Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería de Sistemas

Bebot: Detección de bots en Twitter usando procesamiento de lenguaje natural y modelos de predicción

Tesis presentada por los Bachiller:

Cornejo Tejada, Diego Sebastian

ORCID: 0009-0003-8520-0262

Ylaquita Atencio, Jorge Mauricio

ORCID: 0009-0002-2289-5610

para optar el Título Profesional de Ingeniero de Sistemas

Asesor (a):

Dra.Castro Gutierrez Eveling Gloria

ORCID: 0000-0002-0203-041X

Arequipa – Perú

2025

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA DE SISTEMAS

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 02 de Enero del 2024

Dictamen: 009457-C-EPIS-2024

Visto el borrador del expediente 009457, presentado por:

2018700581 - CORNEJO TEJADA DIEGO SEBASTIAN 2018210331 - YLAQUITA ATENCIO JORGE MAURICIO

Titulado:

BEBOT:DETECCIÓN DE BOTS EN TWITTER USANDO PROCESAMIENTO DE LENGUAJE NATURAL Y MODELOS DE PREDICCIÓN

Nuestro dictamen es:

APROBADO

Titulo Profesional/Titulo de Segunda Especialidad/Grado Académico a optar:

INGENIERO DE SISTEMAS

29217790 - TORRES GAMARRA NESTOR
DICTAMINADOR

29601217 - ROSAS PAREDES KARINA
DICTAMINADOR

29413196 - MONTESINOS MURILLO ANGEL FELIPE
DICTAMINADOR

Bebot: detección de bots en Twitter usando procesamiento de lenguaje natural y modelos de predicción

INFORME DE ORIGINALIDAD

4%	6%	3%	1%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	vsip.info Fuente de Internet	2%
2	repositorio.ucv.edu.pe Fuente de Internet	1%
3	Submitted to Universidad Católica de Santa María Trabajo del estudiante	1%
4	cybertesis.unmsm.edu.pe Fuente de Internet	1%
5	hdl.handle.net Fuente de Internet	1%

Excluir citas	Apagado	Excluir coincidencias	< 1%
Excluir bibliografía	Apagado		

Dedicatoria

Esta tesis la decido a mi madre, mi padre y a mis abuelos, gracias por estar siempre apoyándome y no rendirse conmigo en mis peores momentos, esta tesis es un conjunto de todo mi esfuerzo en estos 5 años de carrera y la prueba de todos los esfuerzos que realice al estudiarla. Espero que refleje el todo el amor que les tengo y el aprecio por todas las cosas que han hecho por mi, este trabajo se los decido a ustedes

Diego Cornejo

A mi madre, por siempre estar en los mejores y más importantes momentos de mi vida, por sus consejos que han sido de mucha ayuda para mi crecimiento en mi vida entera. Esta tesis es el resultado de todo lo que me has enseñado en la vida, ya que siempre has sido una persona muy alegre y entregada al trabajo, pero más importante que todo eso has sido una gran madre para mí y mis hermanos siempre dándonos lo mejor de ti. Por eso y muchas cosas más te dedico este trabajo de tesis. Gracias por todo, un beso al cielo; te amo.

Jorge Ylaquita

.

.

Agradecimiento

Agradecer principalmente a mi familia, en especial a mis abuelos Jaime Atencio y Ruth García por su apoyo incondicional al impulsarme y no dejar que abandone mi trabajo a mis tíos Jaime Atencio y César Atencio por haberme brindado su apoyo para seguir adelante a pesar de las adversidades que se nos presentaron a todos, a mi asesora de la tesis Eveling Castro por su dedicación y paciencia que nos brindo en esta etapa de realización de la tesis, a mi padre Javier Ylaquita por el apoyo material y económico en los estudios y no abandonarlos.

Jorge Ylaquita

Gracias a mi madre Jania Tejada, a mi padre Milko Rolando y a mis abuelos Rolando Tejada y Marcela Deglane por están siempre conmigo y criarme de la mejor manera posible, el mismo hecho que halla logrado graduarme es gracias a ustedes y su completo soporte a mi persona. Gracias a nuestra asesora la Doctora Eveling por tener tanta paciencia con nosotros y ayudarnos a brindar el mejor trabajo posible, del mismo modo a todos los ingenieros que nos enseñaron todos nuestros conocimientos haciendo posible la realización de este trabajo

Diego Cornejo

RESUMEN

En Twitter, surge el concepto de cuentas Bots, que son programas que realizan tareas de forma automática, son usadas para el engaño de identidad. El crecimiento y la evolución de estos Bots es preocupante para los consumidores de esta red, siendo estos Bots usados como spammers, estafadores y ciberacosadores. Este estudio presenta el proyecto "BeBot: Detección de Bots en Twitter Usando Procesamiento de Lenguaje Natural y Modelos de Predicción". El objetivo ha sido desarrollar un sistema para detectar Bots en Twitter utilizando técnicas de web scraping, procesamiento de lenguaje natural (PLN) y modelos de predicción como Random Forest, BERT, ELMO y SVM, implementados en Python. Se adoptó la metodología CRISP-DM, que consta de fases como comprensión del negocio, comprensión de los datos, preparación de los datos, modelado de los datos, evaluación y despliegue. El sistema propuesto logró una precisión del 94% en la detección de Bots en la plataforma de Twitter, demostrando la viabilidad de combinar el procesamiento de lenguaje natural con técnicas de BERT Embeddings. Además, se desarrolló una plataforma web para los usuarios de Twitter, que normaliza los perfiles mediante limpieza de datos y BERT Embeddings, utilizando el modelo desarrollado. Este proyecto representa una contribución al campo de la detección de Bots en Twitter al combinar de manera innovadora el procesamiento de lenguaje natural y los modelos de predicción. En trabajos futuros, se planea mejorar el sistema mediante el uso de otras bases de datos, explorando diferentes modelos de predicción y técnicas de normalización de datos.

Palabras Clave:

PLN, Twitter, BERT, Bots, Random Forest, SVM, Redes neuronales, Modelos de Predicción.

ABSTRACT

In Twitter, the concept of Bot accounts emerges, which are programs that perform tasks automatically and are used for identity deception. The growth and evolution of these Bots are concerning for users of this platform, as they are used as spammers, scammers, and cyberbullies. This study presents the project "BeBot: Bot Detection on Twitter Using Natural Language Processing and Prediction Models."The objective was to develop a system for detecting Bots on Twitter using techniques such as web scraping, natural language processing (NLP), and prediction models like Random Forest, BERT, ELMO, and SVM, implemented in Python. The CRISP-DM methodology was adopted, consisting of phases such as business understanding, data understanding, data preparation, data modeling, evaluation, and deployment.

The system achieved a 94% accuracy in detecting Bots on the Twitter platform, demonstrating the feasibility of combining natural language processing with BERT Embeddings techniques. Additionally, a web platform was developed for Twitter users, which normalizes profiles through data cleaning and BERT Embeddings, using the developed model.

This project represents a contribution to the field of Bot detection on Twitter by innovatively combining natural language processing and prediction models. In future work, the system aims to be improved by utilizing other databases, exploring different prediction models, and data normalization techniques.

Keywords:

PLN, Twitter, BERT, Bots, Random Forest, SVM, Neural Networks, Prediction Models.

Índice

Índice de tablas

Índice de tablas

INTRODUCCIÓN

La presente investigación se enfoca en temas sobre la detección de Bots en Twitter, usando Procesamiento de Lenguaje Natural (PLN) para encontrar patrones en las interacciones de las personas que existe en Bots creadas por técnicos de IA. En el presente proyecto se usan modelos de predicción que utilizan ciertas características de un perfil para desarrollar un modelo aprovechando las técnicas de predicción del Random Forest. Últimamente, la característica más resultante de los Bots en Twitter es que siguen un patrón que son los ataques hacia cuentas de personalidades o alguna cuenta en específico, estos Bots también son utilizados para generar spam cada día sobre algún tema en concreto, la mayoría de estas cuentas se crean de forma automatizada, pero son operadas por otras personas. Los Bots en Twitter pueden realizar Tweets, hacer ReTweets, seguir a otras cuentas a la par que pueden ser seguidas; todas estas acciones pueden ser realizadas por cualquier otra cuenta, lo que le añade una alta dificultad al momento analizar sus acciones y clasificarlas como Bot o una cuenta real.

El presente documento está organizado en capítulos: CAPÍTULO I. Propone con la el planteamiento del problema, el objetivo principal y los específicos; se realizan las preguntas de investigación y se plantea la hipótesis y sus correspondientes variables, indicadores, la justificación, descripción de la solución y el aporte del presente trabajo de investigación. En el CAPÍTULO II, se presenta los antecedentes de la investigación, el estado del arte y las bases teóricas que se han utilizado para conceptualizar las técnicas usadas. En el CAPÍTULO III se observan los alcances y limitaciones, los aspectos que se van a investigar y el tipo y nivel de investigación, la población, el universo de esta investigación, un plan de análisis estadístico y el análisis con un estudio económico. En el CAPÍTULO IV, se desarrolla la metodología de CRISP-DM describiendo sus fases enfocadas en el desarrollo de la investigación. Finalmente, en el CAPÍTULO V, se describe el rendimiento del modelo creado finalizando la presente investigación con DISCUSIÓN, CONCLUSIONES, RECOMENDACIONES y las REFERENCIAS respectivas.

CAPITULO I PLAN DE LA INVESTIGACIÓN

En el presente capítulo se observa el planteamiento del problema, además se plantea el objetivo general y objetivos específicos que van de la mano con las preguntas de investigación y el proceso que se realizará en el documento; a continuación, se detalla a qué línea y sublínea, las palabras claves y una solución que proponemos para el presente trabajo de investigación.

1. Planteamiento del problema

Según la División de Investigación de Delitos de Alta Tecnología del Perú el fraude informático el 2020 ofreció un 23 por ciento en comparación al 2018 (Pichihua, 2018). Mientras más crecen las redes sociales incrementan las funcionalidades que estas proveen y se vuelven una herramienta donde uno puede ofrecer servicios, realizar encuestas y socializar, pero mientras estas funcionalidades crecen también se incrementan los fraudes informáticos como puede ser phishing, suplantación de identidad y fraude en e-commerce, estas acciones son mayormente realizadas por programas que realiza tareas automáticamente denominados ”roBots” o abreviados Bot.

En el artículo de (Walt & Eloff, 2018) nos explican que las cuentas Bots son usadas para el engaño de identidad principalmente en las redes sociales como Twitter, este crecimiento y la evolución de estos Bots es preocupante para los consumidores de esta red, haciendo que estos Bots sean principalmente usados para spammers, estafadores y ciberacosadores. El sitio de (“TWTR. N - Twitter Inc”, 2022), Twitter estimó que aproximadamente el 5% de los 229.000.000 usuarios activos de Twitter son Bots más esta cifra no convenció a Elon Musk el cual aseguro que el número era cuatro veces mayor dando problemas a la hora de comprar Twitter, ya que gracias a los Bots la cifra de usuario reales que le daba valor a Twitter como compañía no era real. Este

crecimiento exponencial de Bots hace que la difusión de spam como noticias falsas genere un ambiente tenso tanto en los usuarios, como la región en la que se encuentren ubicados alterando y desfigurando la originalidad de nuestra sociedad y nuestros valores culturales, por estas razones es necesario detectar estos Bots y posteriormente eliminarlos de las redes sociales.

2. Objetivos de la Investigación

2.1. Objetivo General

Clasificar usuarios de Twitter como humanos o Bots aplicando técnicas de procesamiento de lenguaje natural y modelos de predicción para evitar la generación de mensajes “spam” o la generación de tareas repetitivas de forma automática.

2.2. Objetivos Específicos

1. Extraer información de perfiles públicos de Twitter utilizando la API de Twitter.
2. Preprocesar la información extraída de Twitter para asegurar la calidad de los datos.
3. Procesar el texto de los tweets empleando modelos avanzados de Procesamiento de Lenguaje Natural como BERT y ELMo.
4. Entrenar modelos de predicción utilizando algoritmos de aprendizaje automático como Random Forest, Máquinas de Vectores de Soporte (SVM) y Redes Neuronales.
5. Clasificar a los usuarios de Twitter como humanos o bots utilizando los modelos entrenados.
6. Validar la precisión y efectividad de la clasificación de usuarios de Twitter como humanos o bots.
7. Desarrollar e implementar una plataforma web denominada BeBot, que utilice la combinación de modelos BERT y Random Forest, con el propósito de ofrecer

una herramienta práctica y accesible para la clasificación de usuarios de Twitter como bots o humanos.

3. Preguntas de Investigación

3.1. Preguntas

1. ¿Es posible detectar Bots en Twitter usando procesamiento de lenguaje natural y modelos de predicción?
2. ¿Es posible elaborar el preprocesamiento de la información extraída de Twitter?
3. ¿Qué modelo de predicción es el más eficiente para el entrenamiento del modelo BERT con la información procesada?
4. ¿Es posible que un modelo de predicción pueda clasificar usuarios de Twitter como humanos o Bots?

3.2. Hipótesis

Dada las características extraídas de las cuentas de usuarios de Twitter utilizando aprendizaje por transferencia con los modelos del Procesamiento de Lenguaje Natural (PLN), es posible combinar estos modelos con un modelo de predicción que pueda ser capaz de clasificar una cuenta de usuario de Twitter como humano o Bot, y así evitar el ruido en los Tweets.

3.3. Variables de investigación

	Variables	Indicadores	Indices
Variables Independientes:	Modelo de Procesamiento de Lenguaje Natural (PLN)	Precisión	Rango de valores: 0% a 100%
		F1-Score	Rango de valores: 0% a 100%
	Modelo de Predicción	Precisión	Rango de valores: 0% a 100%
		F1-Score	Rango de valores: 0% a 100%
		Exactitud	Rango de valores: 0% a 100%
		Recall	Rango de valores: 0% a 100%
Variable Dependiente:	Clasificación de una cuenta de usuario de Twitter como humano o Bot	Humano	Verdadero/Falso
		Bot	Verdadero/Falso

4. Línea y Sublínea de Investigación

- Línea de Investigación: Inteligencia Artificial.
- SubLínea de Investigación: Inteligencia artificial y computacional.

5. Palabras Clave

PLN, Twitter, BERT, Bots, Random Forest, SVM, Redes neuronales, Modelos de Predicción.

6. Solución Propuesta

6.1. Justificación e Importancia

Los medios sociales irrumpieron en la vida cotidiana de muchas personas, con el poder de cambiar no solo las herramientas de comunicación de la gente, sino también las opiniones y a veces las vidas de las personas. Las redes sociales se han convertido en un componente estructural de las herramientas de comunicación para individuos y

organizaciones. Por ejemplo Facebook solo tenía 2.375 millones de usuarios en el primer trimestre de 2019 (“Number of monthly active Facebook users worldwide as of 2nd quarter 2022”, 2022), que representan casi un tercio de la población mundial. Al surgir el poder de las redes sociales, éstas se convirtieron en un arma al alcance de cualquiera. Aunque existen usos adecuados, muchos buscadores de influencia y organizaciones malintencionadas utilizan las redes sociales con fines ocultos. Para aprovechar el poder de los medios sociales, los individuos u organizaciones necesitan ganar influencia en los medios sociales, de ahí que se creen los Bots de las redes sociales(Orabi et al., 2020). Este crecimiento exponencial de Bots hace que la difusión de spam como noticias falsas genere un ambiente tenso tanto en los usuarios, como la región en la que se encuentren ubicados alterando y desfigurando la originalidad de nuestra sociedad y nuestros valores culturales, por estas razones es necesario detectar estos Bots y posteriormente eliminarlos de las redes sociales.

Esta investigación beneficiará a los usuarios de Twitter que posean cuentas de negocios o espacios sociales que hagan y/o participen en actividades sociales. Al implementar esta investigación los usuarios podrán realizar sus actividades sin tener que preocuparse de que sus publicaciones u otros usuarios contengan el apoyo e influencia de estos Bots en forma masiva; las herramientas que se usaron para la detección de estos Bots fueron modelos de predicción supervisados junto con modelos de PLN que se encargaron de analizar todas los Tweets de las cuentas las cuales serán extraídas con la API oficial de

Twitter.

6.2. Descripción de la solución

Se utilizó la API de Twitter para extraer información de cuentas de Twitter y poder usarla en el modelo de predicción. Esta información se definió como texto de Tweets, nombre, usuario, fecha de creación, verificado, idioma, cantidad a los que

seguía, cantidad de seguidores y zona horaria. En cuanto al texto de las publicaciones, se aplicaron unos pasos en los cuales se limpió el texto, se procesó el texto, se extrajeron las reacciones y se aplicó PLN, lo cual nos permitió aprovechar los datos en texto que no se pudieran pasar a números. Las fases por las cuales se procesó la información fueron:

1. Preprocesamiento: en esta etapa se prepararon los datos para poder ser procesados por las siguientes fases, se eliminaron los caracteres innecesarios y se colocó en un formato adecuado para procesar.
2. En la siguiente etapa se obtuvieron las reacciones de los Tweets como los "me gusta", las respuestas y los ReTweets.
3. Finalmente, se aplicó PLN al texto ya procesado, el resultado de aplicar PLN fue usado como una característica para el modelo.

Después de aplicar estos pasos al texto, comienza el entrenamiento del modelo de predicción como Random Forest usando toda la data que se recolectó anteriormente.

6.3. Aporte

En esta investigación, se entrenará un modelo de IA con el propósito de detectar los futuros Bots maliciosos de Twitter y el Spam masivo que podrían ocasionar. El principal aporte consistirá en la aplicación de un innovador método para identificar Bots en Twitter en el idioma español, haciendo uso de técnicas de Procesamiento del Lenguaje Natural (PLN) y modelos predictivos de última generación. Se incorporarán variables novedosas, además de aquellas que ya han sido objeto de investigación en el estado del arte. La evaluación de su relevancia en el modelo, al momento de llevar a cabo la clasificación de las cuentas como Bots, contribuirá al enriquecimiento de la comunidad de Twitter, permitiendo interacciones más auténticas y transparentes en el futuro.

CAPITULO II FUNDAMENTOS TEÓRICOS.

En el presente capítulo revelamos todos los artículos en los que nos enfocamos para la realización de la presente investigación, se presentan los antecedentes de esta investigación, el estado del arte donde se expone los artículos y las bases teóricas que se han utilizado para conceptualizar las técnicas usadas en el proyecto de investigación.

1. Antecedentes de investigación

En el artículo de (Varol et al., 2017) nos habla de los peligros en las redes sociales y como estos nos dejan cada vez más vulnerables a la manipulación. Para demostrar que estas cuentas pueden ser controladas por el software o comúnmente llamado como Bots, mencionan que estos comportamientos pueden ser detectados por técnicas de aprendizaje automático supervisado. Para ello se propone un marco para extraer una gran colección de características de datos y metadatos sobre usuarios de Twitter, incluidos amigos, contenido y el análisis de sentimiento en los Tweets, usando patrones de red y analizando el tiempo de actividad. Cuando se mezclan diferentes proporciones de cuentas anotadas manualmente y el conjunto de datos HoneyPot para obtener una precisión entre 0.90 y 0.94.

Por el contrario (Thiran et al., 2011), usa cuentas ya suspendidas de Twitter, utilizan un conjunto de datos para caracterizar el comportamiento y la vida útil de las cuentas de Spam, las campañas que ejecutan y el abuso generalizado de servicios web legítimos como los acortadores de URL y el alojamiento web gratuito. Los resultados muestran que el 77% de las cuentas de Spam identificadas por Twitter son suspendidas al día siguiente de su primer Tweet.

En cuanto al uso del PLN se tiene antecedentes como (**GloVe**) donde crean un nuevo modelo de regresión log-bilineal global llamado GloVe (“Global Vectors for Word Representation”) que combina las ventajas de las dos principales familias de modelos: la factorización matricial global y los métodos de ventana de contexto local. En este

método usan un modelo específico de mínimos cuadrados ponderados que se entrena palabra por palabra y así hace un uso eficiente de las estadísticas, logrando un 75% en una tarea de analogía de palabras, mientras tanto a diferencia de GloVe los autores de (Peters et al., 2018) desarrollan un nuevo tipo de representación profunda de palabras contextualizadas llamada ELMO ("Embeddings from Language Model"), usada para modelar tanto las características complejas del uso de las palabras como el modo en que estos usos varían en función del contexto lingüístico, a diferencia de GloVe este usa un modelo lingüístico bidireccional profundo. Esta configuración permite realizar un aprendizaje semisupervisado, en el que el BiLM ("Bidirectional Language Model") se preentrena a gran escala y se incorpora fácilmente a una amplia gama de arquitecturas neuronales de PLN existentes, logrando mejorar el mismo modelo de GloVe un 10%.

2. Estado del Arte

Esta investigación comienza con una breve revisión sobre el tema de la detección de Bots en Twitter de (Alothali et al., 2019) que presenta la necesidad de desarrollar tecnologías que puedan detectar Bots en las redes sociales. Presentando técnicas recientes que han surgido y que están diseñadas para diferenciar entre cuentas de Bots y cuentas humanas; las técnicas más resaltadas usadas como base para la presente investigación son Neural Network, Decision Tree y Random Forest; así mismo la medición del rendimiento como Precisión, ROC, Recall, accuracy, Error Rate, F-measure y Confusion Matrix. Alguno de los modelos supervisados como (Walt & Eloff, 2018) identificó identidades falsas usando atributos como el ratio de amigos y seguidores, se usó algoritmos como Random Forest, Adaboost y SVM donde como resultado un F1-Score de 49.75% tomando en cuenta que se usaron Bot creados manualmente; mientras que en este artículo(Bindu et al., 2018) usó una base de HoneyBot donde identifican comunidades sobrepuestas de Bot spamers por medio de HyperGraphs llegando a tener

una precisión de 0.90 con data del 2010. Con el fin de eliminar spam y Tweets malisiosos.

También hay otros artículos que tratan de unir diferentes modelos o métodos para detectar estas cuentas; como en el artículo de (Mao et al., 2022) este artículo usó grafos para poder detectar cuentas falsas de acuerdo a sus acciones como el logeo de usuarios, que comparte o a quien tiene como amigo, la razón por lo que este método se vuelve híbrido es porque usan dos clasificadores para poder calcular el nivel de credibilidad que sirve como un input para el modelo llamado SybilHunter demostrando una confianza del 94% mientras tanto el artículo (Martin-Gutierrez et al., 2021) usa también dos tipos de modelos, uno en el cual usa Embeddings junto con transformers para poder procesar el texto de los Tweets y también usarlo como un input para el modelo de Deep Learning llamado Bot-DenseNet el cual hace sus decisiones encontrando patrones escondidos y así identificar Bots, este método logro obtener un F1-Score de 77%. En cuento a redes neuronales tenemos a (**FakeBots**) que junta estas redes con máquinas de vectores de soporte para detectar Bots, el algoritmo se llama SVM-NN dio una precisión del 0.98, a diferencia de otros modelos este no necesita de una gran cantidad de variables para sacar un buen Score.

Otra técnica que se usa para detectar estos Bots es el método de clustering por ejemplo en el artículo de (Shi et al., 2019) usando la información de la red social de CyVOD de cómo se comportaban los usuarios haciendo énfasis en los clicks o toques se usó el algoritmo K-meas usando un método semi-supervisado donde los centros iniciales de dos clústeres se determinan mediante usuarios semilla etiquetados manualmente y después se agrupan consiguiendo un Score de 93,1%.

El spam, noticias falsas, engaños y la difusión de datos maliciosos en Twitter se ha vuelto preocupante en estos últimos años y para ello se necesitan modelos específicos para poder identificar las cuentas que propagan estos Tweets con fines

maliciosos. (Verma et al., 2022) utiliza un método que combina modelos RoBERT ("Robustly Optimized BERT approach"), BiLSTM y Random Forest, con datos OSN ("Organización Socialista Nacional") que contienen información de cuentas reales y falsos; el resultado obtuvo una precisión de 99% aproximadamente. En el artículo de (Inuwa-Dutse et al., 2018) recurre a un conjunto de datos sobre publicaciones no deseadas mediante un SPD automatizado, y otros datos mediante un SPD manual; en este artículo las palabras claves son importantes. Se aplicó la técnica SMOTE (Synthetic Minority Over-sampling Technique) es una técnica utilizada en el aprendizaje automático para abordar el problema del desequilibrio de clases en los datos. El análisis de correlación juega un papel crucial para lograr un rendimiento óptimo. Las características que se correlacionan perfectamente introducen redundancia y no agregan información adicional a los modelos de clasificación de Random Forest, ExtraTrees, MaxEnt, Gradient Boosting y MLP, SVM, SVM + MLP Features; en este caso la precisión máxima obtenida fue de 98%. Por último, en el artículo de (Ilias & Roussaki, 2021) se especializó en detectar cuentas maliciosas, tratan con un conjunto de datos desequilibrados que lo solucionan aplicando la técnica de SMOTE, se plantearon métodos de Machine Learning para detectar Bots en Twitter, estos se dividen en tres enfoques: a) Machine Learning supervisado: extracción de características, b) Machine Learning supervisado en redes neuronales y c) Machine Learning no supervisado en aprendizaje por refuerzo. para clasificar a los usuarios de Twitter en usuarios reales y Bots. Se probó un conjunto de algoritmos, que incluyen máquinas de SVM, TreeDesicion, Random Forest, AdaBoost, Lr, k-NN; para utilizar las características más efectivas. Para finalizar, se observó el uso de Logistic Regression y Random Forest que son adoptados para la selección y clasificación de características, que respectivamente demostró una precisión de 0.9906, lo que supera los enfoques de vanguardia existentes considerados.

Un ataque particularmente representativo es el ataque Sybil, las cuentas de Sybil son cuentas en las que se crean muchas actividades maliciosas, lo que representa una seria amenaza para la seguridad de los usuarios normales. Muchos mecanismos de detección de Sybil existentes tienen condiciones previas o suposiciones, por ejemplo, limitar el número de Bots de ataque. El impacto negativo del ataque Sybil no solo amenazará la seguridad de las redes sociales, sino que también dañará la relación de confianza entre los usuarios. Para hacerle frente a estos ataques existen artículos como (Zhou & Chen, 2020) en la cual se propone un clasificador para la predicción de víctimas, que puede mejorar en gran medida la eficiencia de la detección de Sybil. Se extrajeron seis características del conjunto de datos, se modelaron estas características para obtener un clasificador y así predecir quién
es la víctima, por cada usuario se calculó la probabilidad de convertirse en una víctima. Se Seleccionaron aleatoriamente 1000 usuarios del conjunto de datos recolectados. Las reglas de la etiqueta asignaron 97 víctimas y 903 no víctimas, el clasificador fue evaluado por TPR (tasa de verdaderos positivos), FPR (tasa de falsos positivos) en el cual la precisión que alcanzó fue de 0.98. Otra manera de detectar los engaños que se han ocasionado en Twitter se muestra en el artículo de (BalaAnand et al., 2019) que nos propone usar un algoritmo EGSLA (Algoritmo de Aprendizaje Semi-supervisado Basado en Gráficos) que utiliza técnicas de clasificación basadas en gráficos para distinguir a los usuarios falsos de un grupo de usuarios de Twitter, se realizó un sistema para recopilar contenido público generado por el usuario (Tweets, longitud del Tweet, fracción de la URL y el tiempo medio entre cada Tweet) de Twitter y almacenarlo en la base de datos. Una vez que se entrenan los modelos de ML regulados, se evalúa su efectividad en EGSLA, se comparan los resultados de detección con los obtenidos por otras técnicas de reconocimiento de usuarios falsos de última generación: teoría de juegos, KNN, SVM y árbol de decisión el rendimiento del método propuesto, que mostró una precisión del

0.90. (Al-Qurishi et al., 2018), presentan una solución que predice la confiabilidad de los nodos de Sybil y los defiende. Además, la detección de Sybil no es un proceso que pueda depender de características basadas en una sola estructura, que serían propensas a ataques a través de la colusión de Sybil. Se han usado modelo de DEEP- Regression Model y Feature Extracción utilizando Content-based features y Structure based features. Se creó un conjunto de datos D1 Y D2 donde D1 obtuvo una precisión considerable del 0.9992 con datos a partir de Tweets de mediados de diciembre de 2015 con las palabras clave “Elección de EE. UU., Elección de 2016”.

En el artículo de (Rostami & Karbasi, 2020) utilizan métodos de selección de características que se clasifican en tres grupos, incluidos el método de filtro, el método de envoltura y el método híbrido. Los resultados fueron cuentas falsas entre dos conjuntos de datos que muestran comportamientos similares entre sí, pero sus comportamientos son diferentes a los comportamientos de los usuarios humanos, que usan Mention para sus Tweets. Se realizaron Test donde el más sobresaliente usó algoritmos de Random Forest y

SVM que pudieron lograr resultados óptimos que son cercanos entre sí entre un 97.6% y 98%. La investigación de (Mazza et al., 2022) que proponen un análisis cuantitativo a gran escala, que se basa tanto en conjuntos de datos publicados por Twitter como por investigadores en los últimos años, para caracterizar la diferencia entre trolls, Bots sociales y humanos en Twitter. Estos utilizaron un clasificador de bosque aleatorio para diferenciar entre cuentas humanas y trolls. La prueba del clasificador entrenado con el mismo conjunto de datos obtuvo como resultado una puntuación F1-Score de 99%; el estudio aprovecha los datos relacionados con el perfil del usuario y los Tweets/reTweets compartidos por el usuario obtenidos de la API REST de Twitter para Bots sociales y cuentas humanas. En el artículo de (Beskow & Carley, 2018) nos hablan sobre el etiquetado de Bots por el nombre y propone una "tool-box"para realizar la detección en

varios niveles de granularidad de datos, utilizando un modelo de detección de cadenas aleatorias aplicada a los nombres de usuario para filtrar las transmisiones de Twitter para las cuentas de Bots, para medir el rendimiento utilizaron Log. regresion(0.996), Naive Bayes(0.969) y SVM(0.996) en el que todo obtuvieron más de 0.96 de precisión y F1-Score de 96%. En otro artículo estudian el comportamiento dentro de los OSN(Online Social Network) al usar cuentas falsas llamados Bots sociales. De una recopilacion de 10 millones de reTweets creaon una técninca RTbust (Retweet-Buster) que usa la extracción de funciones y creación de clústeres sin supervisión; el artículo nos cuenta que esta técnica puede llegar a identificar automáticamente patrones significativos a partir de los datos. RTbust obtiene resultados de detección de F1-score 87%.

En el artículo de (Ryffel et al., 2019) presentaron una investigación con relación al BERT y texto de Twitters el cual nos ayuda a tener un mejor concepto de cómo utilizar el BERT y acomodar a este nuevo modelo. En este trabajo se procesó el texto de los Tweets y lo usaron como una característica para predecir la probabilidad de que un tweet forme parte del dominio médico o no médico. Utilizaron diferentes modelos de aprendizaje como Logistic Regression que ajusta los datos a una regresión mediante una función logística, Random Forest para unas clasificaciones de árboles de decisiones, Naive Bayes un algoritmo probabilístico y Support Vector Machines (SVM), que sitúa las clases objetivo en un espacio multidimensional y las separa con un hiperplano. Este artículo recopilamos y lanzamos a la comunidad de investigación un conjunto de Tweets que se pueden usar como punto de referencia para la categorización de Tweets en médicos y no médicos, y desarrollamos un algoritmo de clasificación efectivo basado en aprendizaje automático y BERT. Otro ejemplo que se tiene es (Barbon et al., 2018) donde se utilizó el modelo de Random Forest logrando un 0.94 de precisión utilizando Transformación Wavelet discreta para obtener un patrón de estilo de escritura incrustado en los contenidos de los correos, usando diferentes datasets con diferentes

parametros en el mismo modelo lograron descartar modelos que podría tener un overfiting por el alto F1-Score que obtuvieron. Sobre los modelos de Random Forest que sirven para la detección de Bots tenemos como ejemplo una investigación de (Loyola-Gonzalez et al., 2019) crearon un modelo de clasificación utilizando un nuevo modelo basado en patrones y análisis de sentimiento con el contenido de los Tweets, para combinar este método con el Random Forest; el enfoque basado en patrones de contraste obtuvo resultados de clasificación superiores a 0,90 de AUC (Appropriate Use Criteria) y 0,91 de MCC (coeficiente de correlación de Matthews).

En una investigación reciente en el artículo de (Abuhassan et al., 2022) surgió la necesidad de clasificar personas con trastorno alimentario en Twitter y el modelo de BERT acompañado del LSTM (Memoria a largo plazo y corto plazo) obtuvo la mayor efectividad donde el modelo logra una F1-Score del 98% y una precisión del 0.9837 demostrando que este modelo da muy buena viabilidad de detectar a los individuos y los distingue de otras categorías utilizando sus datos biográficos. El modelo que realizaron nos sirve bastante para alcanzar el objetivo de poder utilizar el modelo de BERT para la detección de Bots en esta la presente investigación, ya que nos dan pruebas de una buena efectividad. En el artículo reciente de (Blanco & Lourenço, 2022) propone un nuevo enfoque de aprendizaje profundo para comprender mejor cómo se transmiten los sentimientos optimistas y pesimistas en las conversaciones de Twitter sobre COVID-19. Usaron LNN, CNN y Bi-LSTM, donde BERT es utilizado para extraer características semánticas y se aplican varias arquitecturas de redes neuronales para la extracción y clasificación de características. Para los modelos basados en BERT, se aplicó el tokenizador BERT para transformar el texto preprocesado y esta representación se introdujo en el codificador de transformador del BERT.

Muchas de las investigaciones implementan datasets antiguos, pero como ellos mismos indican, los Bots evolucionan y se adaptan a los tiempos, por lo que usar dataset

de años pasados podría influenciar al momento de detectar el modelo presentado, por esa razón es que (Feng et al., 2022) decidió crear el mayor dataset hasta ahora con un millón de usuarios para luego reimplementar 35 modelos representativos de detección de Bots en Twitter y evaluarlos en 9 conjuntos de datos, incluido TwiBot-22, para promover una comprensión holística del progreso de la investigación. En su trabajo implementan modelo de predicción basado en gráficos que dieron un mejor rendimiento con este dataset. Un dataset algo antiguo pero robusto es el de (Feng et al., 2021) que propuso el modelo TwiBot-20 que deseaba eliminar los problemas de datos escasos para entrenar y comprar la detección de los Bots, este fue el punto de referencia de la mayoría de artículos relacionados al tema de detección de Bots, este contenía 229 573 usuarios, 33 488 Tweets, 8 723 736 metadata del usuario y 455 958 relaciones de seguimiento; TwiBot-20 podría evaluar mejor las medidas de detección de Bots en el sentido de que contiene Bots diversificados y usuarios genuinos, lo que exige que los detectores de Bots capturen conjuntamente diferentes tipos de Bots en lugar de limitarse a un escenario de detección de Bots específico.

Hablando de NLP con BERT en el trabajo de (Gao et al., 2019) usan tres variaciones del modelo BERT base. El modelo TD-BERT usado alcanzo un nuevo rendimiento de vanguardia, en comparación con los métodos tradicionales, los modelos basados en la incrustación y las anteriores aplicaciones de BERT aunque no llego a una conclusión estadística significante. En (Li et al., 2020) se enfocan más en la parte Embedding de las palabras, introduciendo un nuevo método denominado GBCN (Gradient Boosted Chinese Restaurant Process Network) usado principalmente para mejorar y controlar la representación BERT para el análisis de sentimientos. Este modelo BERT lo aplicaron a un proceso de análisis de sentimiento que dio como resultado un F1-Score de 88%. BERT no es el unico modelo capaz de procesar el texto, modelos como GloVe que son usados en (Wei & Nguyen, 2019) logran un 98% de F1-Score usando

redes neuronales para entrenar un dataset del 2017 de Twitter, comparado a las otras técnicas esta logran un mejor resultado. Otra opción que se tiene a BERT o GloVe es ELMO el cual los autores de (Heidari et al., 2020) usan para los Tweets que extraen de las cuentas de Twitter junto a la metadata para entrenar una red neuronal con un F1-Score del 94%.

Modelos como BERT pueden ser usados para tokenizar el texto y para clasificar como lo hacen en (Heidari & Jones, 2020) donde se enfocan en sacar temas del texto para luego entrenar a una red neuronal donde consiguen un 0.94 de precisión. Mientras (Dukic et al., 2020) usan BERT para probar que la inclusión de características adicionales junto con incrustaciones contextualizadas mejora el rendimiento del modelo logrando un 0.83 de precisión. Aunque la mayoría de investigaciones usan PLN para procesar los Tweets que contienen texto sin embargo (Mohammad et al., 2020) obtienen las características del texto y su transformación de una forma cualitativa entrenando una Red Neuronal con una precisión de 0.90.

3. Bases Teóricas de la Investigación

En primer lugar para poder identificar Bots en Twitter debemos entender como estas cuentas engañan a las personas y por qué la gente cree en ellos, (SRIJAN, 2018) nos explica en su investigación que esta información falsa puede ser clasificada en intención y conocimiento, la intención denominada desinformación que tiene como motivo mover a las masas y ganar dinero mientras que la segunda el conocimiento puede estar basada en opiniones y falsas afirmaciones las cuales no serán tomadas en esta presente investigación. También nos habla de los actores los cuales pueden ser personas o Bots y las razones por las que se cae en estos engaños, los cuales pueden ser la poca habilidad del usuario para distinguir la farsa, los ataques a un mismo grupo de personas con cualidad similares o la información falsa que parezca atrayente. Principalmente, se tomará en cuenta los aspectos más importantes como los actores

implicados en la difusión de la información falsa, las razones para engañar con éxito a los lectores, la cuantificación del impacto de la información falsa, la medición de sus características en diferentes dimensiones, y finalmente, los algoritmos desarrollados para detectar la información falsa, estas características serán necesarios a la hora de recolectar las keywords y desarrollar el algoritmo de análisis.

3.1. Web Scraping

Definido por (Stenhouse, 2017), menciona que la mayoría de los datos que se muestran en las páginas web solo se pueden ver a través de un navegador, haciendo que la única forma de realizar esta práctica es manualmente, lo que puede requerir mucho tiempo y esfuerzo para recopilar grandes cantidades de datos. Esta técnica se basa en extraer datos de múltiples páginas web. Los datos se pueden almacenar en bases de datos para fines específicos. Web Scraping automatiza este proceso. En lugar de recopilar datos manualmente, las aplicaciones de web Scraping realizan la misma tarea en mucho menos tiempo. Un web scraper imita las acciones de los usuarios humanos y el acceso a los datos en la web para extraer contenido relevante. En general, los webs scrapers constan de dos componentes principales. Un rastreador web escanea los enlaces en la página y extrae datos de las páginas visitadas.

3.2. Procesamiento de lenguaje natural (PLN)

PLN es un subcampo de la informática que se centra en permitir que los ordenadores entiendan el lenguaje de forma ”natural”, como lo hacen los humanos. Por lo general, esto se refiere a tareas como la comprensión del sentimiento del texto, el reconocimiento del habla y la generación de respuestas. PLN se ha convertido en un campo que evoluciona rápidamente y sus aplicaciones han representado una gran parte de los avances de la inteligencia artificial (II, 2018).

3.3. Random Forest

Random Forest (Breiman, 2001) es un enfoque de aprendizaje en conjunto para la clasificación, en el que los ”aprendices débiles” colaboran para formar ”aprendices fuertes”, utilizando una gran colección de árboles de decisión descorrelacionados (el bosque aleatorio). Sin embargo, en lugar de desarrollar una solución basada en la salida de un único árbol profundo, el bosque aleatorio agrega la salida de un número de árboles poco profundos, formando una capa adicional al bagging. El bagging construye N predictores, utilizando árboles sucesivos e independientes, mediante el uso de muestras del conjunto de datos. Los N predictores se combinan para resolver un problema de clasificación o estimación a través de un promedio. Aunque los clasificadores individuales son aprendices débiles, todos los clasificadores combinados forman un aprendiz fuerte. Mientras que los árboles de decisión individuales experimentan una alta varianza y un alto sesgo, el bosque aleatorio promedia múltiples árboles de decisión para mejorar el rendimiento de la estimación.

Un árbol de decisión, en términos de conjunto, representa un clasificador débil. El término bosque denota el uso de un número de árboles de decisión para tomar una decisión de clasificación. El algoritmo del bosque aleatorio puede resumirse según (Mariette, 2015), el libro resume este algoritmo de la siguiente manera:

1) Para construir B árboles, seleccione n muestras bootstrap del conjunto de datos original. 2) Para cada muestra bootstrap, crezca un árbol de clasificación o de regresión. 3) En cada nodo del árbol - m variables predictoras (o subconjunto de características) se seleccionan al azar de entre todas las variables predictoras (subespacio aleatorio). - La variable predictora que proporciona la mejor división realiza la división binaria en ese nodo. - El siguiente nodo selecciona aleatoriamente otro conjunto de m variables de entre todas las variables predictoras y realiza el paso anterior. 4) Dado un nuevo conjunto de datos a clasificar, tome el voto mayoritario de

todos los subárboles B. Al promediar el conjunto de árboles, se puede reducir la varianza de la estimación final. El bosque aleatorio ofrece una buena precisión y funciona eficazmente en grandes conjuntos de datos. Es un método eficaz para estimar datos que faltan y mantiene la precisión, incluso si falta una gran parte de los datos. Además, random puede estimar la importancia relativa de una variable para la clasificación.

3.4. BERT

En el artículo de (McCormick, 2019) define a BERT como un codificador Transformer bidireccional que toma como entrada una o dos oraciones que al entrenarlo para el uso de predicción de valores usando tokens especiales para diferenciarlos, estos token siempre aparece al principio del texto y se configura específicamente para realizar las tareas de clasificación. Ambos tokens siempre son necesarios, incluso si solo tenemos una oración, e incluso si no estamos usando BERT para la clasificación. Así es como BERT fue pre-entrenado, y eso es lo que BERT espera ver. El modelo preentrenado se puede ajustar para tareas supervisadas posteriores y se ha demostrado que produce resultados de última generación en una serie de puntos de referencia de la PLN. Estos modelos se pueden usar para extraer características de lenguaje de alta calidad de sus datos de texto, o puede ajustar estos modelos en una tarea específica con sus propios datos para producir predicciones de vanguardia.

3.5. ELMO

ELMO (Embeddings from Language Models) definido por (Peters et al., 2018) es una técnica de procesamiento de lenguaje natural (PLN) que utiliza redes neuronales profundas para generar representaciones de palabras contextualizada. ELMO toma una oración como entrada y produce una representación vectorial para cada palabra en la oración. Estas representaciones se pueden utilizar como características para entrenar modelos de aprendizaje automático para tareas como clasificación de texto,

reconocimiento de entidades y traducción automática; este proceso permite que la red neuronal capture la estructura sintáctica y semántica de las oraciones.

ELMO ha demostrado un rendimiento sobresaliente en una variedad de tareas de
PLN, lo que sugiere que las representaciones de palabras contextualizadas sean una herramienta poderosa para el procesamiento de lenguaje natural.

3.6. SVM

Investigaciones sobre SVM (Support Vector Machine) como de (Walt & Eloff, 2018) y (Inuwa-Dutse et al., 2018) explican que SVM un algoritmo de aprendizaje supervisado utilizado en la clasificación y regresión de datos; donde el objetivo principal es encontrar el hiperplano que mejor separa los datos en dos o más clases. SVM utiliza vectores de soporte, que son los puntos de datos que se encuentran más cerca del hiperplano. El algoritmo trata de maximizar la distancia entre los vectores de soporte y el hiperplano, lo que se conoce como máxima margen. Este enfoque ayuda a SVM a encontrar una solución óptima que sea más generalizable a datos nuevos. SVM es una técnica poderosa y ampliamente utilizada en la clasificación de texto, reconocimiento de imágenes, detección de fraude, análisis de sentimientos y muchas otras aplicaciones de aprendizaje automático. Una de las principales ventajas de SVM es que funciona bien en conjuntos de datos pequeños o de alta dimensionalidad y puede manejar datos no lineales utilizando funciones de kernel.

3.7. Metodología CRISP-DM

En el artículo de (Purbasari, 2021) tomamos como referencia la metodología usada. El actual modelo de proceso proporciona una visión general del ciclo de vida de un proyecto. Contiene las fases de un proyecto, sus respectivas tareas y las relaciones entre estas tareas. A este nivel de descripción, no es posible identificar todas las relaciones. Esencialmente, pueden existir relaciones entre cualquier tarea de minería de

datos en función de los objetivos, los antecedentes y el interés del usuario y, sobre todo, de los datos. La secuencia de las fases no es rígida. Siempre es necesario avanzar y retroceder entre las distintas fases. Las flechas indican las dependencias más importantes y frecuentes entre las fases. Las fases de esta metodología son:

a) Comprensión del negocio. Esta fase inicial se centra en comprender los objetivos y requisitos del proyecto desde el punto de vista de la empresa.

b) Comprensión de los datos. La fase de comprensión de los datos comienza con una recopilación inicial de datos y continúa con actividades para familiarizarse con los datos, identificar los problemas de calidad de los datos, descubrir las primeras percepciones de los datos o detectar subconjuntos interesantes para formar hipóla presente investigación de información oculta.

c) Preparación de los datos. La fase de preparación de los datos abarca todas las actividades para construir el conjunto de datos final (datos que se introducirán en la(s) herramienta(s) de modelización) a partir de los datos brutos iniciales. Incluye la transformación y limpieza de los datos para las herramientas de modelización.

d) Modelado de los datos. En esta fase se seleccionan y aplican diversas técnicas de modelización y se calibran sus parámetros hasta alcanzar los valores óptimos.

Normalmente, existen varias técnicas para el mismo tipo de problema de minería de datos. Algunas técnicas tienen requisitos específicos sobre la forma de los datos. Por lo tanto, a menudo es necesario volver a la fase de preparación de los datos.

e) Evaluación. En esta fase del proyecto se ha construido un modelo (o modelos). Al final de esta fase, debe tomarse una decisión sobre el uso de los resultados.

f) Despliegue. La creación del modelo no suele ser el final del proyecto. Aunque el objetivo del modelo sea aumentar el conocimiento de los datos, los

conocimientos adquiridos deberán organizarse y presentarse de forma que el cliente pueda utilizarlos.

CAPÍTULO III MARCO METODOLÓGICO

En este capítulo se presenta el cómo se va a realizar la presente investigación, los alcances y limitaciones que indican la precisión, los aspectos que se han investigado y los que quedan fuera de esta investigación, también se especifica el tipo y nivel de investigación, se proyecta la población, muestra y universo que abarcará esta investigación, se elabora un plan de análisis estadístico de los datos presentados donde se muestran los datos recolectados y el análisis realizado, para finalizar realizamos un estudio económico en el cual se plantean las horas de trabajo y el costo que se tomó realizar la presente
investigación.

1. Alcances y limitaciones

1.1. Alcances

Para la definición del alcance de este proyecto de la presente investigación se tuvo en cuenta lo siguiente:

Lugar o espacio: Este estudio se centrará en la detección de Bots en la plataforma de Twitter. Se aclara que este enfoque estará limitado exclusivamente a Twitter y no se extenderá a otras redes sociales como Facebook u otras plataformas en línea. El trabajo se llevará a cabo en la ciudad de Arequipa, Perú.

Tiempo: El presente trabajo será llevado con la data extraída del 3 de enero del 2023 con un dataset perteneciente al año 2017(Cresci et al., 2017), el trabajo será realizado en 200 días de trabajo.

Viabilidad Técnica: Este modelo de detección de Bots es viable, ya que se tiene acceso a dataset(Cresci et al., 2017) con un ID de Twitter que fueron etiquetados de

Bots o humanos por investigaciones previas mencionadas en el estado del arte, los datos como la metadata y los Tweets pueden ser extraídos por medio del API de Twitter.

Viabilidad Económica: El financiamiento de este proyecto será asumido por los autores de este proyecto de la presente investigación.

Viabilidad Operativa: En la presente investigación los autores han adquirido las habilidades del área de especialización de PLN y modelos de IA.

1.2. Limitaciones

Los Tweets que se usen para entrenar el modelo serán los que se tome al momento de la recolección de datos, de igual forma con la metadata.

Ya que este modelo de PLN solo acepta texto que esté en inglés, solo se tomará en cuenta las cuentas de Twitter que hablen esos idiomas. Solo se tomará en cuenta las cuentas de Twitter del idioma inglés, y las que tengan las cinco últimas publicaciones de cada cuenta, debido a que la API de Twitter tiene restricciones.

2. Tipo y Nivel de Investigación

2.1. Tipo de Investigación

El presente enfoque se considera cuantitativo, ya que se planea usar un modelo de predicción para detectar un usuario como Bot.

Se considera que es una investigación de tipo transversal, por el hecho de que solo se usarán los datos que se recolecten al momento de la extracción de la metadata y los Tweets.

Pero ya que se tardó una semana en recolectar la información también sería longitudinal.

2.2. Nivel de Investigación

La presente investigación es de nivel aplicativo, debido a que se utilizará un dataset que servirá para el entrenamiento de un modelo de detección de Bots que será desplegado en una página web para medir la eficacia del modelo en un ambiente real.

3. Universo, Población y Muestra

El universo será los usuarios que usan la red Social de Twitter que según la página oficial de (Statista, 2023) que proporciona una previsión de usuarios activos de Twitter del año 2023 es de 368 millones de usuarios. La población y muestra es definido de acuerdo al dataset que seleccionamos para esta investigación, el cual cuenta con 18 796 usuarios de Twitter de los cuales 13 822 son Bots y 4 974 humanos. Para el entrenamiento de los modelos de predicción, se usó la cantidad total de usuarios analizados. Además, se extrajeron los últimos 20 Tweets de cada usuario al momento de la extracción, que se realizó en el año 2023. Para tener una comparación con un dataset más moderno y mucho más grande, se tomó una muestra del dataset TwiBot usado en el paper (Subrahmanian
et al., 2016).

4. Métodos, Técnicas e Instrumentos de Recolección de Datos

Se usará el lenguaje de programación Python junto a la API de Twitter para recolectar y procesar los datos de acuerdo con los presentes requisitos para procesarla con el modelo PLN BERT, PLN ELMO de la librería Tensorflow combinado con los algoritmos de Random Forest, SVM (Support Vector Machine), y Redes neuronales de la librería
SKLearn.

5. Plan de análisis estadístico de los datos

De manera similar, se empleó el lenguaje de programación Python para llevar a cabo el análisis estadístico de los datos depurados. En este proceso, se utilizó la librería Pandas para la manipulación eficiente de los datos y la librería Pyplot para la visualización clara y concisa de la información relevante.

La elección de Python como herramienta principal se fundamenta en su versatilidad y capacidad para manejar datos de manera eficiente. Pandas, en particular,

brindó las funcionalidades necesarias para estructurar, limpiar y preparar los datos para el análisis estadístico subsiguiente. Por su parte, Pyplot facilitó la generación de gráficos y visualizaciones que son esenciales para comprender y comunicar los patrones y tendencias en los datos.

Este enfoque en Python y las librerías asociadas permitió una implementación ágil y eficaz del plan estadístico, optimizando tanto la manipulación de datos como la representación visual de los resultados. Lo mencionado se usó para poder hacer una descripción detallada del conjunto de datos utilizado, incluyendo su tamaño, características, variables y fuentes, se realizó un análisis exploratorio de los datos para comprender su distribución, tendencias, valores atípicos y relaciones entre variables, para la representación de estos se hizo uso de gráficos, estadísticas descriptivas y técnicas de visualización

6. Estudio Económico

Para desarrollar este estudio se utilizó la herramienta de Microsoft Project que permite un cálculo más exacto del costo que requiere un proyecto, en el cual toma en cuenta que se trabaja aproximadamente en 2 horas por días durante 4 meses con un aproximado de 219 horas de trabajo. Dando un costo total de S/. 2.190,00.

Figura 1

Costo y duración del Proyecto

	HRS	FECHA INICIO	FECHA FIN	COSTO	
DESARROLLO DEL PROYECTO	**219**	**15/08/2022**	**26/11/2022**		
INVESTIGACIONES Y ESTADO DEL ARTE					
Investigación de herramientas	20	15/08/2022	22/08/2022	S/	200.00
Investigación de Artículos	20	23/08/2022	30/08/2022	S/	200.00
Investigación de Algoritmos	20	31/08/2022	07/09/2022	S/	200.00
	60			S/	600.00
DESARROLLO					
Web Scraping de Twitter	10	08/09/2022	11/09/2022	S/	100.00
Definición de variables	6	12/09/2022	14/09/2022	S/	60.00
Base de datos	8	15/09/2022	16/09/2022	S/	80.00
Desarrollo de algoritmos	20	17/09/2022	25/09/2022	S/	200.00
Validación de algoritmos	20	26/09/2022	03/10/2022	S/	200.00
Procesamioento de datos	20	04/10/2022	11/10/2022	S/	200.00
Modelos de predicción	20	12/10/2022	19/10/2022	S/	200.00
Comparación de precisión	20	20/10/2022	27/10/2022	S/	200.00
	124			S/	1,240.00
Interfaz					
Desarrollo de la Web	30	12/11/2022	03/12/2022	S/	300.00
	30			S/	300.00
Despliegue					
Lanzamiuento en el Sitio Web	5	21/11/2022	26/11/2022	S/	50.00
	5			S/	50.00
TOTAL				S/	2,190.00

Nota: La figura 1 representa el costo y duración del proyecto. (Fuente: Elaboración propia).

CAPÍTULO IV DESARROLLO DE LA PROPUESTA

En este capítulo se desarrolla el trabajo de investigación propuesto en el capítulo 1 aplicando la metodología de CRISP-DM y se describen las fases que se han trabajado durante todo el tiempo del proyecto.

1. Metodologia

El diagrama de trabajo fue implementado en la metodología CRISP-DM de acuerdo con sus fases.

1.1 Comprensión del negocio

En esta fase se comprendió los objetivos del trabajo de investigación propuesto que se transformó en una solución que resolvió el problema dividiéndose en tres partes:

a) Determinación de los objetivos. El objetivo principal se centró en clasificar las cuentas de Twitter como Bots o humanos utilizando PLN y modelos de predicción

b) Evaluación de la situación. Se contaba con diferentes tipos de conjuntos de datos que incluían campos de ID y clasificación como Bot o humano. Esto permitió utilizar la API de Twitter para extraer todos los datos necesarios para la clasificación. Sin embargo, se debía tener en cuenta que esta API tenía un límite de 900 solicitudes cada 15 minutos, lo que implicaba restricciones de tiempo y cantidad. Siguiendo la figura **??**, se realizó el preprocesamiento utilizando un lenguaje de programación como Python y se utilizó una base de datos como MySQL para una extracción y almacenamiento de datos rápidos. Además, se requería un mínimo de 8GB de RAM para algoritmos como Random Forest y redes neuronales, mientras que BERT y ELMO requerían una mayor cantidad de memoria.

c) Determinación del objetivo del aprendizaje automático. Se usó un modelo BERT para traducir el texto en una matriz que pudiera ser utilizada por el modelo de predicción. Se usó un modelo ELMO para traducir el texto en una matriz que pudiera ser utilizada por el modelo de predicción. Se desarrolló un modelo de predicción basado en Random Forest capaz de detectar si una cuenta era un Bot o un humano según las características indicadas. Se desarrolló un modelo de redes neuronales que detectó si una cuenta era un Bot o un humano según las características indicadas.

1.2. Comprensión de los datos

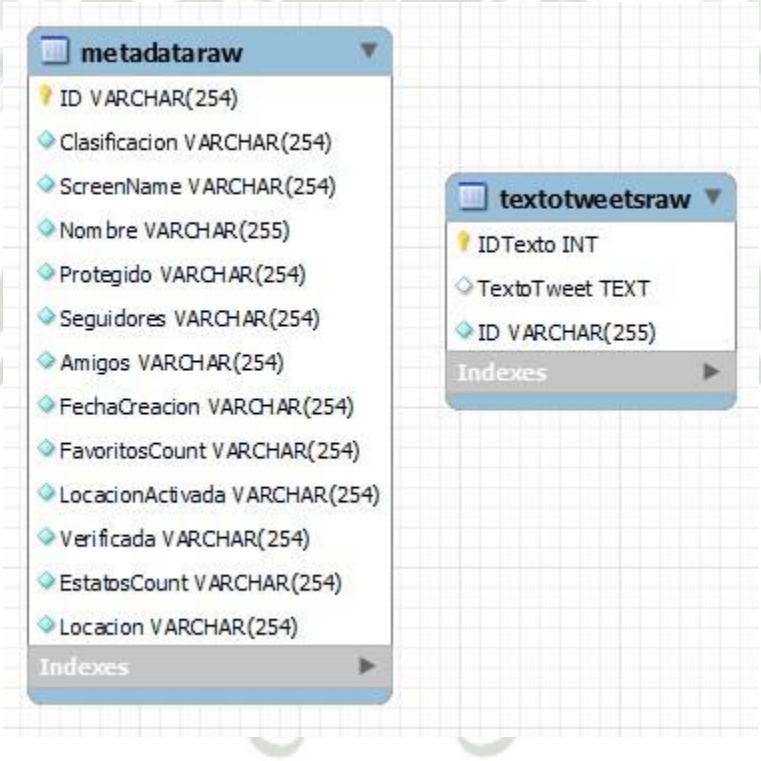
a) Recolección de los datos iniciales. Se utilizaron 20,000 cuentas del conjunto de datos de (Feng et al., 2022), las cuales incluían el ID de usuario y la etiqueta de clasificación. La información de estas cuentas se extrajo utilizando la API de Twitter mediante la biblioteca Tweepy. Se realizaron dos extracciones diferentes: una para la

obtención de la metadata, que incluye toda la información principal del perfil, y otra para el texto de los tweets de los usuarios, de los cuales se tomaron solo los últimos 20.

Toda esta información se almacenó en una base de datos MySQL. Dado que la API de Twitter devuelve la información en formato JSON, se transformó en un DataFrame para facilitar su procesamiento y se ingresó en una base de datos relacional, como se muestra en la figura 2. Se eligió esta base de datos debido a su capacidad para procesar y almacenar datos de forma rápida, además de su fácil integración con Python.

Figura 2

Modelo de Base de datos Raw



Nota: La figura 2 es un modelo de la Base de Datos Raw que se utilizó para almacenar la metada del usuario; (Fuente: Elaboración propia).

Para asegurar que los datos sin procesar se guardaran adecuadamente, se optó por almacenarlos en una base de texto en lugar de una base de datos relacional. Esta decisión se tomó para evitar posibles problemas al guardar los datos.

Es importante mencionar que no se incluyeron las cuentas protegidas o bloqueadas por Twitter, ya que no se podía extraer información de ellas. Se consideró que solo las cuentas accesibles y públicas serían incluidas en el estudio.

La Tabla 1 muestra un ejemplo de cómo se almacenó parte de la información en la base de texto. Este enfoque permitió preservar los datos sin procesar y asegurar su integridad durante el desarrollo del proyecto.

Tabla 1

Base de datos RAW

ID	Clasificacion	ScreenName	Nombre	Protegido	Seguidores	Amigos	FechaCreacion	FavoritosCount	LocacionActivada	Verificada	EstatosCount	Locacion
1000023384531636224	bot	AnoukvanMaris	Anouk van Maris, PhD	0	460.0	357.0	2018-05-25 14:38:49	918.0	0	0	188.0	Bristol Robotics Laboratory
1000206874468409344	human	arrrgr	Arthur Grishkevich | co-founder Quorum | growth	0	149.0	162.0	2018-05-26 02:47:56	2690.0	0	0	1730.0	
100037838	human	underbellie	Kelly Hogaboom	0	619.0	411.0	2009-12-28 20:19:20	1298.0	0	0	24407.0	Planet of the Dinosaurs
1000422226493980672	bot	DrSreelekhaS	Dr. Sreelekha S	0	16.0	417.0	2018-05-26 17:03:40	49.0	0	0	149.0	India
1000458307	bot	tabulmagd	Tarek	0	46.0	631.0	2012-12-09 23:32:10	148.0	0	0	28.0	
1000654232	bot	sendil_sendil	sendil	0	22.0	469.0	2012-12-10 01:44:31	22.0	0	0	0.0	SF
1000836728	human	brettmebane	Brett Mebane	0	92.0	448.0	2012-12-10 03:45:37	3364.0	0	0	894.0	
1000969151043616768	human	teoh_anthony	Anthony Teoh	0	943.0	750.0	2018-05-28 05:16:57	1321.0	1	0	663.0	Hong Kong
1001057931956314112	bot	m1chellemejia	Michelle Mejia	0	108.0	521.0	2018-05-28 11:09:44	15909.0	1	0	15492.0	Dubai, United Arab Emirates
1001160971262550017	human	john_bruning	John Bruning	0	1293.0	436.0	2018-05-28 17:59:11	3228.0	0	0	3159.0	Minneapolis, MN
1001231232733134848	human	jay__ashish	Jay__ashish(Dr.Ashish Gupta)	0	152.0	727.0	2018-05-28 22:38:22	1609.0	1	0	727.0	Gorakhpur, India
10013302	bot	gary8345	Gary	0	146.0	143.0	2007-11-06 22:35:05	173.0	1	0	9337.0	London, England
1001368497404854273	bot	ChuriwalaDev	Dev Churiwala	0	181.0	645.0	2018-05-29 07:43:49	165.0	0	0	34.0	Los Angeles, CA
1001390960	human	ResearchGerm...	Research in Germany	0	137995.0	445.0	2012-12-10 10:31:43	8411.0	1	0	51215.0	Germany
1001421357551407104	bot	WFD_Morocco	WFD Morocco	0	441.0	194.0	2018-05-29 11:13:51	519.0	1	0	578.0	Morocco
1001490297925394434	bot	jensfashions4u	jensfashions4u	0	273.0	395.0	2018-05-29 15:47:48	7.0	0	0	8374.0	South Carolina, USA

Nota: La Tabla 1 muestra un extracto de la Base de datos Raw; (Fuente: Elaboración propia).

b) Describir los Datos. Los datos que se obtuvieron se presentaron en la Tabla 1, que incluía información numérica, booleana y de texto, como la ubicación y los tweets. En el caso de la metadata, se obtuvieron registros de 18,347 usuarios. Sin embargo, se excluyeron 1,653 registros debido a que algunas de las cuentas se volvieron privadas o fueron cerradas.

De estos usuarios, se obtuvieron 214,224 registros de texto. Es importante tener en cuenta que no todas las cuentas tenían 20 tweets, y algunas cuentas no tenían ningún tweet registrado.

Estos resultados mostraron la disponibilidad y cantidad de datos obtenidos durante el proceso de extracción y resaltaron la variabilidad en la cantidad de tweets por cuenta.

Tabla 2

Modelo de datos

Variables	Tipo
CuentaProtegida	Booleano
Seguidores	Int
Amigos	Int
FechaCreacion	Date
Cantidad de Favoritos	Int
Locación Activada	Booleano
Cantidad de Estados	Int
Locación	String
Tweets(20)	String

Nota: La Tabla 2 muestra las variables y el tipo de variable dentro de la base de datos.

c) Exploración de datos y Verificación de la calidad de los datos. Se utilizó la herramienta Python con la biblioteca pandas para realizar la descripción de los datos, como se muestra en la Tabla 2. Durante este análisis, se encontró que no había valores nulos en la mayoría de las variables, excepto en la variable “locación”. Esto se debió a que muchos usuarios no proporcionaron esta información en sus perfiles.

El uso de pandas en Python permitió realizar una exploración detallada de los datos y obtener información relevante sobre su distribución y características. La identificación de valores nulos en la variable “locación” es importante para tener en cuenta al realizar análisis posteriores que involucren esta variable.

La cantidad no nulos representa la cantidad de datos presentes y disponibles para el análisis en lugar de valores ausentes o nulos.

Tabla 3

Descripción de la Data

Variable	Cantidad no nulos	Tipo
ID	18347	object
Clasificacion	18347	object
ScreenName	18347	object
Nombre	18347	object
Protegido	18347	int32
Seguidores	18347	int32
Amigos	18347	int32
FechaCreacion	18347	object
ID	18347	object
FavoritosCount	18347	int32
LocacionActivada	18347	object
Verificada	18347	int32
EstatosCount	18347	int32
Locacion	18347	object

Nota: La Tabla 3 muestra una descripción mas detallada de la data extraida de un usuario de Twitter. (Fuente: Elaboración propia)

También se realizó las estadísticas básicas como la cantidad, el promedio, la desviación estándar, el mínimo con el máximo y los percentiles 25,50 y 75. Como se puede ver no se encontró ningún dato fuera de lo común, en el caso de protegido y verificada son datos booleanos, por lo que esta estadística solo nos ayuda a verificar que los datos estén limpios.

Tabla 4

Data Procesada

	Protegido	Seguidores		FavoritosCount		
	AmigosVerificada	EstadosCount				
count	18347	18347	18347	18347	18347	18347
mean	0	40426.7	1803.757	15357.49	0.060882	16606.46
std	0	1007161	10856.07	52488.09	0.23912	55389.67
min	0	0	0	0	0	0
25%	0	65	131	90	0	140
50%	0	425	458	1177	0	1565
75%	0	2694.5	1369.5	8440	0	9980.5
max	0	1.07E+08	822807	2461124	1	1239989

Nota: La Tabla 4 muestra la data a la que se aplicará el procesamiento del Modelo BERT y Random Forest. (Fuente: Elaboración propia)

1.3. Preparación de datos

En este caso se utilizaron todos los campos, ya que se desconoce que campos son los que están correlacionados con la variable dependiente.

a) **Selección de datos.** En la fase de selección de datos, se utilizaron todos los campos disponibles, ya que no se tenía conocimiento previo de qué campos estaban correlacionados con la variable dependiente.

b) **Limpieza de los datos.** Se procesó la información recolectada previamente para eliminar cualquier factor que pudiera introducir ruido en el análisis. En el caso de la metadata, se realizaron cambios de tipo de datos para que fueran compatibles con la base de datos. Por ejemplo, la variable "verificada"se cambió de un formato booleano

(True/False) a un formato numérico (0/1). Además, se reemplazaron los valores nulos de la variable "locación"por el texto "nulo". En relación con los tweets, que consisten en texto sin procesar, se aplicaron diversas funciones de limpieza, como las siguientes:

1. Normalización: Se utilizó la función “replace” para reemplazar las vocales con acentos u otros caracteres especiales, tanto en mayúsculas como en minúsculas.

2. Remoción de signos: Mediante el uso de la librería Python “re”, se eliminaron los signos innecesarios, como “ -, !, =, , / ”. Asimismo, se eliminaron las menciones a otros usuarios en los tweets, las cuales se caracterizan por el símbolo “@” seguido de sus nombres de usuario.

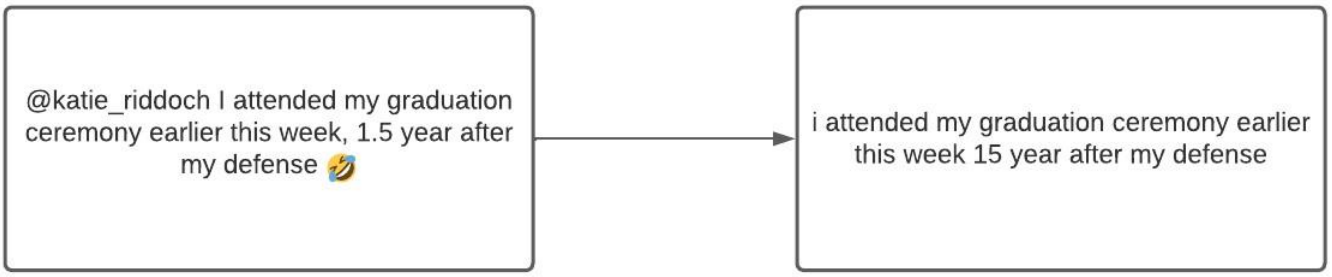
3. Eliminación de emojis: Utilizando la función “encode”, se identificaron los emojis presentes en el texto y se eliminaron aquellos que comenzaban con “//U”.

4. Detección de idiomas: Se empleó ’spacylangdetect’, que es un pipeline de detección de idiomas personalizable para spaCy. Esto permitió filtrar las cuentas según el idioma deseado.

En la figura 3 se puede observar cómo, después de aplicar estas funciones de limpieza, el texto queda limpio y listo para ser ingresado en los modelos de predicción.

Figura 3

Base de Datos Limpios



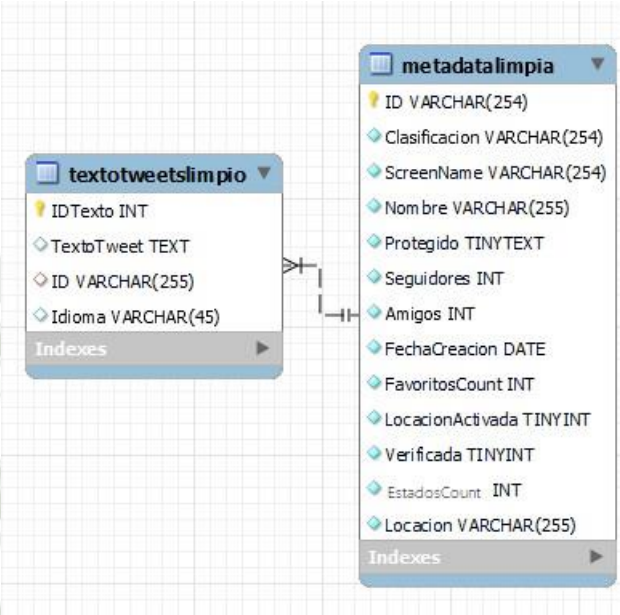
Nota: La figura 3 el texto después de ser procesado; (Fuente: Elaboración propia).

c) Estructuración de los datos. Finalmente, los textos previamente limpios fueron almacenados en la base de datos, asignándoles sus respectivos campos y estableciendo una relación de tipo ’1 a muchos’ con la tabla de texto. Además, se añadió un campo para almacenar el idioma de cada texto.

Esta estructuración permitió organizar de manera adecuada los datos procesados y facilitó su posterior acceso y análisis.

Figura 4

Base de Datos Limpios



Nota: La figura 4 muestra la base de datos limpia; (Fuente: Elaboración propia).

1.4. Fase de modelado

En esta etapa se eligen las técnicas de modelado más apropiadas para el BERT y el Random Forest.

a) **Selección de la técnica de modelado.** Para este caso se optó por utilizar el modelo de transformador BERT Embedding para transformar los Tweets en matrices, así como el modelo ELMO. Estos modelos son ampliamente utilizados en la detección de Bots. Mediante la técnica de Text Embedding, se logró convertir datos no estructurados, como el texto de los Tweets, en datos estructurados representados por matrices.

Estas matrices, junto con la metadata extraída de las cuentas, se utilizaron como entrada en los modelos de predicción. Para la clasificación de Bots y cuentas humanas, se empleó el algoritmo Random Forest debido a su alto rendimiento en el procesamiento de grandes volúmenes de datos.

b) Generación del plan de prueba. Para comparar los resultados, se utilizaron las métricas definidas en el apartado de Indicadores. Entre las métricas más utilizadas se encuentran la Accuracy, Precision, Recall y F1-score, cada una proporcionando una perspectiva única sobre el rendimiento del modelo. Estas métricas son calculadas aparte de la matriz de confusión la cual es una herramienta fundamental en la evaluación de modelos de clasificación, ya que proporciona una representación detallada del rendimiento del modelo en términos de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Vamos a desglosarla en el contexto de un problema de detección de bots en una plataforma en línea.

La matriz de confusión proporciona una visión detallada de cómo se distribuyen los errores del modelo, ofreciendo una perspectiva más completa de su capacidad de clasificación. Esta herramienta nos permite identificar y analizar los verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos, proporcionando información crucial para evaluar el rendimiento del modelo de manera exhaustiva.

- Verdaderos Positivos (TP): Casos correctamente identificados como positivos.
- Falsos Positivos (FP): Casos incorrectamente identificados como positivos.
- Verdaderos Negativos (TN): Casos correctamente identificados como negativos.
- Falsos Negativos (FN): Casos incorrectamente identificados como negativos.

Vamos a ver cómo se interpretan estos valores en el contexto de la detección de bots:

- Verdaderos Positivos (TP): Estos son los casos donde el modelo predice correctamente que un usuario es un bot. Un alto número de verdaderos positivos indica que el modelo es efectivo en identificar bots.

- Estos son los casos donde el modelo predice correctamente que un usuario es humano. Un alto número de verdaderos negativos indica que el modelo es efectivo en identificar usuarios legítimos, minimizando la perturbación de usuarios legítimos.

- Estos son los casos donde el modelo predice incorrectamente que un usuario es un bot cuando en realidad es humano. Un alto número de falsos positivos es problemático porque significa que el modelo está bloqueando o marcando usuarios legítimos como bots, lo cual puede llevar a una mala experiencia del usuario y potencialmente a la pérdida de usuarios legítimos.

- Falsos Negativos (FN): Estos son los casos donde el modelo predice incorrectamente que un usuario es humano cuando en realidad es un bot. Un alto número de falsos negativos es preocupante porque significa que el modelo está dejando pasar bots, lo que puede resultar en actividades maliciosas que comprometan la seguridad de la plataforma y la experiencia del usuario.

A partir de esta matriz, podemos calcular varias métricas clave:

Accuracy: se calcula como la proporción de predicciones correctas sobre el total de predicciones realizadas, esta métrica es intuitiva y fácil de interpretar, indicando el porcentaje de veces que el modelo acertó en sus predicciones. Sin embargo, su utilidad disminuye en escenarios con clases desbalanceadas, como en la detección de bots, donde la mayoría de los usuarios pueden ser humanos y solo una pequeña fracción son bots. En este caso, una alta exactitud puede ser engañosa si una clase domina significativamente sobre la otra. Por ejemplo, si el 95% de los usuarios son humanos y el modelo siempre predice que un usuario es humano, tendrá una alta exactitud del 95%, pero será inútil para identificar correctamente los bots.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision: por otro lado, se centra en la calidad de las predicciones positivas y se define como Precision. Esta métrica indica la proporción de verdaderos positivos entre todas las predicciones positivas realizadas por el modelo. Es particularmente útil en la detección de bots, donde un falso positivo podría implicar bloquear a un usuario legítimo, causando inconvenientes y potencialmente perdiendo usuarios. Por lo tanto, una alta precisión es crucial para minimizar estos errores y asegurar que solo los bots verdaderos sean marcados.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

Recall: También conocida como sensibilidad. Esta métrica mide la capacidad del modelo para identificar correctamente todos los bots reales. Es especialmente importante en la detección de bots, donde es crítico capturar todos los bots, incluso a costa de tener más falsos positivos. No detectar un bot (falso negativo) podría permitir actividades maliciosas en la plataforma, lo cual podría tener consecuencias graves para la seguridad y la experiencia del usuario. Por lo tanto, una alta recuperación es vital para asegurar que se identifiquen todos los bots.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

F1 Score:es la media armónica de la precisión y la recuperación. Esta métrica proporciona un equilibrio entre precisión y recuperación, siendo útil en situaciones donde se necesita considerar ambos aspectos. La puntuación F1 es particularmente relevante en problemas con clases desbalanceadas, como la detección de bots, ya que proporciona una única métrica que refleja tanto la precisión como la capacidad de

recuperación del modelo. Un alto F1 Score indica que el modelo tiene tanto una buena precisión como una buena recuperación, lo cual es deseable en muchos problemas de clasificación.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

c) Construcción del Modelo. Primero, se extrajo el Text Embedding del texto de los Tweets llamando al modelo de BERT con la biblioteca de TensorFlow como una capa de Keras. Para utilizar el Text Embedding, se necesitaron dos herramientas: el preprocesador, que transformó el texto en el formato requerido por BERT para su procesamiento, y el codificador, que transformó el texto en una matriz.

Figura 5

Invocación de BERT

```
# bert preprocessorhttps://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3
preprocessor = hub.KerasLayer("bert_en_uncased_preprocess_3")
# bert encoder https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-512_A-8/2
# encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-24_H-1024_A-16/4",trainable=True)
encoder = hub.KerasLayer("small_bert_bert_en_uncased_L-12_H-512_A-8_2",trainable=True)
```

Nota: La figura 5 muestra la invocación al preprocesador y al encoder; (Fuente: Elaboración propia).

Una vez cargadas estas herramientas, se inició un bucle en incrementos de 1000 para optimizar el uso de recursos, dado que este modelo consume una cantidad considerable de memoria RAM y poder computacional. Dentro del bucle, se aplicó el preprocesador al texto de los Tweets y luego se utilizó el codificador. A partir del resultado del codificador, se obtuvo la salida de secuencia (sequence-output), que contiene la matriz que se convirtió en un array y se almacenó en una lista.

Figura 6

BERT loop

```
for tweet in range(0,len(TwitterText),1000):
    print(tweet)
# preprocessing dataset adding cls sep etc
# slicing reviews due to unsufficient resources
    inputs = preprocessor(TwitterText[tweet:tweet+1000].TextoTweet)
    # feeding it to model for vectorization
    outputs = encoder(inputs)
    for i in range(0,len(outputs['sequence_output'])):
        ListaEncoding.append(outputs['sequence_output'][i].numpy().sum(axis=0))
```

Nota: La figura 6 muestra el loop donde se usa el Text Embedding; (Fuente: Elaboración propia).

Por último, se guardaron los resultados obtenidos del Text Embedding junto con la metadata para poder ser procesados posteriormente por los modelos de predicción. En el caso de ELMO, se llamó al modelo desde Tensorflow. A diferencia de BERT, ELMO requiere un proceso de preprocesamiento llamado lematización. Este proceso reduce las palabras a sus raíces, de modo que las palabras con diferentes conjugaciones puedan ser identificadas como la misma palabra. Para llevar a cabo la lematización, se utilizó un modelo de Spacy llamado en-core-web-sm, el cual contiene una función que permite lematizar el texto.

Figura 7

Lematizacion

```
def lemmatization(texts):
    output = []
    for i in texts:
        s = [token.lemma_ for token in nlp(i)]
        output.append(' '.join(s))
    return output
```

Nota: La figura 7 muestra la Funcion de Lematizacion; (Fuente: Elaboración propia).

Seguidamente se usa la función donde se saca los “signatures” los cuales contiene las matrices que nos proporciona ELMO el cual es una matriz de 1024 de dimensión. Esta matriz es unida junto a la metadata para entrenar al modelo de prediccion

Figura 8

ELMO

```
def elmo_vectors(x):
    print(x)
    embeddings = elmo.signatures["default"](tf.constant([x])
                    )["elmo"]
    return tf.reduce_mean(embeddings,1)
```

Nota: La figura 8 muestra la función donde se extrae los Text Embedding; (Fuente: Elaboración propia).

Ahora que tenemos esta data procesada procedemos a entrenar los modelos de predicción empezando por el Random Forest el cual lo usamos importándolo de la librería SKLearn, para ser más específicos estamos usando el RandomForestClassifier,

pero antes debemos dividir la data con la función train-test-split la cual divide matrices o arrays en subconjuntos aleatorios de entrenamiento y prueba, en este caso usamos un 70% entrenamiento y un 30% a testing, cabe decir que esta distribución de la data será aplicada para todos los modelos a continuación, como se puede ver en la figura 9 los valores por defecto de la función nos divide la data con los valores de 70% y 30%.

Figura 9

TrainTest

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

Nota: La figura 9 muestra la función con la cual se divide el dataset.

Random forest tiene parámetros por defecto los cuales se le puede cambiar como por ejemplo n-estimators, criterion y max-depth que tienen una gran influencia en como se entrena el modelo pero para obtener los mejores parámetros no se puede estar probando cada combinación de parámetros a mano, esto resulta ineficiente y consuma una gran cantidad de tiempo en especial si se tiene una gran cantidad de datos por esta razón se usa el fine-tunning(ajuste fino) el cual nos ayudó a encontrar los mejores parámetros para nuestro modelo. Primero, se define un Grid de hiperparámetros que serán evaluados utilizando la clase ParameterGrid de scikit-learn. En este caso, los hiperparámetros son 'n-estimators', 'max-features', 'max-depth', y 'criterion'. Se evaluarán 3 valores posibles para 'n-estimators', 5 valores posibles para 'max-features', 5 valores posibles para

'max-depth', y 3 opciones para el criterio 'criterion'.

Figura 10

GridParametros

```
param_grid = ParameterGrid(
                {'n_estimators': [150],
                 'max_features': [5, 7, 9],
                 'max_depth'   : [None, 3, 10, 20],
                 'criterion'   : ['gini', 'entropy']
                }
            )
```

Nota: La figura 10 muestra el Grid de hiperparámetros evaluados; (Fuente: Elaboración propia).

Luego, se iteró a través de cada combinación de hiperparámetros utilizando un loop 'for'. Para cada combinación, se instancia un objeto RandomForestClassifier utilizando los parámetros de la combinación actual y se ajusta el modelo utilizando los datos de entrenamiento (X-train, y-train). El OOB score (out-of-bag score) se utiliza como una medida de evaluación del modelo para cada combinación de hiperparámetros. El OOB score es la precisión media en los datos de prueba para cada observación, calculada a partir de los árboles que no utilizaron esa observación durante la creación del árbol.

Figura 11

GridLoop

```
for params in param_grid:

    modelo = RandomForestClassifier(
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123,
                ** params
             )

    modelo.fit(X_train, y_train)

    resultados['params'].append(params)
    resultados['oob_accuracy'].append(modelo.oob_score_)
    print(f"Modelo: {params} \u2713")
```

Nota: La figura 11 muestra el Grid de hiperparámetros evaluados; (Fuente: Elaboración propia).

Para almacenar los parámetros se conviertió el diccionario resultados en un DataFrame de pandas y se ordenan los resultados en orden descendente de acuerdo con el oob-accuracy obtenido. También se separan los valores de cada hiperparámetro en columnas diferentes para una mejor visualización. Como se observa en la tabla 5 la cual muestra los mejores 5 parámetros ordenados por el score.

Tabla 5

Hiperparametros

oob_accuracy	criterion	max_depth	max_features	n_estimators
0.966816	gini	10.0	NaN	200
0.966650	gini	10.0	NaN	150
0.966318	entropy	50.0	NaN	100
0.966318	entropy	NaN	NaN	100
0.966318	log_loss	20.0	NaN	100

Nota: La Tabla 5 muestra los hiperparametros del ajuste; (Fuente: Elaboración propia)

También se usó el Grid Search pero con validación cruzada, la validación cruzada es una técnica de evaluación de modelos que se utiliza para valorar el rendimiento de un algoritmo de aprendizaje automático a la hora de realizar predicciones sobre nuevos conjuntos de datos en los que no ha sido entrenado. Al igual que el anterior entrenamiento se usó el Grid de hiperparámetros evaluados con los mismos parámetros, luego se usó el GridSearchCV de Scikit-learn para ajustar el modelo, este Grid realiza una búsqueda exhaustiva sobre todas las combinaciones de valores posibles para los hiperparámetros evaluando con validación cruzada.

Figura 12

Validación Cruzada

```
# ============================================================================
grid = GridSearchCV(
        estimator  = RandomForestClassifier(random_state = 123),
        param_grid = param_grid,
        scoring    = 'accuracy',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=3, random_state=123),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )

grid.fit(X = X_train, y = y_train)
```

Nota: La figura 12 muestra el Grid de hiperparámetros evaluados con validación cruzada;

(Fuente: Elaboración propia).

Después de que se completó la búsqueda, se imprimen los mejores hiperparámetros encontrados (Grid.best_params_) y su precisión correspondiente (Grid.best_score_), utilizando la métrica de evaluación configurada (Grids coring). El mejor modelo se guarda automáticamente en una variable y se calcula el error de raíz cuadrada media (RMSE) utilizando la función 'mean_squared_error' de Tensorflow keras se importó la funcion Sequential que representa una secuencia lineal de capas de redes neuronales, a esta secuencia le pudimos agregar capas densas. La primera que se agregó es una capa densa donde se me modifico el input_shape el cual es el número de caracteres que pueda aceptar, esta caá tuvo 12 neuronas y es de tipo 'RELU', que es una función de activación no lineal ampliamente utilizada en redes neuronales profundas. En la segunda capa se utilizó la función de activación antes con 8 neuronas, en la última capa se utilizó la función de activación ya que es muy utilizada en clasificación binaria se utilizó función de perdida es 'binary_crossentropy' la cual mide la diferencia entre las

probabilidades de clase predichas por el modelo y las probabilidades de clase reales. Como optimizador se tiene ADAM el cual se encarga de ajustar los pesos de las conexiones neuronales del modelo para hacer que las predicciones del modelo se acerquen lo más posible a las etiquetas reales, también se usa la métrica de precisión para evaluar el modelo.

Figura 13

Iniciando la Red

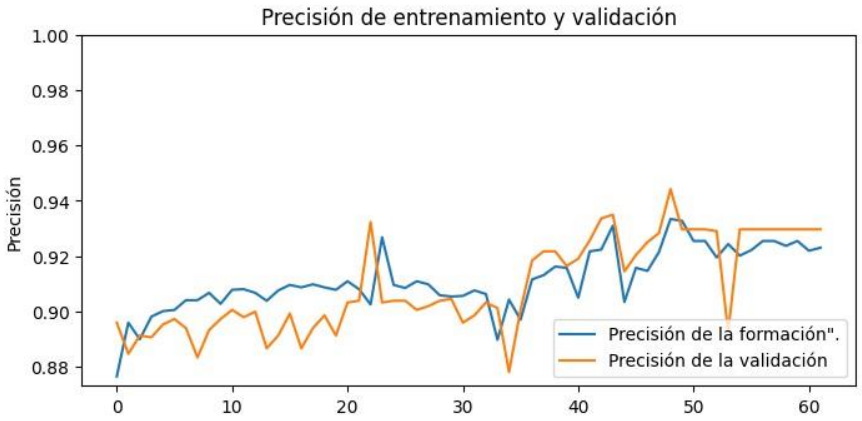
```
model = Sequential()
model.add(Dense(12, input_shape=(520,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Nota: La figura 13 muestra las capas de las neuronas agregadas; (Fuente: Elaboración propia).

Procedimos a entrenar la red neuronal y almacenados el historial de entrenamiento en una variable de donde extraemos la información de como se comportó el entrenamiento en de cada epoch(iteración del entrenamiento), para los parámetros se hizo un loop ’for’ donde se analizó la cantidad para el parámetro ’base_learning_rate’ que se refiere a la tasa de aprendizaje inicial que se utiliza para actualizar los pesos de las conexiones de una red neuronal durante el entrenamiento, también usamos el parámetro ’EarlyStopping’ para que el entrenamiento se detenga cuando se deje de aumentar la precisión. Con esto obtenemos las gráficas como se observa en la figura 14 que nos indica como va mejorando la precisión con la mayor cantidad de epochs.

Figura 14

Gráfica de entrenamiento

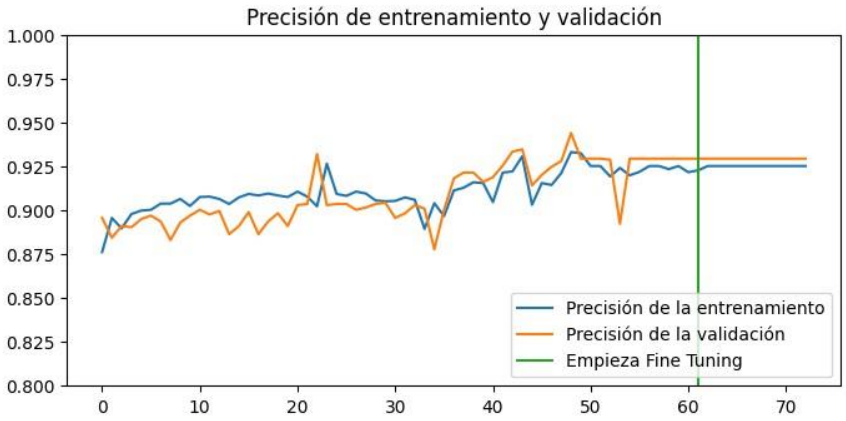


Nota: La figura 14 muestra la evolución del entrenamiento; (Fuente: Elaboración propia).

Una vez tenemos los parámetros óptimos comenzamos con el ajuste fino(fine tunning) como se puede ver en la figura 15 una vez iniciamos el Fine Tunning el entrenamiento se vuelve óptimo y sin variaciones indicando que estos son los parámetros óptimos.

Figura 15

Gráfica de entrenamiento del Fine Tunning



Nota: La figura 15 muestra la evolución del entrenamiento con el Fine Tunning; (Fuente:

Elaboración propia).

Ahora con el último modelo Support Vector Machine (SVM) usamos la misma técnica de definir los parámetros en un Grid en este caso estamos usando los parámetros de 'C', 'kernel', 'gamma', y 'shrinking'. En este modelo usaremos también validación cruzada ya que al ser un algoritmo con bastantes parámetros se beneficia de esta. Con la clase 'StratifiedKFold' la cual nos ayuda a mantener la proporción de clases en los pliegues igual que en el conjunto de datos de entrenamiento, le indicamos 'n_splits' a 3 validaciones cruzadas.

Figura 16

Grid con rango de parámetros

```
# Define the search space
param_grid = {
    # Regularization parameter.
    "C": C_range,
    # Kernel type
    "kernel": ['rbf', 'poly'],
    # Gamma is the Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’.
    "gamma": gamma_range.tolist()+['scale', 'auto'],
    "shrinking":[True,False]
    }
# Set up score
scoring = ['accuracy']
# Set up the k-fold cross-validation
kfold = StratifiedKFold(n_splits=3, shuffle=True, random_state=0)
```

Nota: La figura 16 muestra los parámetros establecidos en el Grid; (Fuente: Elaboración propia).

Como se puede ver en la tabla 6 se obtiene los parámetros óptimos para el modelo con un mejor score

Tabla 6

Hiperparametros SVM

param_C	param_gamma	param_kernel	param_shrinking	precisión
10.0	scale	rbf	True	0.830596
10.0	auto	rbf	True	0.830596
10.0	auto	rbf	False	0.830596
10.0	scale	rbf	False	0.830596
1.0	scale	rbf	False	0.829102

Nota: La Tabla 6 muestra los hiperparámetros del ajuste del modelo SVM; (Fuente: Elaboración propia)

1.5. Fase de evaluación

Esta fase se detalla en el Capítulo V .

1.6. Fase de implementación

Para el lanzamiento, se planifica utilizar una plataforma web compatible con cualquier dispositivo, que sea responsive y accesible para cualquier persona. Esta plataforma permitirá a los usuarios consultar la cuenta de Twitter de una persona y proporcionará información sobre si dicha cuenta es un Bot o no.

Figura 17

Proceso del despliegue de la Página Web

Nota: La figura 17 muestra la etapa de implementación donde se realiza el despliegue de la

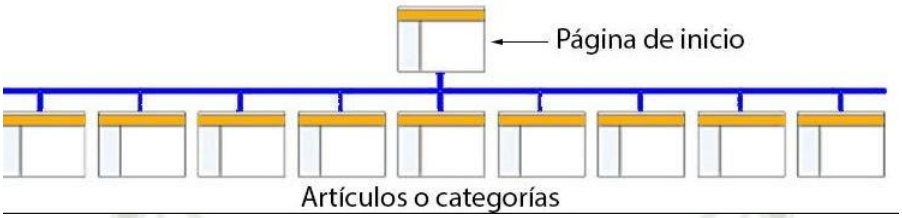
página Web; (Fuente: Elaboración propia).

a) Tipo de arquitectura. El tipo de arquitectura que se utilizó fue la arquitectura horizontal, como se muestra en la figura 17. Esta arquitectura se caracteriza por tener una estructura jerárquica bien definida, donde el sitio web se organiza y subcategoriza

por los distintos servicios que ofrece. Esto facilita a los usuarios la comprensión y navegación de la plataforma.

Figura 18

figura referencial hecha por Dossetenta



Nota: La figura 18 muestra una referencia de como realizar una Página Web Jerarquizada;

(Fuente: "Dossetenta.com").

b) Modelo de la Página Web. Para facilitar la creación de la web se requirió la creación de Mockups para tener una referencia a la cual apuntar. Esta página nos permitirá desplegar BeBot en un backend.

Figura 19

Página Web

Nota: La figura 19 muestra el Login de la Página Web; (Fuente: Elaboración propia).

En la figura 19 se contó con la página principal donde los usuarios podían registrarse ingresando los datos que se les solicitaban en los texbox y realizar el Login con el Botón superior derecho para ingresar a la plataforma. Además, se les daba la opción de tener conocimiento de esta investigación para que las personas pudieran conocer los procesos que realizamos.

Como se ve en la figura 20 la página tiene 4 funciones siendo la primera la de clasificar usuarios, la cual regresa la clasificación de un usuario de Twitter el cual se le ingresa su usuario, la siguiente de analizar sus seguidos selecciona todos los usuarios seguidores que tiene una cuenta y botara una clasificación para cada uno de los usuarios, de igual forma la pestaña de cuentas que siguen hará algo parecido pero las personas que siguen a la cuenta en sí. Finalmente, se tiene un historial con los usuarios que pidió el usuario

Figura 20

Página Web

Nota: La figura 20 muestra el Login de la Página Web; (Fuente: Elaboración propia).

c) Construcción de la Página Web. Para la construcción de la web nos apoyamos en el Framework de Django que utiliza Python y tiene mucha facilidad al conectarse. Para la creación de la web utilizando Django, se usó la herramienta django-admin para crear una carpeta donde se encontraría el proyecto con los ficheros y plantillas básicas, junto con el script que gestionaba el proyecto "manage.py"; este script creó la aplicación para comenzar a conectar con el mapeado URL de la aplicación.

Se configuró de tal manera que la aplicación pudiera utilizar el algoritmo con el modelo de BERT y el entrenamiento del Random Forest.

Figura 21

Página Web

Nota: La figura 21 muestra el estado actual de la Página Web; (Fuente: Elaboración propia).

CAPITULO V ANÁLISIS Y DISCUSIÓN DE RESULTADOS

En este capítulo se presentarán los resultados obtenidos del modelo BERT sumado con las técnicas de predicción del Random Forest y analizarán y validarán estos resultados.

1. Análisis de Resultados

Se realizó un analisis de resultados conseguidos de nuestros modelos hechos con diferentes Dataset de Bots (Cresci2017 y TwitBot_22) los modelos de BERT y ELMO combinado con el algoritmo de Random Forest, SVM y Redes Neuronales para aumentar la precisión de los modelos. Gracias a esta comparación logramos concluir que el modelo BERT + RandomForest usando la DataBase de Cresci2017 da mejores resultados y será seleccionado para usarlo en nuestra plataforma Web.

Tabla 7

Comparación de Modelos - BERT

DB	Modelo	BERT			
		Precision	F1-Score	Recall	Exactitud
Cresci2017	Random Forest	0.94	0.94	0.93	0.96
	SVM	0.92	0.85	0.81	0.90
	Redes Neuronales	0.94	0.89	0.86	0.92
TwitBot_22	Random Forest	0.71	0.71	0.71	0.71
	SVM	0.70	0.60	0.63	0.63
	Redes Neuronales	0.70	0.35	0.50	0.51

Nota: La Tabla 7 muestra la comparación de modelos usando BERT (Fuente: Elaboración propia)

Realizada la comparación elegimos el modelo entrenado con mayor Precisión y F1-Score, se realizó una matriz de confusión el cual nos indica los verdaderos positivos que se define cuando se clasifica un Bot correctamente, los verdaderos negativos que son cuando el modelo clasifica las cuentas de humanos correctamente, los falsos positivos que es cuando

Tabla 8

Comparación de Modelos - ELMO

DB	Modelo	ELMO			
		Precision	F1-Score	Recall	Exactitud
Cresci2017	Random Forest	0.91	0.89	0.88	0.92
	SVM	0.37	0.42	0.50	0.73
	Redes Neuronales	0.84	0.86	0.91	0.88
TwitBot_22	Random Forest	0.72	0.71	0.71	0.71
	SVM	0.71	0.62	0.65	0.65
	Redes Neuronales	0.25	0.34	0.50	0.51

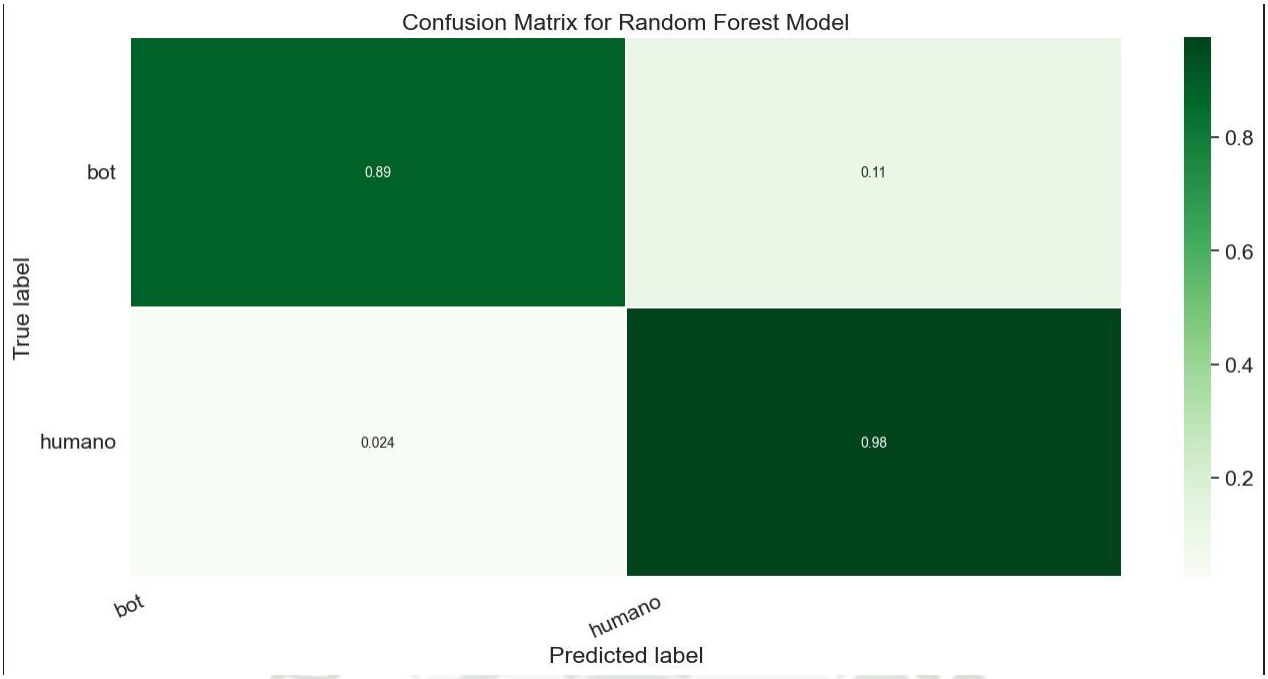
Nota: La Tabla 8 muestra la comparación de modelos usando ELMO (Fuente: Elaboración propia)

un Bot se clasificó como humano y los falsos negativos cuando un humano se clasificó como

Bot. Con estos datos obtuvimos los indicadores que hemos definido.

Figura 22

Matriz de confusión



Nota: La figura 22 muestra una Matriz de Confusión realizada con Random Forest y BERT con una precisión del 89% para la detección de Bots.(Fuente: elaboración propia).

Los mejores hiperparámetros según el obb_accuracy para realizar el entrenamiento del Random Forest, estos se pueden ver en 6, fueron 4:

1. criterion; utilizado para medir la calidad de la partición de los datos en los nodos del árbol. En este caso, se está utilizando el índice Gini; este índice mide la impureza de una partición de los datos en términos de la probabilidad de clasificación incorrecta. Cuanto menor sea el valor del índice de Gini, más homogénea será la partición de los datos y, por lo tanto, mejor será la calidad de la partición
2. max_depth; es la profundidad máxima del árbol de decisión, en este caso como no se especifica el árbol crecerá hasta que todas las hojas sean “puras” o hasta que se alcance el número mínimo de muestras por hoja.

3. max_features; es el número máximo de características que se consideran para
cada partición en un árbol de decisión. En este caso, se está limitando a 20 características.

4. n_estimators; es el número de árboles en el bosque aleatorio. En este caso, se están utilizando 150 árboles.

Tabla 9

Importancia de los predictores

predictor		importancia
3	FavoritosCount	0.160633
6	EstatosCount	0.083377
4	LocacionActivada	0.061668
1	Seguidores	0.041702
2	Amigos	0.013018
8	0	0.001249
9	1	0.001027
5	Verificada	0.000123
0	Protegido	0.000000
7	listaSacar	0.000000

Nota: La tabla 9 muestra los predictores ordenados por la importancia de los usados para entrenar el Modelo con Random Forest. (Fuente: Elaboración propia)

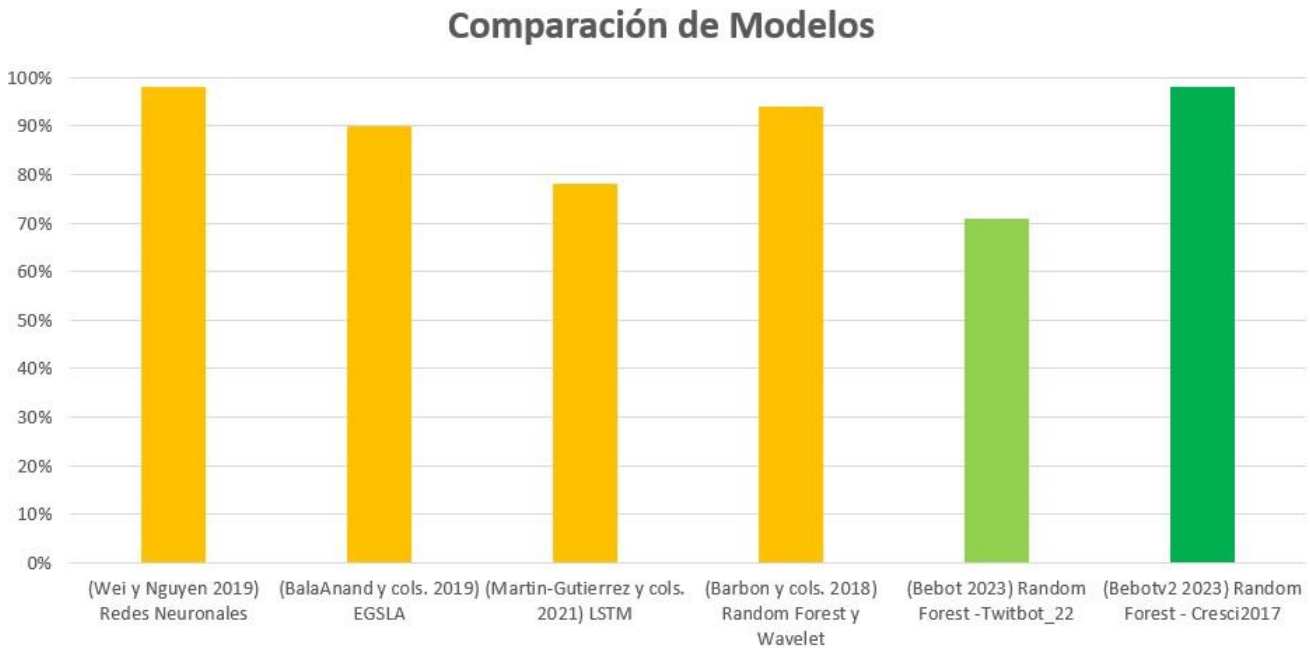
En esta tabla 9 se muestra la importancia de los predictores; esto se refiere a los predictores que más influencian en el modelo para detectar Bots en Twitter, estos se calculan utilizando el atributo “feature_importances”, que es una medida de cuánto contribuye cada predictor a las predicciones precisas del modelo. En esta tabla podemos ver que “FavoritosCount”, representando los “corazones” en Tweets, tuvo una mayor importancia y tuvo una mayor influencia en nuestro modelo, luego otros predictores útiles fueron “EstatosCount” que representa el Estado, “LocacionActivada” si tiene la

Localización activada en su perfil de Twitter, y como últimos puntos más relevantes son los “Seguidores” y “Amigos”.

Para finalizar se comparó la precisión de BeBot con los modelos del estado del arte, en comparación BeBot tuvo una puntuación mucho menor, esto se puede explicar ya que el dataset con el cual se entrenó es uno mucho más actualizado y contiene una mayor cantidad de cuentas. Luego de usar un dataset anterior similar a los datos que se usaron en el estado del arte, se obtuvo una puntuación a la altura de las investigaciones realizadas anteriormente.

Figura 23

Comparación de precisión



Nota: La figura 23 muestra la comparación de Modelos del estado del arte con las versiones

de BeBot (Fuente: Elaboración propia).

2. Discusión

Como se muestra en el capítulo de análisis y evaluación, la primera versión de BeBot con el dataset de TwiBot no llegó a alcanzar el nivel de precisión que sus predecesores nosotros obteniendo 71% y el mejor modelo de (Wei & Nguyen, 2019) un 98%. Una de las razones que podría explicar esta diferencia de resultados es el diferente Dataset que se usó, mientras que en otras investigaciones normalmente se suele usar Dataset de 2017 en este caso estamos usando uno del 2022 en año actual en cuál se realiza la investigación por lo que los datos son muchos más actualizados que los que se suelen usar, considerando que los Bots evolucionan en el tiempo volviéndose muchos más eficientes y difíciles de detectar, por lo que consideramos que tener data que este actualiza y se adapte a los Bots más recientes es mucho más valioso. Otras características es que la muestra es de 20.000 cuentas teniendo 10.000 de Bots y el resto de las cuentas humanas, a diferencia de los Dataset del estado del arte se tiene una distribución mucha más homogénea de Bots y humanos por lo que evitamos casos donde el modelo sufra de Overfitting al momento de ser entrenado como podría ser los otros Dataset. La última característica en tomar en cuenta es la cantidad de Tweets que se está tomando en ejemplos como (Mohammad et al., 2020) toman solo un post para entrenar su modelo, en el caso de esta investigación usamos los últimos 20 Tweets obteniendo mucha más información. Habiendo dado estos puntos podemos inferir que la precisión obtenida es más baja en referente al estado del arte se debe a la data más actualizada y de mayor cantidad con la cual se está trabajando.

Teniendo estos resultados se decidió aplicar el Dataset de Cresci2017, investigar y probar más modelos (BERT ELMO), de los cuales la mejor selección que se hizo fue de el Modelo BERT + Random Forest, entonces se creó una segunda versión de BeBot con este modelo obteniendo resultados al nivel de las investigaciones predecesoras alcanzando una precisión de 94%, el cual da una mejor confianza a la hora de detectar

Bots en Twitter con nuestro modelo. Habiendo dicho esto aún se tiene espacio para mejorar ya sea en el procesamiento del BERT, ya que al poseer que la capacidad de procesamiento necesario se tuvo que combinar los 20 Tweets de una cuenta en un solo párrafo lo que pudo haber causado problemas a la hora de haber sacado las matrices. Trabajar con otros modelos de predicción aparte de los vistos acá también sería una mejora que se tiene para el futuro el cual no se pudo realizar por falta de tiempo. Otro aspecto que no se pudo realizar por falta de tiempo y recursos es la creación de un Dataset que nos pueda asegurar la calidad de las cuentas ya que seriamos nosotros haciendo esta clasificación, para obtener datos se necesita usar diferentes técnicas para atraer estas cuentas que toman mucho tiempo y un alto conocimiento del tema.

Puesto que esta investigación se trabajó usando la data de Twitter es difícil saber si podría ser aplicada a otra red social como Facebook, aunque compartan ciertos campos, el tipo de data ya sea en cantidad y diversidad es muy diferente a la de Twitter.

En esta investigación se puedo comprobar que al usar PLN juntos con modelos de predicción se puede conseguir un 94% de precisión al momento de detectar Bots, este aporte a la comunidad es el uso de BERT Embedding junto con Random Forest usando el

Dataset Cresci2017.

CONCLUSIONES

PRIMERA.- Se comprobó la hipótesis, en esta investigación se combinó el modelo de BERT de Procesamiento de Lenguaje Natural (PLN), con un modelo de predicción como es el RANDOM FOREST. El modelo de predicción de bots demuestra un rendimiento notablemente alto con una precisión del 94%, un F1 score del 94%, un recall del 93% y una exactitud del 96%. Esto indica que el modelo es muy efectivo en identificar bots, manteniendo un buen equilibrio entre precisión y recall. La alta precisión sugiere que el modelo minimiza los falsos positivos, mientras que un recall del 93% muestra que captura la mayoría de los bots reales, aunque hay un pequeño margen de falsos negativos. La alta exactitud global confirma que el modelo realiza correctamente la mayoría de las predicciones. En conjunto, estas métricas reflejan un modelo confiable y robusto para la tarea de predicción de bots. (Tabla 7, página 51)

SEGUNDA.- Se desarrolló el preprocesamiento de la información limpiando las variables numéricas y homologando el texto de forma que el procesamiento de lenguaje natural pueda procesarlo de una mejor manera, como se detalla en la Figura 3. (Página 31)

TERCERA.- Se logró procesar el texto de los Tweets mediante el modelo de procesamiento de lenguaje natural BERT y ELMO, y se extrajeron los Tweets de los usuarios, logrando así una optimización del proceso que se refleja en la precisión del modelo

RANDOM FOREST en la predicción de los Bots. (Página 40)

CUARTA.- Se entrenaron los modelos de procesamiento de lenguaje natural BERT y ELMO, en combinación con los modelos de predicción como Máquinas de Vectores de Soporte (SVM) y Redes Neuronales, eligiendo la combinación con mejor precisión como es BERT y RANDOM FOREST. (Figura 22, página 50)

QUINTA.- Se ha validado la clasificación de los usuarios de Twitter considerados como Bots usando la parte del dataset que se reservó para testing sacando la precisión anteriormente mencionada. (Página 51)

SEXTA.- Se ha desarrollado una plataforma web denominada BeBot, que contiene la combinación de modelos BERT más RANDOM FOREST con el objetivo de clasificar a los usuarios de Twitter como Bots o Humanos, como un prototipo para su fácil uso, esta plataforma es escalable y tiene el potencial de mejorar con el tiempo. (Figura 20, página 48)

RECOMENDACIONES

Se recomienda incrementar el dataset de predicción para tener una mayor variedad de resultados al comparar el modelo BERT con otros modelos. De igual forma, trabajar con diferentes modelos de predicción para obtener resultados cada vez más precisos, como el uso de modelos binarios y nuevos modelos de PLN como ChatGPT, debido a la rápida evolución de dichos modelos.

Es necesario tener una retroalimentación luego de desplegar la página web al usuario y medir la experiencia de usuario. Esta retroalimentación permitirá mejorar la plataforma web y poder mejorar el servicio para futuros trabajos, se recomienda mejorar el rendimiento de BeBot mediante la exploración de otros datasets. Además, se pueden probar técnicas de normalización de datos con el objetivo de continuar mejorando el desempeño del sistema debido al creciente incremento de técnicas cada vez más sofisticadas. En caso se quiera aplicar a otra red social se recomienda tener en cuenta las diferentes características que estas puedan tener ya que al usar Twitter se tiene limitado a la información de las 3 acciones principales de Twitter (retweets, likes, comentarios) mientras que otras redes como Facebook tienen otras acciones como reacciones.

REFERENCIAS

Abuhassan, M., Anwar, T., Fuller-Tyszkiewicz, M., Jarman, H.-N. K., Shatte, A., Liu, C., & Sukunesan, S. (2022). Journal Pre-proof Classification of Twitter users with eating disorder engagement: Learning from the biographies. https://doi.org/10.1016/j.chb.2022.107519

Alothali, E., Zaki, N., Mohamed, E. A., & Alashwal, H. (2019). Detecting Social Bots on Twitter: A Literature Review. *Proceedings of the 2018 13th International Conference on Innovations in Information Technology, IIT 2018*, 175-180. https://doi.org/10.1109/INNOVATIONS.2018.8605995

Al-Qurishi, M., Alrubaian, M., Md, S., Rahman, M., Alamri, A., & Hassan, M. M. (2018). A prediction system of Sybil attack in social network using deep-regression model. *Future Generation Computer Systems*, *87*, 743-753. https://doi.org/10.1016/j.future.2017.08.030

BalaAnand, M., Karthikeyan, N., Karthik, S., Varatharajan, R., Manogaran, G., & Sivaparthipan, C. B. (2019). An enhanced graph-based semi-supervised learning algorithm to detect fake users on Twitter. *Journal of Supercomputing*, *75*, 6085-6105. https://doi.org/10.1007/s11227-019-02948-w

Barbon, S., Campos, G. F., Tavares, G. M., Igawa, R. A., Proença, M. L., & Guido, R. C. (2018). Detection of human, legitimate bot, and malicious bot in online social networks based on wavelets. *ACM Transactions on Multimedia Computing, Communications and Applications*, *14*. https://doi.org/10.1145/3183506

Beskow, D. M., & Carley, K. M. (2018). Its all in a name: detecting and labeling bots by their name. *Computational and Mathematical Organization Theory 2018 25:1*, *25*, 24-35. https://doi.org/10.1007/S10588-018-09290-1

Bindu, P. V., Mishra, R., & Thilagam, P. S. (2018). Discovering spammer communities in twitter. *Journal of Intelligent Information Systems*, *51*, 503-527. https://doi.org/10.1007/s10844-017-0494-z

Blanco, G., & Lourenço, A. (2022). Optimism and pessimism analysis using deep learning on COVID-19 related twitter conversations. *Information Processing and Management*, *59*. https://doi.org/10.1016/j.ipm.2022.102918

Breiman, L. (2001). Random Forests. *45*, 5-32.

Cresci, S., Spognardi, A., Petrocchi, M., Tesconi, M., & Pietro, R. D. (2017). The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. *26th International World Wide Web Conference 2017, WWW 2017 Companion*, 963-972. https://doi.org/10.1145/3041021.3055135

Dukic, D., Keca, D., & Stipic, D. (2020). Are you human? detecting bots on twitter using BERT. *Proceedings - 2020 IEEE 7th International Conference on Data Science and Advanced Analytics, DSAA 2020*, 631-636. https://doi.org/10.1109/DSAA49011.2020.00089

Feng, S., Tan, Z., Wan, H., Wang, N., Chen, Z., Zhang, B., Zheng, Q., Zhang, W., Lei, Z., Yang, S., Feng, X., Zhang, Q., Wang, H., Liu, Y., Bai, Y., Wang, H., Cai, Z., Wang, Y., Zheng, L., ... Luo, M. (2022). TwiBot-22: Towards Graph-Based Twitter Bot Detection. http://arxiv.org/abs/2206.04564

Feng, S., Wan, H., Wang, N., Li, J., & Luo, M. (2021). TwiBot-20: A Comprehensive Twitter Bot Detection Benchmark. *International Conference on Information and Knowledge Management, Proceedings*, 4485-4494. https://doi.org/10.1145/3459637.3482019

Gao, Z., Feng, A., Song, X., & Wu, X. (2019). Target-dependent sentiment classification with BERT. *IEEE Access*, *7*, 154290-154299. https://doi.org/10.1109/ACCESS.2019.2946594

Heidari, M., & Jones, J. H. (2020). Using BERT to Extract Topic-Independent Sentiment Features for Social Media Bot Detection. *2020 11th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2020*, 0542-0547. https://doi.org/10.1109/UEMCON51285.2020.9298158

Heidari, M., Jones, J. H., & Uzuner, O. (2020). Deep Contextualized Word Embedding for Text-based Online User Profiling to Detect Social Bots on Twitter. *IEEE International Conference on Data Mining Workshops, ICDMW*, *2020-November*, 480-487. https://doi.org/10.1109/ICDMW51313.2020.00071

II, T. B. (2018). *Applied Natural Language Processing with Python*. Apress. https://doi.org/10.1007/978-1-4842-3733-5

Ilias, L., & Roussaki, I. (2021). Detecting malicious activity in Twitter using deep learning techniques. *Applied Soft Computing*, *107*, 107360. https://doi.org/10.1016/J.ASOC.2021.107360

Inuwa-Dutse, I., Liptrott, M., & Korkontzelos, I. (2018). Detection of spam-posting accounts on Twitter. *Neurocomputing*, *315*, 496-511. https://doi.org/10.1016/J.NEUCOM.2018.07.044

Li, X., Fu, X., Xu, G., Yang, Y., Wang, J., Jin, L., Liu, Q., & Xiang, T. (2020). Enhancing BERT Representation with Context-Aware Embedding for Aspect-Based Sentiment Analysis. *IEEE Access*, *8*, 46868-46876. https://doi.org/10.1109/ACCESS.2020.2978511

Loyola-Gonzalez, O., Monroy, R., Rodriguez, J., Lopez-Cuevas, A., & Mata-Sanchez, J. I. (2019). Contrast Pattern-Based Classification for Bot Detection on Twitter. *IEEE Access*, *7*, 45800-45817. https://doi.org/10.1109/ACCESS.2019.2904220

Mao, J., Li, X., Luo, X., & Lin, Q. (2022). SybilHunter: Hybrid graph-based sybil detection by aggregating user behaviors. *Neurocomputing*, *500*, 295-306. https://doi.org/10.1016/j.neucom.2021.07.106

Mariette, K. R. A. (2015). *Efficient Learning Machines* (1.ª ed.). Apress Berkeley, CA.

Martin-Gutierrez, D., Hernandez-Penaloza, G., Hernandez, A. B., Lozano-Diez, A., & Alvarez, F. (2021). A Deep Learning Approach for Robust Detection of Bots in Twitter Using Transformers. *IEEE Access*, *9*, 54591-54601. https://doi.org/10.1109/ACCESS.2021.3068659

Mazza, M., Avvenuti, M., Cresci, S., & Tesconi, M. (2022). Investigating the difference between trolls, social bots, and humans on Twitter. *Computer Communications*, *196*, 23-36. https://doi.org/10.1016/j.comcom.2022.09.022

McCormick, C. (2019). BERT Word Embeddings Tutorial.

Mohammad, S., Khan, M. U., Ali, M., Liu, L., Shardlow, M., & Nawaz, R. (2020). Bot detection using a single post on social media.

Ncr & Clinton, J. (1999). CRISP-DM 1.0 Step-by-step data mining guide.

Number of monthly active Facebook users worldwide as of 2nd quarter 2022. (2022). https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/

Orabi, M., Mouheb, D., Aghbari, Z. A., & Kamel, I. (2020). Detection of Bots in Social Media: A Systematic Review. *Information Processing and Management*, *57*. https://doi.org/10.1016/j.ipm.2020.102250

Peters, M. E., Neumann, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep contextualized word representations*. http://allennlp.org/elmo

Pichihua, S. (2018). *Cada mes hay más de 300 denuncias por delitos informáticos*. https://andina.pe/agencia/noticia-cada-mes-hay-mas-300-denuncias-delitos-informaticos-830617.aspx#:~:text=La%5C%20Divisi%5C%C3%5C%B3n%5C%20de%5C%20Investigaci%5C%C3%5C%B3n%5C%20de,4%5C%2C

Purbasari, D. (2021). *CRISP-DM for Data Quality Improvement to Support Machine Learning of Stunting Prediction in Infants and Toddlers*.

Rostami, R. R., & Karbasi, S. (2020). Detecting fake accounts on twitter social network using multi-objective hybrid feature selection approach. *Webology*, *17*. https://doi.org/10.14704/WEB/V17I1/A204

Ryffel, T., Dufour-Sans, E., Gay, R., Bach, F., & Pointcheval, D. (2019). Partially encrypted machine learning using functional encryption. *Advances in Neural Information Processing Systems*, *32*.

Shi, P., Zhang, Z., & Choo, K. K. R. (2019). Detecting Malicious Social Bots Based on Clickstream Sequences. *IEEE Access*, *7*, 28855-28862. https://doi.org/10.1109/ACCESS.2019.2901864

SRIJAN, S. N. K. (2018). False Information on Web and Social Media: A Survey. *arxiv.*

Statista. (2023). El Universo Twitter en cifras.

Stenhouse, N. V. (2017). *Escola Politècnica Superior Memòria del Treball de Fi de Grau HabScraper: herramienta automatizada para la extracción de datos con web scraping*.

Subrahmanian, V. S., Azaria, A., Durst, S., Kagan, V., Galstyan, A., Lerman, K., Zhu, L., Ferrara, E., Flammini, A., & Menczer, F. (2016). The DARPA Twitter Bot Challenge. *Computer*, *49*, 38-46. https://doi.org/10.1109/MC.2016.183

Thiran, P., Library., A. D., ACM-Sigmetrics. & on Data Communication., A. S. I. G. (2011). *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference.* ACM.

TWTR. N - Twitter Inc. (2022). primicias.ec/noticias/tecnologia/twitter-cuantos-bots-tiene-son-problema/#:~:text=Y%20Twitter%2C%20el%20pasado%204,el%20primer%20trimestre%20de%202022.

Varol, O., Ferrara, E., Davis, C. A., Menczer, F., & Flammini, A. (2017). *Online Human-Bot Interactions: Detection, Estimation, and Characterization*.

Verma, P. K., Agrawal, P., Madaan, V., & Gupta, C. (2022). UCred: fusion of machine learning and deep learning methods for user credibility on social media. *Social*

Network Analysis and Mining, *12*.
https://doi.org/10.1007/s13278-022-00880-1

Walt, E. V. D., & Eloff, J. (2018). Using Machine Learning to Detect Fake Identities: Bots vs Humans. *IEEE Access*, *6*, 6540-6549.
https://doi.org/10.1109/ACCESS.2018.2796018

Wei, F., & Nguyen, U. T. (2019). Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. *Proceedings - 1st IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2019*, 101-109.
https://doi.org/10.1109/TPS-ISA48467.2019.00021

Zhou, Q., & Chen, G. (2020). An Efficient Victim Prediction for Sybil Detection in Online Social Network. *IEEE Access*, *8*, 123228-123237.
https://doi.org/10.1109/ACCESS.2020.3007458

ANEXOS Plan de tesis

1. PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1. Planteamiento del problema

Según la División de Investigación de Delitos de Alta Tecnología del Perú, el fraude informático el 2020 creció un 23 por ciento en comparación al 2018 (Pichihua, 2018). Mientras más crecen las redes sociales incrementan las funcionalidades que estas proveen y se vuelven una herramienta donde uno puede ofrecer servicios, realizar encuestas y socializar, pero mientras estas funcionalidades crecen también se incrementan los fraudes informáticos como puede ser phishing, suplantación de identidad y fraude en e-commerce, estas acciones son mayormente realizadas por programas que realiza tareas automáticamente denominados ”robots” o abreviados Bot.

En el artículo de (Walt & Eloff, 2018) nos explican que las cuentas Bots son usadas para el engaño de identidad principalmente en las redes sociales como Twitter, este crecimiento y la evolución de estos Bots es preocupante para los consumidores de esta red, haciendo que estos Bots sean principalmente usados para spammers, estafadores y ciberacosadores. El sitio de “TWTR. N - Twitter Inc”, 2022, Twitter estimó que aproximadamente el 5% de los 229 000 000 usuarios activos de Twitter son Bots más esta cifra no convenció a Elon Musk el cual aseguró que el número era 4 veces mayor dando problemas a la hora de comprar Twitter, ya que gracias a los Bots la cifra de usuario reales que le daba valor a Twitter como compañía no era real. Este crecimiento exponencial de Bots hace que la difusión de spam como noticias falsas genere un ambiente tenso tanto en los usuarios, como la región en la que se encuentren ubicados alterando y desfigurando la originalidad de nuestra sociedad y nuestros valores culturales, por estas razones es necesario detectar estos Bots y eliminarlos de las redes sociales.

1.2. Objetivos de la Investigación

1.2.1. Objetivo General

Detectar Bots en Twitter usando procesamiento de lenguaje natural y modelos de

predicción.

1.2.2. Objetivos Específicos

1. Aplicar la extracción de la información de perfiles públicos de Twitter.
2. Desarrollar el preprocesamiento de la información extraída de Twitter.
3. Aplicar el Procesamiento de Lenguaje Natural(BERT,GloVe) al texto de los

Tweets.

4. Desarrollar el entrenamiento de los modelos de predicción(Random Forest, Redes

Neuronales) con la información procesada.

5. Clasificar usuarios de Twitter(humano o Bot).

1.3. Preguntas de Investigación

1. ¿Es posible detectar Bots en Twitter usando procesamiento de lenguaje

natural y modelos de predicción?

2. ¿Es posible elaborar el preprocesamiento de la información extraída de Twitter?

3. ¿Qué tipo de procesamiento se tendría que aplicar para normalizar el

texto del perfil de usuario extraído?

4. ¿Qué modelo de predicción es el más eficiente para el entrenamiento del modelo

BERT con la información procesada?

5. ¿Es posible que un modelo de predicción pueda clasificar usuarios de

Twitter como humanos o Bots?

1.4. Línea y Sublínea de Investigación

- Línea de Investigación: Inteligencia Artificial.
- SubLínea de Investigación: Inteligencia artificial y computacional.

1.5. Solución Propuesta

1.5.1. Justificación e Importancia

Los medios sociales están destinados a irrumpir aún más en la vida cotidiana de las personas, con el potencial de transformar no solo las herramientas de comunicación, sino también las opiniones e incluso las vidas de individuos. Las redes sociales han asumido un rol estructural en la comunicación tanto para personas como para organizaciones. Por ejemplo, en el primer trimestre de 2019, Facebook contaba con 2.375 millones de usuarios activos ("Number of monthly active Facebook users worldwide as of 2nd quarter 2022", 2022), aproximadamente un tercio de la población mundial. Con la creciente influencia de las redes sociales, han surgido como una herramienta de gran alcance, susceptible de ser utilizada por cualquier individuo u organización. Sin embargo, a pesar de su uso legítimo, muchas entidades malintencionadas y buscadores de influencia recurren a ellas para fines ocultos.

Esta investigación proporcionará beneficios sustanciales a los usuarios de Twitter en diversos ámbitos, ya sea en el contexto empresarial o en espacios sociales, abarcando a quienes interactúan en actividades sociales. Su objetivo principal es permitir que los usuarios puedan realizar sus acciones sin la constante preocupación de que sus publicaciones u otras interacciones sean influenciadas de manera masiva y no auténtica por bots. Para lograr este propósito, se emplearán herramientas avanzadas como modelos de predicción supervisados y técnicas de Procesamiento de Lenguaje Natural (PLN). Estos métodos se aplicarán para analizar exhaustivamente los Tweets de cuentas, las cuales serán extraídas de Twitter mediante el uso de su API. El resultado final será una mayor confiabilidad y autenticidad en la interacción y el contenido de los usuarios en Twitter.

1.5.2. Descripción de la solución

Se utilizará la API de Twitter para extraer información de cuentas de Twitter y poder usarla en el modelo de predicción, esta información se define como texto de Tweets, nombre, usuario, fecha de creación, URL de imagen de perfil, URL de portada, dirección, idioma, cantidad a los que sigue, cantidad que les sigue y zona horaria. En cuanto al texto de las publicaciones, se aplicará unos pasos en los cuales se limpiará el texto, se procesará el texto, se extraerá las reacciones y se aplicará PLN el cual nos permitirá aprovechar los datos en texto que no se puedan pasar a números. Las fases por las cuales se procesará la información son:

1. Preprocesamiento: En esta etapa, los datos se preparan para poder ser procesados en las fases siguientes. Se eliminarán los caracteres innecesarios y se ajustarán al formato adecuado para el procesamiento.
2. Reacciones de los Tweets: En la siguiente etapa, se obtendrán las reacciones de los Tweets, como los "me gusta", las respuestas y los retweets.
3. Procesamiento de Lenguaje Natural (PLN): Finalmente, se aplicará PLN al texto ya procesado. El resultado de esta aplicación de PLN se utilizará como una característica para el modelo.

Tras completar estos pasos de procesamiento del texto, se dará inicio al entrenamiento del modelo de predicción, específicamente el modelo Random Forest, utilizando todos los datos previamente recolectados. Se espera que este modelo pueda lograr la detección de Bots con una precisión de al menos el 90%.

2. FUNDAMENTOS TEÓRICOS.

2.1. Antecedentes de investigación

En el artículo de (Varol et al., 2017) nos habla de los peligros en las redes sociales y como estos nos dejan cada vez más vulnerables a la manipulación. Para demostrar que estas cuentas pueden ser controladas por el software o comúnmente llamado como

Bots, mencionan que estos comportamientos pueden ser detectados por técnicas de Aprendizaje automático supervisado. Para ello se propone un marco para extraer una gran colección de características de datos y metadatos sobre usuarios de Twitter, incluidos amigos, contenido y el análisis de sentimiento en los Tweets, usando patrones de red y analizando el tiempo de actividad. Cuando se mezclan diferentes proporciones de cuentas anotadas manualmente y el conjunto de datos HoneyPot para obtener una precisión entre 0.90 y 0.94.

Por el contrario (Thiran et al., 2011), usa cuentas ya suspendidas de Twitter, utilizan un conjunto de datos para caracterizar el comportamiento y la vida útil de las cuentas de Spam, las campañas que ejecutan y el abuso generalizado de servicios web legítimos como los acortadores de URL y el alojamiento web gratuito. Los resultados muestran que el 77% de las cuentas de Spam identificadas por Twitter son suspendidas al día siguiente de su primer Tweet.

En cuanto al uso del PLN se tiene antecedentes como (**GloVe**) donde crean un nuevo modelo de regresión log-bilineal global llamado GloVe que combina las ventajas de las dos principales familias de modelos: la factorización matricial global y los métodos de ventana de contexto local. En este método usan un modelo específico de mínimos cuadrados ponderados que se entrena palabra por palabra y así hace un uso eficiente de las estadísticas, logrando un 0.75 en una tarea de analogía de palabras, mientras tanto a diferencia de GloVe los autores de (**ELMo**) desarrollan un nuevo tipo de representación profunda de palabras contextualizadas llamada ELMo (Embeddings from Language Model), usada para modelar tanto las características complejas del uso de las palabras como el modo en que estos usos varían en función del contexto lingüístico, a diferencia de GloVe este usa un modelo lingüístico bidireccional profundo. Esta configuración permite realizar un aprendizaje semisupervisado, en el que el biLM se

preentrena a gran escala y se incorpora fácilmente a una amplia gama de arquitecturas neuronales de PNL existentes, logrando mejorar el mismo modelo de GloVe un 10%.

2.1. Estado del Arte

La investigación comienza con una breve revisión sobre el tema de la detección de Bots en Twitter de (Alothali et al., 2019) que presenta la necesidad de desarrollar tecnologías que puedan detectar Bots en las redes sociales. Presentando técnicas recientes que han surgido y que están diseñadas para diferenciar entre cuentas de Bots y cuentas humanas; las técnicas más resaltadas usadas como base para la presente investigación son Neural Network, Decision Tree y Random Forest; así mismo la medición del rendimiento como Precisión, ROC, Recall, accuracy, Error Rate, F-measure y Confusion Matrix. (Walt & Eloff, 2018) identificó identidades falsas usando atributos como el ratio de amigos y seguidores, se usó algoritmos como Random Forest, Adaboost y SVM donde como resultado un F1 Score de 49.75. (Bindu et al., 2018) usó una base de HoneyBot donde identifican comunidades sobrepuestas de Bot spamers por medio de HyperGraphs llegando a tener una precisión de 0,90 con data del 2010. Con el fin de eliminar spam y Tweets malisiosos (Verma et al., 2022) utiliza un método que combina modelos RoBERT, BiLSTM y Random Forest, con datos OSN que contienen información de cuentas reales y falsos; el resultado obtuvo una precisión de 0.99 aproximadamente. (Ilias & Roussaki, 2021) para clasificar a los usuarios de Twitter en usuarios reales y Bots; observó que el uso de Logistic Regression y Random Forest demostró una precisión de 0.9906, lo que supera los enfoques de vanguardia existentes considerados. (Rostami & Karbasi, 2020) realizó varios test, donde los algoritmos que utilizaron fueron Random Forest y SVM que lograron resultados óptimos que son cercanos entre sí entre un 97.6% y 98%, de acuerdo con los criterios de rendimiento detallados. (Mazza et al., 2022) proponen un análisis cuantitativo a gran escala, al utilizar un clasificador de Random Forest para diferenciar entre cuentas humanas y trolls dando como resultado una

puntuación F1 de 0,99. (Barbon et al., 2018) utiliza Random Forest logrando un 94% de precisión usando Transformación Wavelet discreta, con diferentes datasets y diferentes parámetros. (Loyola-Gonzalez et al., 2019) crearon un modelo de clasificación utilizando un nuevo modelo basado en patrones y análisis de sentimiento con el contenido de los Tweets, para combinar este método con el Random Forest; el enfoque basado en patrones de contraste obtuvo resultados de clasificación superiores a 0,90 de AUC (Appropriate Use Criteria) y 0,91 de MCC. (Ryffel et al., 2019) presentaron una investigación con relación al BERT y texto de Twitters; utilizando diferentes modelos de aprendizaje como Logistic Regression, Random Forest, Naive Bayes y Support Vector Machines (SVM), desarrollaron un algoritmo de clasificación efectivo basado en aprendizaje automático y BERT. (Abuhassan et al., 2022) usaron un modelo de BERT acompañado del LSTM logrando la mayor efectividad donde el modelo obtiene una puntuación del 98,19% y una precisión del 98,37%. El modelo que desarrollaron nos sirve bastante para alcanzar la meta de poder utilizar BERT para la detección de Bots en esta tesis. (Blanco & Lourenço, 2022) propone un nuevo enfoque de aprendizaje profundo para comprender mejor cómo se transmiten los sentimientos optimistas y pesimistas en las conversaciones de Twitter sobre COVID-19. Usaron LNN, CNN y Bi-LSTM, donde BERT es utilizado para extraer características semánticas y se aplican varias arquitecturas de redes neuronales para la extracción y clasificación de características. Para los modelos basados en BERT, se aplicó el tokenizador BERT para transformar el texto preprocesado y esta representación se introdujo en el codificador de transformador del BERT. (Feng et al., 2022) decidió crear el mayor dataset hasta ahora con un millón de usuarios para luego reimplementar 35 modelos representativos de detección de Bots en Twitter y evaluarlos en 9 conjuntos de datos, incluido TwiBot-22, para promover una comprensión holística del progreso de la investigación. En su trabajo implementan modelo de predicción basado en gráficos que dieron un mejor

rendimiento con este dataset. (Gao et al., 2019) realizaron una investigación relacionada a modelos con Redes Neuronales, y trabajan redes tradiciones con BERT. El modelo propuesto fue llamado TD-BERT (Target-Dependent BERT), que alcanzó un nuevo rendimiento, comparando el rendimiento con algunos métodos de referencia como Rec-NN, TD-LSTM, ATAE-LSTM, BERT-FC, BERT-pair-QA-M, BERT-PT y AEN-BERT. La precisión promedio de la clasificación se ha llevado a los 70% o 80% con algunas dificultades para ciertas clases de datos. (Li et al., 2020) se enfocan más en la parte embedding de las palabras, introduciendo un nuevo método denominado GBCN (Gradient Boosted Chinese Restaurant Process Network) usado principalmente para mejorar y controlar la representación BERT para el análisis de sentimientos. Este modelo BERT lo aplicaron a un proceso de análisis de sentimiento que dio como resultado un F1
Score de 88.0. BERT no es el único modelo capaz de procesar el texto, modelos como GloVe que son usados en (Wei & Nguyen, 2019) logran un 98% de Score usando redes neuronales para entrenar un dataset del 2017 de Twitter, comparado a las otras técnicas esta logran un mejor resultado. Otra opción que se tiene a BERT o GloVe es ELMo el cual los autores de (Heidari et al., 2020) usan para los Tweets que extraen de las cuentas de Twitter junto a la metadata para entrenar una red neuronal con un Score del 94%. Modelos como BERT pueden ser usados para tokenizar el texto y para clasificar como lo hacen en (Heidari & Jones, 2020) donde se enfocan en sacar temas del texto para luego entrenar a una red neuronal donde consiguen un 94% de precisión. Mientras (Dukic et al., 2020) usan BERT para probar que la inclusión de características adicionales junto con incrustaciones contextualizadas mejora el rendimiento del modelo logrando un 83% de precisión. Aunque la mayoría de investigaciones usan PLN para procesar los Tweets que contienen texto sin embargo (Mohammad et al., 2020) obtienen las características del texto y su transformación de una forma cualitativa entrenando una Red Neuronal con una precisión de 90%.

2.2. Bases Teóricas de la Investigación

En primer lugar para poder identificar Bots en Twitter debemos entender como estas cuentas engañan a las personas y por qué la gente cree en ellos, (SRIJAN, 2018) nos explica en su investigación que esta información falsa puede ser clasificada en intención y conocimiento, la intención denominada desinformación que tiene como motivo mover a las masas y ganar dinero mientras que la segunda el conocimiento puede estar basada en opiniones y falsas afirmaciones las cuales no serán tomadas en esta presente investigación. También nos habla de los actores los cuales pueden ser personas o Bots y las razones por las que se cae en estos engaños, los cuales pueden ser la poca habilidad del usuario para distinguir la farsa, los ataques a un mismo grupo de personas con cualidad similares o la información falsa que parezca atrayente. Principalmente, se tomará en cuenta los aspectos más importantes como los actores implicados en la difusión de la información falsa, las razones para engañar con éxito a los lectores, la cuantificación del impacto de la información falsa, la medición de sus características en diferentes dimensiones, y finalmente, los algoritmos desarrollados para detectar la información falsa, estas características serán necesarios a la hora de recolectar las keywords y desarrollar el algoritmo de análisis.

2.2.2. Procesamiento de Lenguaje Natural (PLN)

PLN es un subcampo de la informática que se centra en permitir que los ordenadores entiendan el lenguaje de forma ”natural”, como lo hacen los humanos. Por lo general, esto se refiere a tareas como la comprensión del sentimiento del texto, el reconocimiento del habla y la generación de respuestas. PNL se ha convertido en un campo que evoluciona rápidamente y sus aplicaciones han representado una gran parte de los avances de la inteligencia artificial (II, 2018).

2.2.3. Random Forest

Random Forest (Breiman, 2001) es un enfoque de aprendizaje en conjunto para la clasificación, en el que los ”aprendices débiles” colaboran para formar ”aprendices fuertes”, utilizando una gran colección de árboles de decisión descorrelacionados (el bosque aleatorio). Sin embargo, en lugar de desarrollar una solución basada en la salida de un único árbol profundo, el bosque aleatorio agrega la salida de un número de árboles poco profundos, formando una capa adicional al bagging. El bagging construye N predictores, utilizando árboles sucesivos e independientes, mediante el uso de muestras del conjunto de datos. Los N predictores se combinan para resolver un problema de clasificación o estimación a través de un promedio. Aunque los clasificadores individuales son aprendices débiles, todos los clasificadores combinados forman un aprendiz fuerte. Mientras que los árboles de decisión individuales experimentan una alta varianza y un alto sesgo, el bosque aleatorio promedia múltiples árboles de decisión para mejorar el rendimiento de la estimación.

2.2.3. BERT

En el artículo de (McCormick, 2019) define a BERT como un codificador Transformer bidireccional que toma como entrada una o dos oraciones que al entrenarlo para el uso de predicción de valores usando tokens especiales para diferenciarlos, estos token siempre aparece al principio del texto y se configura específicamente para realizar las tareas de clasificación. Ambos tokens siempre son necesarios, incluso si solo tenemos una oración, e incluso si no estamos usando BERT para la clasificación. Así es como BERT fue pre-entrenado, y eso es lo que BERT espera ver. El modelo preentrenado se puede ajustar para tareas supervisadas posteriores y se ha demostrado que produce resultados de última generación en una serie de puntos de referencia de la PLN. Estos modelos se pueden usar para extraer características de

lenguaje de alta calidad de sus datos de texto, o puede ajustar estos modelos en una tarea específica con sus propios datos para producir predicciones de vanguardia.

2.2.4. Metodología CRISP-DM

En el artículo de (Ncr & Clinton, 1999) tomamos como referencia la metodología usada. El actual modelo de proceso proporciona una visión general del ciclo de vida de un proyecto. Contiene las fases de un proyecto, sus respectivas tareas y las relaciones entre éstas tareas. A este nivel de descripción, no es posible identificar todas las relaciones. Esencialmente, pueden existir relaciones entre cualquier tarea de minería de datos en función de los objetivos, los antecedentes y el interés del usuario y, sobre todo, de los datos. La secuencia de las fases no es rígida. Siempre es necesario avanzar y retroceder entre las distintas fases. Depende del resultado de cada fase qué fase o qué tarea concreta de una fase debe realizarse a continuación. Las flechas indican las dependencias más importantes y frecuentes entre las fases. Las fases de esta metodología son:

a) **Comprensión del negocio.** Esta fase inicial se centra en comprender los objetivos y requisitos del proyecto desde el punto de vista de la empresa y, a continuación, convertir estos conocimientos en una definición del problema de minería de datos y en un plan preliminar diseñado para alcanzar los objetivos de datos y un plan preliminar diseñado para alcanzar los objetivos.

b) **Comprensión de los datos.** La fase de comprensión de los datos comienza con una recopilación inicial de datos y continúa con actividades para familiarizarse con los datos, identificar los problemas de calidad de los datos, descubrir las primeras percepciones de los datos o detectar subconjuntos interesantes para formar hipótesis de información oculta.

c) **Preparación de los datos.** La fase de preparación de los datos abarca todas las actividades para construir el conjunto de datos final (datos que se introducirán

en la(s) herramienta(s) de modelización) a partir de los datos brutos iniciales. Es probable que las tareas de preparación de datos se realicen varias veces y no en un orden determinado.

d) Modelado de los datos. En esta fase se seleccionan y aplican diversas técnicas de modelización y se calibran sus parámetros hasta alcanzar los valores óptimos.

e) Evaluación. En esta fase del proyecto se ha construido un modelo (o modelos) que parece tener una alta calidad desde la perspectiva del análisis de datos.

f) Despliegue. La creación del modelo no suele ser el final del proyecto. Aunque el objetivo del modelo sea aumentar el conocimiento de los datos, los conocimientos adquiridos deberán organizarse y presentarse de forma que el cliente pueda utilizarlos.