Desarrollo de una herramienta web de detección de Fake News en Ecuador, empleando algoritmos de Machine Learning

Diakov, Artem Dmitrievich

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Sistemas e Informática

Trabajo de titulación, previo a la obtención del título de Ingeniero en Sistemas e Informática

Ing. Quiroz Corrales Dorys Soledad

15 de agosto de 2023

Copyleaks

Plagiarism report

DIAKOV_ARTEM_TESIS Version DQ.docx

Scan details

Scan time:	Total Pages:	Total Words:
August 14th, 2023 at 14:21 UTC	126	31395

Plagiarism Detection

Types of plagiarism		Words
Identical	0.7%	224
Minor Changes	0.4%	135
Paraphrased	0.2%	52
Omitted Words	0%	0

AI Content Detection

Text coverage
- AI text
- Human text

Plagiarism Results: (31)

ML707 - Course Materials - Spring 2022 - LibGuides a... 0.1%

https://mbzuai.libguides.com/spring2022/ml707

Skip to Main Content Mohamed bin Zayed University of Artificial Intelligence Library LibGuides Course Materials -...

Dipòsit Digital de la Universitat de Barcelona: Màst... 0.1%

https://diposit.ub.edu/dspace/handle/2445/170310

Skip navigation ...

Dipòsit Digital de la Universitat de Barcelona: Del fe... 0.1%

https://diposit.ub.edu/dspace/handle/2445/178905

Skip navigation ...

Departamento de Ciencias de Computación

Carrera de Ingeniería de Sistemas e Informática

Certificación

Certifico que el trabajo de titulación: "**Desarrollo de una herramienta web de detección de Fake News en Ecuador, empleando algoritmos de Machine Learning**" fue realizado por el señor **Diakov, Artem Dmitrievich;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 15 de agosto de 2023.

Firma:

DORYS SOLEDAD QUIROZ CORRALES

Firmado digitalmente por DORYS SOLEDAD QUIROZ CORRALES
DN: CN=DORYS SOLEDAD QUIROZ CORRALES, SERIALNUMBER=1108231310O7, OU=ENTIDAD DE CERTIFICACIÓN DE INFORMACIÓN, O=SECURITY DATA S.A. 2, C=EC
Razón: Soy el autor de este documento
Ubicación: CastleCloud S.A.S.
Fecha: 2023.08.15 12:19:32-05'00'
Foxit PDF Reader Versión: 2023.2.0

Quiroz Corrales, Dorys Soledad

C. C. 1714679642.

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Sistemas e Informática

Responsabilidad de Autoría

Yo, **Diakov Artem Dmitrievich** con cédula de ciudadanía n° 1724313042, declaro que el contenido, ideas y criterios del trabajo de titulación: **Desarrollo de una herramienta web de detección de Fake News en Ecuador, empleando algoritmos de Machine Learning** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 15 de agosto del 2023

Artem Diakov

Diakov, Artem Dmitrievich

C.C.: 1724313042

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Sistemas e Informática

Autorización de Publicación

Yo, **Diakov Artem Dmitrievich** con cédula de ciudadanía n°1724313042, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"Desarrollo de una herramienta web de detección de Fake News en Ecuador, empleando algoritmos de Machine Learning"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 15 de agosto del 2023

Diakov, Artem Dmitrievich

C.C.: 1724313042

Dedicatoria

Hoy, con inmensa gratitud, dedico esta tesis de grado a cada uno de ustedes, quienes han sido mi fuente constante de apoyo y amor incondicional a lo largo de este exigente camino académico. A mis queridos padres, su incansable aliento y sacrificio han sido la luz que me ha guiado en los momentos más oscuros. Vuestra confianza en mis habilidades me ha impulsado a superar cada obstáculo. A mis profesores y mentores, que me mostraron el camino del conocimiento, gracias por su paciencia, sabiduría y por despertar en mí una pasión por aprender. Su guía ha sido fundamental en mi formación como persona y profesional.

Agradecimiento

Quisiera expresar mi más profundo agradecimiento a todas aquellas personas que han contribuido de manera significativa en la realización de esta tesis de grado. Sin su apoyo, este logro no hubiera sido posible. Agradezco sinceramente a los profesores y expertos que conformaron el comité evaluador de esta tesis. Sus observaciones y recomendaciones fueron fundamentales para mejorar la calidad del estudio.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

Este estudio presenta el desarrollo de una herramienta web para la detección de noticias falsas (Fake News) en el contexto ecuatoriano, mediante la utilización de algoritmos de Machine Learning e Deep Learning. La problemática de las noticias falsas en Ecuador es especialmente relevante debido a su potencial de manipulación de la opinión pública y la desinformación masiva, lo que se traduce en serias implicaciones políticas y sociales. En la primera fase del estudio, se recopilaron y procesaron grandes bases de datos de noticias internacionales, permitiendo el entrenamiento de modelos de Machine Learning para la detección de Fake News. A continuación, se llevó a cabo el minado de noticias específicas de Ecuador, con el objetivo de adaptar y optimizar estos modelos al contexto local. Un aspecto innovador de este trabajo es el uso de la API de chat GPT-3.5 de OpenAI. Esta estrategia permitió mejorar la precisión de las predicciones al proporcionar a los algoritmos una mayor cantidad de ejemplos de Fake News con las características y matices propios del entorno ecuatoriano. Luego de cual fue entrenada con red neuronal para aprender de asociaciones (Word2Vec) y posteriormente con red neuronal recurrente para aprender de las dependencias de las mismas (LSTM). La herramienta fue desarrollada como una aplicación web utilizando el framework Flask, optando por un diseño centrado en el usuario que facilita su interacción y comprensión. El despliegue se realizó en un servidor EC2 de Amazon, garantizando su accesibilidad y eficiencia. Finalmente, se adquirió un dominio y se aplicó un certificado de seguridad de LetsEncrypt.

Palabras clave: Noticias falsas, Aprendizaje de máquina, Aprendizaje profundo, Web App, LSTM.

Abstract

This study presents the development of a web tool for the detection of Fake News in the Ecuadorian context, through the use of Machine Learning and Deep Learning algorithms. The problem of fake news in Ecuador is especially relevant due to its potential to manipulate public opinion and massive misinformation, which translates into serious political and social implications. In the first phase of the, large databases of international news were collected and processed, allowing the training of Machine Learning study models for the detection of Fake News. Next, the mining of specific news from Ecuador was carried out, with the aim of adapting and optimizing these models to the local context. An innovative aspect of this work is the use of OpenAI's GPT-3.5 chat API. This strategy made it possible to improve the accuracy of the predictions by providing the number of algorithms with a greater number of examples of Fake News with the characteristics and nuances of the Ecuadorian environment. After which it was trained with a neural network to learn from associations (Word2Vec) and later with a recurrent neural network to learn from their dependencies (LSTM). The tool was developed as a web application using the Flask framework, opting for a user-centered design that facilitates interaction and understanding. The use was made on an Amazon EC2 server, guaranteeing its accessibility and efficiency. Finally, a domain was acquired and a LetsEncrypt.

Key Words: Fake news, Machine learning, Deep learning, Web App, LSTM.

Capítulo I

Introducción

Antecedentes

El paro de octubre de 2019 en Ecuador fue un acontecimiento muy significativo en la historia reciente del país. Este evento generó una gran cantidad de protestas en todo el país en respuesta a la decisión del gobierno de implementar una serie de medidas de austeridad, incluyendo la eliminación de subsidios a los combustibles, como parte de un acuerdo con el Fondo Monetario Internacional (FMI). (METRO ECUADOR, 2022)

Las "fake news" o noticias falsas, fueron un componente relevante durante este periodo, ya que afectaron la percepción y el comportamiento de los individuos, y en algunos casos contribuyeron a la escalada de la violencia y la polarización política. (METRO ECUADOR, 2022)

Según el articulo (Larrea et al., 2022) “Durante el paro nacional de 2019 los medios comunitarios y privados, debido a que la percepción de la ciudadanía, consideraba que los grandes medios nacionales impusieron un cerco informativo debido a que la construcción de la noticia, difería de los contenidos publicados por medios alternativos e incluso por ciudadanos ajenos al periodismo y a través de las redes sociales”. Dicha situación se repetía nuevamente en el paro nacional del 2022, el cual, según (METRO ECUADOR, 2022) “se intensifico por la inoportuna operación de los organismos de control estatales”. Bajo esta lógica se puedo observar una reacción comunicativa desestabilizadora.

Con la detención del principal líder de las movilizaciones indígenas, Leonidas Iza, se efectuó un enorme aparataje mediático, dando cabida a la especulación. Por una parte, los medios tradicionales afirmaban que la noticia era real, pero que no se estaban vulnerando sus

derechos, en tanto que, pequeños medios independientes y locales denunciaban que el arresto es ilegítimo y que se están vulnerando los derechos. Ante esto, los manifestantes intensificaron sus posturas y quizás, para muchos, este fue el detonante de la radicalización del paro. (METRO ECUADOR, 2022)

En Ecuador, las noticias falsas tienen un impacto considerable en los conflictos y las protestas sociales. Si una información incorrecta se propaga, puede distorsionar la percepción pública de los eventos de protesta, exacerbando las tensiones y alimentando la polarización entre diferentes sectores de la sociedad. Según (Granda & Paladines, Fanny, 2021). Las noticias falsas fueron usadas para manipular la opinión pública, presentando a los manifestantes como agitadores violentos o al gobierno como desmesuradamente represivo, lo que obstaculiza los intentos de diálogo pacífico y constructivo.

Los paros pueden provocar enfrentamientos entre los manifestantes y las fuerzas de seguridad, lo que lleva a un aumento de la violencia y la inseguridad. Además, los servicios esenciales como la educación, la salud y el transporte público se vieron afectados, atentando a la calidad de vida de la población. Por otra parte, la tensión política y la polarización, se convirtieron en los principales factores desestabilizadores de los gobiernos de turno. En ambos casos, tanto en el 2019 como en el 2022 la respuesta del gobierno al paro, derivó en conflictos internos y luchas por el poder, dejando marcas abiertas que no han podido ser subsanadas hasta la presente fecha. (Torrico, 2022)

A la larga ambos escenarios evidenciaron las desigualdades existentes en la sociedad. Según (Lara, 2022) La división, el clasismo y el racismo afloro en un punto máximo, poniendo a los indígenas en el centro del conflicto, como los impulsadores del caos. Cabe mencionar que las personas que viven al día o tienen trabajos informales se vieron particularmente afectadas

por la pérdida de ingresos durante el conflicto. Además, las regiones más pobres o rurales del país sufrieron la interrupción de los servicios y el comercio.

(Larrea et al., 2022) “los medios de comunicación pueden causar que las audiencias se interesen en temáticas específicas y centren su atención en acontecimientos particulares, incluso ante la facultad de los receptores para interpretar el mensaje. (McCombs, 1972), proponen la teoría del establecimiento de agenda, con la que infieren que los medios a través de su selección de contenidos promueven que el público preste o no atención a los hechos que se debaten en la esfera pública”. Con el presente fundamento se puede entender la relevancia de la agenda comunicacional en las reacciones de los ciudadanos. En tal sentido, si, incluso las noticias verdaderas pueden inferir tanto en las reacciones sociales, las noticias falsas, totalmente manipuladas, se convierten en un problema de gran escala.

(Valenzuela, 2020) En su trabajo de grado titulado “Detección de Fake News mediante técnicas de Deep Learning” Desarrolla un sistema utilizando diferentes métodos como clasificación analizando ventajas y desventajas en cada modelo empleado. El método usado es en el desarrollo fueron nueve. Se realizo con librerías más populares de Machine Learning como TensorFlow, scikit-learn, nltk, nympy, pandas y otros. El resultado obtenido fue un Score en entrenamiento de detección de Fake News fue de 96.66%.

(Alvaro Ibrain Rodriguez, 2019)En el artículo “Fake News Detection using Deep Learning”. Habla sobre una manera de analizar las noticias falsas aplicando Deep Learning o aprendizaje profundo. Las noticias falsas son una tendencia en el mundo actual por que tiene el potencial de desestabilizar gobiernos, según ellos puede poner en peligro a la sociedad. Se menciona también ataques a las elecciones en Estados Unidos donde hacen que los votantes prefieran a un elector en vez de otro, alterando información.

Así mismo en este articulo aplican 3 técnicas sofisticadas diferentes utilizando Redes neuronales. Uno de ellos es BERT creado por Google, LSTM (Long Short Term Memory) y CNN (Convolutional Neural Networks)

(Amoros, 2018) expone el libro titulado: Fake News: La verdad de las noticias falsas". En 2017 aparece la palabra Fake News, para referirse a las noticias falsas que circulan en internet. Una expresión cuyo uso en los últimos doce meses ha aumentado en un 365%. Se pronostica que para el 2022 la mitad de las noticas serán FakeNews. Y no se podrán eliminar ni con la ayuda de las maquinas. Los medios que desarrollan este tipo de noticias pueden caer en la obsesión de información desechable, en tanto que los lectores, de consumirla.

(Fernandez-Garcia, 2017) En su trabajo "Fake News: una oportunidad para la alfabetización mediática", se establece que las noticias falsas han existido durante mucho tiempo, sin embargo, su propagación exponencial a través de las redes sociales es un fenómeno más reciente. En la actualidad, el término "fake news", ampliamente utilizado en varios idiomas, refleja esta realidad. La pérdida de importancia de la fuente original de la noticia y la viralización, un concepto contemporáneo, a menudo reducen el interés en la autenticidad de la noticia y la capacidad crítica de discernir lo falso. Dado que una gran cantidad de la población se informa a través de las redes sociales, estas cuestiones tienen implicaciones políticas significativas, como se ha evidenciado en varios eventos recientes.

(Ashkan Kazemi, 2021) en la obra titulada "Automatic Detection of Fake News": La proliferación de información engañosa en los medios de acceso cotidianos, como las redes sociales, los blogs de noticias y los periódicos en línea, ha dificultado la identificación de fuentes de noticias confiables, aumentando así la necesidad de herramientas computacionales capaces de proporcionar información sobre la confiabilidad del contenido en línea. En este artículo, nos

centramos en la identificación automática de contenido falso en noticias online. Nuestra contribución es doble.

Primero, se presentó dos conjuntos de datos novedosos para la tarea de detección de noticias falsas, que cubren siete dominios de noticias diferentes. Describimos el proceso de recopilación, anotación y validación en detalle y presentamos varios análisis exploratorios sobre la identificación de diferencias lingüísticas en el contenido de noticias falsas y legítimas. En segundo lugar, llevamos a cabo una serie de experimentos de aprendizaje para construir detectores de noticias falsas precisos. Además, proporcionamos análisis comparativos de la identificación automática y manual de noticias falsas.

(Soujanya Poria, 2020) exponen el tema: “Exemplars-guided Empathetic Response Generation Controlled by the Elements of Human Communication”. La mayoría de los métodos existentes para la generación de respuestas empáticas se basan en la emoción del contexto para generar respuestas empáticas. Sin embargo, la empatía es mucho más que generar respuestas con una emoción adecuada. También suele conllevar sutiles expresiones de comprensión y resonancia personal con la situación del otro interlocutor. Desafortunadamente, estas cualidades son difíciles de cuantificar y los conjuntos de datos carecen de las anotaciones relevantes.

(Di Jin, 2020) ¿Presentaron el artículo “How Good Is NLP? A Sober Look at NLP Tasks through the Lens of Social Impact “. En los últimos años se han producido muchos avances en el procesamiento del lenguaje natural (PNL), que han pasado de un campo mayoritariamente teórico a uno con muchas aplicaciones del mundo real. Al observar el creciente número de aplicaciones de otras técnicas de aprendizaje automático y de inteligencia artificial con un impacto social generalizado, anticipamos la creciente importancia de desarrollar tecnologías de PNL para el bien social.

(Deepanway Ghosal, 2020) presenta el título: “Commonsense Inference for Dialogue Explanation and Reasoning”. La inferencia de sentido común para comprender y explicar el lenguaje humano es un problema de investigación fundamental en el procesamiento del lenguaje natural. Explicar las conversaciones humanas plantea un gran desafío, ya que requiere comprensión contextual, planificación, inferencia y varios aspectos del razonamiento, incluido el razonamiento causal, temporal y de sentido común. En este trabajo, se presentó CIDER, un conjunto de datos curado manualmente que contiene explicaciones de diálogo diádico en forma de tripletes de conocimiento implícitos y explícitos inferidos mediante inferencia contextual de sentido común. Extraer explicaciones tan ricas de las conversaciones puede contribuir a mejorar varias aplicaciones posteriores.

(Yiqun Yao, 2021)expone el documento titulado “Multimodal Stress Detection using Emotion Recognition as an Auxiliary”. La capacidad de detectar automáticamente el estrés humano puede beneficiar a los agentes inteligentes artificiales involucrados en la computación afectiva y la interacción humano-computadora. El estrés y la emoción son estados afectivos humanos y se ha demostrado que el estrés tiene importantes implicaciones en la regulación y expresión de la emoción. Aunque se han establecido una serie de métodos para la detección de estrés multimodal, se han tomado pasos limitados para explorar la interdependencia subyacente entre el estrés y la emoción. En este trabajo investigamos el valor del reconocimiento de emociones como tarea auxiliar para mejorar la detección del estrés.

(MeiXing Dong, 2021) expone, ‘’El deseo de cambio es una constante en muchas personas, pero lograrlo puede ser un desafío”. La psicología social ha propuesto varias teorías que explican las características de aquellos que logran mantener su motivación para cambiar,

sin embargo, hay una brecha en los estudios que aplican métodos computacionales para explorar esta etapa crucial de la transformación personal.

Este artículo, se adentra en la investigación de un innovador conjunto de datos, el cual reúne escritos de personas con una intención declarada de cambiar. Este conjunto es especial, ya que engloba tanto a individuos que han demostrado persistencia en su empeño, como a aquellos que no lo han conseguido. Para entender mejor las diferencias entre estos dos grupos, recurrimos a varias técnicas de análisis lingüístico. Nuestro primer objetivo es desentrañar los patrones de escritura que permiten diferenciar a quienes logran sostener su impulso de cambio de aquellos que no lo hacen.

(Ashkan Kazemi, 2021) en el documento titulado “Extractive and Abstractive Explanations for Fact-Checking and Evaluation of News” se exploró la construcción de explicaciones en lenguaje natural para afirmaciones de noticias, con el objetivo de ayudar a las aplicaciones de verificación de datos y evaluación de noticias. Experimentamos con dos métodos: (1) un método de extracción basado en Biased TextRank, un algoritmo basado en gráficos no supervisado y eficaz en función de los recursos para la extracción de contenido; y (2) un método abstractivo basado en el modelo de lenguaje GPT-2.

(Santiago Castro, 2021)exponen: “Fill-in-the-blank as a Challenging Video Understanding Evaluation Framework“ El trabajo hasta la fecha sobre la comprensión de videos informada por el lenguaje ha abordado principalmente dos tareas: (1) respuesta a preguntas de video usando preguntas de opción múltiple, donde los modelos funcionan relativamente bien porque explotan el hecho de que las respuestas de los candidatos están fácilmente disponibles; y (2) subtítulos de video, que se basan en un marco de evaluación abierto que a menudo es inexacto porque las respuestas del sistema pueden percibirse como incorrectas si difieren en forma de la verdad

básica. En este documento, proponemos rellenar los espacios en blanco como un marco de evaluación de comprensión de video que aborda estos inconvenientes de la evaluación anterior y refleja más de cerca los entornos de la vida real donde no se ofrecen múltiples opciones.

En el trabajo de (Fernandez-Garcia, 2017), titulado "FakeNews, ¿Amenaza u oportunidad para los profesionales de la información y la documentación?", se ofrece una visión contextualizada sobre el surgimiento y la relevancia de las noticias falsas en el ámbito de la información y documentación. El autor aborda el papel que pueden desempeñar los profesionales del campo de manera efectiva y eficiente frente a este fenómeno. Se describe una variedad de iniciativas y proyectos emprendidos tanto por bibliotecas y sus trabajadores, como por los sectores educativo y comunicativo, que también están involucrados en la problemática de las noticias falsas y la posverdad. En la conclusión, se sugiere la revisión de prácticas y actividades previamente desarrolladas, además de promover la colaboración con otros sectores profesionales y fortalecer proyectos de formación en competencias digitales y mediáticas.

(Rivero, 2020) Expone el libro titulado “Fake news, trolls y otros encantos: Cómo funcionan (para bien y para mal)” Se trata de un esfuerzo interdisciplinario que involucra una combinación teórica y empírica de los campos de la comunicación, la ciencia política y la estadística. Este trabajo integra el análisis de categorías tanto de textos clásicos como de otros más contemporáneos. Además, establece una discusión crítica con conceptos teóricos formulados en diferentes contextos mediáticos, con el objetivo de adaptarlos y modernizarlos en función de las nuevas formas de diálogo público facilitadas por las redes sociales digitales.

(Álvarez Rufs, 2018) “Estado del arte: posverdad y fake news”. El estudio tiene como finalidad llevar a cabo una indagación Estado del Arte en torno a la Posverdad y las Fake News, abarcando una definición de estos conceptos, su origen, atributos, modo de operación, posibles

efectos, y las sugerencias y movimientos que sean significativos. Esta investigación está organizada en cuatro secciones distintas: La primera sección detalla la metodología aplicada en el estudio, incluyendo un cronograma, las guías instructivas originales y adaptadas, y las referencias bibliográficas. La segunda sección consta del documento estado del arte completo, resultado de la investigación, y se subdivide en dos segmentos para cada tema principal, la Posverdad y las Fake News. La tercera sección brinda conclusiones preliminares del investigador sobre el estado del arte desarrollado. La cuarta y última sección contiene el informe conclusivo de la investigación.

Planteamiento del Problema

Justificación e Importancia

El fenómeno de las Fake News o noticias falsas ha experimentado un crecimiento considerable en la última década, agravado por el auge de las redes sociales y las plataformas de noticias digitales (Ustarroz Molina M. (., 2021). En este entorno, el papel de la veracidad de la información se vuelve crítico, pues las noticias falsas pueden causar daños significativos a nivel social, económico y político. Ecuador no está exento de este fenómeno global y enfrenta el desafío de lidiar con la propagación de información falsa. La necesidad de abordar esta problemática hace esencial la investigación y desarrollo de una herramienta web de detección de Fake News en el país, utilizando algoritmos de Machine Learning.

La importancia de este proyecto radica en su potencial para mejorar la calidad de la información que circula en el espacio digital ecuatoriano. A través de la detección precisa y oportuna de noticias falsas, los ciudadanos estarán mejor equipados para discernir entre información veraz y desinformación, favoreciendo un mejor entendimiento del entorno y toma de decisiones informadas. En el campo político, la herramienta puede contribuir a la integridad de

los procesos democráticos, que a menudo se ven afectados por campañas de desinformación. (Ustarroz Molina M. , 2021)En el ámbito económico, puede ayudar a proteger a las empresas de prácticas comerciales injustas basadas en la propagación de noticias falsas.

El uso de algoritmos de Machine Learning ofrece la oportunidad de abordar el problema de las Fake News de manera innovadora y eficaz. Estos algoritmos pueden entrenarse para identificar patrones y características comunes en las noticias falsas, lo que les permite detectar y clasificar nuevas instancias con alta precisión. A medida que se alimentan con más datos, los algoritmos se vuelven más precisos, proporcionando un sistema robusto y escalable de detección de Fake News. (Cornejo Macías, 2021)

La justificación de este proyecto se encuentra en la creciente necesidad de abordar la problemática de las Fake News en la sociedad ecuatoriana. La desinformación puede afectar negativamente la percepción pública sobre temas importantes, influir en el comportamiento de voto, y fomentar la desconfianza en las instituciones y los medios de comunicación (Allcott, 2017). El desarrollo de una herramienta de detección de Fake News podría combatir estos efectos negativos y promover un espacio digital más seguro y confiable.

Además, el proyecto tiene el potencial de contribuir significativamente al campo de la ciencia de datos en Ecuador. Podría proporcionar un caso de estudio valioso para el uso de Machine Learning en el procesamiento del lenguaje natural y la detección de desinformación. También podría estimular más investigación y desarrollo en este campo, impulsando la innovación y la adopción de tecnologías de inteligencia artificial en el país.

El desarrollo de una herramienta web de detección de Fake News en Ecuador, utilizando algoritmos de Machine Learning, es un proyecto de investigación crucial. No solo tiene el

potencial de mitigar el impacto de las Fake News en la sociedad ecuatoriana, sino que también podría promover la adopción de tecnologías avanzadas de IA y fortalecer el campo de la ciencia de datos en el país. Por tanto, es de gran importancia llevar a cabo esta investigación y desarrollar esta valiosa herramienta para el beneficio de la sociedad ecuatoriana.

Preguntas de problemática.

- **Donde**: En los medios de comunicación electrónica en Ecuador
- **Quienes:** Los ciudadanos en Ecuador
- **Problema:** Alta susceptibilidad por la desinformación causada por los FakeNews
- **El problema:** Los ciudadanos de Ecuador tienen una alta susceptibilidad a los FakeNews en los medios de comunicación electrónicos en Ecuador.

Causas

- La rápida divulgación de la información por medio de redes informáticas.
- La no existencia de filtración, de información falsa de la verídica en medios informáticos.
- El aumento creciente de los “bulíes”.
- Formas de promoción de negocios de manera poco ética.
- Uso de la información falsa para realizar ataques de desestabilización de los gobiernos, empresas, organizaciones o personas.

Consecuencias

- Graves daños a los individuos, demás personas y organizaciones.
- Incitación al caos.
- Incitación a la violencia.

- Desestabilización de la sociedad.
- Desinformación en la población.

Hipótesis

La detección de Fake News en Ecuador utilizando Deep Learning o Machine Learning puede discernir información verdadera de la falsa.

Aporte

Implementar un algoritmo adecuado para entrenar la red neuronal y deducir si la información publicada en diferentes medios es verídica o falsa. Utilizando Deep Learning y Machine Learning. Entrenándolo para el análisis de contexto en el que se desarrollan las palabras.

La información en la época actual viene dada en diversos niveles. La cual es transmitida en los noticieros y medios electrónicos y físicos en todo el mundo. La globalización y la tecnología ayudó a que dicha información se propague rápidamente en todos los medios.

Los noticieros o canales televisión son los que transmiten muchas veces reportajes en sitio utilizando medios visuales y auditivos para su presentación.

Los periódicos locales difunden las noticias en medios físicos. La obtención y transparencia de la información puede ser muchas veces de dudosa procedencia, imparcial o dirigida hacia alguna parte beneficiada. Este tipo de noticias o reportajes por lo general no causan mayores disturbios y caos en la población, pero otros tipos de noticias que nos son transmitidos por fuentes confiables tienen mayor probabilidad de desestabilizar una nación. Sin embargo, contenido de las fuentes clásicas de información son puestas en duda por parte de grupos

contrarios, lo cual, incluso la información real, puede ser utilizada como mecanismo de estallido social.

El desafío que plantean las Fake News reside en su difícil diferenciación de las noticias relevantes y auténticas. Entre las múltiples manifestaciones de este fenómeno, se encuentran aquellas noticias que difunden información no comprobada o respaldada por una fuente confiable o por un experto en el campo en cuestión. Además, se suma la problemática de la manipulación de imágenes y sonidos mediante inteligencia artificial. Este fenómeno puede llegar a involucrar la alteración de rostros y voces de figuras públicas relevantes, permitiendo, literalmente, poner palabras en sus bocas. Esto dificulta aún más la tarea de distinguir entre la realidad y la fabricación de noticias.

En la actualidad, la proliferación de diversas tecnologías está propulsando la recopilación y el procesamiento de la información a un nivel completamente nuevo. Este avance se atribuye a la implementación de tecnologías como el Aprendizaje Automático (Machine Learning), el Aprendizaje Profundo (Deep Learning) y la Inteligencia Artificial. Estas tecnologías se basan en el uso de algoritmos que aplican fórmulas matemáticas para interpretar o simular redes neuronales. Este proceso implica el ajuste de los valores de peso y la retroalimentación constante de los datos hasta alcanzar un valor de salida deseado. De este modo, se logra un aprendizaje y un procesamiento de información cada vez más sofisticado y preciso.

Objetivos

Objetivo General

Desarrollar una herramienta web de detección de Fake News en Ecuador utilizando Machine Learning, recopilando información de diferentes datasets libres.

Objetivos Específicos

- Obtener y recolectar información de Fake News y datasets libres.
- Transformar y traducir los datos obtenidos a idioma español.
- Diseñar el motor de detección de Fake News utilizando Inteligencia Artificial.
- Desarrollar la aplicación web.
- Probar la herramienta con datos externos de redes sociales.

Capítulo II

Marco teórico

Fake News

En la actualidad Fake News es una tendencia mundial, pues la mayoría de los Gobiernos quieren interactuar con la creación de contenido falso. En el escenario político son utilizadas para crear percepciones sobre la opinión pública. No obstante, este fenómeno también se desarrolla a nivel empresarial, deportivo, noticias del espectáculo, creación de cortinas de humo, anarquía, y protesta social. Dichas noticias pueden ser manipuladas en pro del participante y en contra.

Y como todo tiene sus 2 partes de la moneda también los FakeNews son generados por entes desestabilizadores donde quieren aprovecharse de la inestabilidad del país tal hecho que ocurrieron en octubre 2019 en Ecuador. Donde hubo varias imágenes y videos que circulaban por las redes sociales y medios de mensajería como WhatsApp principalmente que mostraban imágenes de contexto incompleto o audio cambiado o inconscientemente alterado donde las personas que graban anuncian lo que ven de una manera equivoca el escenario real.

El caso más reciente y que lamentablemente sigue pasando en el país es la desinformación con respecto a las vacunas. En ellos afirman que el cuerpo luego de la vacuna “se magnetiza” y sostiene objetos metálicos, otros dicen que las vacunas vienen con “chips espías” que recopilan los datos de uno a todo instante, otros sostienen que una vacuna es mejor que la otra (El Comercio,2021) y otro sin número de casos más que está en la lista de los FakeNews. Los actores suelen utilizarlas para generar una imagen positiva sobre sí mismo y/o desacreditar a los adversarios. Afirma (Kuklinski, 2016) que “la comunicación política sabe desde siempre que, entre racionalidad y emoción, predomina la emoción, y que la manipulación, las

medias verdades o directamente las mentiras estratégicas hacen su juego para construir una base electoral o consolidar una idea política". (p. 205).

En el Ecuador se vivió de cerca una alta coyuntura de noticias falsas durante las protestas de octubre 2019, por una parte, medios de comunicación generaban una difusión encuadrada en base a su criterio, la cual era refutada por quienes incitaban la continuidad del caos. La información masiva emitida desde distintos escenarios generaba desconfianza en la población, la cual no lograba comprender a fondo el contexto de la problemática. Mientras más impacto tiene un contenido descontextualizado se vuelve más complicado.

De acuerdo con (López, 2018), un caso emblemático que ilustra este fenómeno a nivel mundial es "la campaña presidencial de Donald Trump. Su ascenso al puesto de presidente de los Estados Unidos se atribuye, en parte, a la manipulación de la verdad. De hecho, el sitio web Politifact reportó que el 70% de las afirmaciones de Trump durante su campaña electoral eran fundamentalmente incorrectas" (p. 82). En el contexto actual, las noticias falsas han adquirido una significativa importancia. En muchos casos, los medios de comunicación se ven inundados por 'haters' y 'trolls' que tienen la capacidad de difundir ampliamente esta información, siendo parte de un extenso sistema de producción de noticias con diversas perspectivas e influencia política.

También se puede observar en elecciones, cómo pueden ser manipuladas, por ejemplo, para obtener votos con partidarios de la anti migración "se inventan noticias de que aumentan delitos cometidos por la población inmigrante" (unamglobal, 2021), con eso logran potenciar los votos por parte de las personas que están en contra del migrante o que no le gusta que mucha gente extranjera venga a su país. Entonces en este caso vemos que es posible focalizar grupos

utilizando medios electrónicos. Abundar a la población que prefiere ciertos tipos de criterio y al final sumando la mayor cantidad de votos.

Fake News también es una buena fuente de ganancia de dinero. Los famosos anuncios que ganan por los clics en los sitios webs muchas veces ponen noticias espectaculares como para enganchar interés de las personas. El usuario al clicar dicho anuncio para tratar de leer más sobre el tema, es dirigido a otro sitio donde poco o nada tiene que ver con el contenido anunciado. Allí es donde se genera mayor parte de dinero. Pero estos no solo son usados en sitios web, en páginas sociales también donde al hacer clic en like o compartir y difundir dicha información el creador de Fake News también gana dinero.

“Eco Chamber Effect” (Kai Shu, 2018) es un efecto causado cuando las personas de mismos creencias y pensamientos reaccionan a las noticias en donde si son noticias falsas genera más tendencia en difundir dicha información. Cabe destacar que dicho efecto se produce cuando una noticia falsa es reaccionada por una persona que cree en un cierto tema. Estas personas por lo general tienden psicológicamente a juntar con las personas de las mismas creencias. Con esto empiezan a generar grupos más grandes donde empiezan a buscar fuentes que afirmen su creencia, si esta fue noticia falsa pues genera más tiempo de uso de las redes y genera por ende mucho más dinero.

Las Fake News son un tipo de bulo que consiste en un contenido pseudo-periodístico difundido a través de portales de noticias, prensa escrita, radio, televisión y redes sociales y cuyo objetivo es la desinformación. (UN, 2021)

Inteligencia Artificial

Desde el punto de vista técnico el avance de las tecnologías ha permitido el desarrollo de inteligencia artificial o IA, básicamente es la simulación de una neurona biológica simple con el uso recursivo de programas informáticos. La IA es uno de los campos más innovadores y prometedores de la tecnología actual, y su creciente importancia y aplicación no pueden ser subestimadas. Según (P.Russell, 2016), los autores de "Artificial Intelligence: A Modern Approach", la IA se define como "la ciencia e ingeniería de hacer máquinas inteligentes, especialmente programas de computadoras inteligentes" (p. 1).

Este desarrollo tecnológico tiene la capacidad de transformar la sociedad y la economía de maneras inimaginables, sin embargo, también plantea ciertas cuestiones éticas y regulatorias. Como afirma (Bostrom, 2014) en "Superintelligence: Paths, Dangers, Strategies", "la creación de una inteligencia artificial avanzada puede ser la última invención que la humanidad necesite hacer, si no se administra adecuadamente" (p. 116).

Es interesante destacar cómo la IA está reformulando las relaciones entre humanos y máquinas. En palabras de (Tegmark, 2017) en "Life 3.0: Being Human in the Age of Artificial Intelligence", "la IA está aquí para quedarse, y su impacto en nuestras vidas será profundo y duradero" (p. 32). Esta cita habla del grado de integración que la IA tiene con la sociedad y cómo cambiará la forma en que vivimos y trabajamos.

No obstante, también hay un lado positivo que no se puede ignorar. (Harari, 2015) en "Homo Deus: A Brief History of Tomorrow" argumenta que "la IA tiene el potencial de mejorar la vida humana de formas que aún no podemos imaginar" (p. 403). Es evidente que la inteligencia artificial está redefiniendo los límites de lo que es posible y, con la regulación y la ética adecuadas, tiene el potencial de cambiar nuestro mundo para mejor. Sin lugar a dudas, la IA es

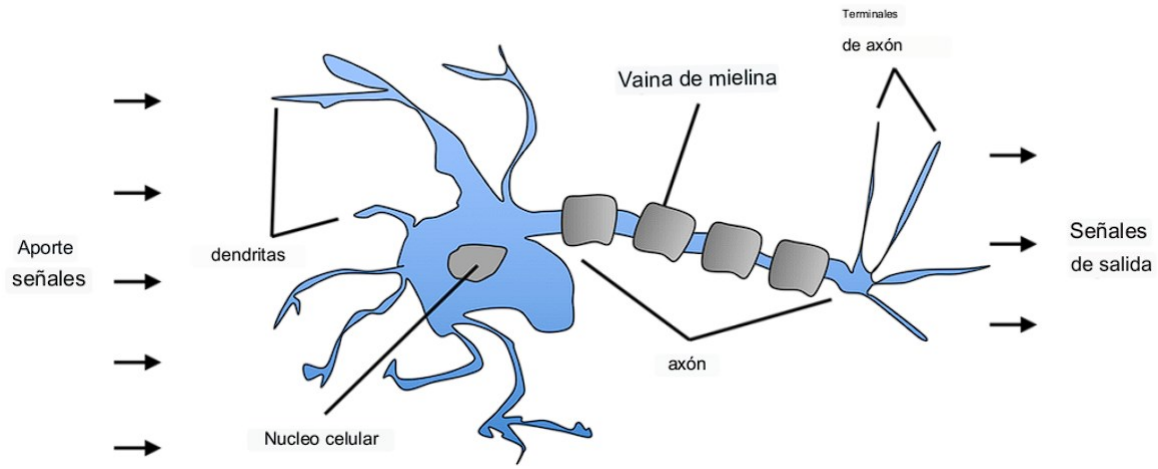
un área de estudio y desarrollo fascinante y compleja que promete cambiar nuestro mundo. A pesar de los posibles desafíos, la oportunidad que ofrece es demasiado grande como para ser ignorada.

Redes Neuronales

Las redes neuronales artificiales (ANN) son sistemas de cálculo que imitan el funcionamiento del cerebro humano. Hay varios tipos de ANN, cada una con su propio conjunto de características y usos.

Figura 1

Estructura de una neurona cerebral.



Nota. Neurona que procesa señales químicas y eléctricas. (Sebastian Raschka, 2022)

Las redes neuronales FeedForward, también conocidas como redes neuronales multicapa (MLP), son uno de los tipos más comunes. Según (Goodfellow I. B., 2016) en "Deep Learning", "las MLP son redes neuronales en las que la información se mueve en una sola dirección, de la entrada a la salida, sin bucles" (p. 189). Son útiles para tareas de clasificación y regresión.

Las redes neuronales recurrentes (RNN) son otro tipo de ANN, especialmente útiles para tratar con secuencias de datos. (Karpathy, 2015) en "Visualizing and Understanding Recurrent Networks" señalan que "las RNN tienen la capacidad única de trabajar con secuencias de datos y recordar información en el tiempo" (p. 24).

Las redes neuronales convolucionales (CNN) son otro tipo significativo de ANN, comúnmente utilizado en el procesamiento de imágenes y videos. (LeCun, 2015) en "Deep Learning" afirmaron que "las CNN han transformado el campo de la visión por computadora, consiguiendo resultados que superan con creces a los métodos tradicionales" (p. 436).

Finalmente, las redes adversariales generativas (GANs) representan una clase innovadora de ANN. (Heaton, 2016) en "Generative Adversarial Nets" argumentan que "las GANs pueden generar muestras nuevas y realistas a partir de datos de entrenamiento, una capacidad que ha revolucionado la generación de imágenes" (p. 267). Hay varios tipos de redes neuronales, cada una con sus propias fortalezas y aplicaciones. Desde MLP hasta RNN, CNN y GANs, estas estructuras de ANN permiten que la IA resuelva una amplia variedad de tareas, desde la clasificación hasta la generación de imágenes y la predicción de secuencias.

Red Neuronal Feed Forward (FFNN)

Las redes neuronales Feed Forward (FFNN), también conocidas como redes neuronales multicapa, son uno de los tipos más comunes de redes neuronales artificiales. Según (Goodfellow I. B., 2016) en "Deep Learning", "las FFNN son redes neuronales en las que la información se mueve en una sola dirección, de la entrada a la salida, sin bucles" (p. 189).

Las FFNN son útiles en una amplia gama de aplicaciones. Según (Haykin, 1998) en "Neural Networks: A Comprehensive Foundation", "las FFNN han demostrado su valor en tareas de reconocimiento de patrones, clasificación, procesamiento de señales y control" (p. 203).

En una FFNN, las capas de neuronas están totalmente conectadas a las capas adyacentes, y la información se propaga desde la capa de entrada a través de las capas ocultas hasta la capa de salida. (Rojas, 2013) en "Neural Networks: A Systematic Introduction", describe este proceso diciendo: "en una red neuronal feedforward, una unidad envía información sólo a aquellas unidades de las que es predecesora directa".

A pesar de su utilidad y eficacia, las FFNN tienen limitaciones. Una crítica común es su incapacidad para manejar secuencias de datos o patrones temporales. Según (Hochreiter, 1997) en "Long short-term memory", "las FFNN son estáticas y carecen de memoria de estado, lo que limita su uso en tareas donde la secuencia o el orden de los datos es importante" (p. 9).

En resumen, las redes neuronales Feed Forward son una herramienta esencial en el campo del aprendizaje automático, con numerosas aplicaciones en una amplia gama de disciplinas. A pesar de sus limitaciones, su capacidad para aprender a partir de los datos y generalizar a partir de experiencias anteriores las convierte en un componente vital de muchos sistemas de inteligencia artificial.

Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes (RNN) son una clase de redes neuronales artificiales diseñadas para manejar datos secuenciales. Según (Goodfellow I. B., 2016) en "Deep Learning", "las RNN se distinguen por su capacidad para operar sobre secuencias de vectores de entrada,

lo que las hace especialmente adecuadas para tareas relacionadas con el tiempo y el orden de los datos" (p. 375).

El elemento clave que diferencia a las RNN de otros tipos de redes neuronales es su memoria interna. Como señala (Graves, 2012) en "Supervised Sequence Labelling with Recurrent Neural Networks", "las RNN almacenan información de estados anteriores en su memoria interna, lo que les permite mantener un tipo de 'contexto' durante el procesamiento de la secuencia" (p. 17).

Sin embargo, las RNN tradicionales sufren el problema del desvanecimiento del gradiente, lo que dificulta su aprendizaje de dependencias a largo plazo. Según (Hochreiter, 1997) "la memoria a corto y largo plazo (LSTM) es una variante de las RNN que aborda este problema mediante el uso de unidades de memoria especializadas" (p. 8). Las LSTM han demostrado ser muy efectivas en diversas tareas de procesamiento del lenguaje natural, como la traducción automática y la generación de texto.

A pesar de sus fortalezas, las RNN también tienen limitaciones. (Chollet, Deep learning with Python. , 2021) en "Deep Learning with Python", señala que "las RNN son computacionalmente intensivas y pueden ser difíciles de escalar a secuencias largas o conjuntos de datos grandes" (p. 198).

En conclusión, las RNN representan un avance significativo en el procesamiento de datos secuenciales. Aunque enfrentan desafíos, variantes como las LSTM han ampliado su utilidad y aplicabilidad.

Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales (CNN) son una subcategoría crucial de las redes neuronales artificiales, que han demostrado un rendimiento excepcional en tareas relacionadas con el procesamiento de imágenes y videos. Según (LeCun, 2015) en "Deep Learning", "las CNN están diseñadas para procesar automáticamente y adaptarse a datos con una topología de cuadrícula, como una imagen" (p. 436).

Una característica esencial de las CNN es la convolución, que consiste en aplicar un filtro a la entrada para extraer características útiles. (Smirnov, 2014) en "Imagenet classification with deep convolutional neural networks" explican que "la convolución permite a las CNN aprender características invariables a la escala, la orientación y otros factores" (p. 1103).

Las CNN también se destacan por su capacidad para reducir la dimensionalidad de los datos a través de la operación de pooling. Según (Zeiler, 2014 September 6-12) en "Visualizing and Understanding Convolutional Networks", "el pooling agrupa las características espaciales, lo que permite que la CNN sea robusta a pequeñas variaciones y reduzca la cantidad de parámetros" (p. 2).

A pesar de las fortalezas de las CNN, también existen desafíos en su implementación. (Szegedy, 2015) en "Going deeper with convolutions", destacan que "las CNN profundas requieren grandes cantidades de datos y tiempo de entrenamiento, lo que puede ser problemático en algunos escenarios" (p. 6).

En resumen, las CNN han demostrado ser una herramienta valiosa para el procesamiento de imágenes y videos, ofreciendo características avanzadas como la convolución y el pooling. A pesar de sus desafíos, su capacidad para extraer características de alto nivel y reducir la

dimensionalidad de los datos, les confiere un lugar prominente en el campo del aprendizaje profundo.

Redes Adversarias Generativas (GANs)

Las Redes Adversariales Generativas (GANs) han emergido como una herramienta poderosa en el aprendizaje automático, transformando la forma en que los sistemas de inteligencia artificial generan datos nuevos y realistas. Según (Goodfellow I. P.-A.-F., 2014) en "Generative Adversarial Nets", las GANs son "modelos generativos que aprenden a crear datos que son similares a los de entrada, a través de un juego competitivo entre dos redes neuronales: un generador y un discriminador" (p. 267).

El generador se entrena para producir muestras realistas, mientras que el discriminador se entrena para distinguir las muestras generadas de las reales. Este juego competitivo da lugar a muestras generadas que son cada vez más difíciles de distinguir de las reales. (Goodfellow I. , 2016) en "NIPS 2016 Tutorial: Generative Adversarial Networks" argumenta que "este enfoque competitivo permite a las GANs generar datos que pueden ser indistinguibles de los datos de entrenamiento" (p. 7).

Las GANs han encontrado aplicaciones en una serie de campos. En la síntesis de imágenes, por ejemplo, (Radford A. M., 2015) en "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" afirman que "las GANs han demostrado ser particularmente eficaces para generar imágenes realistas" (p. 1). Asimismo, en la síntesis de voz, (Oord, 2016) en "WaveNet: A Generative Model for Raw Audio" sostienen que "las GANs han demostrado ser útiles para la generación de audio de alta calidad" (p. 2).

A pesar de sus avances, las GANs no están exentas de desafíos. El entrenamiento de GANs puede ser difícil y a menudo se encuentra con el problema del "modo colapso", en el que el generador produce muestras muy limitadas. (Arjovsky, 2017) en "Towards Principled Methods for Training Generative Adversarial Networks" señalan que "las dificultades en el entrenamiento y los problemas de convergencia siguen siendo obstáculos significativos para el uso más amplio de GANs" (p. 4).

Las GANs representan una innovación importante en el campo del aprendizaje automático y la generación de datos. A pesar de los desafíos, su capacidad para generar datos realistas las convierte en una herramienta poderosa para una variedad de aplicaciones.

Machine Learning

Es un subconjunto de Inteligencia Artificial que incluye técnicas estadísticas abstrusas que permiten a las máquinas mejorar en tareas con experiencia. La categoría incluye Deep Learning. El aprendizaje automático (Machine Learning, ML) es una rama de la Inteligencia Artificial (IA) que se centra en el diseño y la creación de algoritmos que permiten a una computadora aprender de los datos. De acuerdo con Samuel (Samuel, 1959) en su artículo pionero "Some Studies in Machine Learning Using the Game of Checkers", define el aprendizaje automático como "la capacidad de una máquina para mejorar su rendimiento basándose en experiencias anteriores" (p. 601).

El aprendizaje automático tiene el potencial de revolucionar diversas industrias. (Mitchell T. M., 1997) en "Machine Learning", declara que "el aprendizaje automático es un método fascinante para resolver problemas que son demasiado complejos para los humanos, pero no para las computadoras" (p. 2). Esto refleja cómo el ML puede aumentar la eficiencia y reducir la

carga de trabajo en muchos sectores, como la medicina, las finanzas y la tecnología de la información.

El aprendizaje automático también está provocando un cambio de paradigma en la forma en que hacemos ciencia. Según (Breiman, 2001) en "Statistical Modeling: The Two Cultures", "el aprendizaje automático está cambiando la forma en que se abordan los problemas de investigación, ya que está cambiando enfoque de las explicaciones a las predicciones" (p. 199). Este enfoque empírico ofrece nuevas perspectivas y descubrimientos en diversas disciplinas científicas.

Es fundamental tener en cuenta las posibles limitaciones y desafíos éticos que el ML presenta. En palabras de (Domingos, 2015) en "The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World", advierte que "la automatización a través del aprendizaje automático no está exenta de problemas y debe utilizarse de forma responsable" (p. 279). Esta cita resalta la importancia de utilizar ML de manera ética y responsable.

El aprendizaje automático es un campo innovador y en constante crecimiento que tiene el potencial de transformar la forma en que vivimos y trabajamos. Aunque existen desafíos éticos y técnicos, su potencial para mejorar la productividad y la eficiencia es indiscutible.

Deep Learning

Es un subconjunto de Machine Learning que se basa en redes neuronales artificiales con varias capas (conocidas como redes neuronales profundas). El "profundo" en "Deep Learning" no se refiere a cualquier tipo de comprensión más profunda lograda por el enfoque, sino a la utilización de redes neuronales con tres o más capas. Estas redes neuronales intentan simular el comportamiento del cerebro humano para "aprender" de grandes cantidades de datos. Aunque

una red neuronal con una sola capa puede aún hacer un trabajo aproximado para clasificar patrones lineales, las redes neuronales con varias capas hacen un mejor trabajo al clasificar patrones más complejos, y por ello se utilizan en tareas más avanzadas. Ejemplos de Deep Learning incluyen las redes neuronales convolucionales (CNNs), utilizadas en el procesamiento de imágenes, y las redes neuronales recurrentes (RNNs), utilizadas en el procesamiento del lenguaje natural.

El DL ha generado innovaciones significativas en varias áreas, como el procesamiento del lenguaje natural, el reconocimiento de voz y de imágenes, y la medicina. De acuerdo con (LeCun, 2015) en "Deep Learning", afirman que "El Deep Learning ha revolucionado la IA, y su impacto en la sociedad es cada vez más evidente" (p. 436).

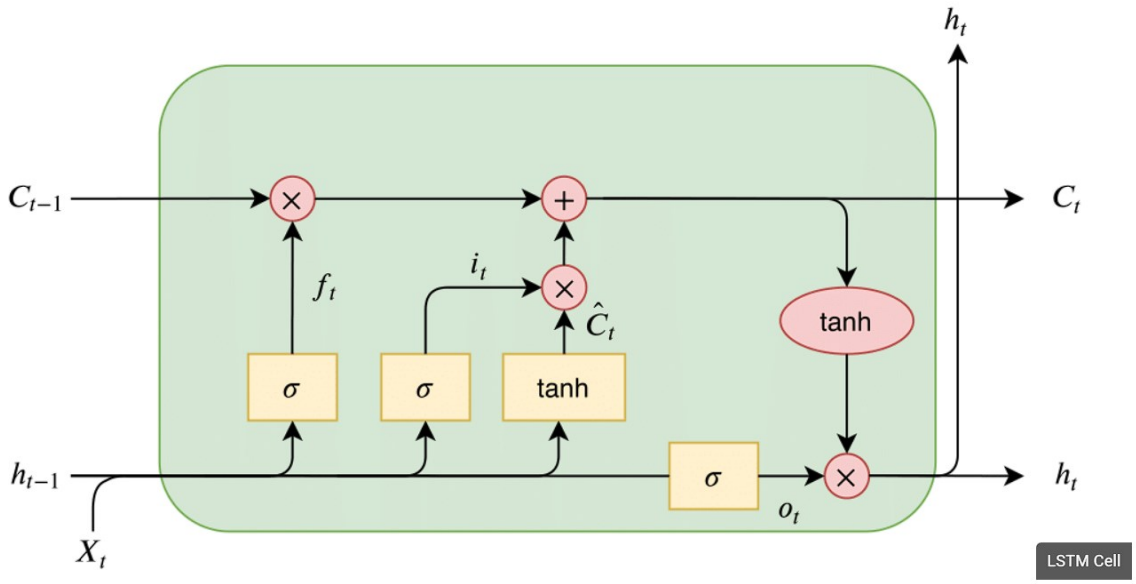
El potencial del Deep Learning para resolver problemas complejos es innegable. Como señala (Hassabis, 2017) en "Neuroscience-Inspired Artificial Intelligence", "El DeepLearning tiene el potencial de superar las limitaciones del aprendizaje automático y acercar a la humanidad un paso más a la Inteligencia Artificial general" (p. 476).**Error! Hyperlink reference not valid.**

Todo Deep Learning Es Machine Learning, pero no todo Machine Learning es Deep Learning. El Deep Learning requiere grandes cantidades de datos y poder de cálculo considerablemente mayor que el Machine Learning tradicional. Los modelos de Deep Learning tienden a ser más precisos que los de Machine Learning tradicional, especialmente cuando los datos son muy complejos o no estructurados.

LSTM (Long Short-Term Memory)

Figura 2

Celda LSTM



Nota. Estructura de una celda LSTM básica. (Ingolfsson, 2021). Contiene puerta de entrada, puerta de salida y puerta de olvido.

LSTM, que significa Long Short-Term Memory o Memoria de Largo Plazo a Corto Plazo, es un tipo de red neuronal recurrente (RNN) desarrollada para resolver problemas asociados con el aprendizaje de dependencias a largo plazo (Hochreiter, 1997). Las RNN son una clase de redes neuronales diseñadas para manejar datos secuenciales o temporales. Sin embargo, las RNN tradicionales tienen problemas para aprender cuando las secuencias son muy largas, ya que sufren de lo que se conoce como el problema del desvanecimiento del gradiente, donde la información útil se pierde durante el proceso de entrenamiento (Bengio, 1994).

Las LSTM abordan este problema a través de un diseño de red más complejo que incorpora una "puerta de olvido" y una "puerta de entrada", que ayudan a la red a decidir qué información olvidar y cuál retener. También incluyen una "puerta de salida" que determina qué

información del estado de la celda actual pasa a la siguiente etapa. Las LSTM pueden aprender dependencias a largo plazo, lo que las hace útiles para tareas como la traducción automática, el reconocimiento de voz y la generación de texto (Gers, 2000).

Metodología ágil de desarrollo SCRUM

Es metodología de desarrollo y uso de las buenas prácticas. La metodología Agile de programación SCRUM es un marco de trabajo para el desarrollo de software que se enfoca en la colaboración, la flexibilidad y la entrega continua de valor. En SCRUM, el equipo de desarrollo trabaja en sprints o iteraciones cortas y enfocadas de trabajo, que suelen durar de dos a cuatro semanas. Durante el sprint, el equipo se reúne diariamente en una reunión diaria de scrum, también conocida como stand-up, para mantenerse al tanto del progreso y resolver cualquier obstáculo que se presente. Al final del sprint, el equipo se reúne en una reunión de revisión del sprint para demostrar el trabajo realizado durante el sprint y obtener retroalimentación. Como el trabajo es realizado por una sola persona la retrospectiva será con su director de tesis.

La metodología ágil de desarrollo Scrum es un enfoque de gestión de proyectos que promueve la colaboración, la adaptabilidad y la mejora continua. De acuerdo con (Schwaber, 2001) en su libro "Agile Software Development with Scrum", "Scrum es un marco de trabajo para el desarrollo y mantenimiento de productos complejos" (p. 13). Scrum se centra en iteraciones cortas y regulares, conocidas como 'sprints', para permitir un control frecuente y ajustes rápidos al producto y al proceso. Según (Cohn, 2010) en "Succeeding with Agile: Software Development Using Scrum", sostiene que "la flexibilidad de Scrum permite a los equipos responder rápidamente a los cambios" (p. 54).

El enfoque Scrum también fomenta la comunicación y la colaboración eficientes. (Rubin, 2012) en "Essential Scrum: A Practical Guide to the Most Popular Agile Process", enfatiza que "Scrum fomenta la transparencia, la inspección y la adaptación para maximizar el valor" (p. 19). No obstante, Scrum no está exento de desafíos. Según (Sutherland, 2014) en "Scrum: The Art of Doing Twice the Work in Half the Time", advierte que "Scrum es simple de entender, pero difícil de dominar" (p. 72).

Scrum es una metodología ágil poderosa que puede impulsar la eficiencia y la productividad en el desarrollo de software. Aunque presenta desafíos, su capacidad para adaptarse y responder a los cambios la convierte en una herramienta invaluable para los equipos de desarrollo.

Ingeniería de Prompts

La ingeniería de prompts es un aspecto crítico del trabajo con modelos de lenguaje generativos, como GPT-4, y puede ser decisiva para obtener las respuestas deseadas. En el contexto de estos modelos, un 'prompt' se refiere a la entrada de texto que se proporciona al modelo para generar una respuesta. Según (Brown, 2020), "la habilidad para construir prompts adecuados es esencial para extraer el comportamiento deseado de un modelo de lenguaje" (p. 23).

Dado que los modelos de lenguaje generativos no tienen conocimiento del mundo más allá de su entrenamiento, la forma en que se enmarcan los prompts puede tener un impacto significativo en la calidad de las respuestas generadas. (Raffel, 2020) en "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", sostienen que "el diseño de prompts efectivos puede ser tan importante como el propio modelo" (p. 8).

Además, la ingeniería de prompts requiere una comprensión profunda tanto del dominio de la tarea como del funcionamiento del modelo. Según Bender, (Bender, 2021) en "On the Dangers of Stochastic Parrots: ¿Can Language Models Be Too Big?", advierten que "la elección de los prompts puede introducir sesgos no deseados en las respuestas del modelo" (p. 35). En conclusión, la ingeniería de prompts es un componente crítico para aprovechar al máximo los modelos de lenguaje generativos. Aunque presenta desafíos, ofrece un medio eficaz para guiar la generación de texto de estos modelos.

Lenguaje de programación Python

Python es un lenguaje de programación de alto nivel reconocido por su legibilidad y versatilidad. Según (Wentworth, 2012) en "How to Think Like a Computer Scientist: Learning with Python", Python es "un lenguaje poderoso y fácil de aprender que es útil tanto para principiantes como para expertos" (p. 1). Python ha ganado popularidad en diversas áreas como la ciencia de datos, el aprendizaje automático y el desarrollo web. (McKinney, 2022) en "Python for Data Analysis" sostiene que "Python ha emergido como una de las herramientas más importantes para los científicos de datos debido a su simplicidad y funcionalidad" (p. 4).

Además, Python también se destaca por su amplia comunidad de desarrolladores y su gran número de bibliotecas y frameworks. (Lutz, 2013) en "Learning Python" resalta que "la extensa biblioteca estándar de Python y los módulos de terceros disponibles hacen de Python una herramienta potente para cualquier tarea" (p. 23). No obstante, Python no está exento de limitaciones. Según (Beazley, 2009) en "Python Essential Reference", señala que "a pesar de sus fortalezas, Python puede no ser el más adecuado para aplicaciones que requieren un alto rendimiento en tiempo real" (p. 20).

Python es un lenguaje de programación potente y accesible que ha tenido un impacto significativo en el campo de la informática y más allá. A pesar de sus limitaciones, su simplicidad y versatilidad lo hacen ideal para una amplia gama de aplicaciones.

Herramientas

ChatGPT

Chatbot GPT (Generative Pretrained Transformer), o ChatGPT, es un modelo de lenguaje impulsado por la arquitectura GPT, que ha sido pre-entrenado en una gran cantidad de texto y luego afinado para tareas de generación de texto en formato de conversación. Según (Radford A. W., 2019) en "Language Models are Unsupervised Multitask Learners", ChatGPT es "un modelo de lenguaje generativo que puede producir respuestas coherentes y relevantes a los prompts de entrada" (p. 31).

ChatGPT se ha utilizado en una variedad de aplicaciones, desde la redacción de correos electrónicos hasta la tutoría en línea. Segun (Brown, 2020) en "Language Models are Few-Shot Learners", argumentan que "ChatGPT puede ser una herramienta útil para aumentar la productividad y la eficiencia en tareas de escritura" (p. 45). A pesar de su utilidad, los desafíos asociados con ChatGPT incluyen la generación de contenido no deseado o sesgado. (Bender, 2021)en "On the Dangers of Stochastic Parrots: ¿Can Language Models Be Too Big?", advierten que "los modelos de lenguaje como ChatGPT pueden reproducir y amplificar los sesgos existentes en los datos de entrenamiento" (p. 50).

Por último, el desarrollo de técnicas de moderación y mejora de la fiabilidad de los chatbots es un área activa de investigación. (Radford A. W., 2019) afirman que "las mejoras en la personalización y control de los modelos de lenguaje son esenciales para maximizar el valor

de las tecnologías como ChatGPT" (p. 56). ChatGPT es una tecnología de IA potente y versátil con muchas aplicaciones útiles, pero también plantea desafíos que deben ser abordados de manera responsable.

Anaconda Navigator

Anaconda Navigator es una interfaz gráfica de usuario incluida en la distribución de Anaconda, que es ampliamente utilizada para la ciencia de datos y la programación en Python y R. Según (McKinney, 2017) en "Python for Data Analysis", "Anaconda Navigator es un tablero para lanzar aplicaciones y gestionar los entornos conda" (p. 23).

Una de las ventajas clave de Anaconda Navigator es su simplicidad de uso. Proporciona un entorno intuitivo para el manejo de paquetes y entornos, permitiendo a los usuarios acceder y administrar una variedad de aplicaciones científicas y de análisis de datos.

Sin embargo, a pesar de su simplicidad, Anaconda Navigator también ofrece un gran potencial para usuarios avanzados. Según (Grus, 2019) en "Data Science from Scratch", "Anaconda Navigator puede manejar paquetes y dependencias complejas, lo que facilita la administración de proyectos de ciencia de datos" (p. 45). A pesar de sus fortalezas, Anaconda Navigator puede tener limitaciones para ciertos usuarios. (Müller, 2016) en "Introduction to Machine Learning with Python", señalan que "para usuarios que prefieren la línea de comandos, Anaconda Navigator puede ser menos atractivo" (p. 31).

Google Colab

Google Colab, o "Colaboratory", es un entorno de Jupyter Notebook gratuito que no requiere configuración y se ejecuta completamente en la nube. Permite a los usuarios escribir y

ejecutar código, guardar y compartir análisis, y acceder a poderosos recursos informáticos, todo de forma gratuita en su navegador (Bonner, 2019)

Como señala (Bonner, 2019), "Colab es un producto increíblemente útil para cualquiera que busque hacer experimentos de codificación rápidos y gratuitos en Python". Su fácil acceso y su naturaleza colaborativa le dan un valor único. Los usuarios pueden compartir sus Notebooks y colaborar en tiempo real, lo que resulta en un aprendizaje efectivo y productivo (Bonner, 2019)

Colab es notable por su capacidad para proporcionar un entorno de computación de alto rendimiento directamente en el navegador. "Google Colab ofrece gratuitamente la potencia de las GPUs de Google, lo que permite a los usuarios ejecutar tareas de aprendizaje automático de manera más eficiente" (Singh, 2022) No obstante, Colab tiene limitaciones. Explica que "el entorno de Google Colab se reinicia después de un período de inactividad, lo que puede resultar en la pérdida de datos si no se guardan adecuadamente". A pesar de este inconveniente, la utilidad de Colab en la investigación y el aprendizaje de ciencias de datos es innegable. (Bonner, 2019) (Singh, 2022)

Google Colab es una herramienta valiosa y accesible que facilita la experimentación y la colaboración en el aprendizaje automático y otras tareas de ciencias de datos. A pesar de algunas limitaciones, su facilidad de uso y acceso gratuito a los recursos de computación de alta potencia lo convierten en una opción atractiva para estudiantes y profesionales de la codificación por igual

Jupyter Notebook

Jupyter Notebook es una aplicación web de código abierto que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Esta

herramienta, como menciona (Kluyver, 2016), "ha ganado popularidad en los campos de la ciencia de datos, el aprendizaje automático y la inteligencia artificial, gracias a su interactividad y flexibilidad". (Gransbury, 2020) afirma que "los Jupyter Notebooks proporcionan un entorno interactivo donde los investigadores pueden experimentar con código, visualizar datos y documentar sus hallazgos todo en un solo lugar". Esta capacidad para explorar, documentar y compartir investigación ha hecho de Jupyter Notebook una herramienta valiosa para la enseñanza y la investigación académica.

Jupyter Notebook no está exento de desafíos. Conforme a: (Pohl, 2021), "los Jupyter Notebooks pueden volverse desordenados y difíciles de seguir si no se mantienen y organizan adecuadamente". Esto subraya la necesidad de adoptar buenas prácticas de codificación y organización al usar esta herramienta. A pesar de los desafíos, los Jupyter Notebooks siguen siendo una herramienta poderosa y versátil para la programación y el análisis de datos. Como señala Jupyter (Jupyter Project, 2022),"Jupyter Notebook es más que una aplicación; es una herramienta que está moldeando cómo trabajamos con datos y código".

Jupyter Notebook es una herramienta de programación de código abierto que ha revolucionado el campo de la ciencia de datos. A pesar de algunos desafíos, su capacidad para facilitar la experimentación interactiva, la visualización de datos y la documentación ha hecho de Jupyter Notebook una herramienta esencial para los investigadores y profesionales de la ciencia de datos.

Google Cloud

Google Cloud es una suite de servicios de computación en la nube que se ejecuta en la misma infraestructura que utiliza Google para sus productos orientados al usuario, como Google Search y YouTube. Rouse (2019) afirma que "Google Cloud es conocido por su infraestructura

confiable y escalable, que permite a las empresas hacer crecer sus soluciones tecnológicas sin preocuparse por la capacidad de hardware".

Uno de los aspectos más destacados de Google Cloud es su capacidad para ofrecer almacenamiento de datos a gran escala y de alta velocidad, lo que es esencial para el manejo de big data y el aprendizaje automático. Conforme a Miller (2020), "los servicios de Google Cloud, como BigQuery y Cloud Storage, permiten a las empresas manejar grandes cantidades de datos de manera eficiente y rentable".

Google Cloud ofrece una gama de servicios de computación en la nube que permiten a las empresas escalar sus soluciones tecnológicas, manejar grandes cantidades de datos y garantizar la seguridad de sus aplicaciones. A pesar de los desafíos que pueden surgir en el proceso de adopción, Google Cloud continúa siendo una plataforma robusta y confiable para la computación en la nube (Rouse, 2019; Miller, 2020; Smith, 2021).

Librerías

Pandas

Pandas es una biblioteca de software escrita para el lenguaje de programación Python que proporciona estructuras de datos y análisis de datos de alto rendimiento y fáciles de usar. (McKinney, 2017) describe a Pandas como "un punto de partida fundamental para el análisis de datos con Python" dada su versatilidad y eficiencia. Una de las características destacadas de Pandas es su capacidad para manejar y manipular datos tabulares. Como apunta (Reback, 2020) "Pandas permite una rápida manipulación y limpieza de datos tabulares estructurados a través de sus objetos DataFrame y Series".

Pandas es apreciado por su amplia gama de funciones que permiten la importación y exportación de datos en una multitud de formatos (Reback, 2020). Esto lo hace compatible con muchos tipos de proyectos y requisitos de análisis de datos. Pero, como indica (Cook, 2019), aunque "Pandas es una herramienta de análisis de datos increíblemente poderosa", puede ser abrumador para los principiantes dada su amplia funcionalidad. Esto destaca la importancia de una educación sólida en las bases de Python y Pandas.

Pandas es una biblioteca de Python esencial para el análisis de datos, destacada por su capacidad para manipular datos tabulares y su amplia funcionalidad. Aunque puede ser un desafío para los principiantes, su dominio es fundamental para cualquier aspirante a científico de datos.

Tensor Flow

TensorFlow es una biblioteca de software de código abierto desarrollada por Google Brain Team para el cálculo numérico utilizando gráficos de flujo de datos. Según, (Allaire, 2018) "TensorFlow es una plataforma integral y flexible para la investigación y producción de machine learning, con una amplia adopción en la academia y la industria". (Géron, 2019) destaca que "TensorFlow es especialmente conocido por sus capacidades en deep learning, pero también es altamente configurable, y se puede utilizar para una variedad de aplicaciones de machine learning". Esta flexibilidad lo convierte en una valiosa herramienta para los científicos de datos.

Una característica notable de TensorFlow es su capacidad para trabajar con múltiples CPUs y GPUs, lo que facilita el procesamiento de grandes volúmenes de datos y cálculos complejos (Allaire & Chollet 2018). Sin embargo, como señala (Howard, 2020), "aunque TensorFlow es poderoso, su complejidad puede ser un desafío para los principiantes en el aprendizaje automático". TensorFlow es una biblioteca de software poderosa y flexible que es

fundamental para muchas aplicaciones de machine learning. A pesar de su complejidad, la versatilidad de TensorFlow y su capacidad para manejar cálculos intensivos lo convierten en una herramienta inestimable en la ciencia de datos.

NumPy

NumPy, o Numerical Python, es una biblioteca para el lenguaje de programación Python que proporciona soporte para matrices grandes y multidimensionales, junto con una colección de funciones matemáticas para operar eficientemente con estas matrices (Walt, 2011). Según (Walt, 2011) "NumPy es la base sobre la cual se construye todo el ecosistema científico de Python".

Una de las principales fortalezas de NumPy es su eficiencia. Como señala (McKinney, 2017), "NumPy es muy eficiente con grandes cantidades de datos y tiene muchas funciones matemáticas incorporadas". Esta eficiencia es crucial para las aplicaciones de análisis de datos que a menudo requieren un manejo eficiente de grandes conjuntos de datos. Otra característica notable de NumPy es su capacidad para integrarse con otras bibliotecas de Python, (Harris, 2020) afirman que "NumPy sirve como un eficiente contenedor de datos multi-dimensionales para el trabajo general con datos numéricos, y es fundamental para otras bibliotecas de Python como Pandas y SciPy".

NumPy tiene su propia curva de aprendizaje. (McKinney, 2017) destaca que "NumPy tiene un API complicado que puede ser difícil de usar, especialmente para los principiantes", NumPy es una biblioteca esencial de Python para el manejo eficiente de datos numéricos y se ha convertido en una herramienta fundamental para el análisis de datos y la ciencia en Python.

Matplot

NumPy, o Numerical Python, es una biblioteca para el lenguaje de programación Python que proporciona soporte para matrices grandes y multidimensionales, junto con una colección de funciones matemáticas para operar eficientemente con estas matrices. Según (Walt, 2011), "NumPy es la base sobre la cual se construye todo el ecosistema científico de Python". Una de las principales fortalezas de NumPy es su eficiencia. Como señala (McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2017) "NumPy es muy eficiente con grandes cantidades de datos y tiene muchas funciones matemáticas incorporadas". Esta eficiencia es crucial para las aplicaciones de análisis de datos que a menudo requieren un manejo eficiente de grandes conjuntos de datos.

Otra característica notable de NumPy es su capacidad para integrarse con otras bibliotecas de Python. (Harris, 2020) afirman que "NumPy sirve como un eficiente contenedor de datos multidimensionales para el trabajo general con datos numéricos, y es fundamental para otras bibliotecas de Python como Pandas y SciPy". A pesar de su poder, NumPy tiene su propia curva de aprendizaje. (McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython., 2017) destaca que "NumPy tiene un API complicado que puede ser difícil de usar, especialmente para los principiantes". (Waskom, 2021)NumPy es una biblioteca esencial de Python para el manejo eficiente de datos numéricos y se ha convertido en una herramienta fundamental para el análisis de datos y la ciencia en Python.

Seaborn

Seaborn es una biblioteca de visualización de datos en Python que proporciona una interfaz de alto nivel para la creación de gráficos estadísticos atractivos e informativos. (Waskom, 2021) sostiene que "Seaborn mejora Matplotlib y hace que la visualización con Python sea más

fácil y atractiva". La biblioteca Seaborn se distingue por sus capacidades para visualizar modelos estadísticos complejos. Como afirma (VanderPlas, 2016), "Seaborn va más allá de Matplotlib y permite visualizar una serie de modelos estadísticos complejos". Esto hace que Seaborn sea una valiosa adición al conjunto de herramientas de un científico de datos.

Seaborn también es apreciada por su integración con la estructura de datos de Pandas. Según (McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2017), "Seaborn integra bien con los DataFrames de Pandas, lo que simplifica la manipulación de datos". Sin embargo, Seaborn no es siempre la herramienta ideal para visualizaciones personalizadas. (Müller, 2016) señalan que "aunque Seaborn ofrece una interfaz más amigable, Matplotlib sigue siendo necesario para personalizaciones finas". En resumen, Seaborn es una biblioteca de visualización de datos en Python que simplifica la creación de gráficos estadísticos, ofreciendo una interfaz más intuitiva que Matplotlib para ciertos tipos de visualizaciones.

Nltk

NLTK, o Natural Language Toolkit, es una biblioteca en Python que proporciona herramientas para el procesamiento de lenguaje natural (PLN). (Bird, 2009) lo describen como un "recurso invaluable para investigadores y estudiantes en áreas de la lingüística computacional y el procesamiento de lenguaje natural". Una de las características más destacadas de NLTK es su robustez. Como señala (Perkins, 2010), "NLTK incluye soporte para una amplia gama de tareas en PLN, desde la tokenización hasta el etiquetado de partes del discurso y la extracción de información".

Otra ventaja de NLTK es su conjunto de recursos lingüísticos. Según (Bird, 2009), "NLTK incluye una variedad de recursos lingüísticos, como corporal y léxicos, que son cruciales para la investigación en PLN". Sin embargo, NLTK no está exento de desafíos. Como advierte (Russell,

2020),"A pesar de su amplia gama de funcionalidades, NLTK puede ser complejo y abrumador para los principiantes". En resumen, NLTK es una biblioteca robusta y versátil para el procesamiento de lenguaje natural en Python, ofreciendo una amplia gama de herramientas y recursos que son esenciales en el campo de la lingüística computacional.

Regex

Las expresiones regulares (Regex) son una herramienta poderosa para manipular y analizar texto. (Friedman, 2006) describe las expresiones regulares como "una secuencia de caracteres que forma un patrón de búsqueda", lo cual permite identificar, buscar y manipular cadenas de texto específicas. Las regex son esenciales para el procesamiento de texto en numerosos lenguajes de programación, incluyendo Python. Según (Mitchell R. , 2015), "Las expresiones regulares son una característica omnipresente en Python y otros lenguajes de programación para la manipulación de texto".

Aunque son útiles, las regex pueden ser difíciles de entender y usar eficientemente. Como destaca (Forta, 2018), "Las expresiones regulares son muy poderosas, pero pueden ser difíciles de leer y comprender, especialmente cuando son complejas". Sin embargo, a pesar de su complejidad, las regex son fundamentales en diversas áreas. Como señala (Goodrich, 2013), "Las expresiones regulares se utilizan en campos como la informática, la bioinformática, la lingüística computacional y la minería de datos". En resumen, las expresiones regulares son una herramienta imprescindible para cualquier programador que trabaje con manipulación de texto, a pesar de su curva de aprendizaje inicial.

WordCloud

WordCloud, o nube de palabras, es una técnica de visualización de datos utilizada en el análisis de texto. Según (Kucher, 2018), una WordCloud es "una representación visual de las

palabras que componen un cuerpo de texto, en la que la importancia de cada palabra se refleja en su tamaño". Las WordClouds pueden ser utilizadas para resaltar las palabras más frecuentes en un conjunto de datos. Como señala (Kumar, 2021), "WordClouds son herramientas eficaces para mostrar palabras de alta frecuencia en un corpus de texto".

Las WordClouds tienen limitaciones. Khonji & Iraqi (2014) advierten que "WordClouds carecen de contexto y no pueden representar relaciones entre palabras". A pesar de estas limitaciones, las WordClouds siguen siendo una herramienta útil para el análisis exploratorio de texto. Según (Raghavan, 2020), "Las WordClouds son útiles para obtener una visión rápida de los temas predominantes en un conjunto de datos".

Keras

Keras es una biblioteca de Python de alto nivel para el aprendizaje profundo. Según (Chollet, Deep Learning with Python. , 2018), el creador de Keras, es "una interfaz de programación de aplicaciones diseñada para la experimentación humana". Una característica central de Keras es su simplicidad y facilidad de uso. (Brownlee, 2017) describe a Keras como "fácil de usar y poderoso, diseñado para permitir la experimentación rápida con redes neuronales profundas".

Además de su simplicidad, Keras también se integra bien con otras bibliotecas. (Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. , 2019) indica que "Keras se puede ejecutar en la parte superior de TensorFlow, permitiendo una fácil interacción con este motor de aprendizaje profundo". No obstante, Keras puede no ser la opción ideal para todos los casos. Como señala (Goodfellow I. B., 2016), "Keras es excelente para prototipos rápidos, pero puede no ser adecuado para casos de uso más complejos o personalizados".

SkLearn

Scikit-learn, comúnmente conocido como sklearn, es una biblioteca de Python utilizada para el aprendizaje automático. Según (Pedregosa, 2011), es "una biblioteca de software de aprendizaje automático de código abierto para Python, construida sobre NumPy, SciPy y Matplotlib". Sklearn es conocido por su variedad de algoritmos y su sencilla interfaz. (Müller, 2016) señalan que "sklearn incluye una serie de algoritmos de clasificación, regresión y clustering, y su interfaz es consistentemente sencilla".

Sklearn es muy útil para el preprocesamiento de datos. Según (Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems., 2019), "sklearn proporciona muchas funciones útiles para preprocesar los datos antes de alimentarlos a cualquier algoritmo de aprendizaje". Sklearn también ofrece herramientas para evaluar y mejorar los modelos. Como indica (Raschka, 2019), "sklearn proporciona funciones para ajustar los parámetros del modelo y evaluar su rendimiento".

PyTorch

PyTorch, desarrollado por Facebook's AI Research lab (FAIR), es un framework de programación de código abierto que facilita la implementación de redes neuronales profundas (Paszke A. G., 2019)(Paszke et al., 2019). Este framework ha ganado popularidad en la comunidad de aprendizaje automático gracias a su capacidad de ofrecer flexibilidad y dinamismo, particularmente útil para la investigación (Lecun, 2019).

Una de las características distintivas de PyTorch es su interfaz tensorial, similar a la de NumPy, pero con capacidades adicionales para operar en GPUs. Además, su motor autograd permite el cálculo automático de gradientes, lo que facilita la optimización de modelos (Paszke A. G., 2019).

La proliferación de noticias falsas ha llevado a la necesidad de herramientas y técnicas avanzadas para identificarlas de manera efectiva. Las redes neuronales profundas, implementadas a través de frameworks como PyTorch, han demostrado ser particularmente efectivas en tareas de procesamiento del lenguaje natural (PLN), incluida la detección de fake news (Shu, 2020). Diferentes arquitecturas, como las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN), han sido adaptadas para analizar el contenido de las noticias y determinar su autenticidad (Zhang, 2020)

La noticia falsa y los medios de comunicación.

Noticias Falsas

En el artículo publicado por (Kasperky, 2023), se evidencia que “Los sitios web que difunden noticias falsas a menudo proliferan imitando a fuentes de información legítimas para obtener confianza. Según investigaciones, las redes sociales son un medio por el cual las declaraciones incorrectas se esparcen velozmente, incluso a un ritmo más acelerado que las noticias verdaderas. La rápida difusión de las noticias falsas se debe, en gran medida, a que suelen estar formuladas para captar la atención y tocar las emociones de la gente, lo que a menudo se logra mediante la presentación de afirmaciones o relatos inusuales que desencadenan enojo o miedo.”

Ciber anzuelos

Los ciber anzuelos, o phishing, son técnicas de fraude en línea que buscan engañar a los usuarios para obtener información confidencial. (Jakobsson, 2007) definen el phishing como "la práctica de engañar a las personas para que compartan datos sensibles como contraseñas y números de tarjetas de crédito". El phishing es una forma de ciberataque que ha ganado

popularidad. Según (Schneider, 2015), "el phishing se ha convertido en una de las formas más comunes de ciberataque debido a su eficacia y facilidad de implementación".

Las técnicas de phishing pueden ser sofisticadas y variadas. Como advierte (Furnell, 2014), "Los atacantes de phishing están utilizando técnicas cada vez más sofisticadas para engañar a sus víctimas". A pesar de los esfuerzos para combatirlo, el phishing sigue siendo una amenaza importante. (Ghafir, 2018) señalan que "el phishing sigue siendo uno de los mayores desafíos en la seguridad cibernética".

Propaganda

La propaganda es una forma poderosa de comunicación que busca influir en las actitudes y comportamientos de las personas. Según (Jowett, 2019), la propaganda es "la manipulación deliberada, sistemática y planificada de las cogniciones, afectos, y el comportamiento para influir en las acciones de una población". La propaganda puede tomar muchas formas y se utiliza en una variedad de contextos. Como señala (Doob, 1950), "la propaganda puede ser tan sutil como la sugerencia o tan obvia como la publicidad directa".

Además, la propaganda se utiliza para lograr diversos objetivos. Según (Ellul, 1965), "la propaganda puede ser utilizada para fomentar el consenso social, manipular a las masas, o incluso para provocar el odio y el miedo". Sin embargo, el uso de la propaganda puede tener consecuencias negativas. (Stanley, 2015) advierte que "la propaganda puede socavar la democracia, fomentar la división y la desinformación".

Periodismo de mala calidad

El periodismo de mala calidad puede describirse como la práctica de informar que se caracteriza por la inexactitud, el sensacionalismo, la falta de rigor y la parcialidad. Según (Kovach,

2014), "el periodismo de mala calidad se caracteriza por la falta de verificación, la inexactitud y la falta de contexto".

Este tipo de periodismo puede tener un impacto negativo en la sociedad. Patterson (2013) afirma que "el periodismo de mala calidad puede socavar la confianza del público en los medios de comunicación y, por extensión, en la democracia misma", el periodismo de mala calidad puede ser impulsado por la economía de la atención. (McChesney, 2013) señala que "el sensacionalismo y la trivialidad pueden ser formas efectivas de atraer la atención del público en un mercado de medios de comunicación cada vez más competitivo". La propagación de las fake news es una forma de periodismo de mala calidad. (Tandoc, 2018) advierten que "la difusión de noticias falsas o engañosas es una manifestación de periodismo de mala calidad".

Encabezados engañosos

Los encabezados engañosos, también conocidos como clickbait, son una estrategia utilizada para atraer la atención de los lectores en línea. Según (Blom, 2015), el clickbait es "una táctica de contenido que promete o mal representa con el fin de atraer clics". Este tipo de práctica puede ser perjudicial para la calidad de la información. (Ifantidou, 2009) advierte que "los encabezados engañosos pueden afectar la comprensión de la noticia por parte del lector y erosionar la confianza en los medios".

El clickbait es utilizado como una táctica para difundir desinformación. (Chen, 2015) señalan que "los encabezados engañosos pueden ser utilizados para atraer a los lectores a noticias falsas o desinformación". Además, el clickbait puede ser un reflejo de las presiones económicas de la era digital. (Potthast, 2018) sugieren que "el clickbait es un síntoma de la economía de la atención en los medios digitales".

Contenido de impostores

El contenido de impostores se refiere a la creación y difusión de información falsa o engañosa por parte de individuos o grupos con intenciones maliciosas. Estos impostores buscan aprovecharse de la credibilidad y confianza del público para difundir desinformación. Según (Kasperky, 2023)

Sátira o parodia

El contenido de impostores se refiere a información presentada bajo identidades falsas, a menudo con el objetivo de desinformar o influir en las percepciones públicas. Wardle y Derakhshan (2017) describen el contenido impostor como "material generado por personas que se hacen pasar por entidades o personas de confianza con el objetivo de difundir información falsa".

Según (Kasperky, 2023) "Existe una tendencia por parte de algunos políticos destacados a descalificar las historias con las que no concuerdan, calificándolas como 'noticias falsas', a pesar de su comprobada veracidad. El término 'noticias falsas' abarca una amplia gama de interpretaciones, dependiendo de la persona que lo utilice, por lo que puede ser objeto de controversia. Además, señala que el gobierno del Reino Unido ha prohibido la etiqueta de 'Fake News', prefiriendo referirse a ellas como desinformación. Esta se refiere a relatos falsos o engañosos, con subyacentes intereses políticos o financieros. En contraposición, la información errónea también alude a historias falsas y engañosas, pero en este caso, diseñadas específicamente para engañar" (Kasperky, 2023)

Recursos de desarrollo de la aplicación

Tabla 1

Características PC de escritorio.

PC DE ESCRITORIO

Especificación.	**Características.**	**Marca**
Procesador	Ryzen 5 3600 3.6GHz 6 Núcleos	AMD
RAM	16GB 3600MHz no ECC	Kingston
HD	256GB SSD	Samsung
Mainboard	Asus TUF Gaming X570Plus	Asus
Case	Ninguno	------
Fuente	600W+ Bronce	EVGA
Sistema Operativo	Windows 10 Home PRO	Windows

Tabla 2

Características de laptop.

LAPTOP Lenovo Ideapad Flex 5

Especificación.	Característica.	Marca
Procesador	Ryzen 5700u, 1.8GHz-4GHz, 8 Núcleos	AMD
RAM	16GB 3600MHz Soldada	Samsung
HD	512GB SSD	Kingston
Mainboard	Lenovo Mainboard	Lenovo
Case	Laptop	------
Fuente	65W power brick	Lenovo
Sistema Operativo	Windows 10 Home	Windows

Nota. Esta tabla muestra los Materiales y Recursos (procesador y gpu etc.) que se utilizaron.

Materiales y programas necesarios para desarrollo el proyecto.

- PC de escritorio.

- Laptop
- Python 3.9
- Cuenta Google para Google Colab
- Cuenta Google Cloud
- Cuenta en Amazon AWS
- Anaconda Navigator
- Nvidia Cuda Toolkit
- TensorFlow
- PyCharm

Capítulo III

Desarrollo

Planificación del Desarrollo de la Herramienta Web de Detección de Fake News

En esta etapa de desarrollo se definió las partes principales que cumplan los objetivos específicos del proyecto, las cuales se llegará al entrenar el modelo final de la aplicación. Se elaboró un esquema que demuestra aquello **Figura 3**. A continuación, se detallan cada uno de los pasos.

Recolección de Datos

Obtener y recolectar información de Fake News y datasets libres.

Actividades a Realizar:

- Investigar y seleccionar fuentes confiables de datasets relacionados con Fake News.
- Descargar o acceder a la información de los datasets seleccionados.
- Traducir y transformar los datos obtenidos al idioma español, asegurando que la información no pierda su contexto original.
- Recopilar noticias de Ecuador.
- Recopilar noticias generadas con API de Chat-GPT 3.5

Entrenamiento del Modelo

Diseñar el motor de detección de Fake News utilizando Inteligencia Artificial.

Actividades a Realizar:

- Efectuar una limpieza y preprocesamiento de los datos recolectados para garantizar su calidad.
- Dividir el dataset en conjuntos de entrenamiento, validación y prueba.
- Seleccionar y definir el modelo de Machine Learning más adecuado para la tarea de detección.
- Entrenar el modelo utilizando el conjunto de entrenamiento.
- Evaluar la precisión y eficacia del modelo con el conjunto de validación y ajustar parámetros según sea necesario.
- Validar el desempeño del modelo con el conjunto de prueba.

Despliegue de la Aplicación

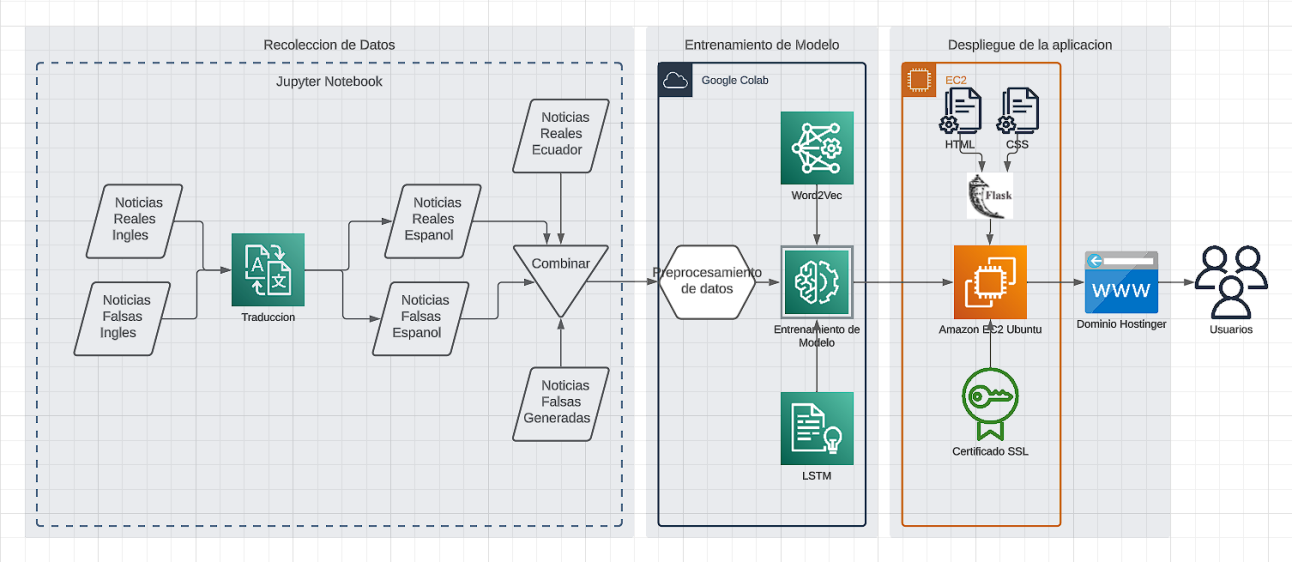
Desarrollar la aplicación web y probar la herramienta con datos externos de redes sociales.

Actividades a Realizar:

- Diseñar y estructurar la interfaz de usuario para la herramienta web, asegurando que sea intuitiva y fácil de usar.
- Integrar el modelo entrenado de Machine Learning en la aplicación WEB.
- Realizar pruebas funcionales de la aplicación para garantizar su correcto funcionamiento.
- Desplegar la aplicación en un servidor EC2 de Amazon adecuado para asegurar su disponibilidad.
- Probar la herramienta utilizando datos externos de redes sociales, evaluando la precisión y eficacia de la detección en escenarios reales.

Figura 3

Diagrama del sistema desarrollado.



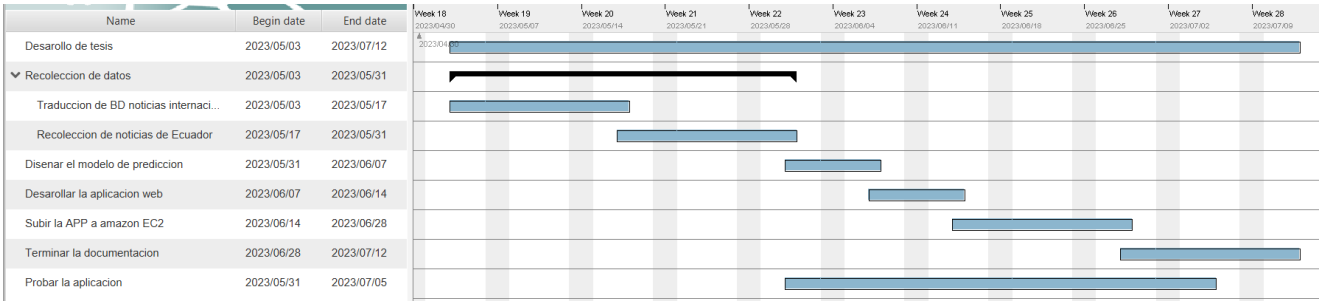
Nota. Esta imagen muestra el desarrollo del proyecto. El cual se basa en tres fases principales que es la recolección de datos, entrenamiento de modelo y despliegue de la aplicación.

Cronograma de desarrollo.

El siguiente cronograma realizado en Gantt Project se puede visualizar cada una de las etapas a seguir. Tal como se detalla en la **Figura 4**.

Figura 4

Gráfico de Cronograma

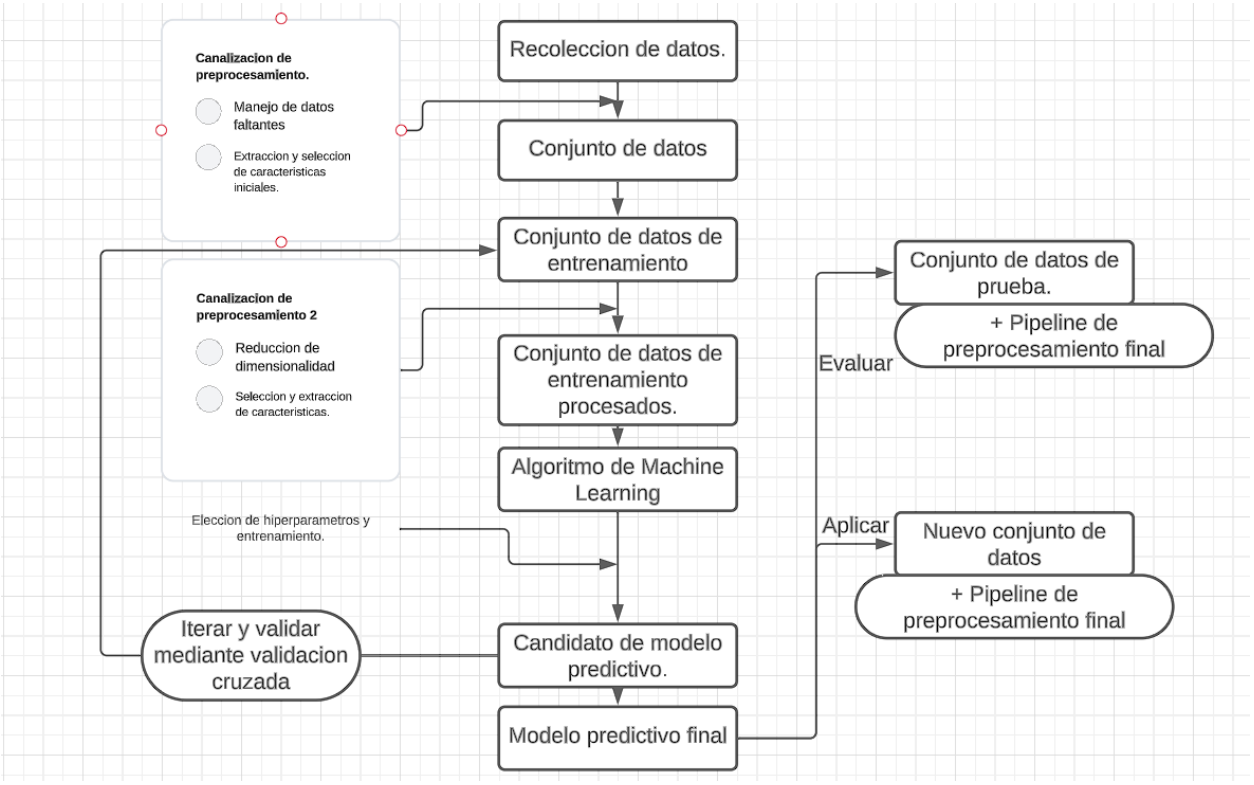


Name	Begin date	End date
Desarollo de tesis	2023/05/03	2023/07/12
Recoleccion de datos	2023/05/03	2023/05/31
Traduccion de BD noticias internaci...	2023/05/03	2023/05/17
Recoleccion de noticias de Ecuador	2023/05/17	2023/05/31
Disenar el modelo de prediccion	2023/05/31	2023/06/07
Desarollar la aplicacion web	2023/06/07	2023/06/14
Subir la APP a amazon EC2	2023/06/14	2023/06/28
Terminar la documentacion	2023/06/28	2023/07/12
Probar la aplicacion	2023/05/31	2023/07/05

Nota. La siguiente imagen muestra Cronograma de desarrollo del proyecto.

Figura 5

Flujograma de construcción de un modelo básico de machine Learning.



Nota. Flujograma básico de desarrollo de un sistema de predicción. Según el libro (Sebastian Raschka, 2022)

Instalación y configuración del ambiente de desarrollo

Para realizar el proyecto de detección de Fake News en Ecuador primero se instaló en ambos equipos tanto de escritorio como portátil, el navegador Anaconda Navigator de la página. Después de descargar se procedió con los pasos de instalación que incluye los siguientes programas: PyCharm y Jupyter Notebook.

Dentro de Anaconda Navigator se ingresó a “Environments” se procedió a abrir la consola y en ella se creó un nuevo entorno se agregó un nombre para poder encontrarlo.

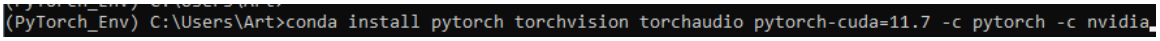
Los entornos de ejecución son muy importantes ya que todas las funcionalidades y librerías dentro de Anaconda estarán aislados de los demás componentes y posibles conflictos que se tengan. En caso de que la instalación tenga algún problema en el momento de ejecutar los scripts, se puede eliminar el entorno y volver a instalarlos de nuevo.

Luego se procedió con la instalación de PyTorch e debe de observar que se tiene que instalar CUDA 11.7 y usar el comando en consola de Conda para poder instalarlo correctamente. Para instalar CUDA 11.7 también es necesario instalar los drivers de equipo correspondiente.

El comando que aparece a continuación se debe de ejecutarlo en la línea de comandos de entorno de Anaconda. El aquel comando sirve para instalar PyTorch este servirá para luego traducir las noticias a idioma español **Figura 6**.

Figura 6

Comandos de entorno



Nota. El comando debe de copiarlo y pegarlo en la línea de comandos de entorno de Anaconda

Para las traducciones con ML locales es importante instalar desde la página oficial Nvidia Drivers para PC de escritorio con la tarjeta gráfica NVIDIA. Desde la página en este caso se instaló drivers para RTX 2060 Super del equipo.

Luego se procedió con la instalación de CUDA Toolkit de la página. Es muy importante que la versión sea versión 11.7 ya que esto permitirá no tener conflictos con PyTorch drivers y librerías.

Obtención y colección de información de Fake News y datasets libres

Se recopilaron noticias de la página de Kaggle (Kaggle, 2017) y se guardaron los archivos de las noticias True.csv y Fake.csv (Ahmed H T. I., 2018) (Ahmed H T. I., 2017) En el disco duro interno del proyecto. Estos archivos serán la base de entrenamiento.

En estos datasets están recolectadas varias noticias y fueron comprobadas por varias fuentes por varios periodistas. El juicio de valor de, si la noticia es verdadera o falsa, lo establecieron los periodistas por lo tanto es la información más fiable y acertada que se logró obtener.

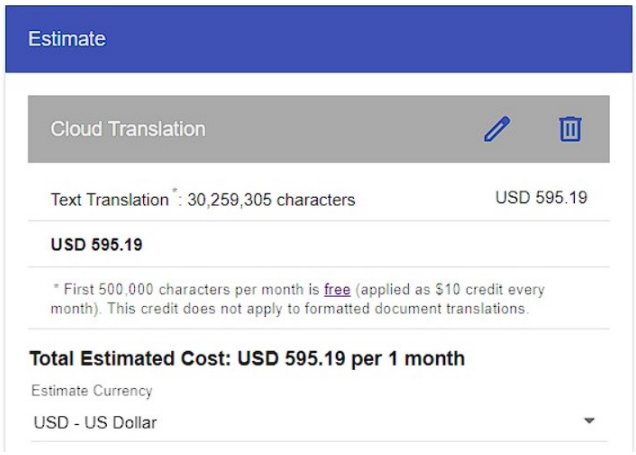
Transformación y Traducción de los datos

Para realizar este proyecto se necesita transformar los datos a la realidad de Ecuador por lo tanto se buscó una manera adecuada de realizarlo. Dicho de otra manera, estos datos son verificados, pero están en idioma inglés. Se procedió encontrar la manera adecuada para traducirlos.

Para empezar, se revisó el servicio de traducción de Google Cloud el cual permite realizar un script para traducir los datasets o archivo de datos recopilados a español **Figura 7**. Pero al revisar los precios y los términos de uso de Google Cloud se encontró con el inconveniente de que los costos pueden ser muy elevados.

Figura *7*

Google Cloud Translation Calculator.

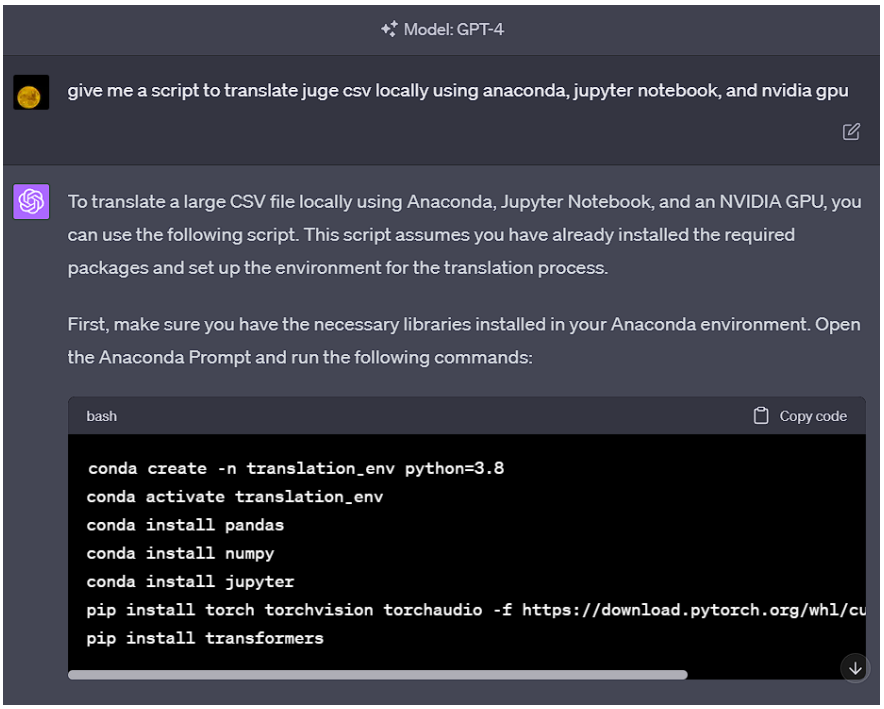


Nota. Revisión de costos de traducción de Google Cloud.

Se ejecutó el navegador Anaconda a continuación Jupyter Notebook. Luego se realizó el prompt a CHAT-GPT para generar script base para realizar la traducción de datasets de las noticias recopiladas **Figura *8***. Luego se procedió a crear un nuevo entorno virtual en Anaconda que se llamará translation_env. Se instaló: Pandas, NumPy, Jupyter, PyTorch y Transformers. Esto permitió tener código base para realizar la traducción de texto.

Figura 8

Script base de GPT-4 para traducción.



Nota. prompt a CHAT-GPT para (https://chat.openai.com/?model=gpt-4) generar script

Figura *9*

Importación de librerías

```
import pandas as pd
from transformers import MarianMTModel, MarianTokenizer
import torch
```

Nota. El código base es generado por Chat-GPT.

Figura 10

Código de traducción de texto

```
# Change 'src_lang' and 'tgt_lang' to the desired source and target language
src_lang = 'en'
tgt_lang = 'fr'
model_name = f'Helsinki-NLP/opus-mt-{src_lang}-{tgt_lang}'
tokenizer = MarianTokenizer.from_pretrained(model_name)
model = MarianMTModel.from_pretrained(model_name).to('cuda')
```

Nota. Importación de MarianMTModel (huggingface, 2023) para procesar con CUDA.

Figura 11

Función que traduce el texto

```
def translate(text, model, tokenizer, max_length=512):
    inputs = tokenizer(text, return_tensors="pt", max_length=max_length, tru
    inputs = {k: v.to('cuda') for k, v in inputs.items()}
    with torch.no_grad():
        outputs = model.generate(**inputs)
    translated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return translated_text
```

Nota. Función translate.

Luego de obtener el código base y guardarlo en CSV se lo modifico para crear el código final que permita leer cada una de las variables de CSV de las noticias y posteriormente traducirlos tal como lo muestra la ***Figura 13***

Figura 12

Código de Guardado de traducción en CSV.

```
# Read the CSV file
csv_path = 'path/to/your/input.csv'
df = pd.read_csv(csv_path)

# Choose the column to translate
column_to_translate = 'column_name'

# Apply the translation function to the chosen column
df[f'translated_{column_to_translate}'] = df[column_to_translate].apply(lamb

# Save the translated CSV
output_csv_path = 'path/to/your/output.csv'
df.to_csv(output_csv_path, index=False)
```

Nota. Guardado de los datos en CSV

Figura *13*

Codigo de función de traducción.

```
def translate_csv(file_path, output_path, source_language, target_language,
    start = timer()
    model_name = f'Helsinki-NLP/opus-mt-{source_language}-{target_language}'
    model = MarianMTModel.from_pretrained(model_name).to(device)
    tokenizer = MarianTokenizer.from_pretrained(model_name)

    df = pd.read_csv(file_path)
    translated_rows = 0

    for index, row in df.iterrows():
        for col in df.columns:
            if pd.notnull(row[col]):
                row[col] = translate_text(str(row[col]), model, tokenizer)
        translated_rows += 1

        # Save the DataFrame to the output file every save_interval rows
        if translated_rows % save_interval == 0:
            df.to_csv(output_path, index=False)
            print(f"Progress: {translated_rows}/{len(df)} rows translated")

    df.to_csv(output_path, index=False)
    print("total time for translation is", timer()-start)
```

Nota. El código muestra el código final de traducción.

Figura 14

Código de traducción y guardado de CSV.

```
if __name__ == '__main__':
    input_file = 'splited_csv/output_5000.csv'
    output_file = 'splited_csv_translated/translated_5000.csv'
    source_language = 'en'
    target_language = 'es'

    translate_csv(input_file, output_file, source_language, target_language)
```

Nota. El código muestra cómo se traduce en bloques de 5000 filas.

Esta traducción de las noticias Fake.csv y True.csv de inglés se realizó en el computador de escritorio ininterrumpidamente utilizando tarjeta gráfica y el tiempo de traducción fue aproximadamente 6 días.

Al traducir los datasets correctamente a español se procede a implementar el modelo de entrenamiento utilizando algoritmo LSTM dentro de Google Colab.

Para realizarlo se abrió entorno virtual Google Colab (Google, n.d.) en el navegador web y se procedió a desarrollar el código.

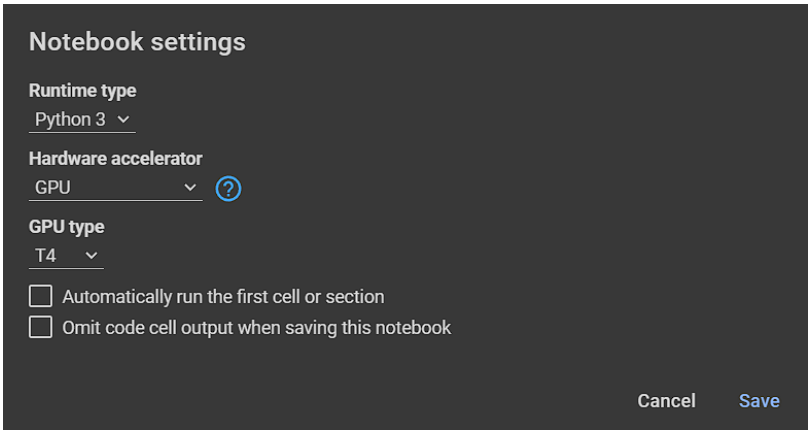
Dentro de Google Colab se escribió cada celda del código y posteriormente se ejecutó cada una de dichas celdas.

Para empezar, se debió de activar el entorno virtual el que permitió acceder a características de procesamiento paralelo para aumentar drásticamente la potencia de procesamiento **Figura 15**. Gratuitamente Colab permite usar por un corto tiempo el GPU TeslaT4 con 16GB en VRAM. Sin embargo, para hacer varios ajustes y pruebas y modificaciones en modelo se puede comprar poder de procesamiento desde la misma página que permite usar el poder de cómputo (compute units) y usar hasta 70GB en RAM y 42 VRAM con GPU A100.

Se pudo ejecutar las siguientes celdas para observar las características del mismo y **Figura *16***.

Figura *15*

Configuración de entorno de ejecución con GPU.



Nota. Antes de proceder con el desarrollo es importante activar GPU en entorno de desarrollo de Colab.

Figura 16

Características de GPU al ejecutar el código especificado.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   48C    P8    11W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Nota. Esto muestra características del GPU.

Se procedió a instalar la librería de TensorFlow **Figura *17***. es la que contiene el modelo de entrenamiento de deep learning de LSTM **Figura 19**. También se procedió importar las demás librerías tales como NumPy, Matplot, Seaborn entre otros **Figura 18**. Estos serán importantes para el resto de sección de entrenamiento.

Figura 17

Celda de ejecución.

```
import tensorflow as tf
tf.test.gpu_device_name()
```

Nota. Se importo TensorFlow

Figura *18*

Celda de ejecución.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import re
from wordcloud import WordCloud
```

Nota. Luego se procedió a importar las librerías ejecutando la siguiente celda.

Figura *19*

Celda de ejecución.

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM, Conv1D, MaxPool1D
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
```

Nota. Importación de herramientas de Machine Learning y Deep Learning.

Entrenamiento de modelo Versión 1.

Carga de los datos.

Primero se cargó el dataset de noticias falsas **Figura *20***. Esto permite acceder a ellos como el objeto fake.

Luego, se procedió a inspeccionarlos para detectar la estructura específica de las filas, con la función *head ()* se obtuvo las primeras filas del archivo y con el comando fake.columns se obtuvieron las cabeceras, en donde la columna “subject” se usó para determinar qué tipos de noticias estaban incluidas en el set de datos **Figura *21***. Se procedió a realizar el diagrama de barras de cada uno de los tipos de noticias. Se puede observar que el dataset contiene: 9050 noticias, de política 6841, noticias de política izquierda 4459, noticias de gobierno 1570, Noticias de USA 783, de oriente medio 778. Por lo tanto, se puede observar que al entrenar el modelo es posible que las noticias políticas tengan una mayor probabilidad de detección.

Figura 20

Celda de ejecución.

```
1 fake = pd.read_csv('/content/drive/MyDrive/TESIS FAKE NEWS/csv_fake_news_spanish
```

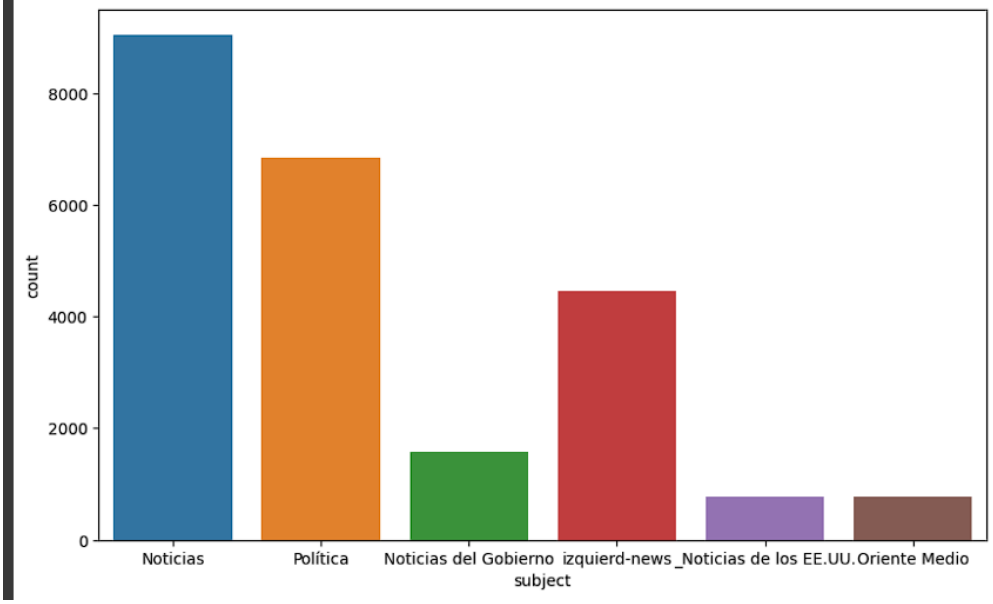
Nota. En la variable fake se cargó dataset de noticias falsas.

Figura 21

Tipos de noticas del dataset.

```
1 plt.figure(figsize=(10, 6))
2 sns.countplot(x = 'subject', data=fake)
```

```
<Axes: xlabel='subject', ylabel='count'>
```



Nota. El dataset muestra tipo de noticias falsas traducidas.

Posteriormente convertimos en columna text a listas esto es importante para poder luego introducir la lista al modelo de ML **Figura *22***. Después se hace una limpieza de valores que contienen NaN en dichas listas para reemplazarlos por strings vacíos. **Figura *23***. lo mismo que vacío eso permite que sea interpretado de adecuada manera y no genere errores en el entrenamiento del modelo. Luego se procedió a unir dichas listas agregando un espacio al final **Figura *24***. Realizamos los mismos pasos para noticias verdaderas **Figura *25***.

Figura 22

Celda de ejecución.

```
#convert all elements in the fake['text'] list to strings
text = ' '.join(str(item) for item in fake['text'].tolist())
```

Nota. Para poder trabajar con datos se tuvo que convertirlos en listas de Python.

Figura 23

Celda de ejecución.

```
#replace all NaN values with empty string
fake['text'] = fake['text'].fillna('')
```

Nota. Hay algunas celdas que estaban vacías y se marcan como NaN la siguiente línea de código permite hacer un barrido en filas de dataset y marcarlos como ‘ ‘.

Figura 24

Celda de ejecución.

```
text = ' '.join(fake['text'].tolist())
```

Nota. Se procedió a unir las listas que creamos en lista de listas. Para eso se ejecutó el siguiente código.

Figura 25

Celda de ejecución.

```
real = pd.read_csv('/content/drive/MyDrive/TESIS FAKE NEWS/Tesis/PROJECT/datasets/Merged news with Ecuado
```

Nota. Luego se procedió a cargar las noticias marcadas como verdaderas y realizar los mismos pasos que se realizaron con noticias falsas.

Los pasos a continuación se utilizó la librería WordCloud la cual crea una imagen visual que permitió observar las palabras más frecuentes en el dataset. Luego de aproximadamente 1 minuto de procesamiento podemos ver las palabras más frecuentes que son “de, que, e, Donald,

los"; como se puede observar hay bastantes palabras propias de la lengua como los conectores, "de, la, el, que, etc.". Los cuales pueden ocupar espacio adicional y tiempo de entrenamiento de modelo y no representan un valor importante para que el modelo ML aporte conclusiones significativas para clasificar las noticias falsas. Por eso se realizó lematización de las palabras **Figura *36***

Análisis de los datos.

Al desarrollar el siguiente código, permite enumerar publicadores desconocidos **Figura *26***, que se encuentran en columna texto de Real.csv. Se procedió a usar la función "split" que permite separar texto en este caso antes de guion medio **Figura *28***. Y posteriormente ubicarlo en una columna aparte. Como resultado final analizando la tabla anterior "ROMA(Reuters) – El Movimiento 5 estrellas… tiene que quedar separado en ROMA (Reuters) y El movimiento 5 estrellas…". Este código se complementa con el código escrito más abajo **Figura *27***.

Figura 26

Celda de ejecución.

```
1 len(unknown_publishers)
```

Nota. Al escribir la siguiente función se puede ver la cantidad de noticias que tienen autores desconocidos.

Figura 27

Salida de Celda de ejecución.

```
1 real.iloc[unknown_publishers]
```

	title	text	subject	date
7	Factbox: Trump en Twitter (Dic 29) - Cálculo d...	Las siguientes declaraciones fueron publicadas...	PoliticsNews	29 de diciembre de 2017
8	Trump en Twitter (Dic 28) - Calendario global	Las siguientes declaraciones fueron publicadas...	PoliticsNews	29 de diciembre de 2017
12	Factbox: Trump en Twitter (Dic 28) - Feria de ...	Las siguientes declaraciones fueron publicadas...	PoliticsNews	28 de diciembre de 2017
13	Trump en Twitter (Dic 27) - Trump, Irak, Siria	Las siguientes declaraciones fueron publicadas...	PoliticsNews	28 de diciembre de 2017
17	Trump en Twitter (Dic 26) - Hillary Clinton, I...	Las siguientes declaraciones fueron publicadas...	PoliticsNews	26 de diciembre de 2017

Nota. Podemos verla organización de los datos.

El código realizado en la **Figura *28*** realiza un barrido de todas las filas que estén marcadas como unknown_publisher agregamos “Unknown” a cada una de ellas si no es el caso buscamos el carácter de separación “ - ” y lo separamos en 2 partes el primero se agrega a columna “Publisher” la segunda a “tmp_text”. En **Figura *29*** se agregaron dichas columnas en noticias reales y se le pasaron las variables respectivas. En este caso el único que nos interesa es “tmp_text” que ahora formara parte de “text” de noticias reales que se agregara como variable a entrenar en ML.

Figura 28

Celda de ejecución.

```
publisher = []
tmp_text = []

for index, row in enumerate(real.text.values):
  if index in unknown_publishers:
    tmp_text.append(row)
    publisher.append('Unknown')

  else:
    record=row.split('-',maxsplit=1)
    publisher.append(record[0].strip())
    tmp_text.append(record[1].strip())
```

Nota. El siguiente código permitió la separación de los datos que están en text y su separación en 2 partes.

Figura 29

Celda de ejecución.

```
real['publisher']=publisher
real['text'] = tmp_text
```

Nota. Se ingresaron las columnas de publisher y text en noticias reales.

El siguiente paso permite cambiar el texto en minúsculas. Este paso es importante ya que se tiene que normalizar la entrada de texto a predecir. **Figura *30***. Posteriormente se agregaron columnas de noticias reales como 1 y noticias falsas como 0 **Figura *31***. Luego se crearon dos objetos que se llamaran real y fake con sus columnas de texto a entrenar y clase de las mismas **Figura *32***.

Figura 30

Celda de ejecución.

```
real['text'] = real['text'].apply(lambda x: str(x).lower())
fake['text'] = fake['text'].apply(lambda x: str(x).lower())
```

Nota. En noticias falsas se procedió a eliminar las filas de noticas que posiblemente pueden estar vacías.

Figura 31

Celda de ejecución.

```
real['class'] = 1
fake['class'] = 0
```

Nota. Se definió columna ‘class’ en la cual 0 es noticia marcada como falsa. Y 1 es noticia marcada como verdadera. Esto permite entrenar el modelo en base a la marca clase.

Figura 32

Celda de ejecución.

```
real = real[['text','class']]

fake = fake[['text','class']]
```

Nota. Se armaron las columnas que sirvieron para entrenamiento del modelo solo se usó texto como la noticia y clase define si la noticia es verdadera o falsa.

Luego se inspecciono el código en **Figura *33*** se observa que ahora noticias reales tienen columnas de texto y su clase respectiva. Después se creó objeto “data” y se le agrega “fake” y “real”. Con esto el objeto “data” ahora contendrá todos las noticias verdaderas y falsas y su respectiva clase. **Figura *34***

Figura 33

Celda de código y ejecución.

Nota. Se revisaron las tablas finales de entrenamiento.

Figura 34

Celda de ejecución.

```
1 data = real.append(fake, ignore_index=True)
```

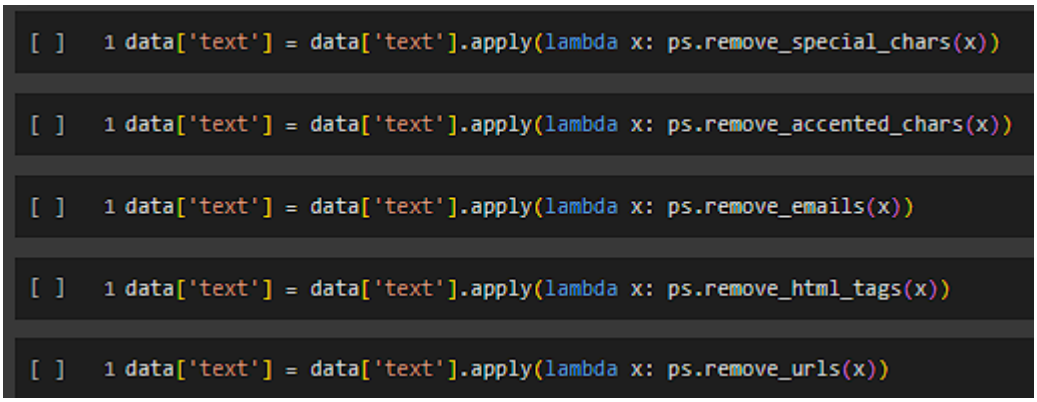
En variable ‘data’ se guardan noticias verderas y falsas con sus respectivas clases.

Preprocesamiento de datos.

En esta fase se preprocesan los datos esto significa limpiarlos de todos los caracteres, acentuaciones, emails, tags HTML, URLs **Figura *35***. También se lematizo al final cada fila de la columna “text”. **Figura *36***

Figura 35

Celda de ejecución.



Nota. Para preprocesar los datos es recomendable quitar caracteres especiales acentuaciones, emails y links HTML o tags.

Figura 36

Celda de ejecución.

```
import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud,STOPWORDS
nltk.download('stopwords')


stop = stopwords.words('spanish')
data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
data.head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

	text	class
0	jefe oficina conservadora republicana congreso...	1
1	personas transgénero podrán alistarse primera ...	1
2	investigación especial vinos rusia campaña ele...	1
3	george papadopoulos, asesor campaña trump, dij...	1
4	presidente donald trump pidió servicio postal ...	1

Nota. El siguiente bloque quita las “stopwords” o traducido como palabras de parada. Las palabras más frecuentes en español como: la, de, el, ellos, etc. Son palabras que pueden perjudicar al entrenamiento. (What are stop words?, n.d.)

Vectorización de las palabras.

La biblioteca gensim es una biblioteca de Python útil para realizar modelado de tópicos y análisis de similitud semántica en documentos. Un elemento principal de gensim es que fue diseñado para trabajar con grandes colecciones de texto, por lo que puede manejar eficientemente conjuntos de datos muy grandes. Figura

La línea de código import gensim carga esta biblioteca en tu entorno Python actual para su uso posterior. A través de gensim, se puede acceder a modelos y métodos de procesamiento de lenguaje natural, incluyendo el modelo Word2Vec.

Figura 37

Celda de ejecución.

Nota. Se importó la librería ‘gensim’ es una librería eficiente para modelamiento de texto no supervisado. Es comúnmente usado en detección de similitudes en un documento.

Se prepararon los datos en lista de listas “X” e “y”, como datos a entrenar. Esto permitirá ingresar al modelo de ML dichos datos a entrenar.

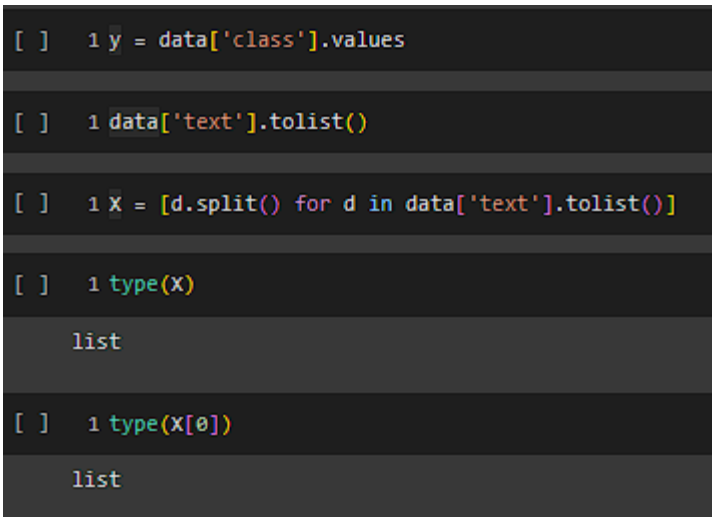
En la **Figura *38*** la primera línea, y = data['class'].values, se asignó las etiquetas de clase de datos (si una noticia es falsa o no) a la variable y. Esto se hizo para que se tenga un conjunto separado de etiquetas a las que el modelo puede apuntar durante el entrenamiento.

La segunda línea, data['text'].tolist(), se convirtió la columna de texto de conjunto de datos en una lista de Python. **Figura *38***

La tercera línea **Figura *38***, X = [d.split() for d in data['text'].tolist()], se creó una lista de listas, donde cada lista interna es una lista de palabras de una noticia. Esto se logró dividiendo cada noticia (en una cadena de texto) en palabras individuales utilizando el método split(). El resultado se almacena en la variable X. E un paso esencial para la preparación de datos para el modelo Word2Vec, que adquiere palabras individuales para aprender los vectores de palabras.

Figura 38

Celda de ejecución



Nota. En esta línea de código se manipularon los datos para prepararlos para el procesamiento posterior.

Ahora se utilizará “Word2Vec” para vectorizar los datos a continuación se explica cada uno de los parámetros de objeto “Word2Vec” **Figura *39***

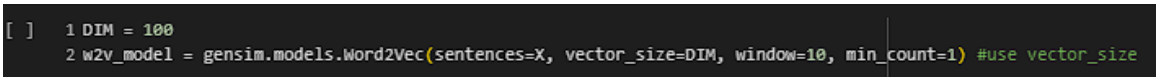
- La variable DIM se definió con el valor 100, lo que significa que se está eligiendo representar cada palabra en corpus como un vector de 100 dimensiones en el espacio de vectores. Este es un hiperparámetro que se puede ajustar las necesidades del proyecto.
- El modelo Word2Vec se inicializo con la función gensim.models.Word2Vec(). Los argumentos que se pasan a esta función son los siguientes:
- sentences=X: se pasa la lista de listas de palabras que se preparó antes. Cada lista interna se considera una "oración" o un documento individual para el modelo Word2Vec.

- vector_size=DIM: Se establece el número de dimensiones de los vectores de palabras que el modelo generará. En este caso, está configurado para coincidir con el valor de DIM que se estableció previamente.
- window=10: Es el tamaño de la "ventana" que Word2Vec utilizo cuando se consideró el contexto de una palabra. En este caso, para cada palabra, el modelo considera las 10 palabras antes y las 10 palabras después como su contexto.
- min_count=1: Este es el número mínimo de veces que una palabra debe aparecer en el corpus para que el modelo la considere. Al configurarlo en 1, modelo debe considerar todas las palabras, incluso si sólo aparecen una vez en todo el corpus.

Al final de esta línea de código, el modelo Word2Vec estará entrenado y listo para transformar cualquier palabra en corpus en un vector de 100 dimensiones. Figura

Figura 39

Celda de ejecución.



Nota. Instanciando y entrenando modelo Word2Vec con la biblioteca gensim.

En la **Figura *40*** se puede observar cómo al entrenar el modelo se obtuvo arreglos de palabras que relacionan cada palabra con la predicción de la siguiente o su semejante. Esto se logra visualizar con una probabilidad de 1 a 0. Donde por ejemplo la palabra “correa” tendrá un 76% de seguir con la palabra “exministro” y así sucesivamente con cada una de las siguientes palabras **Figura *41***.

- Se preparó un tokenizador con la biblioteca de Keras. Tokenizador convirtió texto en tokens numéricos que un modelo de aprendizaje automático que puede

procesar. Tal como se puede observar en la **Figura *42***. En el contexto del procesamiento de lenguaje natural, un token a menudo se refiere a una palabra, pero también puede referirse a una sub-palabra o un carácter. La función Tokenizer() inicializa un nuevo objeto tokenizador.

- Después, con tokenizer.fit_on_texts(X) **Figura *42***, se ajustó el tokenizador a los datos. Esto significa que el tokenizador examino todas las palabras de corpus, que se ha pasado en la forma de X, y construyo un índice de palabras. Cada palabra única en el corpus tendrá un índice numérico correspondiente, y este índice se utilizará para representar la palabra cuando se convirtió en un token. El resultado se puede observar en la **Figura *43***.
- En la **Figura *44*** 'X = tokenizer.texts_to_sequences(X)' reemplaza al texto a enteros en ejemplo anterior es [1,2,3,1,4,5]

Después de ejecutar líneas de código, se obtendrá un tokenizador que está listo para convertir cualquier texto en corpus en una secuencia de tokens numéricos, que luego podrán ser procesados por modelo LSTM.

Figura 40

Celda de ejecución y salida.

```
1 w2v_model.wv["ecuador"]
```

```
array([ 2.9758513 ,  0.53403574, -3.212536  ,  0.2552427 , -0.84223926,
        1.6088291 ,  1.4793164 , -1.016321  , -0.33056805,  1.4050379 ,
        1.3994269 , -1.011899  , -0.89514744, -0.30062824,  1.2992803 ,
       -0.43870655,  2.827291  , -0.71992296, -3.8615525 ,  0.485149  ,
       -0.05570083, -0.50613654, -0.02754175,  0.03776176, -0.7459254 ,
        0.3128388 , -1.1514877 ,  0.09971294, -2.835829  ,  1.4448897 ,
        1.3911756 , -1.6043898 ,  0.06885928,  0.09047494,  2.131668  ,
        0.2906076 ,  2.0475478 , -0.10031062, -0.2563416 ,  1.1207207 ,
       -2.1758544 ,  0.14934951, -0.2973839 ,  1.6523597 , -1.7755029 ,
       -1.9484416 , -1.2201562 ,  0.22411971, -0.6127751 ,  2.692818  ,
```

Nota. Se puede observar como la palabra “ecuador” tiene su único vector luego de pre-entrenar el modelo.

Figura 41

Celda y salida de ejecución.

```
2 w2v_model.wv.most_similar('correa')
```

```
[('exministro', 0.760001540184021),
 ('correa.', 0.7557838559150696),
 ('emco,', 0.7497697472572327),
 ('moreno,', 0.7339462637901306),
 ('rubén', 0.7184755206108093),
 ('lenín', 0.7169250249862671),
 ('aleaga,', 0.7097527980804443),
 ('ordóñez,', 0.7091975212097168),
 ('luis', 0.70329350233078),
 ('exfuncionarios', 0.7015414237976074)]
```

Nota. Se puede observar que la palabra ‘correa’ tiene una probabilidad de ir con ‘exministro’, ‘moreno’ entre otros.

Figura 42

Celda de ejecución.

```
1 tokenizer =Tokenizer()
2 tokenizer.fit_on_texts(X)
```

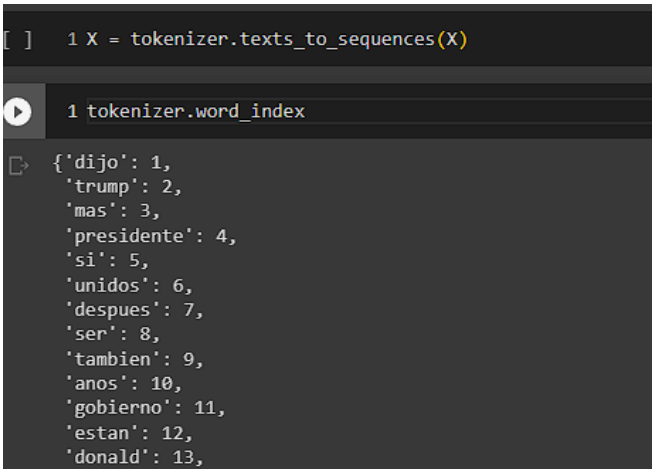
Nota. 'Fit_on_texts': actualiza el vocabulario interno de tokenizador basado en listas de textos 'X'. Este método crea el índice de vocabulario basado en frecuencias. Por ejemplo, la frase 'el gato se comió el pastel' crea el diccionario {'el':1, 'gato':2, 'se':3, 'comio':4,'pastel':5}.

Cuando se llama a la función tokenizer.word_index **Figura *43***, se obtiene diccionario. Este diccionario puede ser útil para entender cómo se han mapeado las palabras a índices numéricos, y se puede usarlo para ver el índice de una palabra específica, o para ver qué palabra corresponde a un índice específico.

En siguiente ejemplo **Figura *44*** se puede observar como la palabra "dijo" tiene índice 1 tokenizer.word_index[dijo]. Del mismo modo, si quisieras ver qué palabra tiene el índice 1, se podrá buscarlo en dicho diccionario.

Figura 43

Tokenización de las palabras.



Nota. El tokenizador convierte en un diccionario cada palabra.

Figura 44

Celda de ejecución.

```
1 tokenizer.word_index
```

```
{'dijo': 1,
 'trump': 2,
 'presidente': 3,
 'si': 4,
 'después': 5,
 'ser': 6,
 'unidos': 7,
 'donald': 8,
 'gobierno': 9,
 'dos': 10,
 'sido': 11,
 'sólo': 12,
 'obama': 13,
 'mientras': 14,
 'clinton': 15,
 'personas': 16,
 'partido': 17,
 'casa': 18,
 'campaña': 19,
 'puede': 20,
```

Nota. se llama a la función tokenizer.word_index, se obtiene diccionario.

En la **Figura *45*** se crea un histograma que muestra la cantidad de palabras eje “y” y cantidad de dichas noticias eje “x”. Se puede observar que 300 noticias tienen aproximadamente 2500 palabras. Esto es útil para tomar decisiones sobre cómo procesar los datos, por ejemplo, cómo acolchar las secuencias antes de alimentarlas al modelo LSTM. Esta visualización ayudará a entender mejor la distribución de las longitudes de las noticias en corpus.

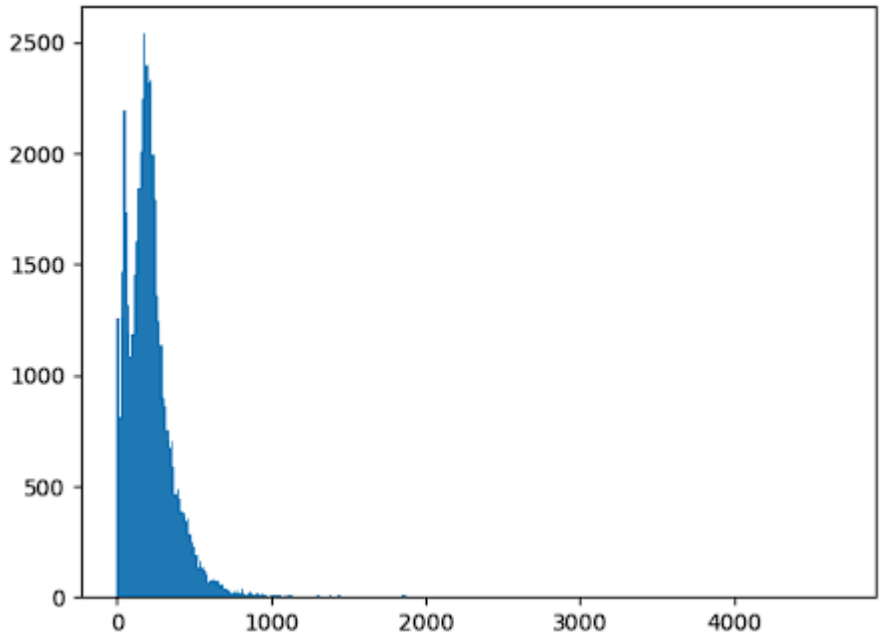
- [len(x) for x in X] es una lista de comprensión que genera una lista de longitudes de todas las noticias en tu corpus X. Cada x en X es una lista de palabras de una noticia y len(x) da el número de palabras en esa noticia.
- plt.hist([len(x) for x in X], bins=350) genera un histograma de estas longitudes de noticias. El parámetro bins determina el número de barras en histograma. En este

caso, se está usando 350 barras. Si la distribución de las longitudes de las noticias es muy variada, un mayor número de barras dará un histograma más detallado.

- Finalmente, plt.show() muestra el histograma.

Figura 45

Celda de salida de ejecución.



Nota. En este fragmento de código, está generando un histograma para visualizar la distribución de la longitud del documento en términos del número de palabras.

Posteriormente, se ha realizado una operación de indexación booleana para filtrar las longitudes que son mayores que 1000 **Figura *46***. El resultado nos[nos>1000] es un array de NumPy que contiene solo las longitudes de las noticias que tienen más de 1000 palabras. Finalmente, al aplicar la función len() a este array, se ha obtenido el número total de noticias en corpus que tienen más de 1000 palabras. Esta información es útil para entender la distribución

de las longitudes de las noticias y tomar decisiones sobre cómo procesarlas antes de alimentarlas al modelo LSTM. Se ha centrado en las noticias que tienen más de 1000 palabras. Primero, se ha utilizado una lista de comprensión para crear una lista de las longitudes de todas las noticias de corpus X. Se ha convertido esta lista en un array de NumPy con la función np.array(), lo que permitió manipular los datos de manera eficiente.

Figura 46

Celda de ejecución.

```
1 nos = np.array([len(x) for x in X])
2 len(nos[nos>1000])
```

Nota. En este segmento de código, se ha trabajado con la longitud de las noticias.

Luego, la función pad_sequences(X, maxlen=maxlen) de Keras se usa para ajustar las secuencias de palabras en nuestras noticias a esta longitud máxima **Figura *47*** . Las secuencias más largas que maxlen serán truncadas y las secuencias más cortas serán rellenadas con ceros al principio para que todas las secuencias tengan exactamente la misma longitud. Esto es necesario porque los modelos de redes neuronales requieren que todas las entradas tengan la misma forma, y pad_sequences nos ayuda a lograr esto.

Por lo tanto, después de estas dos líneas de código **Figura *47***, todas las noticias en corpus X se representarán como secuencias de exactamente 1000 palabras (o tokens) para su uso en el modelo LSTM. Cada noticia será truncada a un máximo de 1000 palabras o rellena con ceros si tiene menos de 1000 palabras.

Figura 47

Celda de ejecución.

```
maxlen = 1000
X = pad_sequences(X, maxlen=maxlen)
```

Nota. En este fragmento de código, primero se definió maxlen = 1000. Esta variable será usada para definir la longitud máxima que puede tener una secuencia de palabras en noticias.

En la **Figura *48*** vocab_size = len(tokenizer.word_index)+1 calcula el tamaño del vocabulario y print(vocab_size) lo que arrojo como 263268 palabras.

vocab = tokenizer.word_index, esta asignando el diccionario word_index a la variable vocab para un uso posterior más conveniente. Este diccionario será útil para convertir palabras a índices y viceversa durante el procesamiento y el análisis de los datos. El tamaño del vocabulario se refiere al número total de palabras únicas en el corpus. Se obtiene al encontrar la longitud del diccionario word_index del objeto tokenizer, que es donde se almacenan todas las palabras únicas y sus índices numéricos. Se agrega 1 a este número porque Keras reserva el índice 0 y no lo asigna a ninguna palabra.

Figura 48

Celda de ejecución.

```
vocab_size = len(tokenizer.word_index)+1
print(vocab_size)
vocab = tokenizer.word_index
```

Nota. En siguiente línea de código se calculó el tamaño del vocabulario y luego se imprime.

En **Figura *49*** la función crea una matriz de pesos inicializada a ceros con el tamaño (vocab_size, DIM). El tamaño de esta matriz coincide con el número total de palabras únicas en corpus (el tamaño del vocabulario) y la cantidad de dimensiones que se eligió para vectores de palabras (DIM).

Luego, la función recorre cada palabra de vocabulario (el diccionario vocab que se obtuvo de tokenizador) y su índice correspondiente. Si la palabra se encuentra en el vocabulario del modelo Word2Vec, se obtiene su vector de palabras utilizando model.wv[word] y se asigna a la fila correspondiente de su índice en la matriz de pesos.

Si la palabra no se encuentra en el vocabulario del modelo Word2Vec, se lanza una excepción KeyError. En este caso, la excepción es capturada y se continúa con la siguiente palabra, dejando la fila de la matriz de pesos correspondiente a esa palabra como ceros.

Luego, la función devuelve la matriz de pesos. Esta matriz de pesos será útil para inicializar la capa de embedding en modelo LSTM, ya que contiene los vectores de palabras previamente aprendidos para todas las palabras en vocabulario.

Figura 49

Celda de ejecución.

```
def get_weight_matrix(model):
    weight_matrix = np.zeros((vocab_size, DIM))
    for word, i in vocab.items():
        try:
            weight_matrix[i] = model.wv[word]
        except KeyError:  # word not in model vocabulary
            continue
    return weight_matrix
```

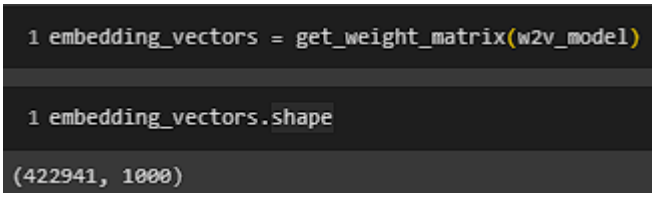
Nota. En este fragmento de código, se está definiendo una función llamada get_weight_matrix que se usa para construir una matriz de pesos a partir de modelo Word2Vec entrenado.

Con `embedding_vectors = get_weight_matrix(w2v_model)`, se llama a la función `get_weight_matrix` con modelo Word2Vec como argumento **Figura *50***. Esta función devuelve la matriz de embedding que contiene los vectores de palabras para todas las palabras en vocabulario. Estos vectores de palabras fueron aprendidos por el modelo Word2Vec de `gensim` que se entrenó anteriormente.

Después, al ejecutar `embedding_vectors.shape`, se obtuvo la forma de esta matriz de embedding. La forma de la matriz será `(vocab_size, DIM)`, ya que la matriz tiene una fila para cada palabra en vocabulario y una columna para cada dimensión en tus vectores de palabras. Esto te permitirá verificar que la matriz de embedding tiene la forma esperada.

Figura 50

Celda de ejecución.



Nota. En estas líneas de código, se generó la matriz de embedding utilizando la función `get_weight_matrix` que se acabó de definir y se está evaluando su forma.

Entrenamiento.

En esta fase se procede a entrenar el modelo. Primero, se ha importado los módulos necesarios, que incluyen `Embedding` para la capa de incrustación de palabras, `Sequential` para el modelo de capas apiladas, `LSTM` para la capa de la red neuronal recurrente, `Bidireccional` para hacer que la LSTM sea bidireccional, `Dense` para las capas completamente conectadas y `Dropout` para regularizar el modelo y evitar el sobreajuste **Figura *51***.

Posteriormente, se ha creado un modelo secuencial, lo que significa que las capas se apilan en secuencia y la salida de una capa se utiliza como entrada para la siguiente.

En la primera capa `Embedding`, se está utilizando la matriz de incrustación que se calculó previamente para inicializar los vectores de palabras. El argumento `trainable=False` indica que estos vectores de palabras no se actualizarán durante el entrenamiento.

A continuación, está añadiendo una capa LSTM bidireccional con 256 unidades. Esta capa procesará las secuencias de palabras y extraerá características útiles para la tarea de detección de noticias falsas.

Después, estás añadiendo una capa densa con una sola unidad y una función de activación sigmoidea. Esta capa proporcionará la salida final del modelo: una probabilidad entre 0 y 1 que indica la probabilidad de que la noticia sea falsa o no.

Finalmente, se ha compilando el modelo con el optimizador Adam, la pérdida de entropía cruzada binaria (que es apropiada para una tarea de clasificación binaria) y se está monitoreando la precisión durante el entrenamiento.

Figura 51

Celda de ejecución.

```
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout

model = Sequential()
model.add(Embedding(vocab_size, output_dim=DIM, weights = [embedding_vectors], input_length=maxlen, trainable=False))
model.add(Bidirectional(LSTM(units=256)))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

Nota. En este bloque de código, ha importado varios módulos necesarios de la biblioteca TensorFlow Keras y se definió la arquitectura del modelo de red neuronal para la detección de noticias falsas.

En la **Figura *52*** se puede observar un resumen de modelo a entrenar:

- La primera fila de la tabla corresponde a la capa de incrustación de palabras. Esta capa tiene una cantidad de parámetros igual al producto del tamaño del vocabulario y la cantidad de dimensiones que se eligió para los vectores de

palabras (DIM). Estos parámetros no son entrenables, como se especificó al agregar la capa de incrustación al modelo.

- La segunda fila corresponde a la capa LSTM bidireccional. Esta capa tiene una cantidad de parámetros que depende del número de unidades en la LSTM y la cantidad de dimensiones en los vectores de palabras. Los parámetros de esta capa son entrenables.
- La última fila corresponde a la capa densa, que es la capa de salida del modelo. Esta capa tiene una cantidad de parámetros que depende del número de unidades en la capa anterior y el número de unidades en la capa densa. Los parámetros de esta capa también son entrenables.
- La última fila de la tabla proporciona el total de parámetros en el modelo, así como el total de parámetros entrenables y no entrenables. Esto puede ser útil para tener una idea de la capacidad del modelo y el costo computacional de entrenarlo.

Figura 52

Celda de ejecución.

```
1 model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1000, 1000)        422941000

 bidirectional (Bidirectiona  (None, 512)              2574336
 l)

 dense (Dense)               (None, 1)                 513

=================================================================
Total params: 425,515,849
Trainable params: 2,574,849
Non-trainable params: 422,941,000
_________________________________________________________________
```

Nota. La función `model.summary()` proporciona un resumen detallado del modelo que acaba de construir. Esta función muestra una tabla que incluye las capas del modelo, la forma de salida de cada capa y el número de parámetros en cada capa.

En la **Figura *53*** se separan los datos de las noticias “X” y variable a entrenar “y” de cada una. Donde se dividen los datos de entrenamiento y de validación:

- El primer argumento que se pasa a la función `train_test_split` es `X`, que es conjunto de datos de entrada (en este caso, las secuencias de palabras de las noticias). El segundo argumento es `y`, que es conjunto de datos de salida (en este caso, las etiquetas de clase de las noticias, indicando si son falsas o no).
- La función `train_test_split` divide aleatoriamente los datos en un conjunto de entrenamiento y un conjunto de prueba. Por defecto, el 75% de los datos se usan para el entrenamiento y el 25% para la prueba, pero se puede cambiar esta proporción usando el argumento `test_size` o `train_size`.

Como resultado, se obtiene cuatro arrays: `X_train` y `y_train`, que contienen las secuencias de palabras y las etiquetas de clase de las noticias en conjunto de entrenamiento, y `X_test` y `y_test`, que contienen las secuencias de palabras y las etiquetas de clase de las noticias en conjunto de prueba. Estos conjuntos de datos se usaron para entrenar y evaluar modelo de detección de noticias falsas **Figura *53***.

Figura 53

Celda de ejecución.

```
1 X_train, X_test, y_train, y_test = train_test_split(X,y)
```

Nota. En esta línea de código, se está utilizando la función `train_test_split` de la biblioteca de aprendizaje automático scikit-learn para dividir el conjunto de datos en dos subconjuntos: un conjunto de entrenamiento y un conjunto de prueba.

Luego la función `model.fit()` es la que realizo el entrenamiento **Figura *54***. Los primeros dos argumentos son datos de entrada (`X_train`) y de salida (`y_train`) respectivamente. Estos datos se usaron para ajustar los parámetros del modelo durante el entrenamiento.

El argumento `validation_split = 0.3` indica que se reservó el 30% de los datos de entrenamiento para validación. Durante el entrenamiento, después de cada época, el modelo se evaluó en este conjunto de validación. Esto es útil para controlar si el modelo está sobre ajustando los datos de entrenamiento.

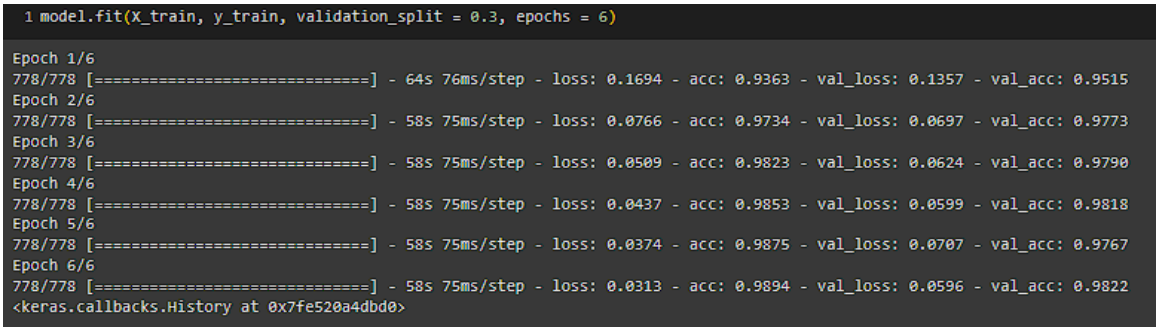
El argumento `epochs = 6` indico que se va a entrenar el modelo durante seis épocas. Una época es un pase completo a través de todo el conjunto de entrenamiento.

Después de ejecutar la línea de código, se pudo observar la pérdida y la precisión del modelo en los datos de entrenamiento y validación después de cada época. Estos números

permitieron seguir el progreso del entrenamiento y decidir cuándo es el momento adecuado para detenerlo.

Figura 54

Celda de ejecución.



Nota. Se procedió a entrena por 6 épocas el modelo que se ha definido previamente utilizando datos de entrenamiento.

La función model.predict() se utiliza para obtener las predicciones del modelo. Se pasa X_test a esta función, que son tus datos de prueba. Para cada ejemplo en X_test, model.predict() genera un número entre 0 y 1, que es la probabilidad predicha por el modelo de que la noticia sea falsa.

A continuación, se comparó estas probabilidades con el umbral de 0.5 utilizando el operador >= **Figura *55***. Esto da como resultado un array de booleanos: True para las noticias que el modelo predice con una probabilidad de más del 50% de que sean verdaderas, y False para las demás.

Luego se utilizó método. astype(int) para convertir el array de booleanos en un array de enteros. Esto convirtió True en 1 y False en 0, con un array de predicciones de clase para noticias de prueba: 0 para las noticias que el modelo predice que son falsas, y 1 para las que predice que

son verdaderas. Estas predicciones se almacenan en y_pred y se pueden comparar con las verdaderas etiquetas de clase para evaluar el rendimiento del modelo.

Figura 55

Celda de ejecución.

```
1 y_pred = (model.predict(X_test) >= 0.5).astype(int)
```

```
371/371 [==============================] - 11s 27ms/step
```

Nota. En esta línea de código, se han generado predicciones con modelo para los datos de prueba.

Se ha utilizado la función `accuracy_score` del módulo `sklearn.metrics` para calcular la precisión del modelo tal como muestra la **Figura *56***. Este método compara las etiquetas reales de las noticias (`y_test`) con las predicciones generadas por el modelo (`y_pred`). La precisión es una métrica de evaluación comúnmente utilizada en problemas de clasificación, que calcula la proporción de predicciones correctas entre todas las predicciones realizadas.

En este caso, el modelo ha obtenido una precisión de aproximadamente 0.981, lo que indica que el 98.1% de las predicciones del modelo en el conjunto de datos de prueba fueron correctas. En otras palabras, el modelo fue capaz de identificar correctamente si una noticia era falsa o verdadera en el 98.1% de los casos en el conjunto de prueba. Esto sugiere que el modelo está realizando un buen trabajo al clasificar las noticias.

Figura 56

Celda de ejecución.

```
1 accuracy_score(y_test, y_pred)
```

```
0.9810126582278481
```

Nota el modelo ha obtenido una **precisión de aproximadamente 0.981**, lo que indica que el 98.1% de las predicciones del modelo en el conjunto de datos de prueba fueron correctas.

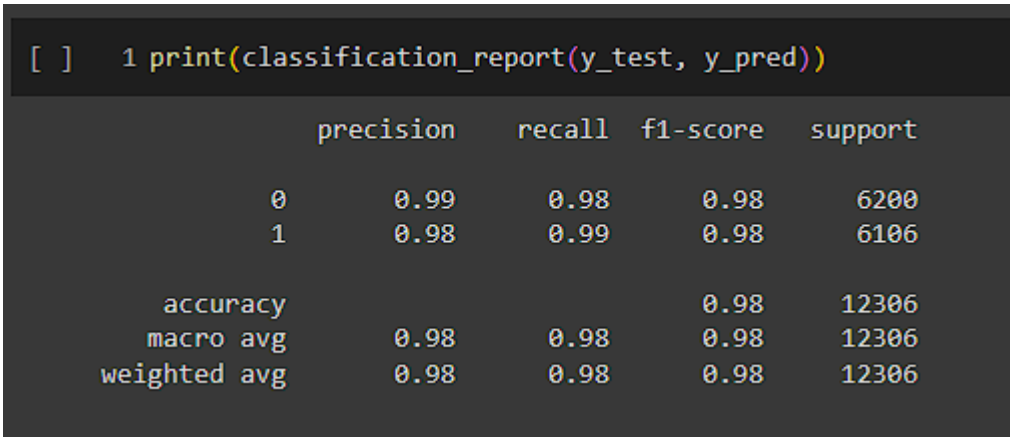
El informe incluye las siguientes métricas para cada clase y su promedio:

- Precisión: de todas las predicciones para una clase, qué proporción fue correcta.
- Recall (sensibilidad): de todas las instancias de una clase en los datos reales, qué proporción fue correctamente identificada por el modelo.
- F1-Score: es una medida que combina la precisión y el Recall en un solo número, dando igual importancia a ambas métricas. Es la media armónica de la precisión y el Recall.

El informe incluye el soporte, que es el número de instancias de cada clase en los datos reales. Por último, se muestra el promedio ponderado de las métricas mencionadas, siendo este ponderado por el soporte de cada clase.

Figura 57

Celda de ejecución.



Nota. El código que se ingreso ha impreso un informe de clasificación, que es una representación textual de las principales métricas de clasificación.

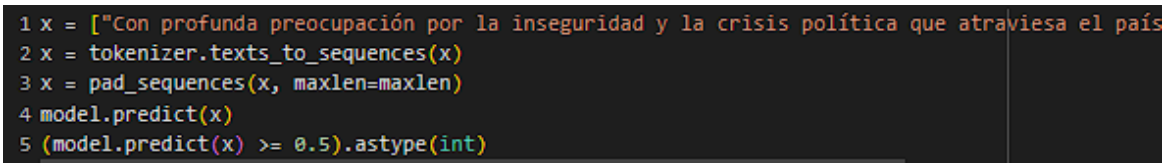
Al finalizar estará entrenado el modelo se pasó a la parte de prueba donde se ingresó el texto de la noticia a predecir si es falsa o no para esto se siguió los siguientes pasos **Figura *58***.

- Inicialmente, se tuvo un texto de noticias en la lista `x`. Este texto pasa por un preprocesamiento usando el `tokenizer` que fue ajustado anteriormente con todos los datos de texto. La función `tokenizer.texts_to_sequences(x)` convirtió el texto en una secuencia de números, donde cada número representa una palabra en el texto.
- A continuación, las secuencias se pasan a la función `pad_sequences` que asegura que todas las secuencias tengan la misma longitud (`maxlen`), agregando ceros en las posiciones necesarias para las secuencias más cortas.
- Finalmente, estas secuencias se pasan al modelo para hacer la predicción. La función `model.predict(x)` devuelve un número entre 0 y 1, que es la probabilidad predicha de que la noticia sea falsa. Posteriormente, se aplica un umbral de 0.5 a esta probabilidad para obtener una predicción de clase binaria: si la probabilidad es mayor o igual a 0.5, la noticia se clasifica como verdadera (1); si es menor, se clasifica como falsa (0). El resultado se convierte en un número entero usando `.astype(int)` y representa la predicción final del modelo para la noticia dada.

Esta metodología reflejo la utilidad del modelo desarrollado, permitiendo su aplicación en la detección de noticias falsas en situaciones reales. En una implementación a mayor escala, este código se integrará en una aplicación web para filtrar las noticias falsas a medida que se el usuario las ingrese.

Figura 58

Código de ejecución.



```
x = ["Con profunda preocupación por la inseguridad y la crisis política que atraviesa el país
x = tokenizer.texts_to_sequences(x)
x = pad_sequences(x, maxlen=maxlen)
model.predict(x)
(model.predict(x) >= 0.5).astype(int)
```

Nota. El código mostrado en estas líneas es una implementación de cómo se puede usar el modelo entrenado para predecir si una noticia es falsa o no. En este caso, se realiza una predicción para una noticia individual, la cual está contenida en la variable `x`.

Luego de entrenar el modelo se tiene que guardar el tokenizador tal como muestra la **Figura *59***. El tokenizador se ha utilizado para convertir las palabras en el texto de las noticias en secuencias numéricas, un formato que el modelo de aprendizaje automático puede entender y procesar.

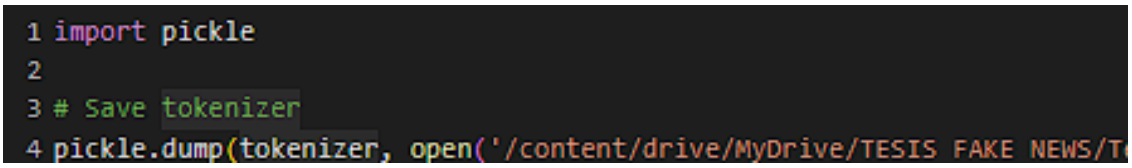
Se utilizó el módulo Pickle de Python para serializar el objeto del tokenizador y guardarlo en un archivo. La serialización es el proceso de convertir un objeto en un formato que se puede almacenar en disco o transmitir a través de una red. En Python, los objetos Pickle pueden contener una amplia gama de tipos de datos de Python.

La función pickle.dump() se utiliza para serializar el objeto del tokenizador. Se pasan dos argumentos a esta función: el objeto que se desea serializar (tokenizer) y un objeto de archivo abierto en modo de escritura binaria.

Como resultado, el tokenizador se almaceno en el archivo 'tokenizer.pkl' en el directorio especificado. Este archivo se puede cargar más tarde para reutilizar el tokenizador en nuevos datos sin tener que volver a entrenarlo, asegurando que las palabras en los nuevos datos se tokenicen de la misma manera que en los datos de entrenamiento.

Figura 59

Celda de ejecución.



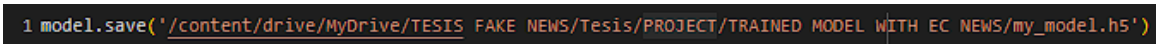
```
import pickle

# Save tokenizer
pickle.dump(tokenizer, open('/content/drive/MyDrive/TESIS FAKE NEWS/T
```

Nota. El tokenizador se ha utilizado para convertir las palabras en el texto de las noticias en secuencias numéricas Esta función permite la persistencia de modelos, facilitando su reutilización en futuras sesiones de trabajo o su distribución.

Una vez que se ha guardado el modelo **Figura *60***, se puede cargar en una nueva sesión de Python utilizando la función `load_model()` de Keras. Esto permitió reutilizar el modelo sin necesidad de volver a entrenarlo, ahorrando tiempo y recursos computacionales.

Figura 60

Celda de ejecución.



```
model.save('/content/drive/MyDrive/TESIS FAKE NEWS/Tesis/PROJECT/TRAINED MODEL WITH EC NEWS/my_model.h5')
```

Nota. El argumento que se pasa a `model.save()`, en este caso `'/content/drive/MyDrive/TESIS FAKE NEWS/my_model.h5'`, especifica la ruta y el nombre del archivo en el que se almacenará el modelo. La extensión `.h5` indica que el modelo se guarda en el formato HDF5, que es un formato de archivo de datos diseñado para almacenar y organizar grandes cantidades de datos.

Con los archivos de tokenizer.pkl **Figura *59*** y my_model.h5 **Figura *60*** se procede a crear la aplicación web con Django para lo cual se debe de montarlo en un servidor web en este caso se lo desarrollo en Amazon EC2.

Entrenamiento modelo versión 2.

Para entrenar la segunda versión se procedió a recopilar o minar los datos con script de automatización desde web.

Se entreno la segunda versión del modelo agregando el dataset con noticias recopiladas de él (El Comercio, 2023), para eso se ha utilizado beautifulsoup4 e Selenium web driver **Figura *61***. Esto se desarrolló en equipo propio utilizando computadora de escritorio y dejando ejecutado el script aproximadamente 1 día al final se recopilo más de 12000 noticias en español de diferentes categorías.

Se abrió el Jupyter Notebook desde Anaconda. Se importaron las bibliotecas necesarias:

- Selenium: Biblioteca que se utilizó para la automatización de navegadores web. Permitió interactuar con diferentes elementos en una página web, como formularios y botones, y también proporciona una forma de recopilar datos de la web.
- WebDriver, Options, y Service de selenium.webdriver.chrome: Estos son componentes específicos de la biblioteca Selenium, que permitieron interactuar con el navegador Google Chrome.
- bs4 (BeautifulSoup): Esta biblioteca se utiliza para extraer datos de archivos HTML y XML. Se convirtió los documentos complejos de la web en árboles de Python, lo que facilita la navegación, búsqueda y modificación de la estructura del documento.
- CSV: Es una biblioteca incorporada de Python que permite leer y escribir archivos CSV.

Se crea una instancia de WebDriver para el navegador Chrome utilizando las Options y Service de selenium.webdriver.chrome. Esto abre un navegador Chrome que puede ser controlado por el script. Esta parte del código no se ve en el fragmento proporcionado, pero es un paso crítico en la utilización de Selenium.

Se utiliza BeautifulSoup para analizar la página HTML cargada por Selenium. Esto se hace para convertir la página en una estructura que se puede recorrer fácilmente en Python. Finalmente, se recogen los datos necesarios de la página web con BeautifulSoup. Estos datos se estructuran y se guardan en un archivo CSV para su posterior análisis.

Figura 61

Celda de ejecución.

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from bs4 import BeautifulSoup
import csv
```

Nota. El WebDriver navega a la URL deseada, que debe ser proporcionada en el código.

Se definió en siguiente bloque de código driver_service utilizando la clase Service de selenium.webdriver.chrome.service **Figura *62***. Esto estableció el servicio de driver que controlará el navegador Chrome. El argumento executable_path se utilizó para proporcionar la ruta al archivo ejecutable del driver de Chrome en el sistema del usuario. En este caso, el archivo se llama chromedriver.exe y se encuentra en la carpeta especificada.

Luego, se define base_url, que representa la URL base de la página web de la cual se extraerán las noticias. Aquí, la URL base corresponde a la sección "Última Hora" del periódico "El Comercio" (El Comercio, 2023) **Figura *62***.

Después, se define csv_file, que es la ruta al archivo CSV donde se guardarán los datos recopilados. Este archivo se llama large_news_ec_2.csv. Por defecto, este archivo se creará (o sobrescribirá si ya existe) en la misma carpeta donde se ejecuta el script.

Finalmente, csv_headers define los encabezados de las columnas en el archivo CSV. En este caso, se recopilarán el "Título" y la "Descripción" de cada noticia tal como muestra la **Figura *62***.

Figura 62

Celda de código

```
# Define the base URL
base_url = "https://www.elcomercio.com/ultima-hora/page/"

# Define the CSV file path and headers
csv_file = "large_news_ec_2.csv"
csv_headers = ["Title", "Description"]
```

Nota. En esta porción del código, se están estableciendo varios parámetros fundamentales para la tarea de recopilación de datos.

Options(): Se creo una instancia de la clase Options de selenium.webdriver.chrome.options, que se utilizó para configurar el navegador Chrome.

options.add_argument("--disable-gpu"): Se añade un argumento a las opciones para desactivar la aceleración de GPU en Chrome. Esta opción permitió ayudar a mejorar la estabilidad del navegador cuando se ejecuta en modo "headless", sobre todo en sistemas Windows.

Se abrió el navegador con la línea webdriver.Chrome(service=driver_service, options=options). Esta línea creo una nueva instancia de la clase webdriver.Chrome, que representa una sesión de Chrome que puede ser controlada a través de Selenium. Se pasan las opciones configuradas anteriormente y el servicio de driver establecido a esta instancia. A partir de este punto, el navegador Chrome está listo para ser controlado a través del script.

Figura 63

Celda de código

```
# Configure the headless browser
options = Options()
#options.add_argument("--headless")  # Run Chrome in headless mode
options.add_argument("--disable-gpu")

# Open the headless browser
driver = webdriver.Chrome(service=driver_service, options=options)
```

Nota. Options(): Se crea una instancia de la clase Options de selenium.webdriver.chrome.options, que se utilizará para configurar el navegador Chrome.

Luego se abrió el archivo CSV especificado anteriormente en modo de escritura ("w"), utilizando el encoding "utf-8". Se establece newline="" para permitir que el módulo csv maneje los finales de línea **Figura *64***. A través del método csv.writer(file), se creó un objeto writer que será usado para escribir en el archivo CSV. Se escribió la primera línea del archivo CSV con writer.writerow(csv_headers), que contiene los encabezados especificados anteriormente.

Se estableció page_number a 1 y news_scraped a 0. Estas variables se usarán para llevar un registro del número de la página actual que se está recolectando y del total de noticias recolectadas, respectivamente.

Se entra en un bucle infinito con while True: **Figura *64***. En cada iteración del bucle, se realizó lo siguiente:

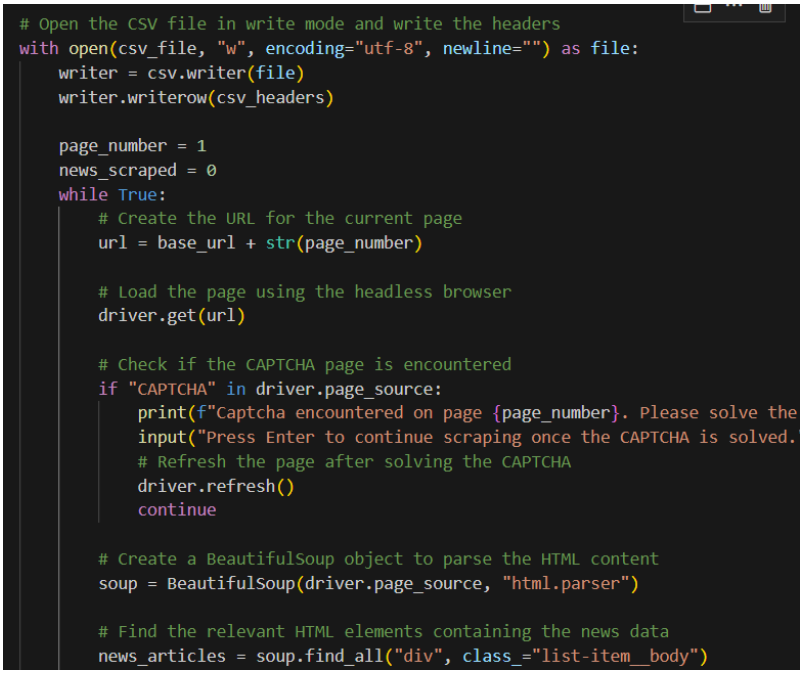
- Se construyo la URL de la página actual añadiendo page_number a base_url.
- Se cargo la página en el navegador con driver.get(url).
- Se verifico si se ha encontrado una página CAPTCHA en el sitio web. Si es así, se imprime un mensaje pidiendo al usuario que resuelva el CAPTCHA y se detiene

el programa hasta que el usuario presione Enter. Luego se recarga la página con driver.refresh() y se salta al inicio del bucle con continue.

- Se creo un objeto BeautifulSoup con el contenido HTML de la página web.
- Se buscan todos los elementos HTML relevantes que contienen los datos de las noticias.

Figura 64

Celda de ejecución.



```
# Open the CSV file in write mode and write the headers
with open(csv_file, "w", encoding="utf-8", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(csv_headers)

    page_number = 1
    news_scraped = 0
    while True:
        # Create the URL for the current page
        url = base_url + str(page_number)

        # Load the page using the headless browser
        driver.get(url)

        # Check if the CAPTCHA page is encountered
        if "CAPTCHA" in driver.page_source:
            print(f"Captcha encountered on page {page_number}. Please solve the
            input("Press Enter to continue scraping once the CAPTCHA is solved.
            # Refresh the page after solving the CAPTCHA
            driver.refresh()
            continue

        # Create a BeautifulSoup object to parse the HTML content
        soup = BeautifulSoup(driver.page_source, "html.parser")

        # Find the relevant HTML elements containing the news data
        news_articles = soup.find_all("div", class_="list-item__body")
```

Nota. En este fragmento del código se realiza el proceso de recopilación de datos a través del recorrido de las páginas de noticias.

Encuentra el elemento HTML que contiene el título del artículo y extrae el texto tal como muestra la **Figura *65***. Aquí el título está contenido en un enlace (a). Almacena el enlace (href) que lleva a la página detallada del artículo y navega hacia ella utilizando driver.get(title_link).

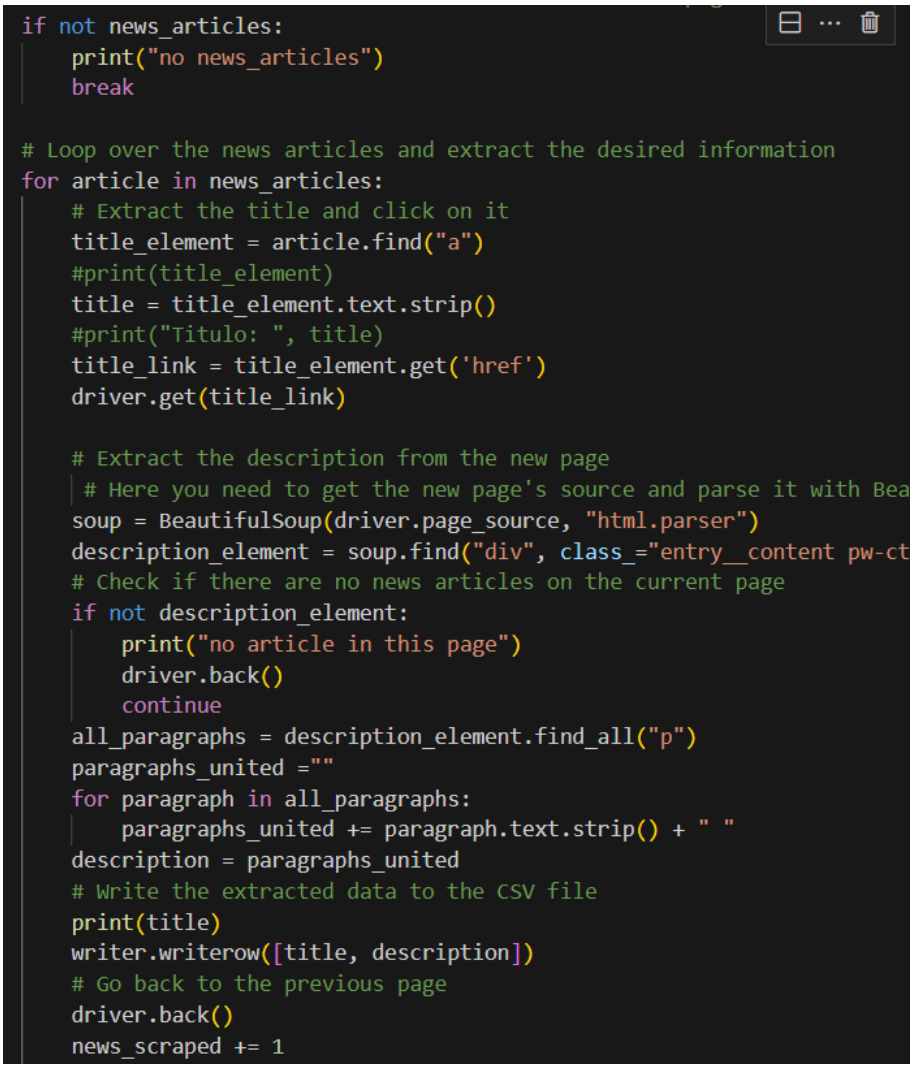
Crea un nuevo objeto BeautifulSoup para analizar la nueva página y encontrar el elemento que contiene la descripción del artículo.

Si no se encontró description_element, se imprimirá un mensaje de alerta, regresa a la página anterior con driver.back() y pasa a la siguiente iteración del bucle for con continue. Encuentra todos los párrafos (p) dentro de description_element y los une en una sola cadena de texto (paragraphs_united), separando cada párrafo con un espacio. Escribe una nueva fila en el archivo CSV con el título y la descripción del artículo mediante writer.writerow([title, description]). Regresa a la página anterior (la página que contiene la lista de artículos) con driver.back(). Incrementa news_scraped en uno e imprime la cantidad total de noticias scrapeadas hasta el momento **Figura *65***.

De esta manera, el código recopilara de manera efectiva el título y la descripción de cada artículo de noticias y los guardara en un archivo CSV para su posterior análisis. El bucle while continuará ejecutándose y scrapeando noticias de cada página hasta que no encuentre más artículos de noticias en una página, momento en el cual se detendrá la ejecución.

Figura 65

Celda de ejecución.



```
if not news_articles:
    print("no news_articles")
    break

# Loop over the news articles and extract the desired information
for article in news_articles:
    # Extract the title and click on it
    title_element = article.find("a")
    #print(title_element)
    title = title_element.text.strip()
    #print("Titulo: ", title)
    title_link = title_element.get('href')
    driver.get(title_link)

    # Extract the description from the new page
    # Here you need to get the new page's source and parse it with Bea
    soup = BeautifulSoup(driver.page_source, "html.parser")
    description_element = soup.find("div", class_="entry__content pw-ct
    # Check if there are no news articles on the current page
    if not description_element:
        print("no article in this page")
        driver.back()
        continue
    all_paragraphs = description_element.find_all("p")
    paragraphs_united =""
    for paragraph in all_paragraphs:
        paragraphs_united += paragraph.text.strip() + " "
    description = paragraphs_united
    # Write the extracted data to the CSV file
    print(title)
    writer.writerow([title, description])
    # Go back to the previous page
    driver.back()
    news_scraped += 1
```

Nota. Este fragmento de código se encarga de extraer los detalles relevantes de cada artículo de noticias y almacenarlos en el archivo CSV.

Al finalizar la extracción de noticias de cada página, el código retrocede a la página anterior con driver.back(), incrementa el contador news_scraped, y registra la cantidad de noticias scrapeadas hasta el momento. Se imprime un mensaje que indica que la página actual se ha recolectado con éxito **Figura *66***.

Posteriormente, se generó otro objeto BeautifulSoup para analizar la página y encontrar el botón "Siguiente". Se asume que este botón es un enlace (a) con el atributo rel="next" y la clase "next page-numbers". Si existe tal botón, se recoge el enlace asociado a él y se navega hacia esa página con driver.get(next_link).

Si no se encuentra el botón "Siguiente", significa que ya no hay más páginas para scrapear, por lo que el bucle while se rompe con break.

Si se encontró el botón "Siguiente" y se navegó a la próxima página, se incrementa page_number en uno para reflejar la nueva página actual. Una vez finalizado el proceso de extracción de datos, se imprime un mensaje que indica que la extracción de datos se ha completado con éxito. Por último, se cierra el navegador controlado por Selenium con driver.quit().

Con estos pasos, el script de extracción de datos navega por todas las páginas del sitio web de noticias, recopilando los detalles de cada artículo de noticias y almacenándolos en un archivo CSV para su análisis posterior. Cuando no hay más páginas para navegar, el script finaliza de manera ordenada, liberando los recursos que utilizaba. El continuo resultado de trabajo se puede visualizar en la **Figura *67***.

Figura 66

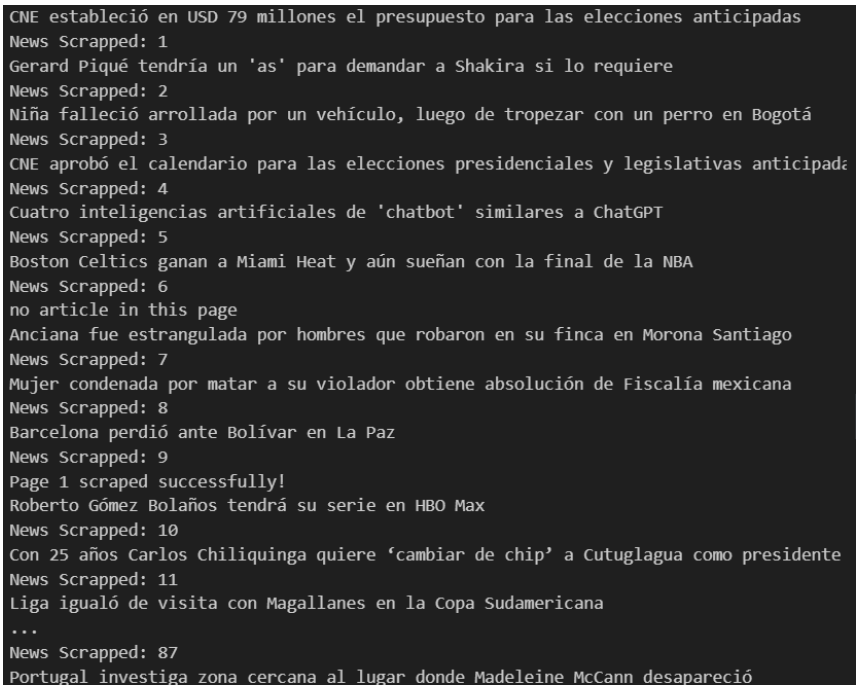
Celda de código

```
            # Go back to the previous page
            driver.back()
            news_scraped += 1
            print(f"News Scrapped: {news_scraped}")
        #driver.back()
        print(f"Page {page_number} scraped successfully!")
        # Find the "Next" button and check if it exists
        soup = BeautifulSoup(driver.page_source, "html.parser")
        next_button = soup.find("a", rel="next", class_="next page-numbers")
        next_link = next_button.get('href')
        driver.get(next_link)
        if not next_button:
            break
        page_number += 1
print("Data extraction completed successfully!")

# Close the headless browser
driver.quit()
```

Nota. Este segmento del código se encarga de la navegación entre páginas y del cierre adecuado del programa de extracción de datos.

Figura 67

Celda de ejecución.



Nota. El proceso de recopilación ha demorado aproximadamente 1 día se han recopilado aproximadamente 12000 noticias de Ecuador.

Luego se procedió a entrenar el nuevo modelo, pero antes se agregó las noticias verdaderas con 2500 noticias de Ecuador seleccionadas al azar y agregados al datasets. Y luego se continuó haciendo los mismos pasos de entrenamiento de modelo V1. Con esto se trató de mejorar las detecciones de noticias verdaderas.

Entrenamiento modelo versión 3

Para desarrollar este modelo se procedió a mejorar el dataset de noticias falsas con API de Chat-GPT 3.5 generando así con prompts noticias inventadas y posiblemente mejorando el modelo de detección de noticias que son procedentes de Transformers y de otras fuentes inventadas o no creíbles.

Para eso se procedió usar la API de Chat-GPT desde (Open AI, 2023) Luego se procedió a programar el script para guardar en CSV aproximadamente 2000 prompts de noticias falsas **Figura *68*** para posteriormente agregarlos a datasets existente. Para esto se registró en la página de OpenAI luego se ingresó los datos de pago y luego se procedió a copiar API KEY. Luego se probó la conexión y se ejecutó el script después de esperar un tiempo prudente se pudo generar 2000 noticias.

Figura 68

Celda de código.

```
# Generate and write the fake news text
for i in range(10):
    try:
        # Make the API request
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=base_prompt,
            temperature=0.7,
        )

        # Extract the generated text and write it to the CSV file
        text_generated = response['choices'][0]['message']['content']
        writer.writerow([text_generated])
        token_count = response.usage['total_tokens']
        print(token_count)
    except Exception as e:
        print(f"An error occurred: {e}")
```

Nota. se procedió a programar el script para guardar en CSV aproximadamente 2000 prompts de noticias falsas para posteriormente agregarlos a dataset existente.

Luego se procedió a agregar al dataset de noticias falsas y entrenar el modelo igual que se lo hizo en V1 otra vez en Google Colab esta vez agregando los datos recopilados de Fake News y Noticias de Ecuador con las noticias traducidas base.

Capitulo IV

Diseño e Implementación del Sistema

Desarrollo de WEB APP.

Se procedió a crear aplicación web con Flask y se tuvo en cuenta varios componentes. A continuación, se cada uno de los pasos del proceso de forma general:

Configuración del entorno.

Primero se preparó el entorno de desarrollo. Se instalo Python y, además, se utilizó un entorno virtual de Python para evitar conflictos de dependencias. Un entorno virtual permitió instalar paquetes de Python que solo están disponibles en este entorno en particular y no afectarán otros proyectos de Python.

Instalación de Flask.

Una vez que se tiene listo el entorno virtual, se procedió a instalar Flask con el comando pip install flask.

Creación de la aplicación Flask.

En un nuevo archivo Python (en este caso FakeNews_LSTM.py), se ha creado una aplicación Flask básica como muestra en el código (**Figura *69***).

- El decorador @app.route('/') indica que la función home se ejecutará cuando se visite la raíz del sitio web.
- Se ha importado diferentes módulos de la biblioteca Flask. Flask se utilizó para crear la instancia de la aplicación, request para manejar las solicitudes HTTP,

render_template para renderizar las plantillas HTML y “session” para mantener la sesión del usuario entre múltiples solicitudes.

Figura 69

Línea de código.

```
from flask import Flask, request, render_template, session
```

Nota. Se ha importado diferentes módulos de la biblioteca Flask.

TensorFlow es una biblioteca de código abierto para el aprendizaje automático y las redes neuronales. Se utilizo aquí para trabajar con redes LSTM para el análisis de textos **Figura *70***.

Figura 70

Línea de código.

```
import tensorflow as tf
```

Nota. TensorFlow es una biblioteca de código abierto para el aprendizaje automático y las redes neuronales.

Para este paso fue necesario acoplar el modelo de detección ya entrenado a la Web App. Se importaron funciones específicas de la biblioteca Keras en TensorFlow (**Figura *71***). pad_sequences es una función que se utiliza para garantizar que todas las secuencias en un lote tengan la misma longitud, rellenando las secuencias más cortas según sea necesario. Tokenizer para vectorizar textos, o convertir los textos en secuencias numéricas, estas son más fáciles de procesar para la red.

Se importo os **Figura *72***, datetime **Figura *73*** y se creó la instancia de la aplicación flask **Figura *74***.

Figura 71

Línea de código.

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
```

Nota. Se importaron funciones específicas de la biblioteca Keras en TensorFlow. pad_sequences es una función que se utiliza para garantizar que todas las secuencias en un lote tengan la misma longitud

Figura 72

Línea de código

```
import os
```

Nota. El módulo os proporciona una forma de usar las funcionalidades dependientes del sistema operativo, como leer o escribir en variables de entorno.

Figura 73

Línea de código

```
from datetime import datetime
```

Se importa la clase datetime del módulo datetime para trabajar con fechas y horas.

Figura 74

Línea de código

```
app = Flask(__name__)
```

Nota. Se creo una instancia de la aplicación Flask. __name__ una variable especial que obtuvo el nombre del módulo de Python actual. Flask utiliza la ubicación del módulo como punto de partida cuando carga recursos asociados, como plantillas.

Se estableció la clave secreta de la aplicación a partir de una variable de entorno **Figura *75***. Esta clave secreta se utiliza para firmar las sesiones de los usuarios. Es importante mantener esta clave secreta para evitar que los atacantes puedan falsificar las sesiones. Luego esta clave se ingresa en el servidor directamente. A través de EC2 instance. En la **Figura *76*** se carga el

modelo que ya se entrenó en Google Colab. Se carga también el tokenizador **Figura *76***. Posteriormente. Se estableció una variable maxlen al valor 1000 **Figura *78***, que es la longitud máxima que la red LSTM puede aceptar para una secuencia de entrada. Esta es la misma que se utilizó en fase de entrenamiento de modelo y debe de ser idéntica. Cualquier secuencia que se introduzca en la red debe tener esta longitud. Si es más corta, se rellenará con ceros; si es más larga, se truncará.

Figura 75

Línea de código

```
app.secret_key = os.environ.get('FLASK_APP_SECRET_KEY')
```

Nota. Este código parece configurar una aplicación Flask que utilizará una red LSTM para detectar noticias falsas.

Figura 76

Línea de código

```
model = tf.keras.models.load_model('my_model.h5')
```

Nota. Se cargo un modelo previamente entrenado con Keras, utilizando la función load_model. El modelo debe estar en formato .h5, que es el formato de archivo estándar para los modelos de Keras. Este modelo es la red LSTM que se ha entrenado para detectar noticias falsas.

Figura 77

Línea de código

```
tokenizer = pickle.load(open('tokenizer.pkl', 'rb'))
```

Nota. Se ha cargado un modelo previamente entrenado con Keras, utilizando la función load_model. El modelo debe estar en formato .h5, que es el formato de archivo estándar para los modelos de Keras. Este modelo es la red LSTM que se ha entrenado para detectar noticias falsas.

Figura 78

Línea de código

```
maxlen = 1000
```

Nota. Se estableció el padding a 1000.

Estas líneas **Figura *79*** definen una ruta en la aplicación Flask para el endpoint '/'. Cuando un usuario visita la URL raíz de la aplicación se ejecuta la función home(). Esta función imprime la cadena "home" a la salida estándar y luego renderiza y devuelve una plantilla HTML llamada 'home_v2.html'. Esta plantilla existe en el directorio de plantillas de la aplicación Flask. Posteriormente se creó la ruta “predict” que ayudara a detectar la noticia.

Figura 79

Línea de código

```
Artiom
@app.route('/')
def home():
    print("home")
    return render_template('home_v2.html')
```

Nota. Función home.

Figura 80

Línea de código

```
@app.route('/predict', methods=['POST'])
```

Nota. Esto define una ruta '/predict' en la aplicación Flask que responde a las solicitudes HTTP POST.

Se verifica que la solicitud sea de tipo POST. En una aplicación Flask, request.method contiene el método HTTP de la solicitud actual. Luego se pasó message desde el form. **Figura *81*** posteriormente. El 'message' se convierte en una lista **Figura *82*** y luego se tokeniza usando

el tokenizador que se cargó anteriormente. Esto convierte las palabras en el mensaje en números correspondientes a los índices de las palabras en el vocabulario. Luego, las secuencias tokenizadas se acolchan hasta una longitud máxima determinada.

Figura 81

Línea de código

```
message = request.form['message']
```

Nota. Se recupera el campo 'message' del formulario enviado en la solicitud POST. Este 'message' contiene el texto de la noticia que se evaluará.

Figura 82

Línea de código

```
data = [message]
seq = tokenizer.texts_to_sequences(data)
padded = pad_sequences(seq, maxlen=maxlen)
```

Nota se crea data, seq y padded

En la **Figura *83*** se obtiene el score o la puntuación que ingreso el usuario luego se le multiplica por 100 para dar el porcentaje y se establece a 2 decimales la misma. En la **Figura *84*** si es mayor a 0.5 la respuesta la convierte en 1 o noticia verdadera de lo contrario a falsa. En **Figura *86*** se devuelve la predicción a la plantilla HTML como variables.

Figura 83

Línea de código

```
prediction_score = model.predict(padded)[0][0]  # Get the prediction score
prediction_percentage = round(prediction_score * 100, 2)  # Calculate percentage
```

Nota. Se hace una predicción utilizando el modelo cargado. La predicción devolverá una probabilidad de que la noticia sea falsa, que se convierte en un porcentaje.

Figura 84

Línea de código

```
my_prediction = (model.predict(padded) >= 0.5).astype(int)
```

Nota. Salida de predicción como numero entero.

Figura 85

Línea de código

```
user_prediction = request.form['user_prediction']
```

Nota. Se recupera el campo 'user_prediction' del formulario, que contiene una predicción realizada por el usuario sobre si la noticia es falsa o no.

Figura 86

Línea de código

```
# Redirect to the next step
return render_template('result_v2.html', prediction=my_prediction, percentage=prediction_percentage
```

Nota. Finalmente, se renderiza la plantilla 'result_v2.html', pasando la predicción del modelo, el porcentaje de predicción y el mensaje original como variables a la plantilla.

Ejecución de la aplicación

Se ha ejecutado aplicación con el comando Python FakeNews_LSTM.py observo un mensaje indicando que el servidor está en marcha y se accedió a la aplicación en localhost:5000 **Figura *88***. Se ingreso una pregunta de prueba y se observó la siguiente plantilla devuelta del resultado de detección de la noticia. **Figura *89***

Prueba de la aplicación.

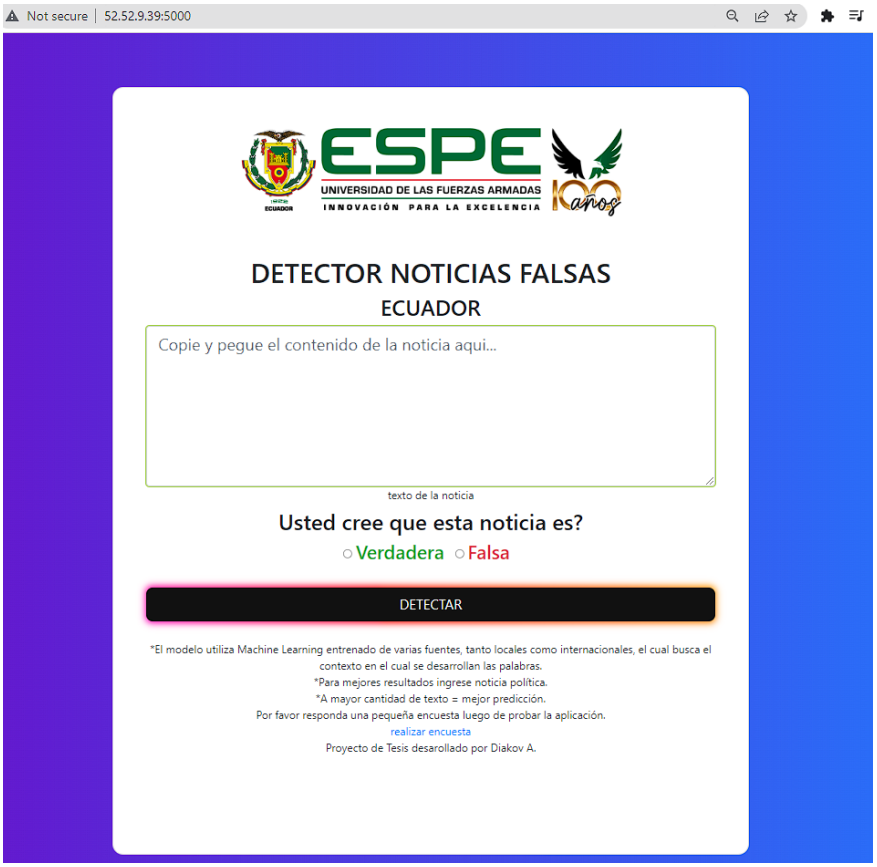
Figura 87

URL

Nota. Se ingreso en Chrome a la dirección:

Figura 88

Aplicación ejecutada en Chrome



Nota. Se puede observar la aplicación desarrollada.

Figura 89

Aplicación retornando resultado de detección.

Nota. Se ingresa la noticia copiada de la página www.elcomercio.com.

Expansión de la aplicación

En este punto, se pudo empezar a expandir la aplicación. Flask es un micro-framework, lo que significa que es ligero y modular.

Despliegue de la aplicación

Finalmente, una vez que se está satisfecho con la aplicación, se puede desplegar a un servidor en producción. La mejor opción depende de las necesidades específicas del proyecto.

Algunas opciones populares incluyen servidores como AWS, Google Cloud, Heroku, entre otros (Kennedy, 2016).

Configuración de servidor EC2 en AWS.

Para comenzar, es esencial establecer una cuenta en AWS y configurar una instancia de Amazon EC2 (Elastic Compute Cloud). EC2 permite la creación de instancias de servidor virtual en la nube de AWS, lo cual fue ideal para la implementación del servidor. **Figura *90***

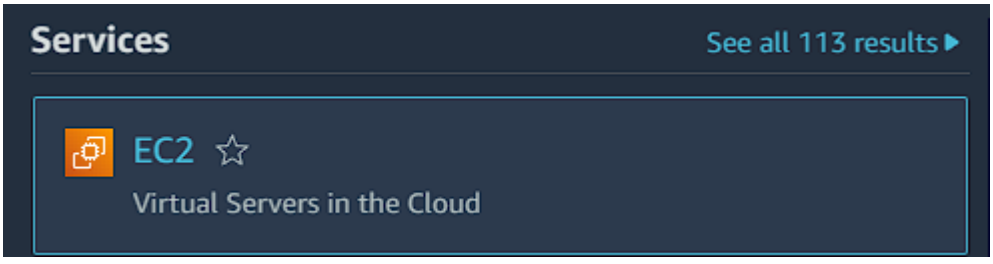
Una vez creada la cuenta AWS y seleccionada la opción EC2, se elige el tipo de instancia, Amazon Linux 2 AMI (Amazon Machine Image). A continuación, se ajustan los detalles de la instancia y se configura el grupo de seguridad para permitir el tráfico entrante y saliente a la instancia.

Luego se creó una pair key para poder acceder a la instancia. Que se ingresara desde PuTTy en Windows. Se agrego una regla de acceso al puerto 5000 **Figura *94***.

Después se seleccionó 8GB en almacenamiento interno. **Figura *95*** y se activó la instancia.

Figura 90

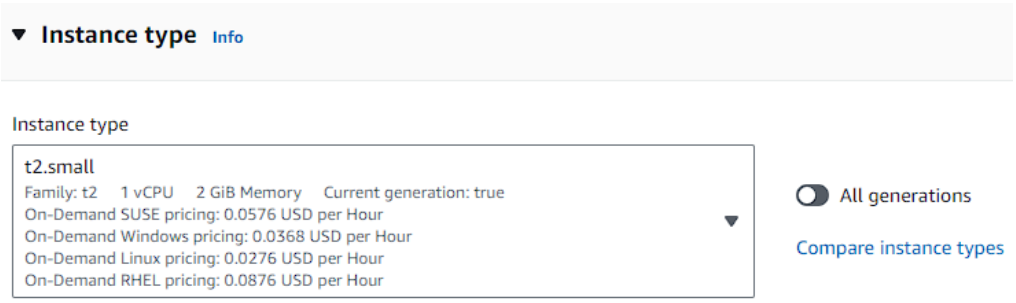
Servicio EC2 de Amazon AWS



Nota. Se escribe en barra de búsqueda de AWS “EC2”

Figura 91

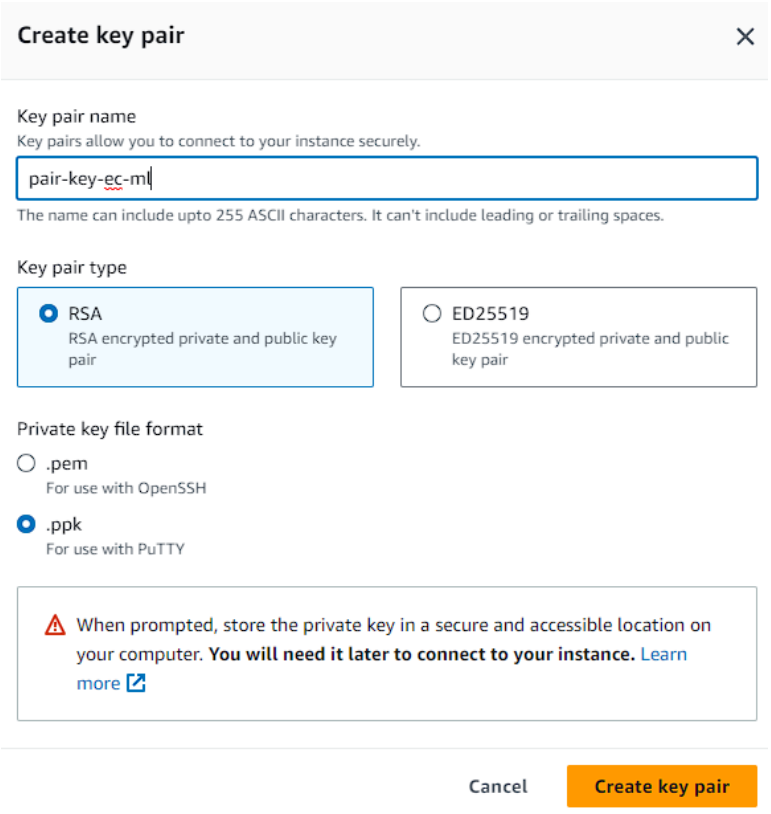
Servicio EC2 de Amazon AWS



Nota. En tipo de instancia se seleccionó t2.small que permite tener la base de procesamiento para la ejecución de proyecto.

Figura 92

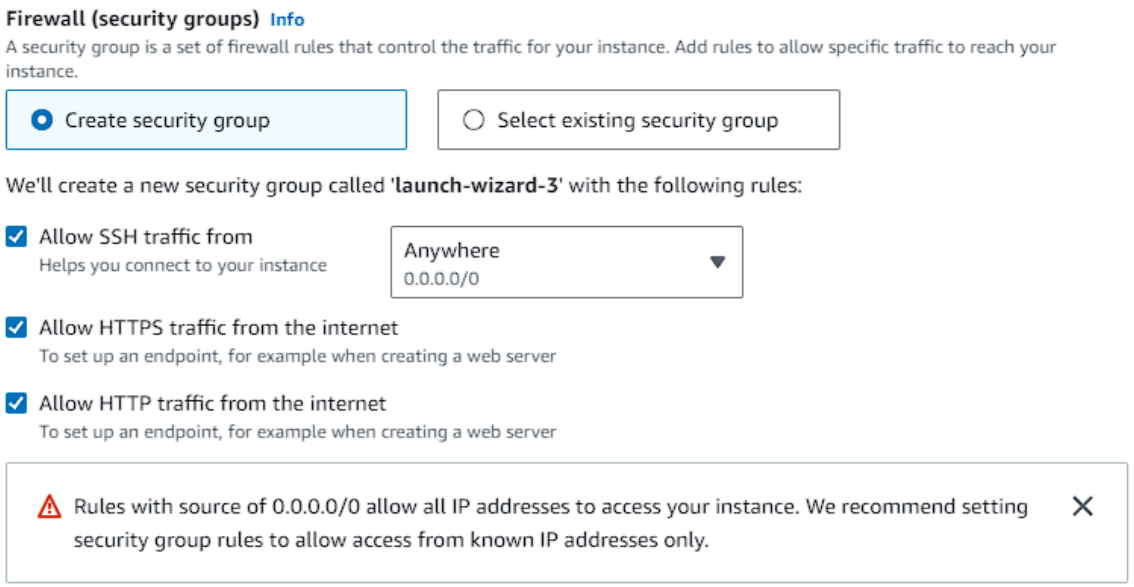
Servicio EC2 de Amazon AWS



Nota. Para poder acceder a la instancia EC2 se realizó desde consola de AWS o a través de PuTTy usando Windows.

Figura 93

Servicio EC2 de Amazon AWS



Nota. Se configuro la configuración de red. Se activa el tráfico en sección de firewall a HTTPS y HTTP desde internet.

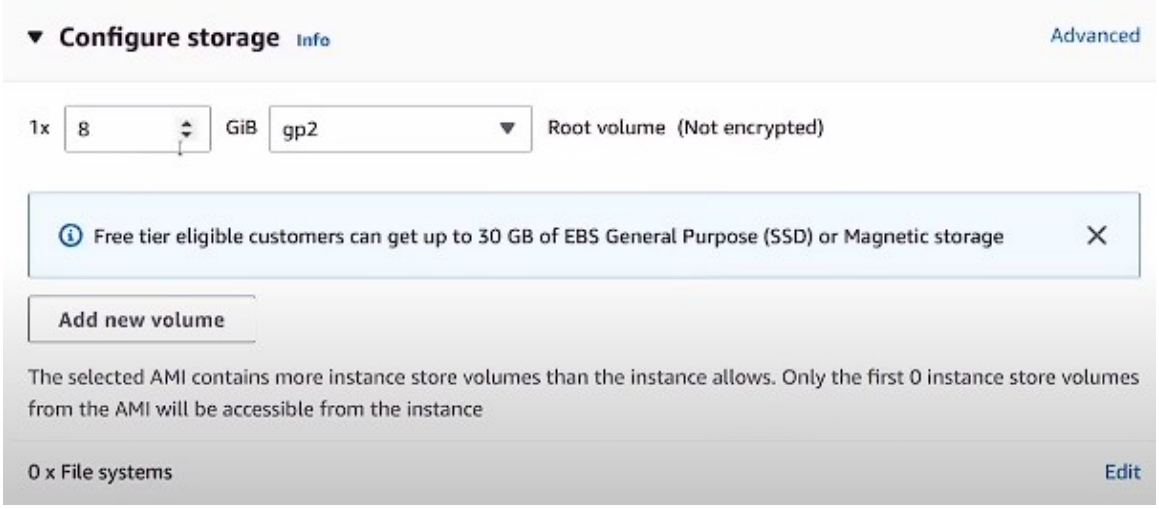
Figura 94

Servicio EC2 de Amazon AWS

Nota. Se ingreso a edit security group y se agrega una nueva regla. Como el proyecto se lo realizo en Flask se debe de tener abierto el puerto 5000. En “source type” se cambió a “Anywhere”

Figura 95

Servicio EC2 de Amazon AWS



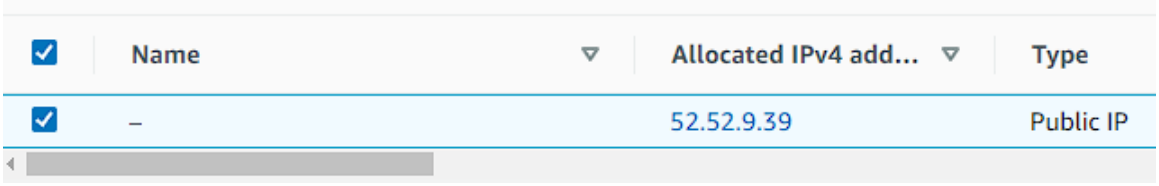
Nota. Por último, se seleccionó 8GiB en gp2 en SSD de almacenamiento.

Para poder obtener una virtualización de IP fija para el proyecto se activó el “Elastic IP” este permitirá acceder a la misma IP del proyecto, aunque la dirección cambie **¡Error! No se encuentra el origen de la referencia.**. Después se visualizó la IP fija que genero Elastic IP para el proyecto **Figura *96***.

Configuración de Elastic IP.

Figura 96

Servicio EC2 de Amazon AWS



Nota. Se puedo observar la instancia de servidor creada.

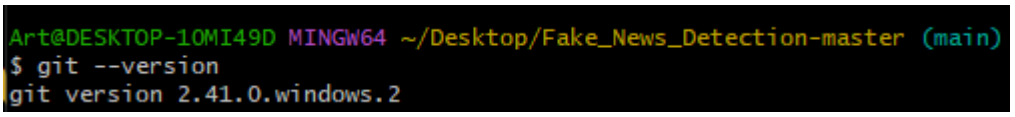
Carga de proyecto a repositorio GitHub.

Verificar la instalación de Git.

Ahora se procede a subir el proyecto desde el equipo al repositorio GitHub para lo cual creamos preparamos el proyecto con verificación de Git instalado **Figura *97***.

Figura 97

Preparación de push a GitHub



Nota. Se aseguro de que Git esté instalado en la máquina de desarrollo. Se ejecuto el siguiente comando para verificarlo:

Crear un nuevo repositorio en GitHub.

Se creo el repositorio y posteriormente se lo procedió a cargar utilizando los comandos subsecuentes.

Agregar archivos al repositorio.

Figura 98

Código en servidor Ubuntu en EC2

```
git status
```

Nota. Primero, se debe verificar el estado de los archivos en el repositorio con

Primer commit.

Desde el equipo local agregamos el proyecto al repositorio se hace el primer commit luego ponemos en stage el proyecto y se hace un push al repositorio remoto

Configuración de servidor Ubuntu.

Luego de crear la instancia se la ejecuto para configuración de la misma En el servidor de Amazon EC2. Y se descarga el proyecto desde repositorio remoto.

Clonar el repositorio remoto.

Figura 99

Comandos en EC2

```
git clone https://github.com/ArtlordReload/Fake_News_Detection_EC
```

Nota. Una vez que Git esté instalado, se clono el repositorio remoto con el comando:

Ejecución de servidor Flask.

Posteriormente se ejecutó la aplicación Flask **Figura *100***. El proyecto Flask ahora debe estar en ejecución en el servidor EC2 y debería poder accederse a través del navegador utilizando la dirección IP del servidor EC2 y el puerto designado (el puerto predeterminado es el 5000 para Flask).

Figura 100

Comandos en EC2

```
flask run --host=0.0.0.0
```

Nota. Finalmente, ejecute el servidor Flask con el comando:

Desarrollo de aplicación WEB

Compra de dominio en Hostinger.com

Se adquirido un dominio en Hostinger.com. Este involucra una serie de pasos:

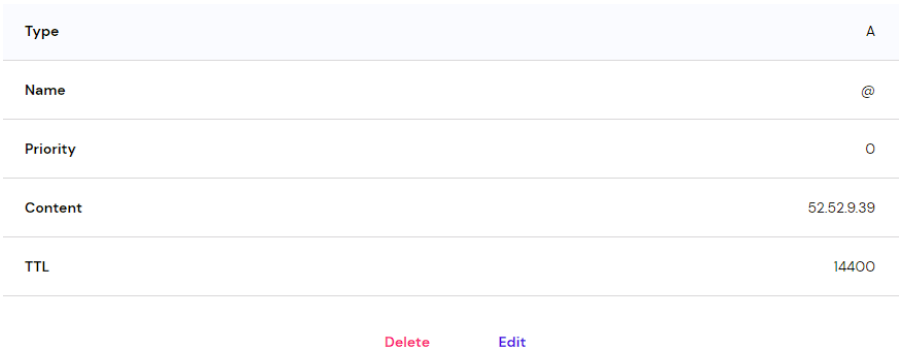
- Se ingreso al el Sitio Web: Inicialmente, se dirigió a la página oficial de Hostinger.com.

  Figura *101*

- Se busco el dominio: En la página principal de Hostinger, En barra de búsqueda. Se escribió el nombre del dominio detectornoticiasfalsasecuador.com
- El dominio que se ingreso está disponible, y presento una opción para comprarlo. Se escogió como .com como la extensión más popular.
- Se lo añadió al carrito: Se selecciono el dominio, Se lo añadió al carrito de compras.
- Se registro en Hostinger
- Se realizo el pago por medio de cuenta propia.
- Se confirmo el Pedido: se recibió un correo electrónico de confirmación de Hostinger. Este correo contenía detalles de compra.
- Se configuro el dominio: A continuación, se procedió a configurar dominio a través del panel de control de Hostinger.
- Se agrego un A record. Este apuntara a la aplicación web en servidor EC2.

Figura 101

Configuración en Hostinger.



Nota. Se agrego un record A así mismo la IP del servidor vinculada al dominio.

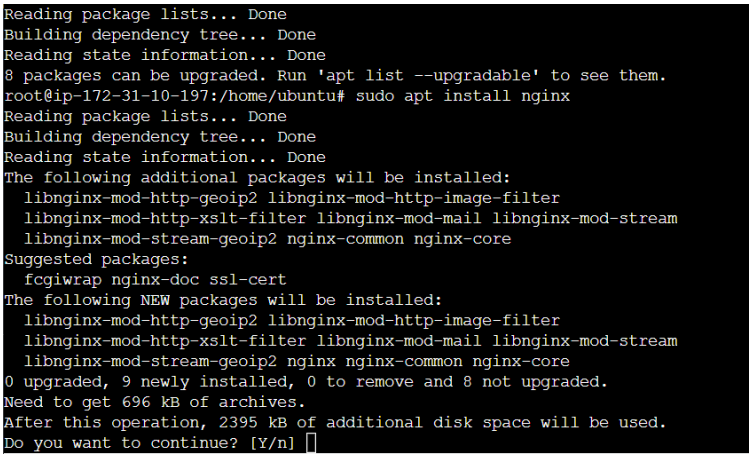
Configuración de servidor NGIX en Ubuntu server.

Se configuro Nginx **Figura *102***: Una vez instalado Nginx, se lo configuro para que pueda servir la aplicación de detección de noticias falsas. Se creo un nuevo archivo de configuración de Nginx para la aplicación en el directorio /etc/nginx/sites-available/ y luego se logró un enlace simbólico a archivo desde el directorio /etc/nginx/sites-enabled/. Por último, reinicio Nginx para que los cambios surtan efecto.

Luego se accedió a la página detectornoticiasfalsasecuador.com la cual ya está correctamente servida a través de dominio. **Figura *103***

Figura 102

Comandos de configuración en EC2 Ubuntu server.



Nota. Se instalo el NGINX server para servir de manera apropiada la aplicación.

Figura 103

WEB App desplegada en WEB

Nota. Se probo la configuración: Se dirigió a la página desde navegador. detectornoticiasfalsasecuador.com. Se comprobó el funcionamiento de la aplicación.

Instalación de Certificado SSL.

Para este paso se instaló el certificado SSL el cual permite de que la pagina este cifrada según los estándares de seguridad se ejecutó la siguiente línea de código la que permitió la instalación de la misma **Figura *104***

Figura 104

WEB App desplegada en WEB

Nota. Al ejecutar el siguiente comando en el servidor se instala el certificado gratuito de certbot.

Capitulo V

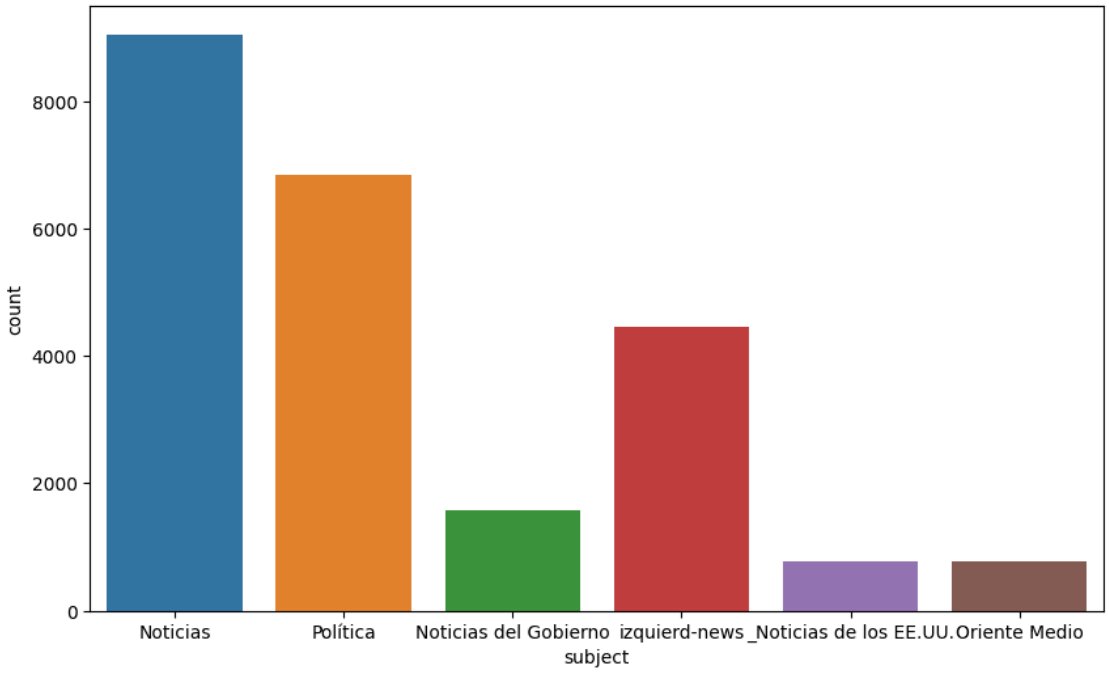
Pruebas y Encuestas

Contenido de Dataset para entrenamiento.

Se analizaron el tipo de datos encontrados en noticias falsas originales del dataset de noticias traducidas las que se muestran en la Figura 105. El total fue de 24623 noticias falsas. En dataset de noticias verdaderas fue de 21418. Para dar un total de 46041 de noticias totales traducidas. Posteriormente se agregó 3205 noticias de Ecuador recogidas al azar del dataset de 12000 noticias recopiladas. Y 1120 agregados por el Chat-GPT API el dataset de modelo final entrenado tiene 49246 noticias con sus respectivas etiquetas **Tabla *3***.

Figura 105

Contenido de tipo de noticias falsas traducidas.



Nota. Cantidad de noticias falsas traducidas 24623

Tabla 3

Desglose de dataset utilizado en el modelo entrenado versión 3.

Noticias de dataset.	Verdaderas	Falsas
Noticias Traducida	21418	23503
Noticias de Ecuador	3205	-
Noticias Generadas por CHAT-GPT	-	1120
Total	24623	24623

Nota. se trató de equilibrar los datos de noticias verdaderas y falsas.

Pruebas locales de modelo entrenado.

Se han realizado las pruebas a los 3 modelos realizados. Los datos validados no eran usados del dataset de entrenamiento. Fueron realizadas con noticias Verdaderas al azar de diferentes páginas web de noticias locales e internacionales. Las noticias Falsas fueron probadas con generación de prompts con Chat-GPT. El idioma usado fue español. El tamaño de texto fue de entre 200 a 1000 palabras. Tal como se puede observar en la **Tabla 4**

Como se puede observar de la **Tabla 4** **Tabla *5*** y **Tabla *6*** el modelo mejora considerablemente al agregarle al dataset noticias de recopiladas de Ecuador y aun mas con el Chat-GPT.

Tabla 4

Pruebas realizadas a la aplicación agregadas al entrenamiento versión 1.

MODELO ENTRENADO VERSION 1			
NOTICIA VERDADERA #	ACIERTOS	NOTICIA FALSA #	ACIERTOS
1	1	1	0
2	0	2	1
3	1	3	0
4	0	4	0
5	1	5	1
6	0	6	0
7	0	7	1
8	0	8	1
9	0	9	0
10	0	10	1
TOTAL	3/10	TOTAL	5/10
PUNTUACION FINAL 8/20			

Nota. Si la noticia es catalogada o se cree que es verdadera y el programa devuelve como verdadero el resultado se acierta como 1 de lo contrario como 0. Si es catalogada la noticia como falsa o se cree que

debe de ser falsa y el resultado de la aplicación acierta a que es falsa se acierta como 1 de lo contrario 0. El resultado de puntaje es la suma de todos los aciertos. Las noticias verdaderas son diferentes de las falsas.

Tabla 5

Pruebas realizadas a la aplicación agregadas al entrenamiento versión 2.

MODELO ENTRENADO VERSION 2			
NOTICIA VERDADERA #	ACIERTOS	NOTICIA FALSA #	ACIERTOS
1	1	1	1
2	1	2	0
3	1	3	1
4	1	4	1
5	1	5	1
6	0	6	0
7	1	7	1
8	1	8	0
9	1	9	1
10	1	10	0
TOTAL	9/10	TOTAL	5/10
PUNTUACION FINAL 14/20			

Nota. Si la noticia es catalogada o se cree que es verdadera y el programa devuelve como verdadero el resultado se acierta como 1 de lo contrario como 0. Si es catalogada la noticia como falsa o se cree que debe de ser falsa y el resultado de la aplicación acierta a que es falsa se acierta como 1 de lo contrario 0. El resultado de puntaje es la suma de todos los aciertos. Las noticias verdaderas son diferentes de las falsas.

Tabla 6

Pruebas realizadas a la aplicación agregadas al entrenamiento versión 3.

MODELO ENTRENADO VERSION 3			
NOTICIA VERDADERA #	ACIERTOS	NOTICIA FALSA #	ACIERTOS
1	1	1	1
2	1	2	1
3	1	3	0
4	1	4	1
5	1	5	1
6	1	6	1
7	1	7	1
8	1	8	1
9	1	9	1
10	1	10	1
TOTAL	10/10	TOTAL	9/10
PUNTUACION FINAL 19/20			

Nota. Si la noticia es catalogada o se cree que es verdadera y el programa devuelve como verdadero el resultado se acierta como 1 de lo contrario como 0. Si es catalogada la noticia como falsa o se cree que

debe de ser falsa y el resultado de la aplicación acierta a que es falsa se acierta como 1 de lo contrario 0. El resultado de puntaje es la suma de todos los aciertos. Las noticias verdaderas son diferentes de las falsas.

Conclusión entre diferentes versiones de modelos entrenados.

El primer modelo entrenado únicamente de dataset traducido de Ingles a español dio una base para noticias en su mayoría de USA y de Noticias Internacionales. También es posible que contenga errores de traducción a español. Por lo que la predicción a realidad de Ecuador no da buenos resultados.

El segundo modelo entrenado dio una mejor versión ya que se agregaron al dataset de noticias minadas de los sitios de Ecuador catalogadas como reales. Por lo tanto, la predicción en noticias reales es bastante notable.

En el tercer modelo entrenado fue mucho más preciso debido a que se equilibraron los datos de noticias reales y falsas. Al agregarle noticias de Ecuador y noticias generadas por Chat-GPT mejoro notablemente el índice de predicción.

Cabe destacar que las noticias probadas reales fueron de medios locales como CNN en español, El Comercio, Metro Ecuador. Y las falsas por su lado fueron las generadas por Chat-GPT. No se han hecho pruebas en otros ambientes por lo tanto el índice de predicción podría ser bajo o nulo en algunos escenarios.

Resultados de Encuestas a Usuarios Finales sobre la Herramienta Web de Detección de Fake News

Para validar la eficacia y facilidad de uso de la herramienta, se llevó a cabo un proceso de prueba y retroalimentación con diferentes usuarios. Estos comprenden una mezcla diversa de compañeros de estudio, amigos, docentes y algunos profesionales en áreas de desarrollo,

aunque es esencial mencionar que no todos eran expertos en el ámbito de noticias o en la detección de información falsa.

Primera Etapa de Retroalimentación:

La primera versión de la aplicación recibió comentarios críticos en cuanto a su interfaz, que fue descrita por algunos como "pobre" o no intuitiva. Ante estas observaciones, se proporcionaron instrucciones básicas para guiar a los usuarios sobre cómo interactuar con la plataforma y qué tipo de información ingresar. Las respuestas de esta etapa se encuentran entre las **Figura *106*** y **Figura *125***.

Segunda Etapa de Retroalimentación:

Con base en los comentarios recibidos, se realizó una revisión integral de la herramienta. En esta segunda versión, se implementó una interfaz mejorada y se añadió una pregunta adicional en la encuesta para captar sugerencias sobre posibles mejoras. Las respuestas correspondientes a esta etapa se pueden consultar en las **Figura *116*** a **Figura *119***.

Una sugerencia predominante fue la posibilidad de insertar enlaces de noticias para su análisis, en lugar de copiar y pegar el texto. Si bien esta es una característica valiosa, es importante destacar que el propósito central de la herramienta es analizar el contenido textual de la noticia y no necesariamente la fuente o página web en sí. Sin embargo, esta recomendación será considerada para futuros desarrollos.

Tercera Etapa de Retroalimentación:

En la versión 3, se incorporó una pregunta dirigida específicamente a aquellos usuarios que habían experimentado con al menos una de las dos versiones anteriores. Se buscaba

conocer su percepción sobre si hubo mejoras en el modelo de detección de noticias falsas a lo largo de las versiones. Esta información está ilustrada en la **Figura *125***.

Los resultados fueron alentadores: más del 80% de los encuestados percibieron una mejora en la capacidad de detección de la herramienta en esta última versión.

Conclusión

La retroalimentación de los usuarios ha sido esencial para guiar las mejoras en la herramienta. A través de las diferentes etapas de encuestas, se ha logrado refinar no solo la interfaz sino también la eficiencia del modelo subyacente. Se prevé que futuras iteraciones se beneficiarán aún más de esta retroalimentación y continuarán mejorando en función de las necesidades del usuario y las demandas del entorno de noticias en Ecuador.

Encuesta de Aplicación y Modelo Entrenado Versión 1

Figura 106

Encuesta Modelo V1 Pregunta 1

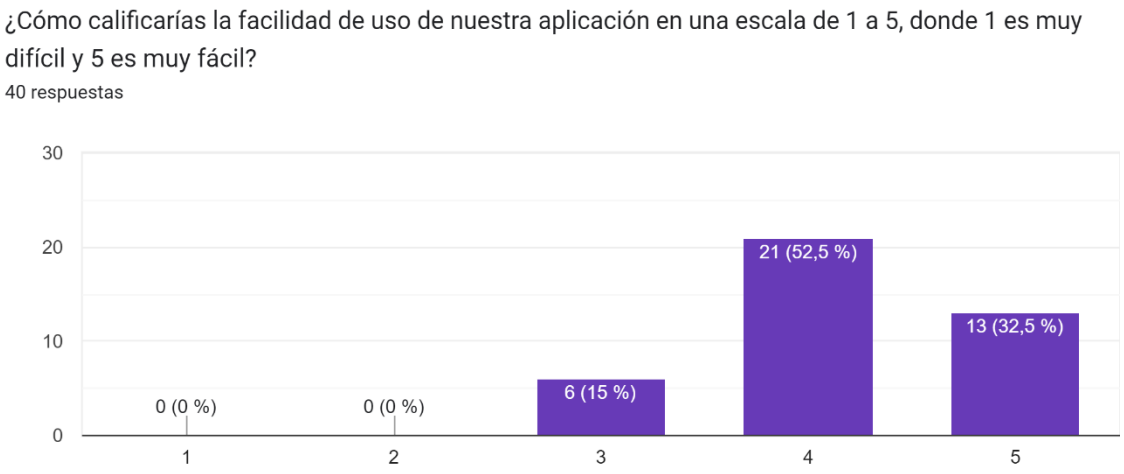


Figura 107

Encuesta Modelo V1 Pregunta 2

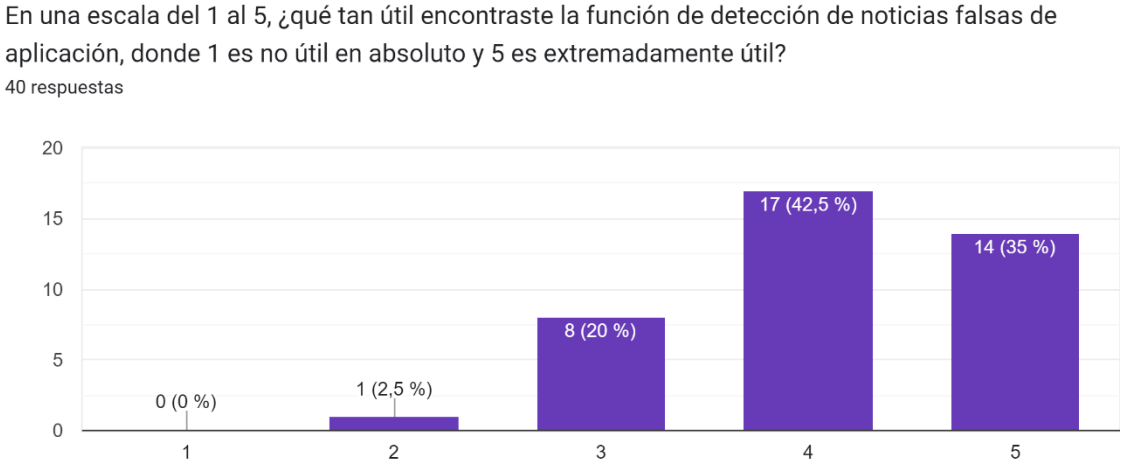


Figura 108

Encuesta Modelo V1 Pregunta 3

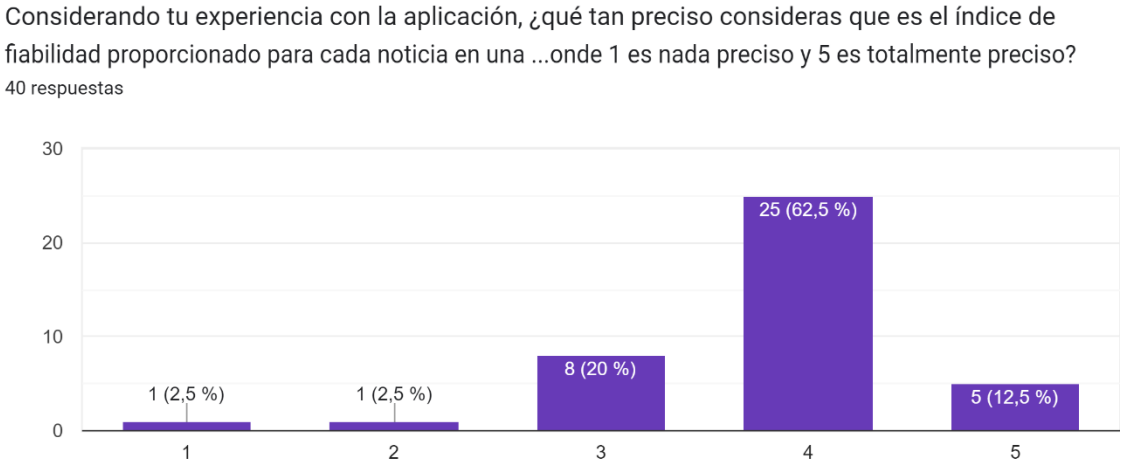


Figura 109

Encuesta Modelo V1 Pregunta 4

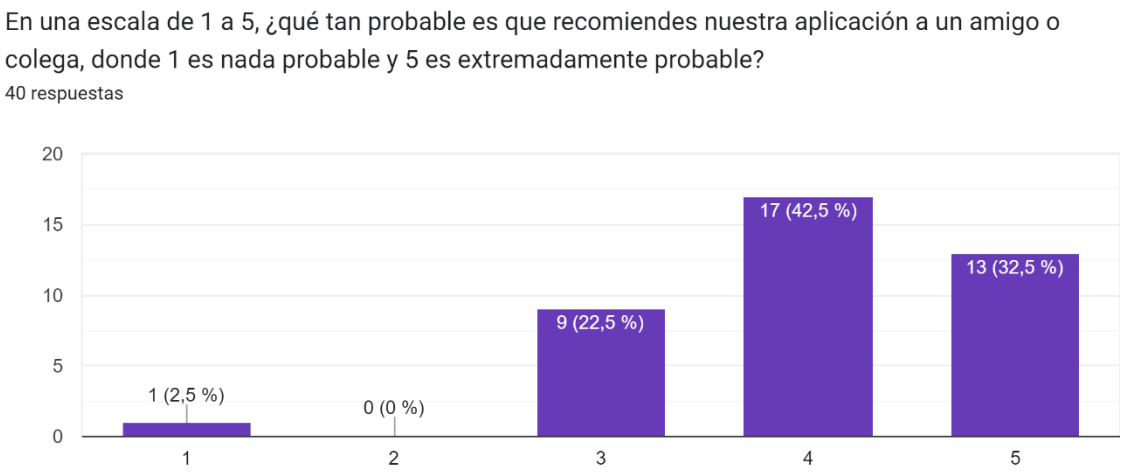
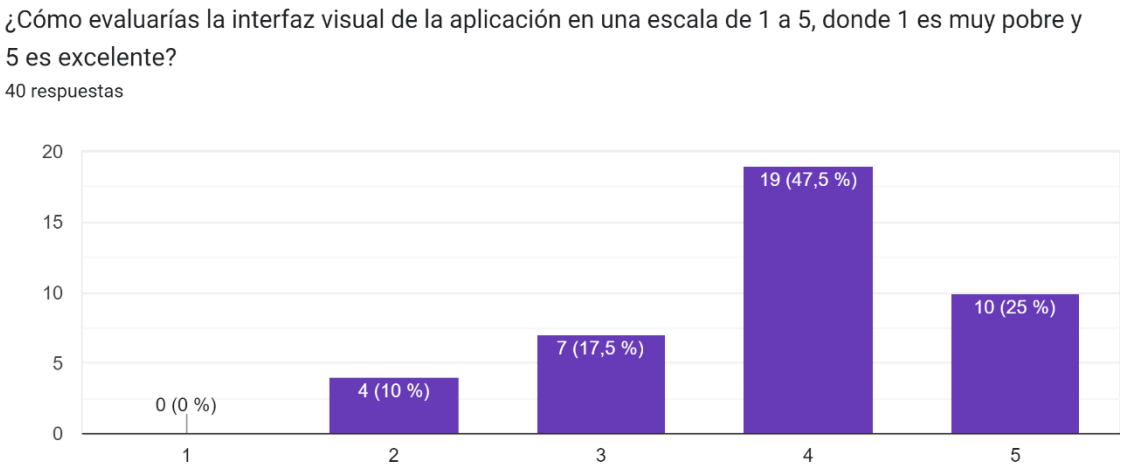


Figura 110

Encuesta Modelo V1 Pregunta 5



Encuesta de Aplicación y Modelo Entrenado Versión 2

Figura 111

Encuesta Modelo V2 Pregunta 1

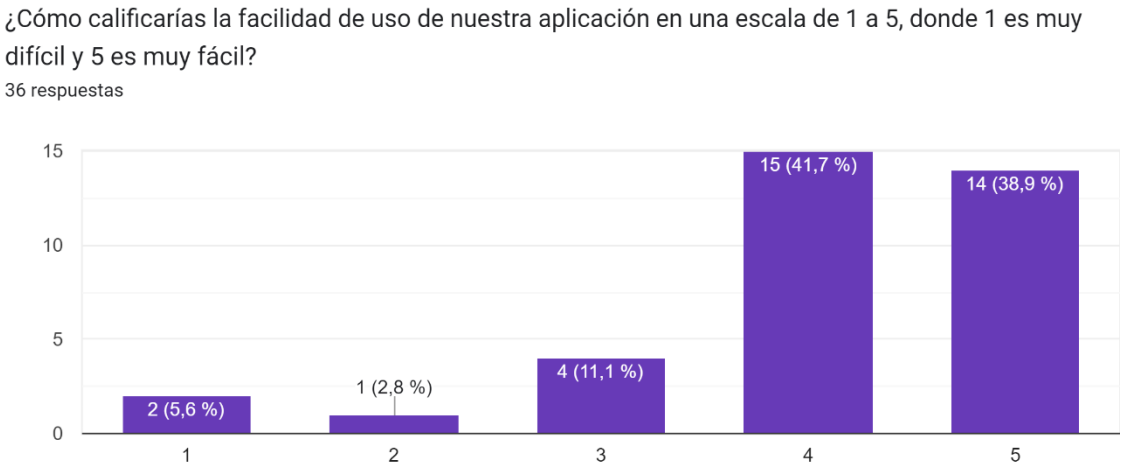


Figura 112

Encuesta Modelo V2 Pregunta 2

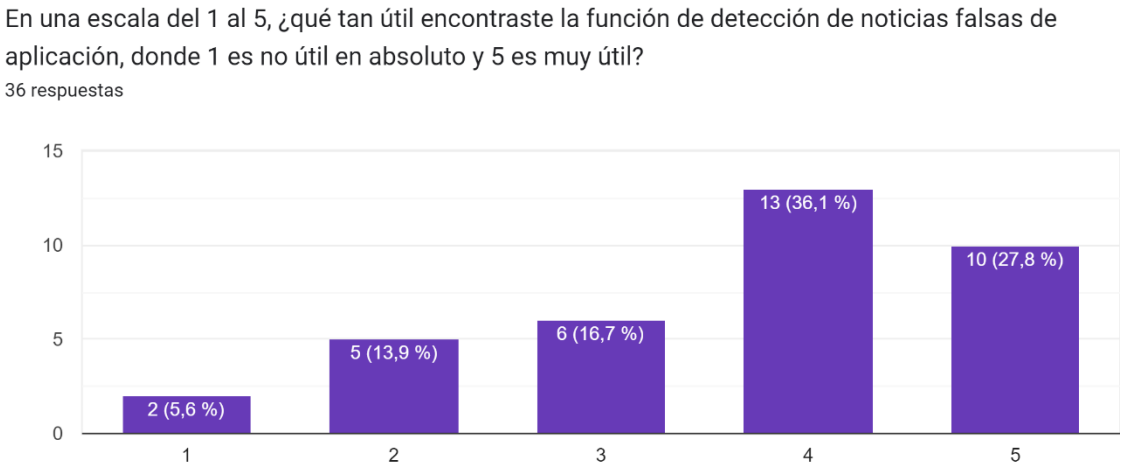


Figura 113

Encuesta Modelo V2 Pregunta 3

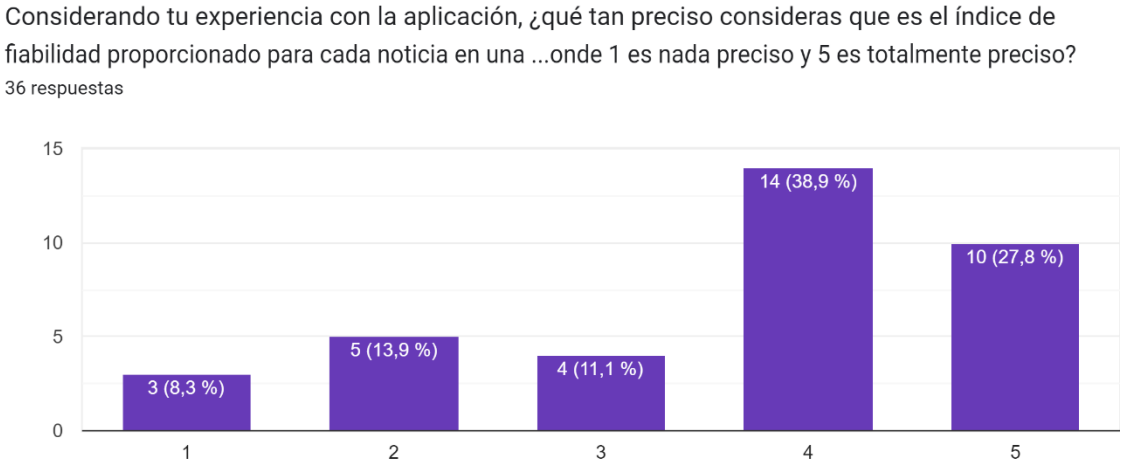


Figura 114

Encuesta Modelo V2 Pregunta 4

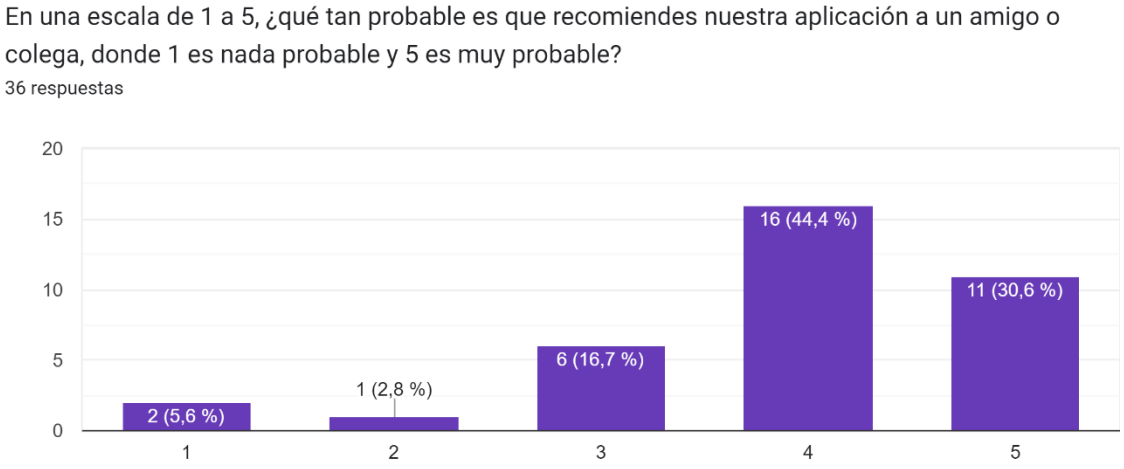
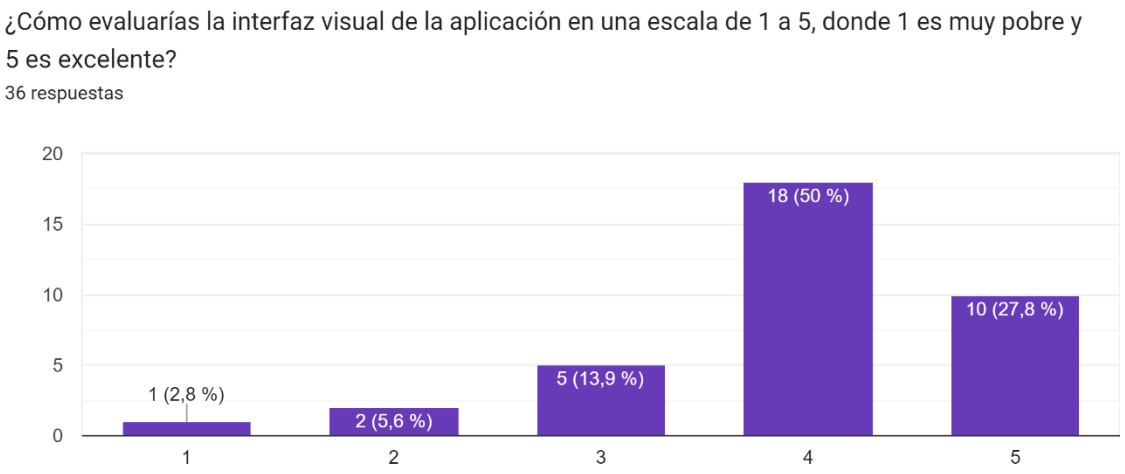


Figura 115

Encuesta Modelo V2 Pregunta 5



Preguntas orientadas a la mejora de la aplicación.

Figura 116

Preguntas de sugerencias a los usuarios para mejorar la versión 3 de la aplicación.

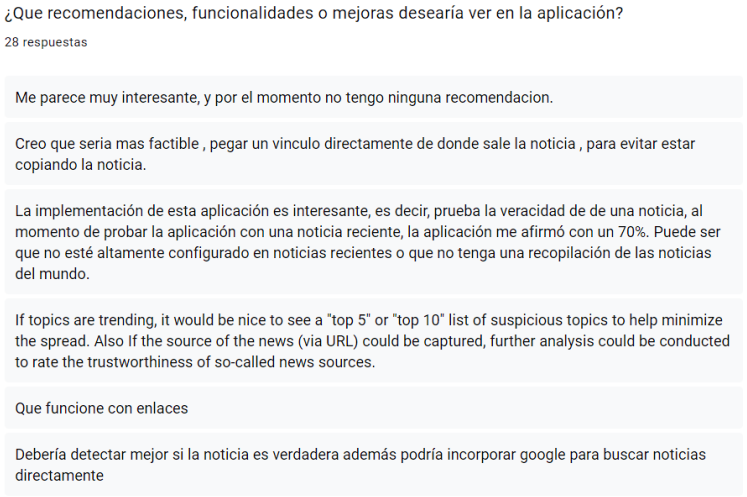


Figura 117

Preguntas de sugerencias a los usuarios para mejorar la versión 3 de la aplicación.

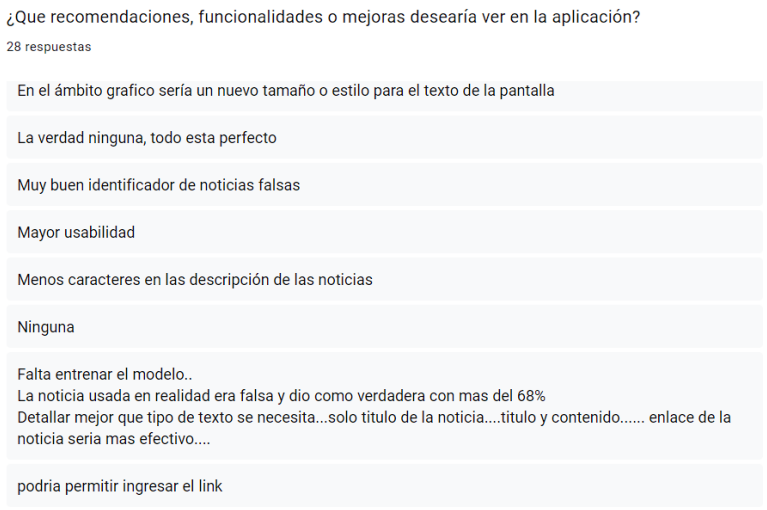


Figura 118

Preguntas de sugerencias a los usuarios para mejorar la versión 3 de la aplicación.

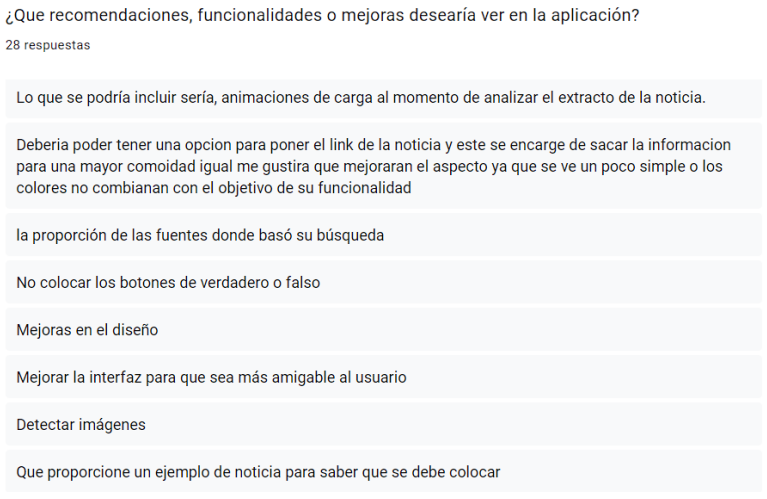
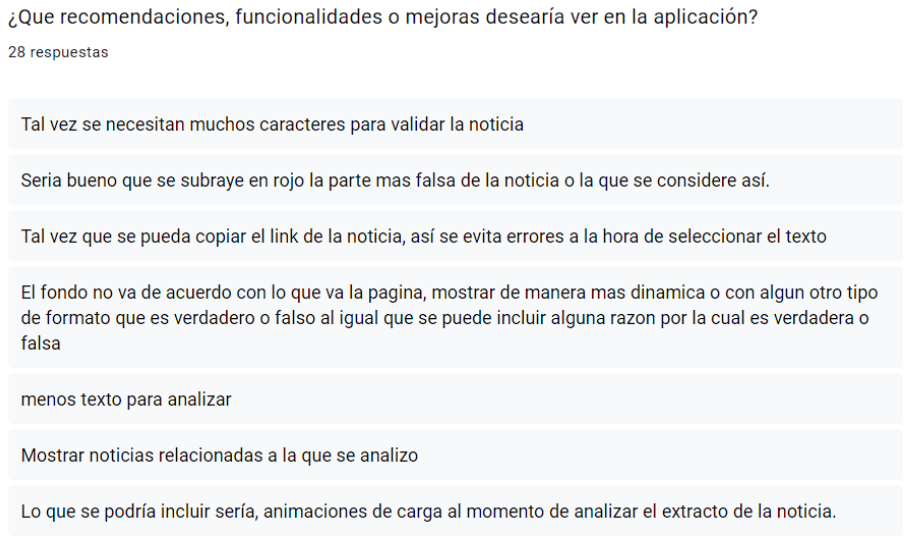


Figura 119

Preguntas de sugerencias a los usuarios para mejorar la versión 3 de la aplicación.



Encuesta de Aplicación y Modelo Entrenado Versión 1

Figura 120

Encuesta Modelo V3 Pregunta 1

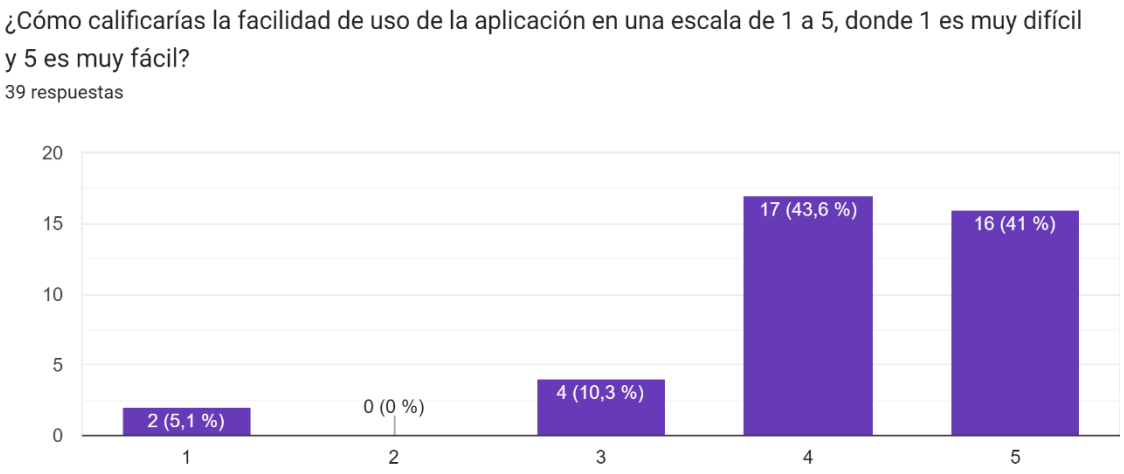


Figura 121

Encuesta Modelo V3 Pregunta 2

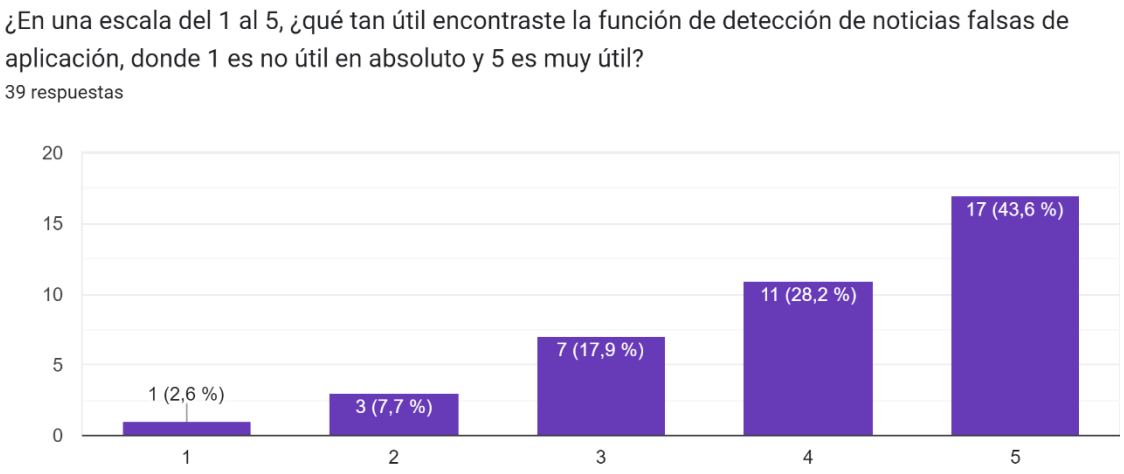


Figura 122

Encuesta Modelo V3 Pregunta 3

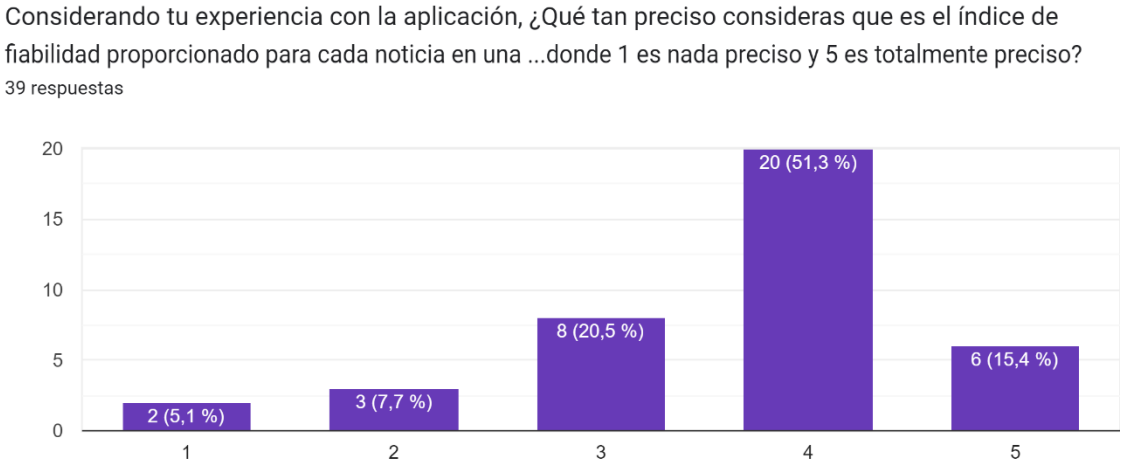


Figura 123

Encuesta Modelo V3 Pregunta 4

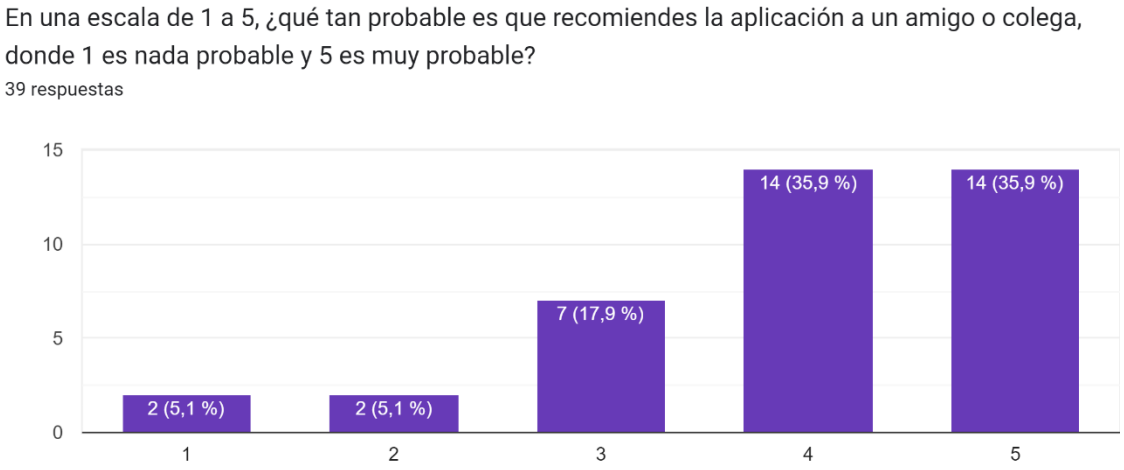


Figura 124

Encuesta Modelo V3 Pregunta 5

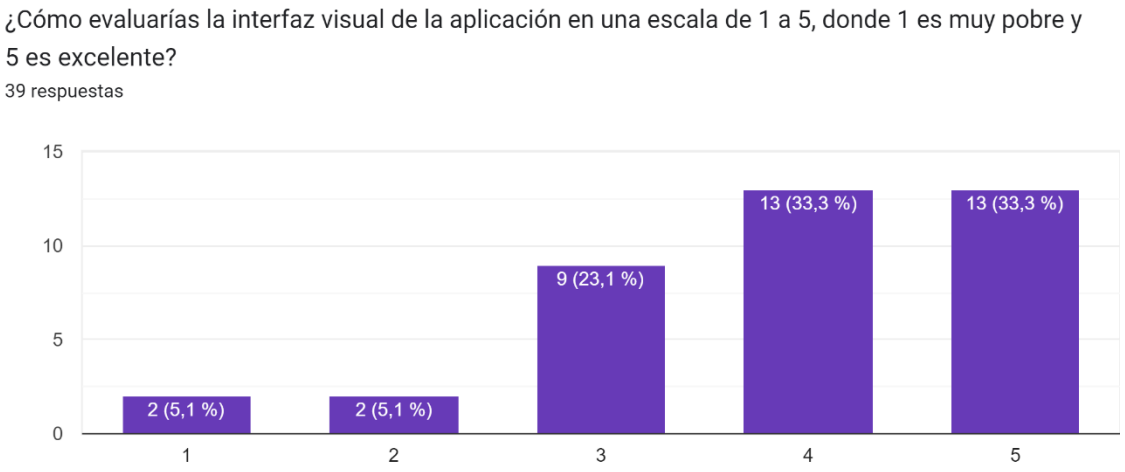
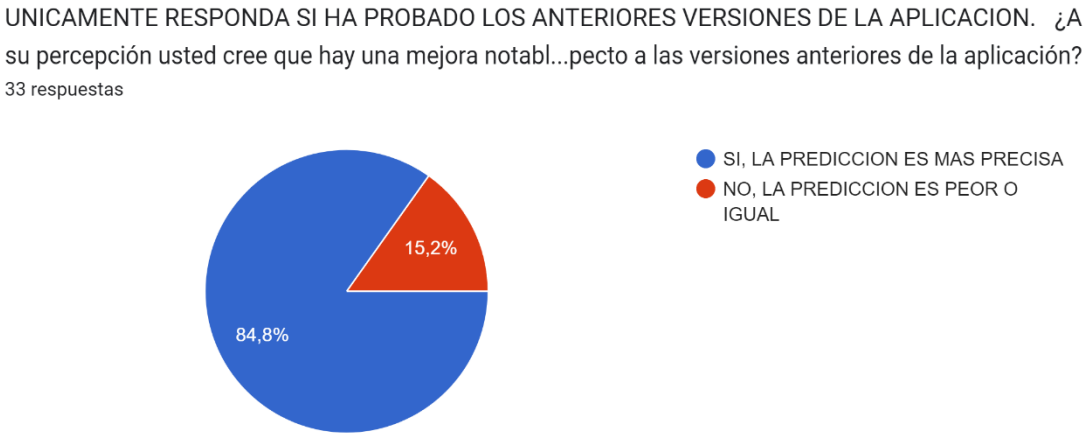


Figura 125

Comparación de la predicción del modelo V3 con respecto a modelos V1 y V2



Nota. Como se muestra a continuación la siguiente pregunta está dirigida a si el último modelo entrenado tiene una mejor aceptación para clasificar noticias. El cual el 28 de 33 usuarios afirmaron que el modelo de predicción mejoro.

Análisis de comparación entre encuestas por cada modelo.

Tabla 7

Análisis descriptivo de encuestas en Versión 1 de la aplicación.

Pregunta	1	2	3	4	5
Media	4.175	4.1	3.8	4.025	3.875
Error Estándar	0.106744	0.128103	0.125064	0.140911	0.14406
Mediana	4	4	4	4	4
Moda	4	4	4	4	4
Desviación estándar	0.675107	0.810191	0.790975	0.891196	0.911114
Variación de la muestra	0.455769	0.65641	0.625641	0.794231	0.830128
Curtosis	-0.72878	-0.44469	3.298885	1.752757	-0.22618
Skewness	-0.22335	-0.4933	-1.25663	-0.96565	-0.59885

Nota. Cada número corresponde a su respectiva pregunta.

Tabla 8

Análisis descriptivo de encuestas en Versión 2 de la aplicación.

Pregunta	1	2	3	4	5
Media	4.055556	3.666667	3.638889	3.916667	3.944444
Error Estándar	0.177927	0.199205	0.211393	0.175368	0.159087
Mediana	4	4	4	4	4
Moda	4	4	4	4	4
Desviación estándar	1.067559	1.195229	1.268357	1.052209	0.954521
Variación de la muestra	1.139683	1.428571	1.60873	1.107143	0.911111
Curtosis	2.164048	-0.40563	-0.43425	1.639642	1.633836
Skewness	-1.45841	-0.68508	-0.77552	-1.2281	-1.13676

Nota. Cada número corresponde a su respectiva pregunta.

Tabla 9

Análisis descriptivo de encuestas en Versión 3 de la aplicación.

Pregunta	1	2	3	4	5
Media	4.153846	4.025641	3.641026	3.923077	3.846154
Error Estándar	0.158171	0.174204	0.162168	0.177646	0.17823
Mediana	4	4	4	4	4
Moda	4	5	4	4	5
Desviación estándar	0.98778	1.087904	1.012739	1.1094	1.113044
Variación de la muestra	0.975709	1.183536	1.025641	1.230769	1.238866
Curtosis	3.649146	0.210574	0.884851	0.761647	0.405235
Skewness	-1.70538	-0.95781	-0.96734	-1.06022	-0.88772

Conclusiones

Conclusión Objetivo General

La herramienta web desarrollada para la detección de Fake News en Ecuador, empleando algoritmos de Machine Learning, ha evidenciado un avance significativo en la lucha contra la desinformación en el país. La combinación de datasets libres y técnicas avanzadas de aprendizaje automático ha permitido una solución robusta, precisa y accesible para el público en general.

Conclusión Objetivos Específicos:

Obtener y recolectar información de Fake News y datasets libres:

Se logró una amplia recolección de datos de Fake News de diversas fuentes, proporcionando un panorama amplio de la desinformación circulante. Estos datasets no sólo han servido para el entrenamiento del modelo, sino también como referencia para entender el panorama actual de las noticias falsas en Ecuador.

Transformar y traducir los datos obtenidos a idioma español:

Para garantizar la precisión y aplicabilidad local, se llevó a cabo un proceso minucioso de transformación y traducción de los datos, asegurando que las noticias estuvieran en un lenguaje comprensible y culturalmente relevante para los ecuatorianos.

Diseñar el motor de detección de Fake News utilizando Inteligencia Artificial:

A través del uso de la arquitectura LSTM, se diseñó un motor de detección eficiente y confiable. Esta elección no solo asegura un alto índice de fiabilidad, sino que también ofrece oportunidades para futuras actualizaciones y adaptaciones según la evolución del panorama de Fake News.

Desarrollar la aplicación web:

La interfaz de la aplicación es intuitiva y amigable, permitiendo que tanto el público general como los profesionales en áreas específicas puedan interactuar y beneficiarse de ella. Su desarrollo se basó en prácticas estándar de diseño web, asegurando la adaptabilidad y escalabilidad de la herramienta.

Probar la herramienta con datos externos de redes sociales:

Las pruebas realizadas con datos externos de redes sociales confirmaron la efectividad y precisión de la herramienta. Estos test resaltaron la capacidad de la herramienta de adaptarse y responder ante la constante evolución de las Fake News, en particular en plataformas donde la información fluye rápidamente.

La herramienta desarrollada no solo aborda un problema contemporáneo de gran relevancia, sino que también pone de manifiesto las posibilidades y capacidades del Machine Learning y la Inteligencia Artificial en el campo de la verificación de información. La aplicación tiene un gran potencial de crecimiento, y con la integración de nuevas funciones, como la extracción de texto de imágenes y el análisis de sentimientos, su relevancia y utilidad se verán incrementadas en el futuro cercano.

Recomendaciones.

Modelo de Predicción

Se observó una mayor precisión en la detección de fake news al emplear el modelo LSTM Bidireccional. Por ello, se recomienda continuar utilizando y explorando este modelo específico en futuros desarrollos o mejoras.

Aunque aumentar la dimensión de vectores puede mejorar la precisión, es esencial mantener un equilibrio. Un modelo excesivamente grande podría sobrecargar los recursos del servidor. En futuros entrenamientos, se aconseja ajustar las dimensiones de vectores en proporciones que no comprometan la eficiencia del sistema.

Calidad del Dataset:

La eficiencia del modelo depende en gran medida de la calidad y cantidad de datos con los que se entrena. Se propone la creación de una asociación en Ecuador, similar a PolitiFact, compuesta por periodistas reputados que puedan discernir y validar la información. Este esfuerzo colectivo podría mejorar sustancialmente la base de datos para la detección de noticias falsas en el contexto ecuatoriano.

Seguridad Web:

Es imperativo aplicar una certificación SSL en la aplicación web para garantizar la seguridad y confidencialidad de los datos de los usuarios. Además, en el caso de escalabilidad o amenazas de seguridad, considerar medidas adicionales como la protección contra ataques DDoS.

Interfaz de Usuario:

Durante las pruebas, algunos usuarios manifestaron dificultades en comprender la interacción con la herramienta. Se sugiere rediseñar la interfaz para que sea más intuitiva y amigable.

Aunque algunos usuarios mostraron preferencia por el uso de enlaces en lugar de texto, se debe recalcar que el núcleo de esta herramienta es el análisis textual. Se podría considerar la

inclusión de un breve tutorial o guía para el usuario, explicando la importancia y metodología del análisis basado en texto.

Preprocesamiento de Datos:

El tokenizador, elemento esencial para el análisis de texto, requiere un preprocesamiento adecuado de los datos ingresados. Para optimizar su funcionamiento y reducir posibles errores, se recomienda implementar una función que filtre y acepte únicamente letras del alfabeto español y números en la entrada de datos.

Retroalimentación Activa y Capacitación

Dado que el panorama de las noticias cambia constantemente, sería beneficioso implementar un sistema donde los usuarios puedan reportar errores o falsos negativos/positivos detectados por la herramienta. Esto no solo permitiría afinar y recalibrar el modelo con el tiempo, sino también fomentaría la participación activa de la comunidad. Además, considera ofrecer talleres o seminarios webs educativos para capacitar a los usuarios sobre cómo detectar fake news por sí mismos, complementando el uso de la herramienta.

Integración con Plataformas Sociales

Las redes sociales son uno de los principales canales de propagación de noticias falsas. Sería beneficioso explorar asociaciones o integraciones con estas plataformas, permitiendo que los usuarios verifiquen directamente las noticias desde su feed o timeline. Esta integración podría incluir un plugin o extensión que destaque o marque noticias con un alto índice de ser falsas, brindando una experiencia de navegación más segura y educativa.

Bibliografía

Ahmed H, T. I. (2017). Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. . *Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments.*, págs. 127-138.

Ahmed H, T. I. (2018). Detecting opinion spams and fake news using text classification. *Journal of Security and Privacy, Volume 1, Issue 1,*.

Allaire, J. &. (2018). Deep Learning with R. *Manning Publications Co.*

Allcott, H. &. (2017). Social media and fake news in the 2016 election. . *Journal of economic perspectives, 31(2),* , 211-236.

Álvarez Rufs, M. (2018). *Estado del arte: Posverdad y fake news.*

Alvaro Ibrain Rodriguez, L. L. (2019). Fake News detection using Deep Learning. *IFCA*, 10.

Amoros. (2018). *Fake News: La verdad de las noticias falsas.* Barcelona : Atlas .

Arjovsky, M. &. (2017). Towards principled methods for training generative adversarial networks. . *arXiv preprint*. https://doi.org/arXiv:1701.04862.

Ashkan Kazemi, Z. L.-R. (2021). Extractive and Abstractive Explanations for Fact-Checking and Evaluation of News. *arxiv.org*, 6.

Beazley, D. M. (2009). *Python essential reference. .* Addison-Wesley Professional.

Becker, K. (2017). *Manifiesto por el Desarrollo Ágil de Software.* Obtenido de Agile Manifesto: http://agilemanifesto.org/iso/es/manifesto.html

Bender, E. M.-M. (2021). On the dangers of stochastic parrots: Can language models be too big?🦜. In Proceedings of the 2021. *ACM conference on fairness, accountability, and transparency*, 610-623.

Bengio, Y. S. (1994). Learning long-term dependencies with gradient descent is difficult. . *IEEE transactions on neural networks, 5*(2), 157-166.

Bird, S. K. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. . *O'Reilly Media, Inc.*

Blom, J. N. (2015). Click bait: Forward-reference as lure in online news headlines. *Journal of Pragmatics*(76), 87-100.

Bonner, A. (2019). *www.towardsdatascience.com*. Obtenido de the beginners guide to google colab: https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c

Bostrom, N. (2014). *Superintelligence: Paths, Dangers, Strategies. .* Oxford University Press.

Breiman, L. (2001). *Statistical Modeling: The Two Cultures.* Statistical Science.

Brown, T. M. (2020). Language models are few-shot learners. *Advances in neural information processing systems , 33*, 1877-1901.

Brownlee, J. (2017). *Deep Learning With Python. .* Machine Learning Mastery.

Chen, Y. C. (2015). Misleading online content: Recognizing clickbait as false news. . *In Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection,*, 15-19.

Chollet, F. (2018). *Deep Learning with Python. .* Manning Publications Co.

Chollet, F. (2021). *Deep learning with Python. .* Simon and Schuster.

Cohn, M. (2010). *Succeeding with agile: software development using Scrum. .* Pearson Education.

Cook, D. (. (2019). *Pandas*. Obtenido de 10 minutes to pandas.: www.pandas.pydata.org/10-minutes-to-pandas

Cornejo Macías, G. S. (2021). Diseño de un modelo de aprendizaje automático Machine Learning para clasificar texto en variables PEST (Doctoral dissertation. *Universidad de Guayaquil*.

Deepanway Ghosal, N. M. (2020). COSMIC: COmmonSense knowledge for eMotion Identification in Conversations. *arxiv.org*, 12.

Di Jin, Z. J. (2020). Deep Learning for Text Style Transfer: A Survey. *arxiv.org*, 47.

Domingos, P. (2015). *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. .* Basic Books.

Doob, L. W. (1950). *Public opinion and propaganda. .* Henry Holt.

El Comercio. (2023). *El comercio*. Obtenido de Ultima Hora: www.elcomercio.com

Ellul, J. (1965). *Propaganda: The Formation of Men's Attitudes.* Vintage Books.

Fernandez-Garcia, N. (2017). Fake news: una oportunidad para la alfabetizacion mediatica. *www.nuso.org*, 12.

Forta, B. (2018). Learning Regular Expressions. *Addison-Wesley Professional.*

Friedman, M. (2006). Regular Expression Pocket Reference: Regular Expressions for Perl, Ruby, PHP, Python, C, Java and .NET. . *O'Reilly Media, Inc.*

Furnell, S. (. (2014). Cybersecurity: Is anyone really secure?. *Computer Fraud & Security,*(6), 8-13.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, Inc.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. .* O'Reilly Media, Inc.

Gers, F. A. (2000). Learning to forget: Continual prediction with LSTM. Neural computation. *MIT Press, 12*(10), 2451-2471.

Ghafir, I. S. (2018). Security threats to critical infrastructure: the human factor. *The journal of supercomputing, 74*(10), 4986-5002.

Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. . *arXiv preprint*. https://doi.org/ arXiv:1701.00160.

Goodfellow, I. B. (2016). *Deep Learning. .* MIT Press.

Goodfellow, I. P.-A.-F. (2014). *Generative adversarial nets.* Advances in neural information processing systems.

Goodrich, M. T. (2013). Data Structures and Algorithms in Python. *Wiley*.

Google. (s.f.). *Google Colab*. Obtenido de Colab Research: https://colab.research.google.com

Granda, M., & Paladines, Fanny. (2021). Gobierno del Ecuador frente a las fake news en el paro nacional de octubre 2019. *ProQuest*, 15.

Gransbury, I. (2020). *DataCamp*. Obtenido de Introduction to Jupyter Notebooks.: www.datacamp.com/introduction-to-jupyter-notebooks

Graves, A. &. (2012). *Supervised sequence labelling .* Springer Berlin Heidelberg.

Grinberg, M. (2018). *Flask web development: Developing web applications with Python.* O'Reilly Media, Inc.

Grus, J. (2019). *Data science from scratch: first principles with python. .* O'Reilly Media.

Harari, Y. N. (2015). *Homo Deus: A Brief History of Tomorrow.* Harper.

Harris, C. R. (2020). Array programming with NumPy. Nature. *Nature*, 357-362. https://doi.org/https://doi.org/10.1038/s41586-020-2649-2

Hassabis, D. K. (2017). *Neuroscience-Inspired Artificial Intelligence. .* Neuron.

Haykin, S. (1998). *Neural networks: a comprehensive foundation.* Prentice Hall PTR.

Heaton, J. (. (2016). Deep learning. *The MIT Press, 19*(1-2), 305-307. https://doi.org/ISBN: 0262035618.

Hochreiter, S. &. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780.

Howard, J. (2020). *Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning.* . Obtenido de Coursera.: www.coursera.org/introduction-to-tensor-flow

huggingface. (2023). Obtenido de Language Technology Research Group at the University of Helsinki: https://huggingface.co/Helsinki-NLP

Ifantidou, E. (2009). Newspaper headlines and relevance: Ad hoc concepts in ad hoc contexts. . *Journal of Pragmatics, 4*(41), 699-720.

Ingolfsson, T. M. (10 de November de 2021). *Thorir Mar Ingolfsson*. Obtenido de Insights into LSTM architecture: https://thorirmar.com/post/insight_into_lstm/

Jakobsson, M. &. (2007). *Phishing and countermeasures: Understanding the increasing problem of electronic identity theft. .* John Wiley & Sons.

Jowett, G. S. (2019). *Propaganda & persuasion. .* SAGE Publications.

Jupyter Project. (2022). *Jupyter Project.* Obtenido de www.jupyter.org: www.jupyter.org/about-jupyter-notebook

Kaggle. (2017). Obtenido de Fake News: https://www.kaggle.com/c/fake-news/data

Kai Shu, D. M. (2018). FakeNewsTracker: a tool for fake news collection, detection, and visualization. *Springer*, 12.

Karpathy, A. J.-F. (2015). Visualizing and understanding recurrent networks. . *arXiv preprint* . https://doi.org/arXiv:1506.02078.

Kasperky. (1 de 07 de 2023). *Kaspersly latam*. Obtenido de Cómo identificar noticias falsas: https://latam.kaspersky.com/resource-center/preemptive-safety/how-to-identify-fake-news

Kennedy, M. (2016). *miguelgrinberg*. Obtenido de The Flask mega-tutorial.: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

Kluyver, T. R.-K. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. *IOS Press*, 87-90.

Kovach, B. &. (2014). *The Elements of Journalism: What Newspeople Should Know and the Public Should Expect. .* Three Rivers Press.

Kucher, K. P. (2018). The State of the Art in Sentiment Visualization. . *Computer Graphics Forum, 37*(1), 71–96. . https://doi.org/https://doi.org/10.1111/cgf.13217

Kuklinski, P. (28 de noviembre de 2016). *La microfisica de la posverdad*. Obtenido de digitalisimo: http://digitalismo.com/la-microfisica-de-la-posverdad/

Kumar, P. &. (2021). *Exploratory data analysis using Python: A practical approach for business and data science.* BPB Publications.

Lara, R. U. (Diciembre de 2022). Paro nacional indígena y movilización social en Ecuador. *El trayecto de Octubre 2019 a Junio 2022. Nº 34*, pág. 56.

Larrea, A., Galarza, A., Chamorro Loyo, A., Guerrero Pasquel, M., Cadena , A., & Mestanza , E. (2022). Incidencia de los medios de comunicacion, redes sociales y fake news durante las manifestaciones sociales de octubre 2019 en Ecuador. *ECOSACADEMIA*, 17.

Lecun, Y. (2019). Deep learning hardware: past, present, and future. (arXiv preprint). https://doi.org/arXiv:1903.11621.

LeCun, Y. B. (2015). *Deep Learning. .* Nature.

López, N. (2018). Campaña presidencial de Donald Trump. In Libro de actas 2017: 1º Congreso de la Red Iberoamericana de Investigadores en Publicidad, Capítulo Argentino, Homenaje a Antonio Caro Almela 2º Congreso de Publicidad . Universidad Nacional de Tucuman.

Lutz, M. (2013). *Learning python: Powerful object-oriented programming.* O'Reilly Media, Inc.

McChesney, R. W. (2013). *Digital Disconnect: How Capitalism is Turning the Internet Against Democracy. .* The New Press.

McCombs, M. E. (1972). The Agenda-Setting Function of Mass Media. *Oxford: Oxford University Press.*

McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. *O'Reilly Media, Inc.*

McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.* O'Reilly Media, Inc.

McKinney, W. (2022). *Python for data analysis.* O'Reilly Media, Inc.

MeiXing Dong, X. X. (2021). Room to Grow: Understanding Personal Characteristics Behind Self Improvement Using Social Media. *arxiv.org*, 10.

METRO ECUADOR. (21 de JUNIO de 2022). *METRO ECUADOR*. Obtenido de cuanto duro el paro nacional de octubre de 2019: https://www.metroecuador.com.ec/noticias/2022/06/21/cuanto-duro-el-paro-nacional-de-octubre-de-2019/

Mitchell, R. (2015). Web Scraping with Python: Collecting More Data from the Modern Web. . *O'Reilly Media, Inc.*

Mitchell, T. M. (1997). *Machine Learning. .* McGraw-Hill.

Müller, A. C. (2016). *Introduction to machine learning with Python: a guide for data scientists.* O'Reilly Media, Inc.

Oord, A. V. (2016). Wavenet: A generative model for raw audio. *arXiv preprint* . https://doi.org/arXiv:1609.03499.

Open AI. (2023). *OpenAI*. Obtenido de Guides GPT: https://platform.openai.com/docs/guides/gpt.

P.Russell, S. (2016). *Artificial Intelligence: A Modern Approach.* Pearson.

Paszke, A. G. (2019). *Pytorch: An imperative style, high-performance deep learning library.* Advances in neural information processing systems.

Paszke, A. G. (2019). *PyTorch: An imperative style, high-performance deep learning library. .* Advances in neural information processing systems, 32.

Pedregosa, F. V. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*(oct-12), 2825-2830.

Perkins, J. (2010). Python Text Processing with NLTK 2.0 Cookbook. *Packt Publishing Ltd.*

Pohl, I. L. (2021). Best Practices for Jupyter Notebook for Effective and Efficient Collaboration. Frontiers in Big Data. *DOI.org*, 4.

Potthast, M. K. (2018). Clickbait Detection. *In European Conference on Information Retrieval*, 215-228.

Radford, A. M. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. . *arXiv preprint* . https://doi.org/arXiv:1511.06434.

Radford, A. W. (2019). *Language models are unsupervised multitask learners.* OpenAI blog.

Raffel, C. S. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. . *The Journal of Machine Learning Research, , 21*(1), 5485-5551.

Raghavan, V. &. (2020). *Visualizing Qualitative Data: The Word Cloud. .* The Political Methodologist. .

Raschka, S. &. (2019). *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2.* Packt Publishing Ltd.

Reback, J. M. (2020). Pandas. *Zenodo*. https://doi.org/https://doi.org/10.5281/zenodo.3509134

Rivero, E. A. (2020). Fake News, trolls y otros encantos. Cómo funcionan (para bien y para mal) las redes sociales. En N. A. Ernesto Calvo. Revista mexicana de opinión pública.

Rojas, R. (2013). *Neural networks: a systematic introduction. .* Springer Science & Business Media.

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process. .* Addison-Wesley.

Russell, M. (2020). Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Instagram, GitHub, and More. *O'Reilly Media, Inc.*

Samper, J. J. (2017). Introducción a los sistemas expertos. *Scalahed, 1*(1), 1-10. Obtenido de https://gc.scalahed.com/recursos/files/r161r/w21782w/Introduccion%20a%20los%20Sistemas%20Expertos.pdf#:~:text=Los%20sistemas%20expertos%20se%20pueden%20considerar%20como%20el,que%20tenga%20almacenado%20y%20algunos%20m%C3%A9todos%20de%20inferencia.

Samuel, A. L. (1959). *Some Studies in Machine Learning Using the Game of Checkers. .* IBM Journal of Research and Development.

Santiago Castro, R. W. (2021). Fill-in-the-blank as a Challenging Video Understanding Evaluation Framework. *arxiv.org*, 8.

Schneider, F. B. (2015). *Invitation to Computer Science.* Cengage Learning.

Schwaber, K. &. (2001). *Agile software development with Scrum.* Prentice Hall PTR.

Sebastian Raschka, Y. (. (2022). Training Simple Machine Learning Algorithms for Classification. En D. Dzhulgakov, *Machine Learning with PyTorch and Scikit-Learn* (pág. 775). PACKT.

Sepúlveda, J. M. (s.f.). Propuesta de aplicación de Scrumban para gestionar el proceso de generación de proyectos de I+D+i con el modelo Canvas: estudio preliminar. *Universidad EAFIT,* *82*(1), 1-43. Obtenido de https://repository.eafit.edu.co/handle/10784/11355#.WhfcTcZryUl

Shu, K. W. (2020). *Beyond news contents: The role of social context for fake news detection. .* ACM Transactions on Knowledge Discovery from Data (TKDD), 14(2), 1-26.

Singh, A. (2022). *Medium*. Obtenido de Pros and Cons of Google Colab for Data Scientists: www.medium.com/pros-and-cons-of-google-colab-for-data-scientists

Smirnov, E. A. (2014). Comparison of regularization methods for imagenet classification with deep convolutional neural networks. . *Aasri Procedia, 6*, 89-94.

Soujanya Poria, N. M. (2020). Recognizing Emotion Cause in COnversations. *arxiv.org*, 15.

Stanley, J. (2015). *How Propaganda Works. .* Princeton University Press.

Sutherland, J. &. (2014). *Scrum: the art of doing twice the work in half the time.* Currency.

Szegedy, C. L. (2015). *Going deeper with convolutions.* Proceedings of the IEEE conference on computer vision and pattern recognition.

Tandoc, E. C. (2018). Defining “Fake News”. *Digital Journalism, 6*(2), 137–153.

Tegmark, M. (2017). *Life 3.0: Being Human in the Age of Artificial Intelligence. .* Knopf.

Theodo. (2017). *How to set up a server on Amazon EC2. Recuperado de*. Obtenido de https://www.theodo.fr/blog/2016/03/how-to-set-up-a-server-on-amazon-ec2/

Torrico, M. (2022). Giro a la derecha. Un nuevo ciclo político en América Latina. *FLACSO Mexico.*

UN. (13 de 08 de 2021). *UN*. Obtenido de United Nations: https://news.un.org/en/audio/2018/05/1008682

unamglobal. (06 de 07 de 2021). Obtenido de Unam Global: https://unamglobal.unam.mx/que-son-las-fake-news/

Ustarroz Molina, M. (. (2021). Del fenómeno de la desinformación: marco conceptual y análisis comparativo del marco legal en la Unión Europea. *Deposit Digital*, 3.

Ustarroz Molina, M. (2021). Del fenómeno de la desinformación: marco conceptual y análisis comparativo del marco legal en la Unión Europea. *Deposit UB*, 3.

Valenzuela, J. C. (2020). Deteccion de Fake News mediante tecnicas de Deep Learning. 202.

VanderPlas, J. (2016). Python Data Science Handbook: Essential Tools for Working with Data. . *O'Reilly Media, Inc.*

Walt, S. V. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 22-30.

Waskom, M. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, 60. https://doi.org/ https://doi.org/10.21105/joss.03021

Wentworth, P. E. (2012). *How To think like a computer scientist: Learning with Python 3 documentation.* Open Book Project.

What are stop words? (s.f.). Obtenido de Kavita: https://kavita-ganesan.com/what-are-stop-words/#.ZKXWh3bMJPY

Yiqun Yao, M. P. (2021). MUSER: MUltimodal Stress Detection using Emotion Recognition as an Auxiliary Task. *arxiv.org*, 12.

Zeiler, M. D. (2014 September 6-12). Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014: 13th European Conference. *Proceedings, Part I 13* (págs. 818-833). Zurich, Switzerland: Springer International Publishing.

Zhang, Y. Z. (2020). *Fake news: a survey of research, detection methods, and opportunities. .* ACM Computing Surveys (CSUR), 53(5), 1-37.