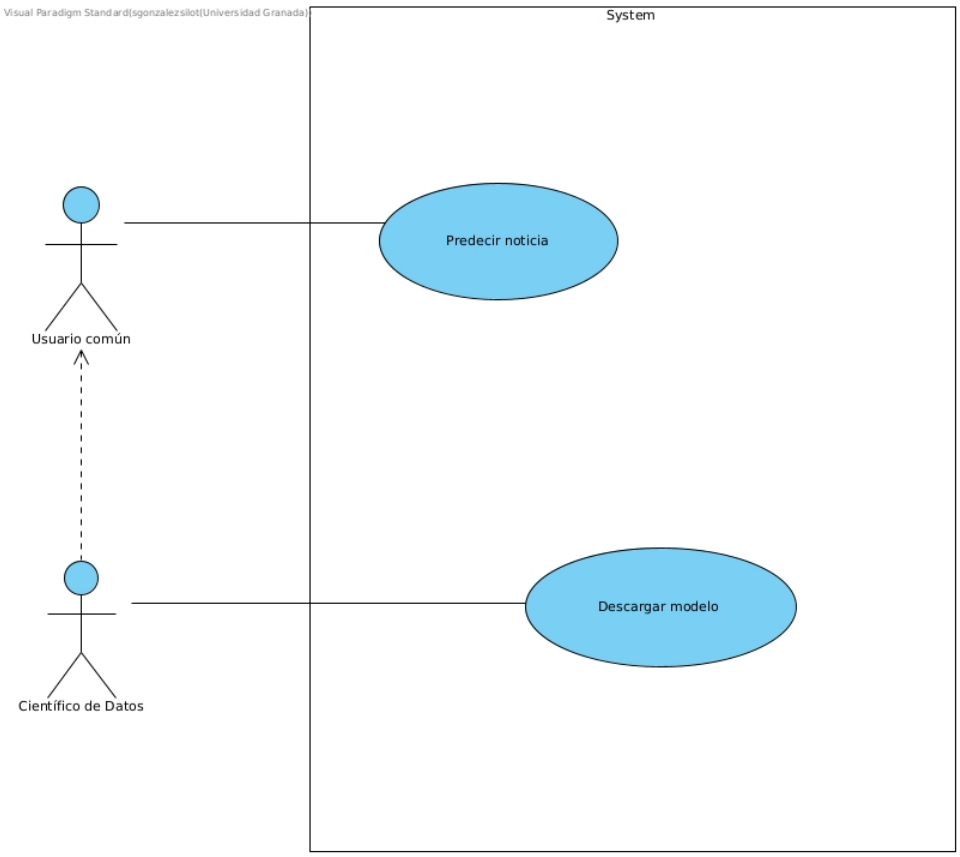


Figura 4.1: Diagrama de casos de uso.

4.2.1. Descripción de los actores

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

Actor	Usuario común				ACT-1
Descripción	Usuario común con habilidades básicas de informática y redes sociales.				
Características	Tiene un conocimiento básico de informática. Tiene un buen manejo de aplicaciones web como usuario. No tiene conocimientos específicos de Inteligencia Artificial.				
Relaciones					
Referencias	Predecir noticia				
Autor	Santiago González Silot	**Fecha**	24/03/22	**Versión**	1.0

Tabla 4.1: Descripción del actor "Usuario común".

Actor	Científico de datos				ACT-2
Descripción	Usuario con habilidades avanzadas de informática, ciencia de datos y PLN.				
Características	Tiene un conocimiento avanzado de informática, ciencia de datos y PLN. Es capaz de elegir que modelo es más adecuado para su uso. Puede realizar ciertas acciones exclusivas.				
Relaciones	Es un caso específico de Usuario común.				
Referencias	Predecir noticia, Descargar modelo				
Autor	Santiago González Silot	**Fecha**	24/03/22	**Versión**	1.0

Tabla 4.2: Descripción del actor "Científico de datos".

4.2.2. Descripción de los casos de uso

A continuación se muestra la narrativa de los casos de usos 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	Descargar modelo				CU-1
Actores	Científico de Datos				
Tipo	Primario				
Referencias					
Precondición	Tener suficiente espacio de almacenamiento en el ordenador personal				
Postcondición	El usuario obtiene el archivo del modelo que desea descargar.				
Autor	Santiago González Silot	**Fecha**	24/03/22	**Versión**	1.0

Propósito
Descargar un modelo entre los disponibles.

Resumen
Científico de datos solicita el modelo en uso para su posterior descarga.

Curso Normal			
1	Científico de datos: Solicita la descarga del modelo.		
		2	Proporciona el acceso a la descarga.
3	Científico de datos: Descarga y guarda el archivo en su ordenador.		

Otros datos			
Frecuencia esperada	Alta	**Rendimiento**	Menos de 1 minuto
Importancia	Alta	**Urgencia**	Moderada
Estado		**Estabilidad**	Moderada

Tabla 4.3: Descripción del CU-1 Descargar modelo.

Caso de Uso	Predecir noticia				CU-2
Actores	Usuario común				
Tipo	Primario				
Referencias					
Precondición					
Postcondición	El usuario obtiene la predicción de la noticia introducida.				
Autor	Santiago González Silot	**Fecha**	24/03/22	**Versión**	1.0

Propósito
Predecir la veracidad de una noticia.

Resumen
Usuario introduce una noticia y recibe el resultado de su veracidad.

Curso Normal			
1	Usuario común: Solicita la predicción de la noticia		
		2	Muestra las opciones para la predicción
3	Usuario común: Elige la opción que desea		
		4	Muestra la casilla para introducir el texto
5	Usuario común: Introduce el texto		
		6	Calcula la predicción y devuelve el resultado

Otros datos			
Frecuencia esperada	Alta	**Rendimiento**	Menos de 1 minuto
Importancia	Alta	**Urgencia**	Moderada
Estado		**Estabilidad**	Moderada

Tabla 4.4: Descripción del CU-2 Predecir noticia.

4.2.3. Diagramas de secuencia del sistema

Finalmente para mostrar aun mayor detalle sobre los casos de uso, se muestran a continuación los diagramas de secuencia de los casos de uso correspondientes. En estos diagramas de secuencia se muestran las líneas de vida de los procesos y los mensajes intercambiados entre ellos.

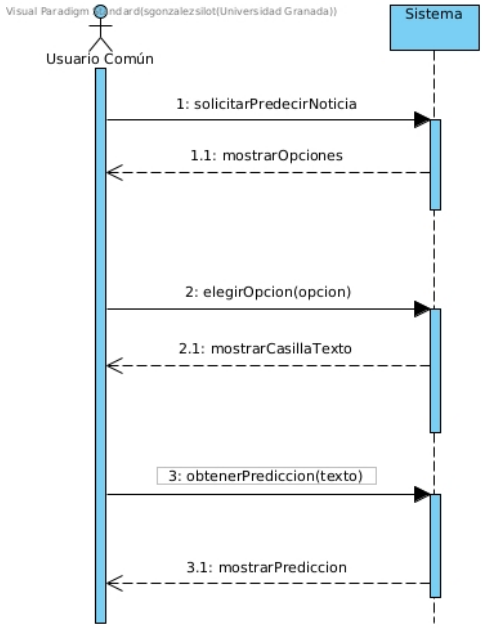


Figura 4.2: DSS-Predecir noticia.

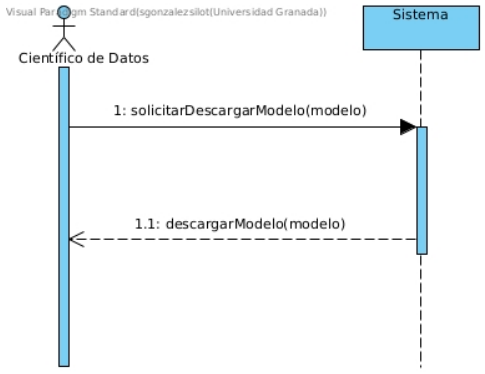


Figura 4.3: DSS-Descargar Modelo.

Capítulo 5

Diseño

Una vez realizado todo el análisis necesario para el desarrollo de la aplicación web, el siguiente paso en un proceso de ingeniería del software es el de diseñar la arquitectura, componentes software e interfaz de la aplicación. Es un proceso clave para realizar un sistema de calidad y con una buena escalabilidad para posteriores mejoras y adiciones al sistema.

Por tanto en la Sección 5.1 se muestra el diseño de la arquitectura software junto a las componentes software. Finalmente en la Sección 5.2 se realiza un boceto inicial de la interfaz para posteriormente mostrar la interfaz final.

5.1. Diseño de la arquitectura

Durante el diseño de la arquitectura del sistema, se han analizado 2 posibilidades principales para la implementación de la aplicación web de detección de *fake news*. Estas 2 posibilidades se analizan a continuación.

1. La primera opción es el desarrollo de un sistema con la capacidad de se ser re-entrenado y utilizarlo en producción.
2. La segunda opción es la utilización del sistema cargándose en memoria tras una serialización previa.

Si bien la primera opción da mayor funcionalidad y permite que el sistema pueda ser entrenado, la segunda opción tiene una implementación más sencilla y simple dadas las condiciones de tiempo y capacidad de computación disponibles durante el desarrollo de este trabajo. Además, atendiendo al objetivo del trabajo, la segunda opción lo cumple en su totalidad, ya que se busca desarrollar un sistema para la detección de *fake news*, es decir, que el sistema de la opción de predecir la veracidad de la noticiad y esto la

segunda opción lo cumple totalmente. Cabe destacar que si el usuario desease entrenar el modelo tiene la posibilidad de descargarlo desde la propia aplicación web para posteriormente entrenarlo.

5.1.1. Diseño de las componentes software

A continuación se procede a mostrar el diagrama de clases de la aplicación web junto a la explicación y justificación de las distintas decisiones tomadas.

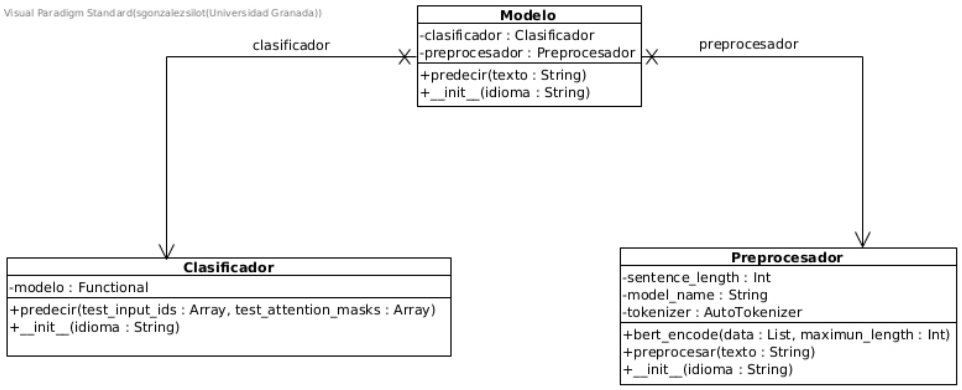


Figura 5.1: Diagrama de clases del sistema.

Para esta aplicación se ha decidido contar con 3 clases software las cuales se describen a continuación:

- **Preprocesador**: Esta clase software se encarga de realizar el preprocesado necesario dado el modelo de BERT que se esté utilizando. Además también realiza la tokenización y truncamiento necesario para BERT. Para ello utiliza la API de HuggingFace para obtener la última versión disponible del tokenizador del modelo base.

- **Clasificador**: Esta clase software se encarga de cargar en memoria el modelo de BERT dado un archivo serializado según el modelo que se vaya a utilizar. Además también incluye la funcionalidad de predecir la noticia dada una entrada correctamente preprocesada.

- **Modelo**: Esta clase software es la principal del sistema, consiste de 2 atributos principales los cuales son un objeto de la clase Preprocesador y de la clase Clasificador. Todas las llamadas que realiza la aplicación web son a través de esta clase. Se basa en un constructor que inicializa el Preprocesador y el Clasificador pasando como argumento si se va a predecir una noticia en español o inglés y de una función para predecir que se encarga de preprocesar el texto utilizando el Preprocesador

y de predecir utilizando el Clasificador. Finalmente devuelve como resultado un dato de tipo *string* según la veracidad de la noticia.

Por tanto con esta arquitectura y componentes software, se tiene un sistema que cumple totalmente con los objetivos y además, dado el diseño orientado a objetos utilizado durante su desarrollo, este sistema está pensado para la futura incorporación de distintos modelos de BERT o RoBERTa u otros modelos del lenguaje. Además también sería posible agregar nuevos idiomas y preprocesados para el texto. Por tanto se ha logrado desarrollar un sistema con una gran escalabilidad.

5.1.2. Diagramas de secuencia del sistema

Si bien en el capítulo anterior se definieron dos diagramas de secuencia los cuales corresponden a los dos casos de uso del sistema, en este capítulo se muestran la versión de dichos diagramas para la fase de diseño, visualizándose la vida de los objetos involucrados en cada caso de uso. A continuación en la figura 5.2 se puede observar el diagrama de secuencia de "Predecir noticia" y en la figura 5.3 el diagrama de secuencia de "Descargar modelo". Como se puede observar, la implementación es más clara y simplificado gracias al uso de las distintas clases software explicadas anteriormente.

Caso de uso Predecir noticia

Para este diagrama de secuencia como se puede observar en la figura 5.2, el usuario elige si va a predecir noticias según las distintas opciones, es decir, según el idioma. Posteriormente introduce en la vista el texto a predecir. La vista solicita la predicción al controlador, el cual a su vez se lo solicita al modelo. El modelo en primer lugar preprocesa el texto con una llamada al preprocesador. Una vez preprocesado introduce el input al clasificador el cual devuelve la predicción numérica. Finalmente el modelo transforma esta predicción numérica a un *string* y se la devuelve al controlador y finalmente aparece en la vista en un recuadro de texto.

Caso de uso Descargar modelo

Este diagrama de secuencia a diferencia del anterior es considerablemente más sencillo ya que tan solo actúan la vista y el controlador. Como se puede observar en la figura 5.3, el usuario solicita la descarga del modelo a la vista, la vista solicita el modelo al controlador el cual devuelve la ruta de descarga. Finalmente el archivo comienza a descargarse en el navegador web en el que se esté ejecutando la vista.

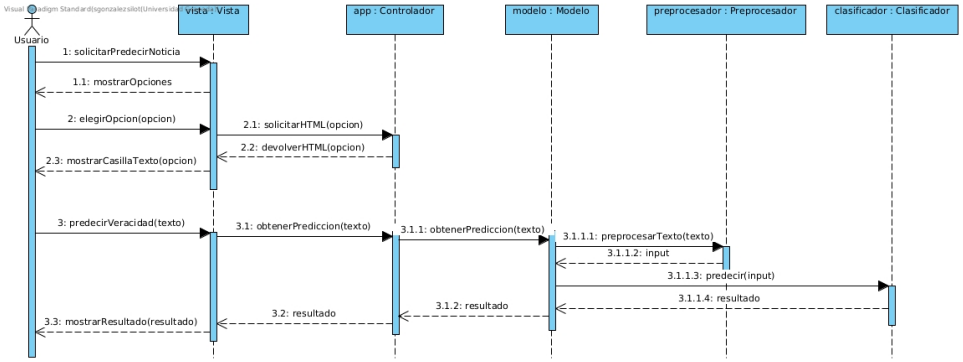


Figura 5.2: DSS de diseño - Predecir noticia.

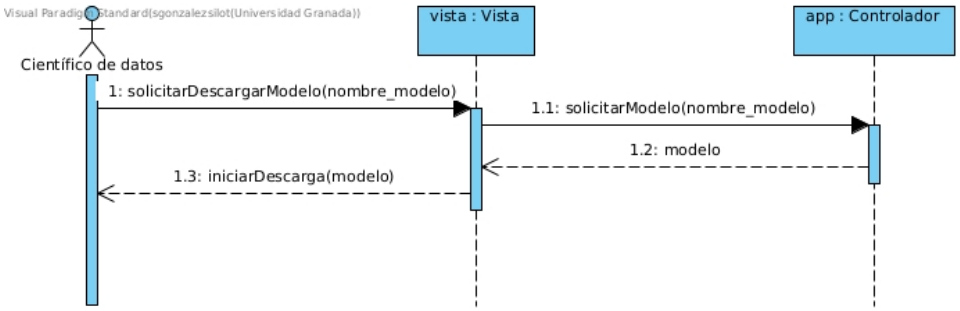


Figura 5.3: DSS de diseño - Descargar modelo.

5.2. Diseño de la interfaz

En primer lugar como se puede observar en la figura 5.4, el objetivo de esta interfaz web es que sea limpia, clara, fácil de entender y minimalista. Por ello desde la fase de diseño se ha intentado eliminar botones y opciones innecesarias las cuales no aportasen una mayor calidad a la aplicación. Cabe destacar que la elección del color morado para alguna de las letras se debe a que es la combinación del color azul y rojo, colores los cuales suelen asociarse con verdadero y falso, respectivamente. Finalmente el resultado tras la implementación es el que se puede observar en la figura 5.5

Además como se puede observar en la figura 5.6 se ha agregado una página principal la cual muestra 3 opciones para elegir, las cuales son las 2 versiones de detección de *fake news* tanto para inglés como para español y un enlace al Github personal. Además como se puede apreciar en las figuras 5.7 los enlaces responden al posar el ratón en el enlace.

Figura 5.4: Boceto de la página principal.

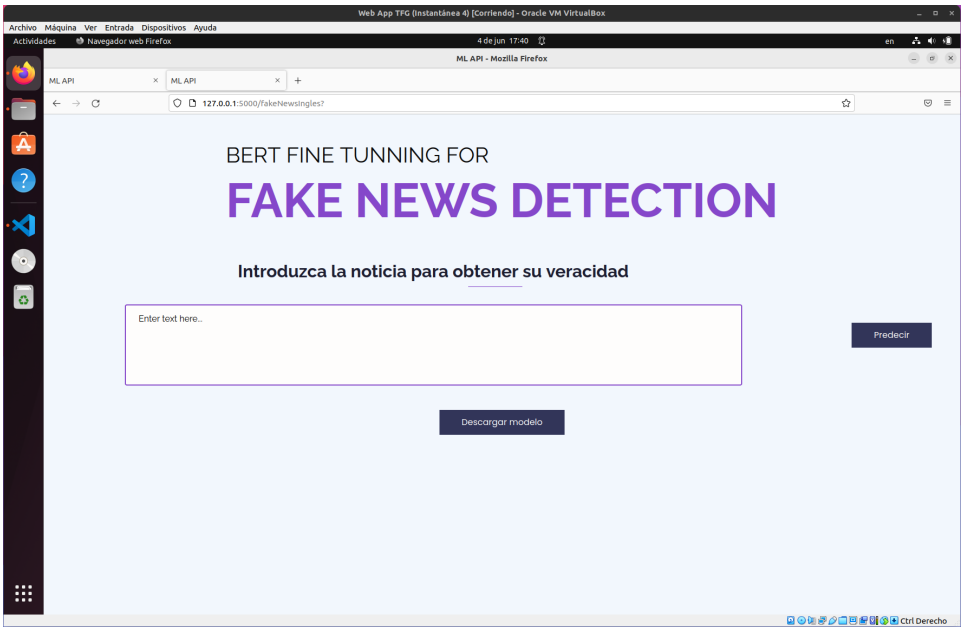


Figura 5.5: Resultado final de página principal.

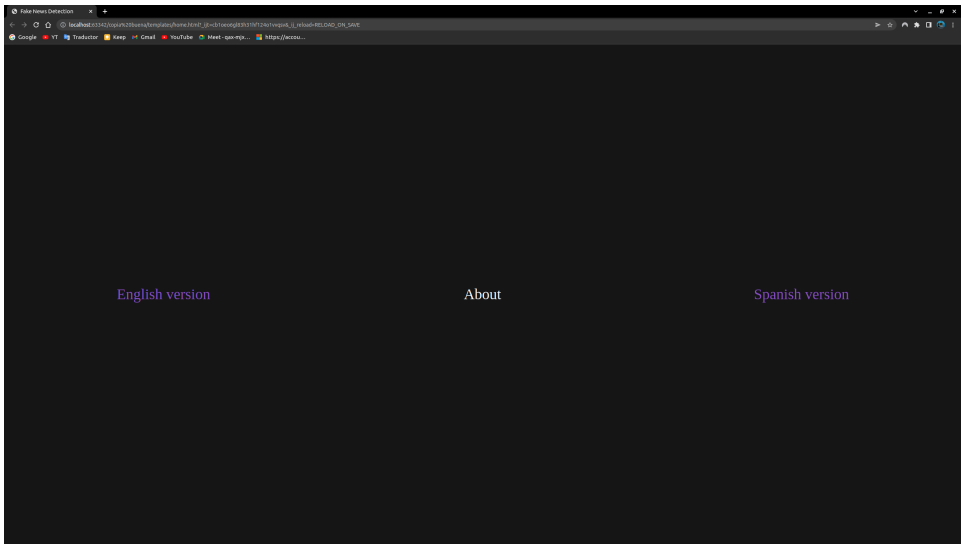
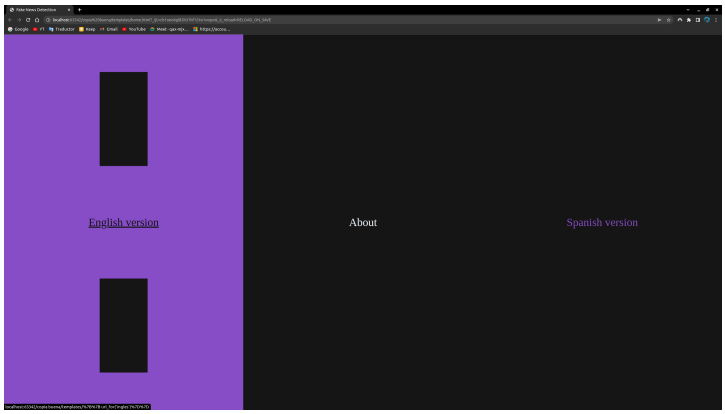
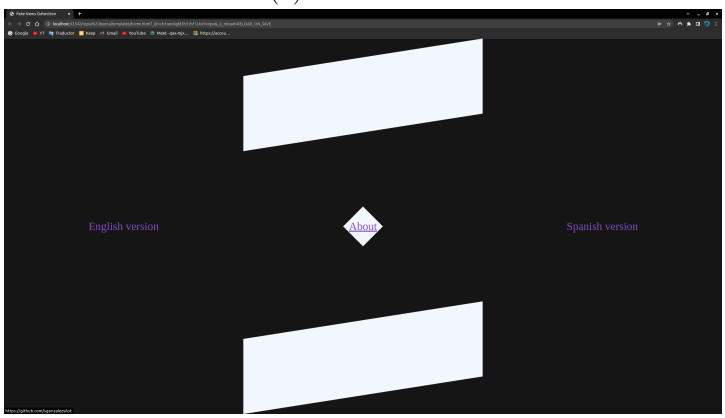


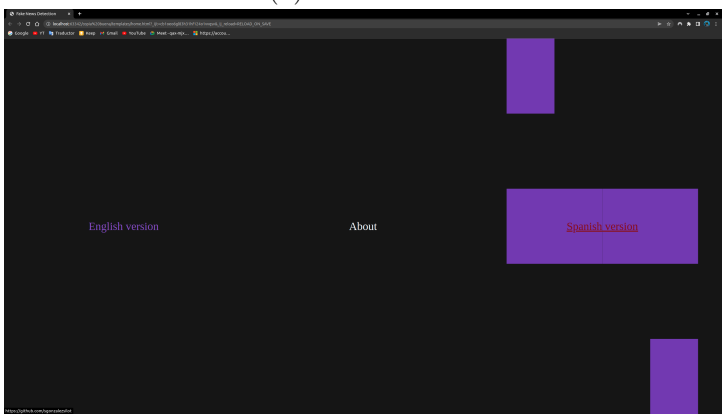
Figura 5.6: Resultado final de página inicial.



(a) Activación 1



(b) Activación 2



(c) Activación 3

Figura 5.7: Captura de las distintas activaciones de los enlaces de la página inicial.

Capítulo 6

Implementación y Pruebas

Una vez analizado el problema de detección de *fake news* y decidido el diseño de sistema, se procede a explicar la implementación realizada del sistema en sí.

Además para comprobar el correcto funcionamiento del sistema de detección de *fake news* se van a desarrollar distintas pruebas típicas utilizadas en procesos de ingeniería del software. Con esto se dará por finalizado todo el trabajo relacionado con el sistema de detección de *fake news*.

Por tanto en la Sección 6.1 se explicará como se ha implementado el sistema. Finalmente en la Sección 6.2 se explicarán las pruebas realizadas para comprobar el correcto funcionamiento del sistema.

6.1. Implementación

Para la realización de la aplicación web se ha optado por un patrón Modelo-Vista-Controlador, utilizando Python con Flask[1] para el apartado del modelo y controlador y HTML con CSS para el apartado de la vista. Se ha decidido utilizar esta arquitectura y estos lenguajes ya que es una combinación típica a la hora de implementar aplicaciones web para modelos de *Machine Learning* de forma cómoda y sencilla atendiendo a la necesidad básica de la aplicación de solicitar al modelo de *Machine Learning* que haga una predicción dada una entrada.

Ya que se ha optado por el diseño orientado a objetos explicado en el capítulo de diseño, se ha decidido separar cada clase en un archivo distinto con el mismo nombre de la clase y un archivo principal llamado "app.py" el cual corresponde al controlador de la aplicación web, el cual está gestionado con Flask. Además se han introducido todos lo archivos HTML en

[1] https://flask.palletsprojects.com/en/2.1.x/

su correspondiente carpeta “templates”, los CSS en su carpeta “static” y finalmente los modelos serializados se han guardado en la carpeta “models”. Esta estructura de ficheros se puede observar en la figura 6.1.

Toda la aplicación web junto al *notebook* de Python utilizado para el desarrollo de los modelos se encuentra disponible en el siguiente enlace de Github.

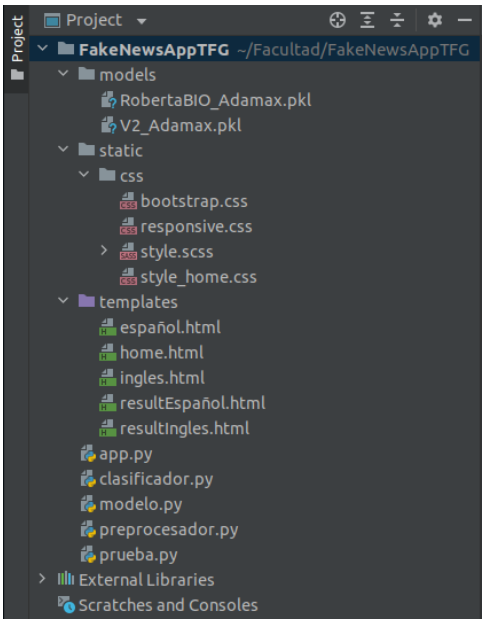


Figura 6.1: Estructura de ficheros.

6.2. Pruebas

Para comprobar el correcto funcionamiento del sistema se realizará un conjunto de pruebas de caja negra las cuales comprobarán las distintas entradas que puede tener el sistema y la idoneidad de la salida que este ofrece. En concreto se realizarán tres pruebas de caja negra para el requisito funcional “predecir noticia” y una prueba de caja negra para el requisito funcional “descargar modelo”.

6.2.1. Predecir noticia

Prueba 1. Introducción de una 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. Como se puede observar en la figura 6.2 el sistema avisa correctamente al usuario frente a una entrada vacía.

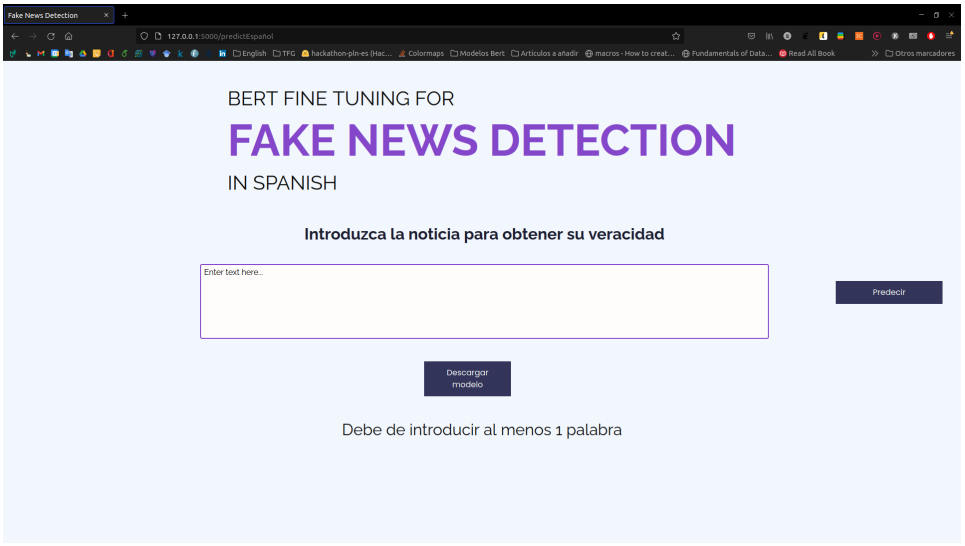
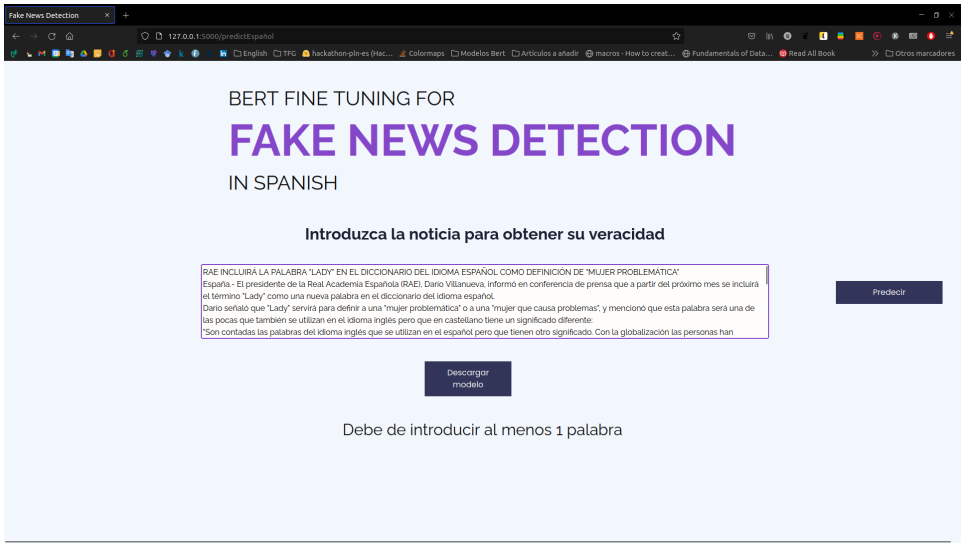


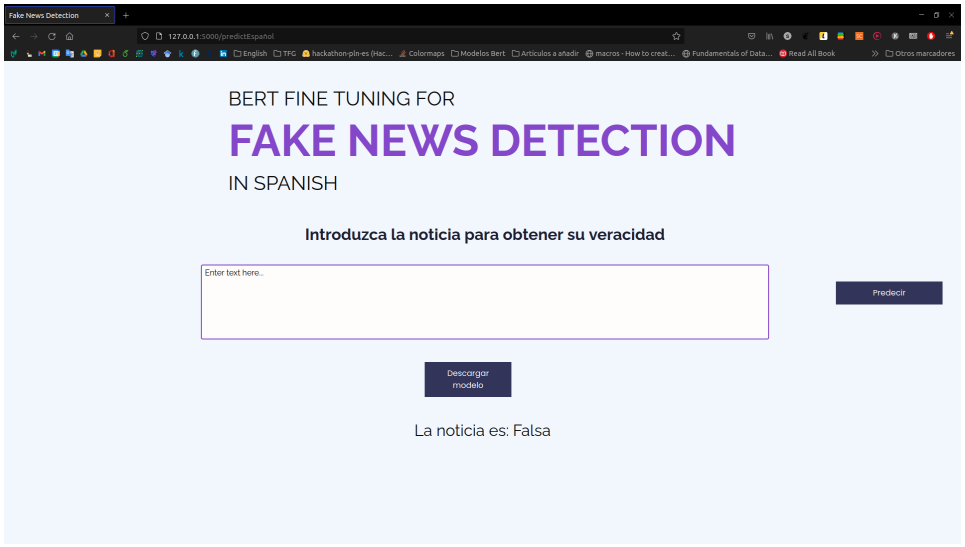
Figura 6.2: Prueba del requisito "Predecir noticia" - Prueba 1.

Prueba 2. Introducción de una noticia falsa.

La segunda prueba de caja negra para este requisito funcional consistirá en introducir una noticia falsa para comprobar si el sistema predice correctamente la noticia y notifica al usuario correctamente. Como se puede observar en la figura 6.3 el sistema notifica correctamente al usuario que la noticia es falsa.



(a) Antes de introducir la noticia.

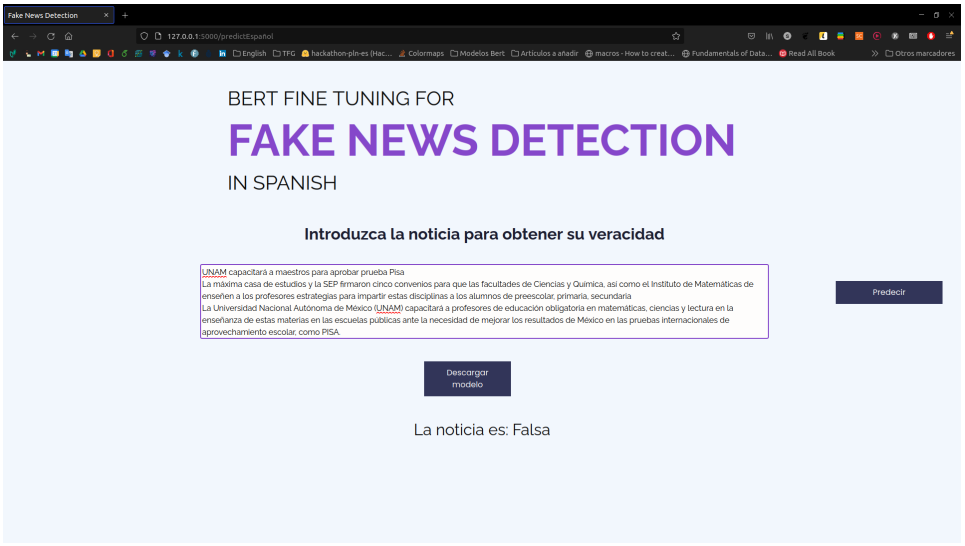


(b) Después de introducir la noticia.

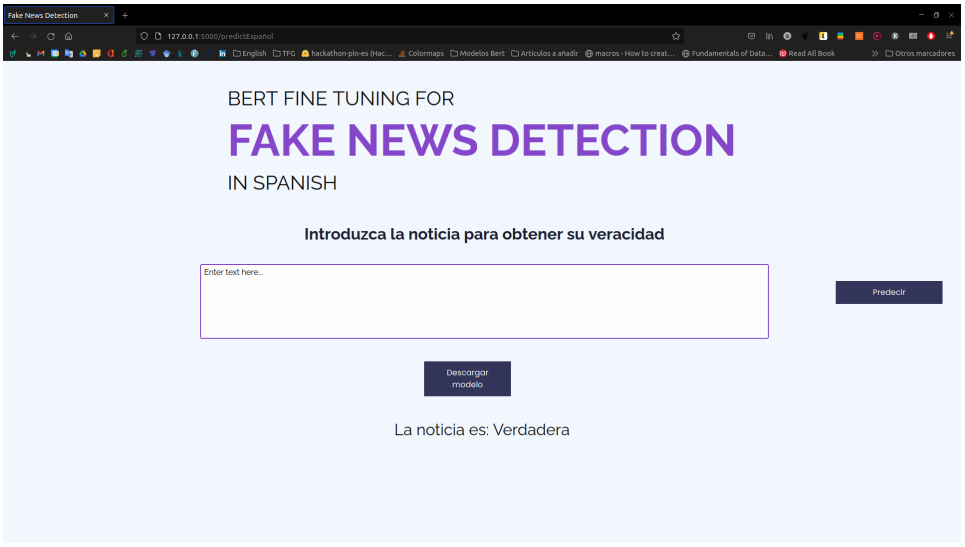
Figura 6.3: Prueba del requisito “Predecir noticia” - Prueba 2.

Prueba 3. Introducción de una noticia verdadera.

La tercera prueba de caja negra para este requisito funcional consistirá en introducir una noticia verdadera para comprobar si el sistema predice correctamente la noticia y notifica al usuario correctamente. Como se puede observar en la figura 6.4 el sistema notifica correctamente al usuario que la noticia es verdadera.



(a) Antes de introducir la noticia.



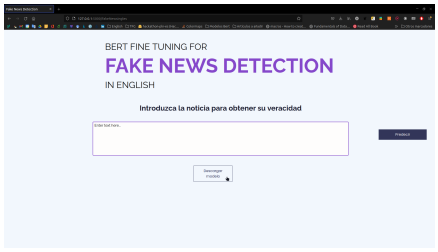
(b) Después de introducir la noticia.

Figura 6.4: Prueba del requisito “Predecir noticia” - Prueba 3.

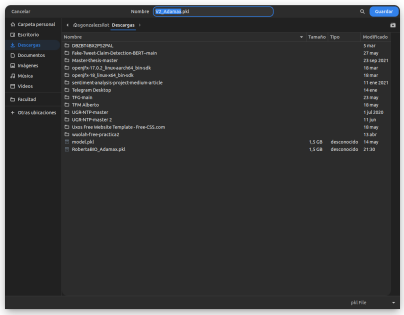
6.2.2. Descargar modelo

Prueba 4. Descarga del modelo correcto.

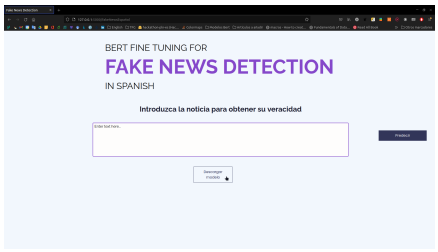
La primera y única prueba de caja negra para este requisito funcional consistirá en solicitar al sistema la descarga del modelo que está utilizando, a fin de ver si el sistema logra descargar el modelo que está utilizando y no el del otro idioma. Como se puede observar en la figura 6.5 el sistema descarga correctamente el modelo del idioma que está utilizando.



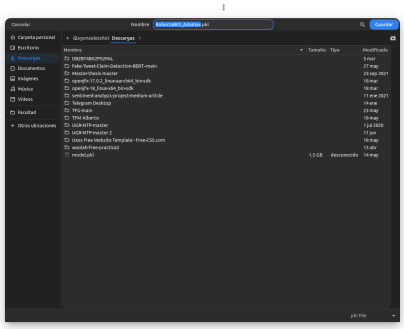
(a) Antes de pulsar el botón.



(b) Después de pulsar el botón.



(c) Antes de pulsar el botón.



(d) Después de pulsar el botón.

Figura 6.5: Prueba del requisito “Descargar modelo” - Prueba 4.

6.3. Conclusiones

A lo largo de esta segunda parte del trabajo se ha desarrollado una aplicación web para comprobar el correcto funcionamiento de los modelos y a su vez presentar al usuario una forma cómoda de interactuar con ellos. Se ha desarrollado esta aplicación web siguiendo un proceso típico de ingeniería del software.

En primer lugar en el análisis se han definido los requisitos y los distintos

diagramas UML que definen el sistema. Posteriormente se ha tratado el diseño de la arquitectura del sistema y las distintas componentes software. Además se ha realizado a su vez el diseño de la interfaz web. Finalmente en este capítulo se han tratado los distintos detalles de implementación del sistema junto a las pruebas de caja negra las cuales validan y verifican el correcto funcionamiento del sistema.

Por tanto, con todo esto se ha presentado una solución web para el usuario común para la detección de noticias tanto en inglés como en español.

Parte III

Conclusiones finales

Capítulo 7

Conclusiones y trabajo futuro

Para finalizar el trabajo se han de analizar los resultados obtenidos y compararlos en función de los objetivos, con esto se podrá saber por un lado si se han cumplido y por otro poder analizar cual es el trabajo futuro que se ha de realizar para seguir avanzando en la tarea de detección de *fake news*.

En este capítulo final del trabajo se comentarán las conclusiones del trabajo en la Sección 7.1 además de posibles trabajos futuros en la Sección 7.2.

7.1. Conclusiones

Si bien la tarea de detección de *fake news* es una tarea amplia y en constante avance e investigación, en este trabajo se han analizado los distintos puntos clave de este problema para su resolución.

- En primer lugar se ha realizado un análisis de los principales conjuntos de datos disponibles para la detección de noticias falsas. Tras este análisis se puede observar la necesidad de la creación de más y mas grandes conjuntos de datos por parte de la comunidad científica. Aunque el principal problema se debe a que es una tarea laboriosa al ser necesaria la anotación manual de la veracidad de las noticias.

  Además, muchos de los conjuntos de datos existentes no se centran en detectar la veracidad analizando el texto si no utilizando diversas características, como la fuente de la noticia, relación del titular con el cuerpo u otras técnicas.

  Si bien es relativamente escaso el número y calidad de los conjuntos

de datos disponibles para la detección de *fake news*, el problema se ve intensificado si se refiere a noticias en castellano, ya que el número de estos conjuntos de datos es menor aún y muchos de ellos incluyendo erróneamente como *fake news* noticias de índole satírica.

- En segundo lugar se han construido una serie de modelos tanto para un conjunto de datos en español como para uno en inglés. Estos modelos representan el estado del arte tanto del PLN como de la detección de *fake-news* al estar utilizando modelos potentes y robustos como lo son *BERT* y *RoBERTa*.

  En el caso del conjunto de datos en inglés (CONSTRAINT AAAI) si bien no se logra mejorar el resultado del ganador de la competición: 98.69 % de F1-score (obtenido con un triple *Ensemble* de modelos de BERT), se logra obtener un resultado muy cercano: 98.41 % con un modelo mucho más sencillo como lo es el *fine-tuning* realizado.

  En el caso del conjunto de datos en español (IberLEF) tampoco se logra mejorar el resultado del ganador de la competición: 76,66 % de F1-score (obtenido con un modelo de alta complejidad, explicado en la Sección 3.3.2), se logra obtener un resultado relativamente cercano: 73.77 % con un modelo mucho más sencillo como lo es el *fine-tuning* realizado.

- Finalmente, con el fin de presentar los resultados del proyecto y su correcto funcionamiento, se ha desarrollado una aplicación web y se ha probado su correcto funcionamiento, y se ha presentado como solución para el usuario común para la detección de *fake news* tanto en inglés como en español. Toda la aplicación web junto al *notebook* de Python utilizado para el desarrollo de los modelos se encuentra disponible en el siguiente enlace de Github[1].

7.2. 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.

- **Creación de un modelo *cross-lingue***. Ya que en la sociedad actual cada vez es más común el conocimiento de más de un idioma, y más en concreto del inglés y el español en conjunto, se propone como posible

[1] `https://github.com/sgonzalezsilot/WebAppTFG`

trabajo futuro la implementación de un modelo *cross-lingue* el cual sea capaz de detectar noticias tanto en inglés como en español. Una posible aproximación sería utilizando un modelo *XLM-BERT*.

- **Creación de un modelo de lenguaje propio**. Si bien durante este trabajo se han utilizado distintos modelos de BERT entrenados por distintas empresas, universidades y organizaciones, otro paso para intentar lograr mejores resultados sería la creación de un modelo propio de BERT. Para esto se debería de utilizar una gran cantidad de datos y entrenar el modelo base de BERT con estos datos durante un gran periodo de tiempo, lo cual es el principal motivo por el que no se ha decidido incluir este apartado en este trabajo.

- **Mejora de la evaluación, análisis y comparación de modelos de lenguaje.** Si bien se han realizado en total 28 pruebas con distintos modelos de BERT y RoBERTa y distintas configuraciones, un posible trabajo futuro sería la inclusión de más modelos de BERT y RoBERTa a este conjunto de pruebas junto al uso de otro tipo de optimizador los cuales también aportan buenos resultados para *Embeddings* como puede ser AdamW [25].

- **Predicción de noticias a través de URL**. En la interfaz web presentada, se permite únicamente la introducción de noticias a través de una caja de texto. Una posible mejora es la inclusión de la posibilidad de detección de *fake news* a través del identificador *URL* del *tweet* o noticia en cuestión. Para ello sería necesario el uso de la *API* de Twitter[2] y de un el desarrollo de un software de *web-scrapping*.

[2]https://developer.twitter.com/en/docs

Bibliografía

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Sarah A Alkhodair, Steven HH Ding, Benjamin CM Fung, and Junqiang Liu. Detecting breaking news rumors of emerging topics in social media. *Information Processing & Management*, 57(2):102018, 2020.

[3] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36, 2017.

[4] Mariette Awad and Rahul Khanna. *Machine Learning*.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Iz Beltagy, Arman Cohan, and Kyle Lo. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676, 2019.

[8] Alessandro Bondielli and Francesco Marcelloni. A survey on fake news and rumour detection techniques. *Information Sciences*, 497:38–55, 2019.

[9] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*, 2020.

[10] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[12] Daelemans, Walter and Hoste, Veronique. Evaluation of machine learning methods for natural language processing tasks. In *LREC 2002 : third international conference on language resources and evaluation*, page 6. European Language Resources Association (ELRA), 2002.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Nicholas DiFonzo and Prashant Bordia. Rumor, gossip and urban legends. *Diogenes*, 54(1):19–35, 2007.

[15] Salma El Anigri, Mohammed Majid Himmi, and Abdelhak Mahmoudi. How bert's dropout fine-tuning affects text classification? In *International Conference on Business Intelligence*, pages 130–139. Springer, 2021.

[16] William Ferreira and Andreas Vlachos. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*. ACL, 2016.

[17] Anna Glazkova, Maksim Glazkov, and Timofey Trifonov. g2tmn at constraint@aaai2021: Exploiting CT-BERT and ensembling learning for COVID-19 fake news detection. *CoRR*, abs/2012.11967, 2020.

[18] Helena Gómez-Adorno, Juan Pablo Posadas-Durán, Gemma Bel Enguix, and Claudia Porto Capetillo. Overview of fakedes at iberlef 2021: Fake news detection in spanish shared task. *Procesamiento del Lenguaje Natural*, 67:223–231, 2021.

[19] Nir Grinberg, Kenneth Joseph, Lisa Friedland, Briony Swire-Thompson, and David Lazer. Fake news on twitter during the 2016 us presidential election. *Science*, 363(6425):374–378, 2019.

[20] Asier Gutiérrez-Fandiño, Jordi Armengol-Estapé, Marc Pàmies, Joan Llop-Palao, Joaquin Silveira-Ocampo, Casimiro Pio Carrino, Carme Armentano-Oller, Carlos Rodriguez-Penagos, Aitor Gonzalez-Agirre, and Marta Villegas. Maria: Spanish language models. *Procesamiento del Lenguaje Natural*, 68(0):39–60, 2022.

[21] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] Benjamin Horne and Sibel Adali. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 759–766, 2017.

[24] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[25] Ricardo Llugsi, Samira El Yacoubi, Allyx Fontaine, and Pablo Lupera. Comparison between adam, adamax and adam w optimizers to implement a weather forecast based on neural networks for the andean city of quito. In *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, pages 1–6, 2021.

[26] Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. Timelms: Diachronic language models from twitter. *arXiv preprint arXiv:2202.03829*, 2022.

[27] Lluís Marquez and Jordi Girona Salgado. Machine learning and natural language processing, 2000.

[28] Tanushree Mitra and Eric Gilbert. Credbank: A large-scale social media corpus with associated credibility annotations. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 9, pages 258–267, 2015.

[29] Martin Müller, Marcel Salathé, and Per E Kummervold. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. *arXiv preprint arXiv:2005.07503*, 2020.

[30] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, 2020.

[31] Parth Patwa, Mohit Bhardwaj, Vineeth Guptha, Gitanjali Kumari, Shivam Sharma, Srinivas PYKL, Amitava Das, Asif Ekbal, Shad Akhtar, and Tanmoy 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 (CONSTRAINT)*. Springer, 2021.

[32] Parth Patwa, Shivam Sharma, Srinivas PYKL, Vineeth Guptha, Gitanjali Kumari, Md Shad Akhtar, Asif Ekbal, Amitava Das, and Tanmoy Chakraborty. Fighting an infodemic: Covid-19 fake news dataset, 2020.

[33] Mohammad Taher Pilehvar and Jose Camacho-Collados. Embeddings in natural language processing: Theory and advances in vector representations of meaning. *Synthesis Lectures on Human Language Technologies*, 13(4):1–175, 2020.

[34] Juan-Pablo Posadas-Durán, Helena Gómez-Adorno, Grigori Sidorov, and Jesús Jaime Moreno Escobar. Detection of fake news in a new corpus for the spanish language. *Journal of Intelligent & Fuzzy Systems*, 36(5):4869–4876, 2019.

[35] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[36] J Ross Quinlan. Learning decision tree classifiers. *ACM Computing Surveys (CSUR)*, 28(1):71–72, 1996.

[37] Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264*, 2017.

[38] Aránzazu Román-San-Miguel, Nuria Sánchez-Gey Valenzuela, and Rodrigo Elías Zambrano. Las fake news durante el estado de alarma por covid-19. análisis desde el punto de vista político en la prensa española. *Revista latina de comunicación social*, (78):359–391, 2020.

[39] Victoria L Rubin, Yimin Chen, and Nadia K Conroy. Deception detection for news: three types of fakes. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4, 2015.

[40] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004.

[41] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan 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.

[42] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36, sep 2017.

[43] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China national conference on Chinese computational linguistics*, pages 194–206. Springer, 2019.

[44] Eugenio Tacchini, Gabriele Ballarin, Marco L Della Vedova, Stefano Moret, and Luca de Alfaro. Some like it hoax: Automated fake news detection in social networks. *arXiv preprint arXiv:1704.07506*, 2017.

[45] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*, 2018.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[47] Andreas Vlachos and Sebastian 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, 2014.

[48] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[49] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

[50] Xinyi Zhou and Reza Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys (CSUR)*, 53(5):1–40, 2020.

[51] Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. Detection and resolution of rumours in social media: A survey. *ACM Comput. Surv.*, 51(2), feb 2018.

[52] Arkaitz Zubiaga, Maria Liakata, Rob Procter, Kalina Bontcheva, and Peter Tolmie. Towards detecting rumours in social media. In *Workshops at the Twenty-Ninth AAAI conference on artificial intelligence*, 2015.

[53] Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989, 2016.