UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

TESIS

IMPLEMENTACIÓN DE UN DATASET PARA LA EVALUACIÓN DE MODELOS DE ANÁLISIS DE SENTIMIENTOS EN LA CLASIFICACIÓN DE TWEETS

Para optar al Título Profesional de:
Ingeniero Informático y de Sistemas

Presentado por:
Bach. Karen Alexsandra Carbajal Bacilio
Bach. Claudia Francesca Suarez Mariscal

Asesor:
M.Sc. Yeshica Isela Ormeño Ayala

Financiado por CONCYTEC-UNSAAC-FONDECYT

CUSCO - PERÚ
2023

Agradecimientos

Agradecemos a Dios, por ser nuestra fortaleza en los momentos más difíciles y permitirnos llegar hasta esta etapa.
Agradecemos a nuestra casa de estudios, la Universidad Nacional de San Antonio Abad del Cusco, que nos albergo en sus aulas durante cinco años y nos impartió los conocimientos necesarios para nuestra formación profesional.
Agradecemos a la M.Sc. Yeshica Isela Ormeño Ayala, quien fue nuestra guía y soporte en este periodo, abriéndonos las puertas a la investigación científica.
Agradecemos a todos nuestros docentes, quienes por medio de sus sabias enseñanzas afianzaron nuestros conocimientos y permitieron desarrollar nuestra vocación durante los años de formación universitaria.
Agradecemos a aquellas personas que se involucraron de manera simbólica en esta tesis por medio de roles de 'etiquetadores' o simplemente con sugerencias o consejos para un mejor desarrollo de esta tesis.
Agradecemos a todos nuestros compañeros y amigos de aulas universitarias que por medio de vivencias permitieron afianzar una amistad genuina que perdurara por el resto de nuestra vida profesional.
Agradecemos al Laboratorio de Algoritmos y Análisis de datos de la Universidad Nacional de San Antonio Abad del Cusco, por haber puesto a nuestra disposición sus equipos para desarrollar esta tesis.

Dedicatorias

Este trabajo esta dedicado a toda mi familia; en especial a mi madre, Gladis, por ser mi modelo de esfuerzo y perseverancia; a mi padre, Gabriel, por brindarme su apoyo incondicional en cada una de mis decisiones; a mis hermanos, Marcelo y Silvia, por ser cómplices de muchas experiencias; y a mis mascotas, por ser seres invaluables en mi vida.

Claudia

Dedico este trabajo a mis padres, Wilbert y Vicentina, por apoyarme siempre; a mis amigos y amigas, que aportaron su granito de arena para lograr esta meta; y a mis dos compañeros felinos, por ser parte de mi vida

Alexsandra

Dedicamos este trabajo a nuestro querido país, Perú, porque creemos que nuestra nación es soberana, independiente y tolerante.

Claudia y Alexsandra

Resumen

El Análisis de Sentimientos (AS) permite evaluar opiniones o emociones en diferentes redes sociales respecto a determinados temas. Este tipo de análisis viene siendo empleado para identificar las tendencias que se manifiestan en coyunturas de interés social, tales como la pandemia, político, economía, etc., lo que permite analizar la información que circula en las redes sociales y conocer si realmente esta asociada a la realidad. El objetivo de esta investigación se centra en evaluar el etiquetado de tweets mediante Pysentimiento, un modelo transformer especializado para el analisis de texto, y posteriormente utilizar este dataset para entrenar modelos de aprendizaje supervisado no especializados en la tarea del análisis de sentimiento. Para ello se propone la construcción de dos datasets, de escenario político y pandemia, mediante una metodología que incluye a Twint, para la extracción de tweets, y a Pysentimiento, para etiquetar los datos automáticamente. También se incluye una etapa de verificación manual de una porción de los datos, para evaluar el impacto de contar con datos etiquetados manualmente en la fase de entrenamiento de los modelos. En tanto para realizar el análisis de sentimientos, los datos son pre-procesados y transformados previamente a la aplicación de cuatro modelos clasificadores de Machine Learning(ML): Naive Bayes (NB), Super Vector Machine (SVM), Random Forest (RF) y Multilayer Perceptron (MLP). Así mismo se propone una fase experimental para aumentar el rendimiento de estos modelos, de los cuales se obtiene que: (1) se debe trabajar basado en el análisis de las clase positiva y negativa (de acuerdo a los datasets obtenidos) con una proporción estratificada para el train/test, (2) la verificación manual mejora los resultados para ambos escenarios, (3) Naive Bayes es el mejor clasificador con 88,4 % de exactitud, 83,4 % de precisión, 89,8 % de exhaustividad y 86,5 % de F1-score, mientras que para el escenario pandemia Super Vector Machine es considerado como el mejor clasificador con 85 % de exactitud, 86,1 % de precisión, 77,8 % de exhaustividad y 81,7 % de F1-score.

Palabras claves: Análisis de sentimientos, Twint, Pysentimiento, Naive Bayes, Super Vector Machine, Random Forest, Multilayer Perceptron.

Abstract

Sentiment Analysis (AS) allows one to evaluate opinions or emotions in different social networks regarding certain topics. This type of analysis has been used to identify the trends that are manifested in situations of social interest, such as the pandemic, politics, economy, etc., which makes it possible to analyze the information that circulates on social networks and find out if it is associated with reality. The objective of this research is focused on evaluating the tagging of tweets using Pysentiment, a specialized transformer model for text analysis, and later using this dataset to train supervised learning models not specialized in the task of sentiment analysis. For this, the construction of two datasets, of political scenario and pandemic, is proposed through a methodology that includes Twint, for the extraction of tweets, and Pysentimiento, to label the data automatically. A stage of manual verification of a portion of the data is also included, to evaluate the impact of having manually labeled data in the training phase of the models. While carrying out the sentiment analysis, the data is pre-processed and transformed before the application of four Machine Learning (ML) classifier models: Naive Bayes (NB), Super Vector Machine (SVM), Random Forest (RF), and Multilayer Perceptron (MLP). As a result of the execution of the experiments, it is obtained that: (1) one must work based on the analysis of the positive and negative classes (according to the datasets obtained) with a stratified proportion for the train/test, (2) manual verification improves the results for both scenarios, (3) Naive Bayes is the best classifier with 88.4 % accuracy, 83.4 % precision, 89.8 % completeness, and 86.5 % reliability. F1-score, while for the pandemic scenario, Super Vector Machine is considered the best classifier with 85 % accuracy, 86.1 % precision, 77.8 % completeness, and 81.7 % of F1- score.

Keywords: Sentiment Analysis, Twint, Pysentimiento, Naive Bayes, Super Vector Machine, Random Forest, Multilayer Perceptron.

Introducción

Actualmente, el Análisis de Sentimientos posee la capacidad de evaluar opiniones o emociones en diferentes redes sociales respecto a determinados temas. Este tipo de análisis viene siendo empleado para identificar las tendencias que se manifiestan en coyunturas de interés social, tales como la pandemia, político, economía, etc., lo que permite analizar la información que circula en las redes sociales y conocer si realmente esta asociada a la realidad. Así mismo, esta tarea debe lidiar con diferentes tipos de información contenida en el *tweet* (hiperlinks o emoticones), con la volatilidad del idioma y el parafraseo, resolver problemas como la subjetividad de palabras, contexto de expresiones, determinar expresiones de ironía y sarcasmo, desambiguación, negaciones, entre otros.

Para construir un *dataset* con *tweets* netamente crudos requiere una fase de etiquetado para determinar el sentimiento de la expresión. Este proceso habitualmente se realiza de forma manual, pero posee mayor costo de tiempo y recursos porque es realizado por varias personas para lograr la confiabilidad esperada. Para este proceso podría ser reemplazado por un etiquetado automático utilizando un modelo especializado para el análisis de texto. Al revisar la literatura, no se encontró una metodología referencial que contemple este proceso desde la recolección de los datos hasta el etiquetado, por tanto mediante esta investigación se incorporan herramientas, de extracción de *tweets* y etiquetado de *tweets*, que permitan establecer el proceso para conseguir una correcta implementación de un *dataset*.

El objetivo de esta investigación se centra en evaluar el etiquetado de *tweets* mediante *Pysentimiento*, un modelo *transformer* especializado para el análisis de texto, y posteriormente utilizar este *dataset* para entrenar modelos de aprendizaje supervisado no especializados en la tarea del análisis de sentimiento. Para ello se propone la construcción de dos *datasets*, de escenario político y pandemia, mediante una metodología que incluye a *Twint*, para la extracción de *tweets*, y *Pysentimiento*, para etiquetar los datos automáticamente. También se incluye una etapa de verificación manual de una porción de los datos, para evaluar el impacto de contar con datos etiquetados manualmente en la fase de entrenamiento de los modelos. En tanto para realizar el análisis de sentimientos, los datos son pre-procesados y transformados previamente a la aplicación de cuatro modelos clasificadores de *Machine Learning*(ML): Naive Bayes (NB), *Super Vector Machine* (SVM), *Random Forest* (RF) y *Multilayer Perceptron* (MLP). Así mismo se propone una fase experimental para aumentar el rendimiento de estos modelos, de los cuales se obtiene que: (1) se debe trabajar basado en el análisis de las clase positiva y negativa (de acuerdo a los *datasets* obtenidos) con una proporción estratificada para el *train/test*, (2) la verificación manual mejora los resultados para ambos escenarios, (3) Naive Bayes es el mejor

clasificador con 88,4 % de exactitud, 83,4 % de precisión, 89,8 % de exhaustividad y 86,5 % de F1-score, mientras que para el escenario pandemia *Super Vector Machine* es considerado como el mejor clasificador con 85 % de exactitud, 86,1 % de precisión, 77,8 % de exhaustividad y 81,7 % de F1-score.

La presente tesis tiene la siguiente estructura:

- **Capitulo 1 Aspectos generales:** Capítulo que comprende: el problema de investigación, antecedentes, justificación, objetivo general y los objetivos específicos, alcances y limitaciones, la metodología usada, los resultados esperados y el cronograma de actividades.
- **Capitulo 2 Marco teórico:** Se presentan los conceptos importantes relacionados a la investigación, estos conceptos clarifican la base para el desarrollo del proyecto.
- **Capítulo 3 Desarrollo del proyecto:** Se desarrolla la construcción de *dataset*, así como el proceso del análisis de sentimientos para el escenario político y pandemia.
- **Capítulo 4 Análisis de resultados:** Capıtulo donde se muestran los resultados obtenidos durante el entrenamiento de cuatro modelos de *Machine Learning* para el análisis de sentimientos.
- **Capıtulo 5 Conclusiones, recomendaciones y trabajos futuros:** Se desarrolla las conclusiones del trabajo de investigación, recomendaciones y trabajos futuros.

Índice

Índice de figuras

Índice de tablas

Abreviaturas

API Application Programming Interface.
AST Análisis de Sentimientos de Tweets.

BERT Bidirectional Encoder Representations from Transformers.

FN Falso Negativo.
FP Falso Positivo.

IDF Inverse document frequency.

ML Machine Learning.
MLP Multilayer perceptron.

NB Naive Bayes.
NEG Negativo.
NEU Neutro.
NLTK Natural Language Toolkit.

PLN Procesamiento de Lenguaje Natural.
POS Positivo.

re Regular Expression Operations.
RF Random Forest.

SVM Super Vector Machine.

TF Term frequency.

VM verificación manual.
VN Verdadero Negativo.
VP Verdadero Positivo.

Capítulo 1

Aspectos Generales

1.1. Problema de la investigación

1.1.1. Planteamiento del problema

En las redes sociales, la opinión de cada uno de los usuarios es la actividad central para determinar diferentes comportamientos. A diario, la interacción entre los seres humanos y las redes sociales van aumentando, por ejemplo, Twitter alberga información vital de usuarios que expresan puntos de vistas de acontecimientos en tiempo real (Greenwade, 1993). De este modo, esta gran cantidad de datos contribuye a que El Análisis de Sentimientos de Tweets (AST) sea un nuevo enfoque de investigación en las redes sociales, ya que es conducido para el estudio de opiniones, actitudes, valoraciones o emociones hacia diferentes productos, eventos, organizaciones, temas, personajes, etc (Liu, 2012).

El AST no es una tarea trivial, debido a que lidia con diferentes tipos de información contenida en el *tweet* (hiperlinks o emoticones); también se debe resaltar la volatilidad del idioma y el parafraseo, debido a que los usuarios expresan sus ideas en pocas o muchas palabras (Giachanou and Crestani, 2016). En medio de esta problemática, se requiere asociar este análisis con el Procesamiento de Lenguaje Natural (PLN) que implica la tarea de encontrar el significado semántico del contenido del texto (Kanakaraj and Guddeti, 2015), que a su vez debe resolver problemas como la subjetividad de palabras, contexto de expresiones, determinar expresiones de ironía y sarcasmo, desambiguación, negaciones, entre otros (Becker and Tumitan, 2013).

Para efectuar ciencia de datos, se debe de contar con un *dataset* que recopile la información necesaria para alimentar a los modelos de clasificación. En el ámbito de extracción datos, los investigadores pueden optar por (1) construir un *dataset* o (2) utilizar *datasets* de repositorios. Para el primer aspecto, Twitter dispone de la información a través de las APIs por medio de un registro de autenticación con diferentes niveles de acceso. Algunas de estas pueden tener un costo de suscripción, pero la mayoría son libres; sin embargo en el search API, de nivel standard y libre acceso, solo permite: (a) recuperar 500 mil *tweets* por mes, (b) aplicar 5 filtros básicos, (c) realizar 25 peticiones en 15 minutos (máximo de 100 *tweets* por petición) y (d) obtener *tweets* de los ultimos 7 días. Para el segundo aspecto, la mayoría de *datasets* que se encuentran en los repositorios están en el idioma ingles,no están actualizados y albergan múltiples escenarios.

Entre tanto construir un *dataset* con *tweets* netamente crudos requiere una fase de etiquetado, en la cual se determina el sentimiento de la expresión. Este proceso habitualmente se realiza de forma manual, pero posee mayor costo de tiempo y recursos económicos porque es realizado por varias personas para lograr la confiabilidad esperada; de este modo este proceso podría ser reemplazado por un etiquetado automático de acuerdo a un modelo especializado para el análisis de texto. Al revisar la literatura, no se encontró una metodología referencial que contemple este proceso desde la recolección de los datos hasta el etiquetado, por tanto mediante esta investigación se incorporan herramientas, de extracción de *tweets* y etiquetado de *tweets*, que permitan establecer el proceso para conseguir una correcta implementación de un *dataset*.

Finalmente, en nuestra investigación emplea el análisis de sentimientos aplicando cuatro modelos de aprendizaje supervisado para la clasificación de texto, que son: Naive Bayes, Super Vector Machine, Random Forest y Multilayer Perceptron, sin embargo aun no se conoce el rendimiento de estos utilizando un *dataset* etiquetado por medio de especializado para el análisis de texto.

1.1.2. Formulación del problema

¿Cómo implementar un *dataset* para la evaluación de modelos de análisis de sentimientos en la clasificación de *tweets*?

1.2. Antecedentes

El análisis de sentimientos con datos de Twitter viene siendo una área muy explorada recientemente, debido a que Twitter es una red social que viene proporcionando públicamente sus datos mediante las APIs. En un trabajo reciente, Antonakaki (Antonakaki et al., 2021) incluye una metodología que engloba las principales etapas que deben ser aplicadas al análisis de sentimientos. La investigación recopila trabajos propuestos que se enfocan en diferentes formas de extraer *tweets* sin utilizar APIs, así la evaluación de diferentes formas de emplear el etiquetado de *tweets*.

1.2.1. Extracción de datos

De acuerdo con la literatura, existen dos maneras de llegar a conseguir los datos: (1) obtener *dataset* publico o (2) creación de un nuevo *dataset*. Para la primera opción, muchos *datasets* se encuentran disponibles en repositorios públicos como son Kaggle [1], TASS Dataset [2], entre otros. Para el segundo, se cuenta con el API de Twitter y diferentes herramientas de web scraping. A continuación, se presenta algunas propuestas que utilizan estas herramientas:

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

[2] http://tass.sepln.org/tass_data/download.php

- **Kausar, M. A., Soosaimanickam, A., & Nasar, M. (2021). Public sentiment analysis on Twitter data during COVID-19 outbreak. International Journal of Advanced Computer Science and Applications, 12(2).**

  Este trabajo aplica el análisis de sentimientos en *tweets* publicados en inglés con el objetivo de comprender cómo las personas de diferentes países reaccionaron frente a la situación de la pandemia (Kausar et al., 2021). Los *tweets* recopilados son de los 10 países con mayor número de infectados en el mundo entre Junio del 2020 a Julio del 2021. La extracción de datos es a través de la API de Twitter utilizando el paquete RTweet por medio de *hashtags*. Posteriormente estos son pre-procesados, luego se calcula el sentimiento usando el paquete syuzhet y finalmente los sentimientos encontrados son analizados.

  Como resultado, este trabajo proporcionó un buen análisis sobre los sentimientos de las personas hacia el Covid-19 en distintos lugares del mundo. Este trabajo podría ser utilizado en trabajos futuros que deseen analizar opiniones sobre el Covid-19 en determinadas regiones y posteriormente comparar dichos sentimientos.

- **Dangi, D., Dixit, D. K., & Bhagat, A. (2022). Sentiment analysis of COVID-19 social media data through machine learning. Multimedia Tools and Applications, 1-23.**

  Este trabajo se basa en la evaluación de cinco modelos: regresión logística, *Random Forest*, Naive Bayes multinomial, *Super Vector Machine* y Decision Tree. Para identificar ventajas o desventajas, son recopilados *tweets* con Twint Scraping Tools[3]. Los *tweets* son recuperados por medio de palabras claves, entre Marzo y Junio del 2021, bajo dos contextos, el primer conjunto de datos pertenece a los que fueron publicados durante la cuarentena y el segundo conjunto de datos, posterior a la cuarentena. Luego estos son etiquetados como positivo, negativo o neutro por medio de la polaridad hallada por TextBlob (Dangi et al., 2022).

  El resultado es un framework llamado Análisis de sentimiento de los datos de las redes sociales de Twitter, el cual analizo y predijo el sentimiento de los *tweets* en función de Covid-19 mediante los clasificadores de aprendizaje automático ya mencionados. Además propone como trabajo futuros, incluir clasificadores de aprendizaje profundo para procesar más estos datos y extraer resultados más precisos.

- **Thelwall, M., Kousha, K., & Thelwall, S. (2021). Covid-19 vaccine hesitancy on English-language Twitter. Profesional de la información (EPI), 30(2).**

  Este trabajo aplica el análisis de sentimiento a una muestra aleatoria de *tweets* en ingles respecto al Covid-19 que dudan sobre la vacuna. Los principales temas discutidos fueron las conspiraciones, la velocidad de desarrollo de la vacuna y la seguridad de la vacuna. Estos fueron clasificados en dos grupos, por un lado los que tenían una posición dubitativa a vacunarse y el otro grupo, presentaba una posición rotundamente negativa a vacunarse

[3]https://github.com/twintproject/twint

(Thelwall et al., 2012). El software gratuito Mozdeh fue utilizado para extraer *tweets* generales de Covid-19 en un rango de tiempo y posteriormente se seleccionaron utilizando cuatro consultas: *corona virus, coronavirus, covid-19 y covid19.*

Como resultado, este trabajo identifico *tweets* de desinformación sobre la vacuna contra el Covid-19 permitiendo abordar las influencias negativas y comprender cómo se propagan. También se identifico los perfiles que poseen los usuarios que propagan dicha desinformación.

- **Sandoval-Almazán, R., & Valle-Cruz, D. (2016, June). Understanding network links in Twitter: A Mexican case study. In Proceedings of the 17th International Digital Government Research Conference on Digital Government Research (pp. 122-128)**

  Este trabajo tuvo como propósito agrupar perfiles de usuario de acuerdo a sus opiniones emitidas en Twitter, así como evaluar el impacto de las redes sociales durante acontecimientos específicos (Sandoval-Almazán and Valle-Cruz, 2016). Son obtenidos una muestra de 1592 *tweets* en español, entre el 24 y 26 de setiembre, utilizando la herramienta Netlytic, a partir de estos *tweets* se efectúa una clusterización que permite establecer 5 grupos: simpatizantes, ciudadanos, periodistas, funcionarios gubernamentales y otros. Netlytic software se utilizado para la recolección de datos y para la aplicación del algoritmo del cluster.

  Como resultado, este trabajo logro establecer perfiles de usuarios de acuerdo a las opiniones emitidas, lo cual permite desarrollar políticas públicas de acuerdo a las preocupaciones de los ciudadanos. Además, se aporta la metodología utilizada en este proceso.

- **Müller, M., Salathé, M., & Kummervold, P. E. (2020). Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. arXiv preprint arXiv2005.07503.**

  Este trabajo propone COVID-Twitter-BERT (CT-BERT), un modelo basado en BERT, entrenado previamente en un gran corpus de mensajes de Twitter sobre el tema de Covid-19 (Müller et al., 2020). CT-BERT está optimizado para ser usado con contenido respecto al Covid-19, en particular de las redes sociales, logrando mejores resultados en comparación con su modelo base. CT-BERT se entrena utilizando un corpus de *tweets* sobre el coronavirus, recopilados a través de la plataforma Crowdbreaks, el cual utiliza un conjunto de palabras clave relacionadas con Covid-19 en el idioma en inglés.

  Este trabajo concluye en que CT-BERT mejora significativamente los resultados de clasificación frente a BERT-LARGE. Asi mismo CT-BERT presenta mejorías en la evaluación con *datasets* de otras redes sociales. Finalmente, consideran como trabajo futuro dedicarse al finetuning de CT-BERT para el trabajo con otras tareas de procesamiento de texto.

- **Swaminathan, R. T., Balaji, V., & Subramanian, S. Chennai Floods 2021 Sentiment Analysis of Twitter Data using Tweepy and TextBlob.**

  Este trabajo utilizo el análisis de sentimientos para la clasificación de *tweets* durante las inundaciones de Chennai en el 2021 por el lapso de 14 días (Swaminathan et al.,). La herramienta Tweepy es empleada para la extracción de datos, mientras que TextBlob calcula la puntuación de sentimiento con base en la subjetividad y polaridad, que se requiere para el etiquetado. Los autores aseveran que esta metodología puede ser aplicada de forma eficiente sobre grandes cantidades de datos y en diversos escenarios.

  Este trabajo concluyo que las redes sociales es una fuente vital para el análisis de datos. Además se estableció una metodología que puede ser empleada por investigadores con intención de estudiar de manera eficiente los sentimientos sobre grandes cantidades de datos.

En conclusión, existen diferentes herramientas para la extracción de *tweets* que establecen una conexión con las APIs de Twitter, tales como RTweet o Tweepy. Sin embargo, Twint Scraping Tools opta por no utilizar las APIs, lo cual permite recolectar *tweets* más antiguos que da acceso a los *tweets* de años pasados durante eventos suscitados. De lo analizado, se observa que es viable utilizar una herramienta que pueda lidiar con esta limitación, como es el caso de Twint.

1.2.2. Etiquetado de datos

Para el etiquetado de *tweets*, según Giachanou (Giachanou and Crestani, 2016) se pueden establecer los siguientes enfoques (1) etiquetado manual, (2) distant supervision y (3) modelos de Machine Learning. A continuación, se describen recientes propuestas que incluyen este enfoque

- **Mujahid, M., Lee, E., Rustam, F., Washington, P. B., Ullah, S., Reshi, A. A., & Ashraf, I. (2021). Sentiment analysis and topic modeling on *tweets* about online education during COVID-19. Applied Sciences, 11(18), 8438.**

  La investigación de Mujahid realiza la construcción de un *dataset* relacionado a la educación durante pandemia (Mujahid et al., 2021), este contiene 17155 *tweets* para evaluar el desempeño de modelos de clasificación mediante las métricas de exactitud, precisión, exhaustividad, y F1 score. Se realiza un filtrado por medio palabras claves y se seleccionan los atributos de username, location y text. También se emplea el pre-procesamiento para limpiar los datos y eliminar datos irrelevantes. Para el etiquetado se usa TextBlob, una librería de Python, para obtener puntuaciones con base en la polaridad de los *tweets* y clasificarlos como positivo, negativo y neutro. Como resultados, se identifican los principales problemas para poder tomar acciones que mejoren la educación en linea.

- **Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using Distant Supervision. CS224N project report, Stanford, 1(12), 2009.**

  Este trabajo desarrolla la idea de usar *tweets* con emojis para el distant supervised learning (Go et al., 2009). Proponen un método automatizado para extraer el sentimiento del *tweet* (positivo o negativo). Como solución, se utiliza Distant Supervision, mediante el cual los emojis serán considerados como una 'etiqueta'. Por ejemplo *')'*, indica que el *tweet* contiene un sentimiento positivo y *'('*, indica que el *tweet* contiene un sentimiento negativo. Esto es una mejora significativa con respecto a las horas que tomaría etiquetar los datos de forma manual; puesto que los datos de entrenamiento son *tweets* que incluyen emojis. Como resultado, Naive Bayes, Maximum Entropy y *Super Vector Machine* obtuvieron una exactitud mayor al 80 % siendo entrenados con un *dataset* de *tweets* que contengan emojis.

- **Van Atteveldt, W., Van der Velden, M. A., & Boukes, M. (2021). The validity of sentiment analysis Comparing manual annotation, crowd-coding, dictionary approaches, and machine learning algorithms. Communication Methods and Measures, 15(2), 121-140.**

  Este trabajo utiliza un *dataset* de titulares económicos holandeses para comparar el rendimiento de 4 métodos para el etiquetado de datos con anotación manual, el crowdcoding, diccionarios y Machine Learning (Van Atteveldt et al., 2021). Se concluye que (1) El mejor rendimiento aún se logra con el etiquetado humana o crowd coding; (2) Ninguno de los diccionarios utilizados se acerca a niveles aceptables de validez; y (3) el aprendizaje automático supera sustancialmente a los métodos basados en diccionarios, pero no alcanza el rendimiento humano. Como resultado de estos hallazgos, se destaca la importancia de validar siempre los métodos de análisis de texto automático antes de su uso. Además, proporciona un enfoque automatizado recomendado para proyectos de análisis de sentimiento en texto para garantizar la eficiencia y validez.

- **Tejani, A. S., Ng, Y. S., Xi, Y., Fielding, J. R., Browning, T. G., & Rayan, J. C. (2022). Performance of Multiple Pretrained BERT Models to Automate and Accelerate Data Annotation for Large Datasets. Radiology Artificial Intelligence, 4(4), e220007.**

  Este trabajo se desarrolla y evalúa BERT en diferentes cantidades de datos para el etiquetado de datos general. Se seleccionan 1004 informes de un total de 69 095 informes anónimos de radiografías. Los modelos pre-entrenados (BERT, PubMedBERT, DistilBERT, Roberta y DeBERTa), fueron validados mediante diferentes tamaños del *dataset* (Tejani et al., 2022). Finalmente se llega a la conclusión de que los modelos de BERT pre-entrenados pueden realizar un entrenamiento adicional con conjuntos de datos en menor tiempo, lo que genera un modelo final de alta precisión que acelera la anotación autónoma de grandes conjuntos de datos.

- **Nakov, P., Rosenthal, S., Kiritchenko, S., Mohammad, S. M., Kozareva, Z., Ritter, A., ... & Zhu, X. (2016). Developing a successful SemEval task in sentiment analysis of Twitter and other social media texts. Language Resources and Evaluation, 50(1), 35-65.**

  Este trabajo recolecta *tweets* de diferentes temas populares. Se realizo una limpieza de datos irrelevantes, para posteriormente etiquetar el *dataset* usando la plataforma de Amazon's Mechanical Turk [4]. Cada *tweet* es etiquetado por cinco anotadores como positivo, negativo o neutro; bajo este método se puede llegar a etiquetar una cantidad limitada de *tweets* (Nakov et al., 2016). La calidad del etiquetado se mide respecto a la concordancia entre los resultados de los anotadores, por otro lado tanto anotadores como plataforma obtienen un beneficio monetario por disponibilizar los datos. La creación del *dataset* se realiza con el fin de desarrollar y avalar una tarea de análisis semántico aplicado al análisis de sentimientos.

El etiquetado automatizado es un método que ahorra tiempo frente al etiquetado manual, sin embargo un método automatizado no siempre sera preciso debido a que este incluye una tasa de error (Giachanou and Crestani, 2016). Muchos investigadores optan por el etiquetado manual, pero en caso se posea un *dataset* de gran tamaño, se deben evaluar otras alternativas como Distant Supervision o modelos pre-entrenados como BERT, DeBERTa y DistilBERT. Para esta investigación se opta por utilizar un modelo de etiquetado automático e incluir una fase de verificación manual, que permitirá detectar errores propios de un modelo automatizado.

1.3. Justificación

Esta investigación propone extraer datos de Twitter, sin hacer uso de las APIs, ya que existen otras herramientas que nos permiten acceder de manera simplificada a una gran cantidad de datos sin limites de tiempo. Por otro lado, se busca reemplazar el etiquetado manual por un etiquetado automático a través de Pysentimiento, un modelo transformer especializado para el análisis de texto, esto permite reducir el costo de tiempo y recursos. Adicionalmente, se evalúa este etiquetado por medio de modelos de aprendizaje supervisado que permitan analizar el rendimiento en la tarea de clasificación de texto de los *tweets*.

El impacto social de este trabajo se centra en el análisis de opiniones en Twitter para diferentes escenarios en tiempo real como son las campañas políticas, campañas publicitarias, campañas de concientización, etc. Este tipo de trabajo puede servir como precedentes para trabajos que se enfoquen en el estudio de las tendencias en las redes sociales, ya sea desde diversos puntos de vista como comerciales, marketing, sociales, etc.

Las implicaciones practicas conllevan a que la metodología sea escalable para otro idioma y otros escenarios. Además la disponibilización de los *datasets* contribuye a que los datos sean empleados para otras investigaciones que involucren el análisis de sentimiento en el lenguaje español, facilitando otras

[4] https//www.mturk.com/

formas de clasificación para el etiquetado a partir de la clasificación desarrollada en el presente trabajo.

Su valor teórico es mostrado mediante el uso de técnicas de PLN, así como la empleabilidad de modelos de Machine Learning durante la fase de experimentación. Por medio de las métricas de evaluación se espera conocer el comportamiento de nuestros datos con cada modelo y también conocer mejor configuración que se adapte al dataset.

1.4. Objetivos

1.4.1. Objetivo General

Implementar un *dataset* para la evaluación de modelos de análisis de sentimientos en la clasificación de *tweets*.

1.4.2. Objetivos Específicos

- Proponer una metodología para la construcción del dataset.
- Utilizar la metodología general para el proceso de análisis de sentimientos de *tweets*
- Aplicar técnicas de Procesamiento de Lenguaje Natural en el pre-procesamiento para realizar el análisis de sentimientos con los modelos de clasificación
- Entrenar y aplicar modelos de clasificación Naive Bayes (NB), Super Vector Machine (SVM), Random Forest (RF) y Multilayer perceptron (MLP), y obtener el mejor clasificador
- Evaluar modelos de aprendizaje supervisado para el análisis de sentimientos de *tweets* con *datasets* etiquetados mediante Pysentimiento
- Determinar los mejores criterios para obtener el conjunto de datos que provea mayor conocimiento a los modelos de clasificación durante el entrenamiento
- Analizar el impacto de incluir datos verificados manualmente en el proceso de aprendizaje de los modelos de clasificación

1.5. Alcances y Limitaciones

1.5.1. Alcances

- El *dataset* creado contendrá *tweets* en español extraídos de la red social Twitter
- La extraccion de *tweets* es realizada mediante Twint, herramienta de web scraping
- Se etiquetara el *dataset* de forma automatizada, asi mismo se incluye una etapa de verificación manual solo para una porción de datos

- El etiquetado de *tweets* se realiza considerando solo tres clases: Positivo, Negativo y Neutro (Go et al., 2009), (Nakov et al., 2016), (Giachanou and Crestani, 2016), (Zahra et al., 2018), (Müller et al., 2020), (Mujahid et al., 2021),(Dangi et al., 2022). Cabe recalcar que no se toman en cuenta sentimientos como amor, odio, tristeza, euforia, admiración, envidia, esperanza enojo, impaciencia, satisfacción, culpa, preocupación, gratitud, etc. ya que la herramienta usada para el etiquetado no realiza este tipo de clasificación; este requiere otro tipo de modelos (clusterización) que no están dentro del aprendizaje supervisado
- Se utilizaran las métricas de exactitud, precisión, exhaustividad y F1-score para la evaluación de los modelos (Hossin and Sulaiman, 2015), (Giachanou and Crestani, 2016), (Grandini et al., 2020)
- Se extraen los *tweets* mediante una herramienta de web scrapping (Twint)
- Se utiliza un modelo pre-entrenado para el etiquetado de datos (Pysentimiento)
- Para la fase de pre-procesamiento de *tweets* se realizara la normalización de mayúsculas y minúsculas, duplicidad de caracteres, eliminación de *retweets*, tratamiento de caracteres especiales(urls, *usernames*, *hashtags*, signos de puntuación) (Cerón-Guzmán and León-Guzmán, 2016)
- Para el tratamiento de *retweets*, se utiliza la similitud de coseno que permite eliminar aquellas oraciones similares (Rahutomo et al., 2012)

1.5.2. Limitaciones

- Se hace uso de la herramienta Twint para recuperar *tweets* mediante web scrapping
- Fueron analizados aquellos *tweets* publicados a partir de 2020.
- No se consideraron elementos como fotos, vídeos, links, imágenes, para el análisis de sentimientos
- No se realiza tratamientos especiales para los siguientes caracteres tildes y números. Por otro lado, tampoco se consideran las jergas o risas debido a que su presencia es mínima, dado que los *tweets* corresponden a expresiones formales

1.6. Metodología

La metodología que se presenta a continuación esta asociada al cronograma que se presenta en la sección 1.8, el cual muestra un tiempo estimado para cada una de las etapas. De la misma forma, la Figura 1 muestra gráficamente la metodología propuesta, donde se expone la secuencia de etapas que se siguen para alcanzar el objetivo trazado.

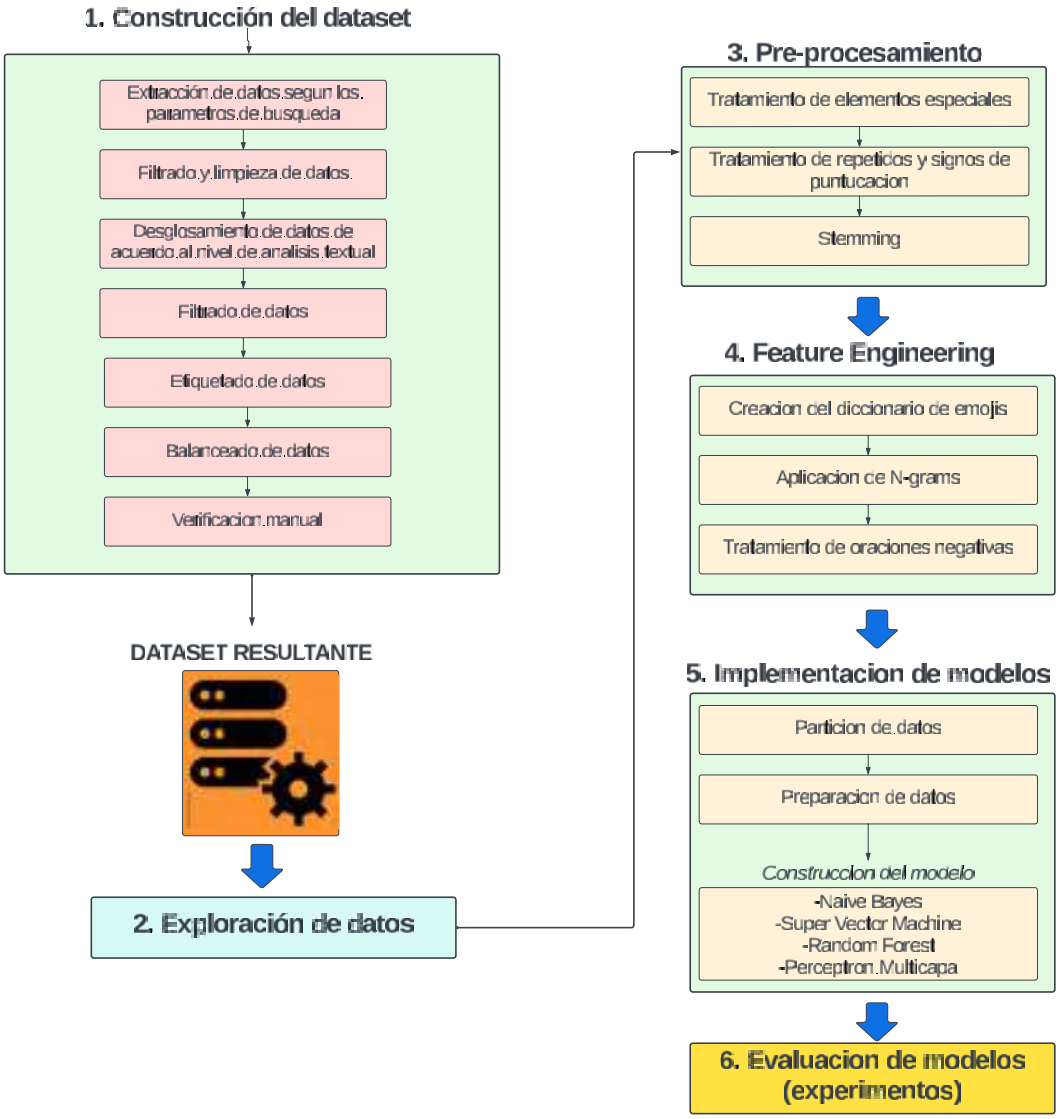

Figura 1: *Metodología propuesta*
Fuente *Elaboración propia*

1. **Construcción del *dataset*** Esta etapa esta dividida en dos fases (1) Extracción con *hashtags* y (2) Extracción con *usernames*.
   - **FASE 1: Extracción con *hashtags*:** Esta fase comprende (1) la extracción de *tweets* por medio de *hashtags* con Twint, (2) el filtrado y limpieza de datos y (3) el etiquetado de datos. De acuerdo a los resultados se evalúa proceder con otra extracción de *tweets* mediante *usernames*.
   - **FASE 2: Extracción con *usernames*:** Esta fase comprende (1) la extracción de *tweets* por medio de *usernames* con Twint, (2) el filtrado y limpieza de datos, y (3) el etiquetado de datos.

   Los datos obtenidos mediante cada fase son unidos, posteriormente se realiza el balanceado de datos, el desglosamiento de *tweets* y finalmente la verificación manual de un porcentaje de los datos. Como resultado, se

obtiene un *dataset* político y un *dataset* pandemia.

2. **Exploración de datos:** Esta etapa se enfoca en indagación del *dataset* mediante el análisis de la distribución de datos.
3. **Pre-procesamiento:** Esta etapa incluye el (1) tratamiento de elementos especiales como: *hashtags*, *usernames* y URLs. Así mismo se procede al (2) tratamiento de repetidos y signos de puntuación y finalmente se emplea (3) *stemming*, la técnica encargada de reducir las palabras a su raíz.
4. **Feature Engineering:** Esta fase incluye la creación de nuevas características para un mejor de análisis de los datos. A partir de ello, se incluye la (1) creación del diccionario de emojis, la (2) aplicación de N-grams y finalmente, el (3) tratamiento de las oraciones negativas.
5. **Implementación de modelos:** Primero se efectúa la (1) partición de datos en train, validación y test. Luego se realiza la preparación de datos y finalmente los modelos son construidos por medio de la librería ***NLTK.classify*** que contiene los cuatros modelos de clasificación: NB, SVM, RF y MLP.
6. **Evaluación de resultados:** Durante esta etapa se evalúan seis aspectos que podrían optimizar los resultados obtenidos hasta el momento. Son comparados los resultados de cada modelo, mediante las siguientes métricas: exactitud, precisión, exhaustividad y F1-score.

1.7. Resultados esperados

- Dos *datasets* de *tweets* etiquetados entre sentimiento positivo, negativo y neutro con escenario político y pandemia.
- Metodología para la construcción de un *dataset* que pueda ser utilizado en el análisis de sentimientos.

1.8. Cronograma de actividades

El cronograma de actividades se muestra en la figura 2.

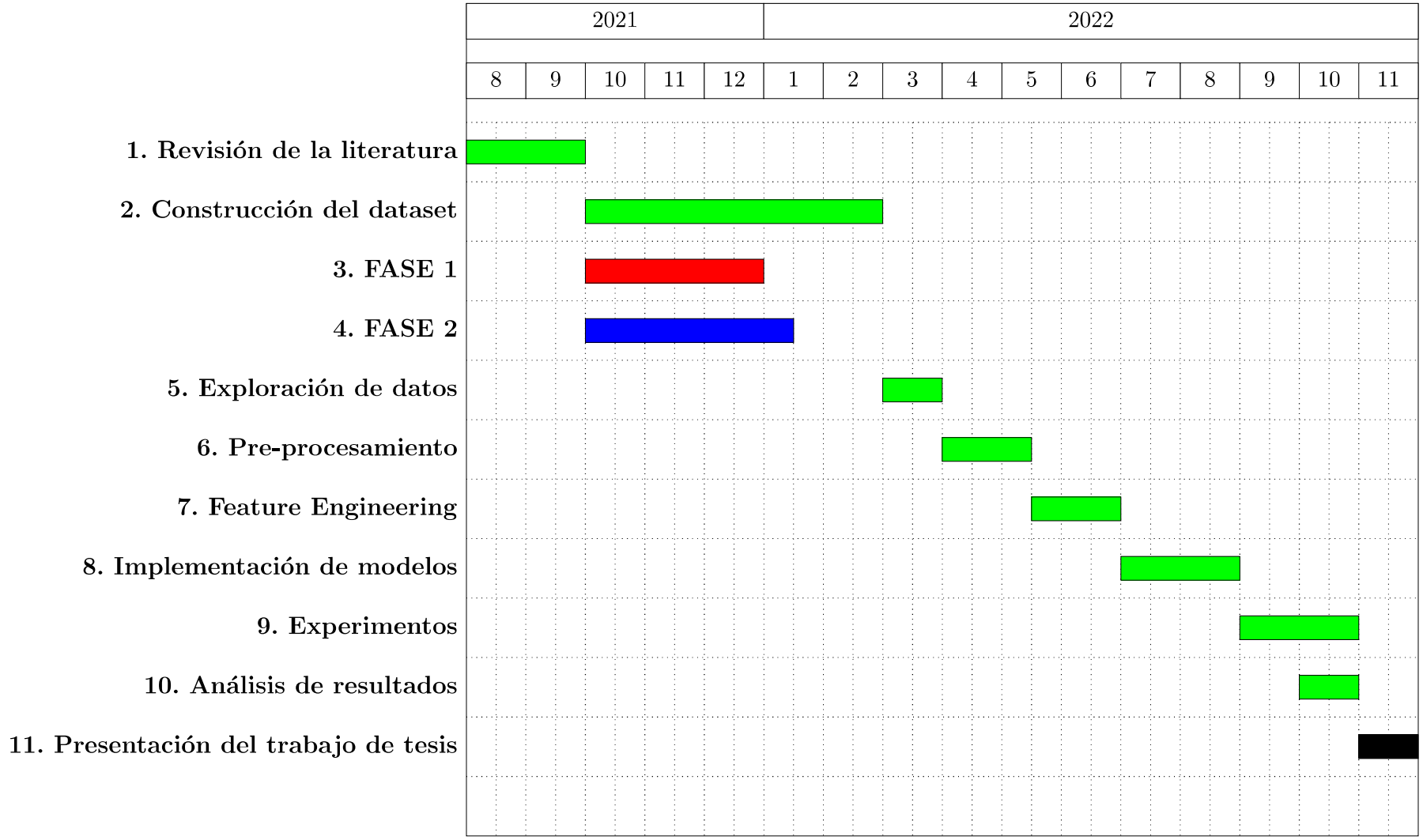

Figura 2: *Cronograma de trabajo*
legend***Fuente:*** *Elaboración propia*

Capítulo 2

Marco Teórico

2.1. Tweet

Un *tweet* es un texto iniciado por un usuario de Twitter, con un límite de 280 caracteres, pero se pueden concatenar en una serie de *tweets* secuenciales como un hilo, con un identificador de orden (1/8, 2/8, etc.). Un *tweet* puede contener imágenes, videos, GIF, *usernames* y *hashtags*, lo que permite la indexación del contenido según palabras clave o frases, que funcionan para vincular el *tweet* a contenido similar (O'Glasser et al., 2020).

2.2. Análisis de sentimientos

2.2.1. Definición

El análisis de sentimientos o minería de opiniones es el área de estudio donde se determina el comportamiento de la persona con respecto a cualquier tema. Este puede estar enfocado en diferentes tareas como la clasificación de *tweets* o resumen de texto (Mittal and Patidar, 2019).

Toda opinión está compuesta de por lo menos dos elementos fundamentales, un objeto y un sentimiento sobre este objeto. Dicho objeto puede ser una entidad, un aspecto de una entidad, o un tema, que representa un producto, una persona, una organización, una marca, un evento, etc. Un sentimiento representa una actitud, opinión o emoción en una escala que representa una validación positiva, neutra o negativa del significado de este sentimiento (Cambria et al., 2017). Formalmente, una opinión corresponde a una quíntupla (Becker and Tumitan, 2013) (e_i, a_{ij}, s_{ijkl}, h_k, t_l) donde:

- e_i: es el nombre de la entidad;
- a_{ij}: es un aspecto de la entidad e_i (opcional);
- s_{ijkl}: es el sentimiento en relación al aspecto a_{ij} de la entidad e_i;
- h_k: es la persona que expresó el sentimiento, también llamada fuente de opinión;
- t_l: es el instante en que la opinión fue expresada por h_k.

Oración	$\mathbf{e_i}$	$\mathbf{a_{ij}}$	$\mathbf{s_{ijkl}}$	$\mathbf{h_k}$	$\mathbf{t_l}$
Un gran saludo para mis hermanas y hermanos agricultores que alimentan a nuestra patria.	saludo	gran	Positivo	@PedroCastillo	30/05/2021
Desde Fuerza Popular reafirmamos que el país está por encima de cualquier diferencia personal o partidaria.	pais		Positivo	@KeikoFujimori	06/08/2020
Políticos como Vizcarra y su gente de confianza confirma que ellos solo están al servicios de los intereses de los empresarios y no de la gente.	Vizcarra y su gente confianza		Negativo	@PedroCastillo	16/02/2021
Por eso es su negociado con las vacunas chinas, que son las menos efectivas y más caras.	vacunas chinas	menos efectivas y más caras	Negativo	@PedroCastillo	16/02/2021
El Presidente del JNE solicitó a los medios de comunicación que mantengan una cobertura imparcial, neutral y equitativa.	Presidente del JNE	solicitó	Neutro	@JNE$_{Peru}$	05/30/2021

Tabla 1: *Ejemplos de tweet respecto al análisis con quíntupla*
Fuente: *Elaboración propia*

La Tabla 1 muestra ejemplos de como se debe emplear el análisis en cada oración respecto a la quíntupla.

2.2.2. Etiquetado de datos

La etiqueta respecto al sentimiento es la actitud, la evaluación o la emoción subyacentes asociados con una opinión, esta etiqueta puede ser vista de dos formas(Chaudhuri, 2006):

1. **Sentimiento racional**, estos provienen del razonamiento racional, las creencias tangibles y las actitudes utilitarias, además no expresan emociones. La oración *'La Constitución define nuestros derechos como ciudadanos'* o *'La pandemia se inicio en China'* dan idea de un sentimiento racional, los cuales son considerados dentro de la clase neutra.
2. **Sentimiento emocional**, estos provienen de respuestas no tangibles y emocionales a entidades que profundizan en el estado mental psicológico de las personas. Las opiniones en las siguientes oraciones implican un sentimiento emocional: *'Me niego a creer que el presidente traiciono al país'* o *'Estoy contenta porque vacunaron a mi abuela'*. El sentimiento emocional es más fuerte que el sentimiento racional y suele ser más importante en la práctica. A continuación se describen dos aspectos que deben considerados:
   - *Orientación de sentimiento:* Puede ser positivo, negativo o neutral. Neutral generalmente significa la ausencia de sentimiento o ningún sentimiento u opinión, como por ejemplo *stopwords* (preposiciones, pronombres, conjunciones, etc), expresiones cortas como *'hola', 'chau', 'Vivaa', etc*, u oraciones sin sentimiento, como *'Ire a recibir mi vacuna'* o 'Regrese de la protesta'. La orientación de sentimiento también se denomina polaridad, orientación semántica o valencia en la literatura de investigación.
   - *Intensidad de sentimiento:* El sentimiento puede tener diferentes niveles de fuerza o intensidad. Los seres humanos utilizan dos formas de expresar la intensidad de sus sentimientos en el texto. La primera es elegir términos de sentimiento (palabras o frases) es decir usando adjetivos, por ejemplo *'Nuestros valerosos médicos hicieron un gran trabajo'*. La segunda forma es usar intensificadores y disminuidores, que son términos que cambian el grado del sentimiento expresado, como por ejemplo *'La candidata no fue muy objetiva con sus propuestas'*.

El sentimiento es difícil de diferenciar basándose únicamente en el texto debido a que su naturaleza es altamente subjetiva y muchas expresiones habladas o escritas de las personas pueden ser interpretadas de muchas formas (Cambria et al., 2017), por lo cual el sentimiento no puede detectarse con exactitud. Este tipo de oraciones son denominadas ambiguas y muchas veces pueden crear problemas en la comprensión humana, y para un ordenador es aun más complicado. Se mencionan algunos ejemplos a continuación:

- *'Vi a las personas de edad haciendo cola para la vacunación'*, podría ser expresada como algo positivo si la cola era para mantener el orden, o negativo, si las personas de edad no tenían acceso a la atención preferencial.

- *'Vote en blanco en las ultimas elecciones'*, podría ser de sentimiento positivo si el votante considero que ningún candidato merecía su voto o negativo, si el votante ejerció su voto de manera irresponsable.

2.2.3. Niveles de Análisis Textual

El análisis de sentimientos implica muchas fases y subproblemas, entre las cuales se tiene la definición del nivel de análisis del texto. La detección de sentimientos en un texto puede ocurrir en diferentes niveles, y la decisión del nivel está sujeta al contexto y la aplicación. El análisis puede ser a nivel de:

- Documento: En este nivel, la tarea es determinar el sentimiento general del documento, tratado como una sola entidad (Mittal and Patidar, 2019). Este nivel es adecuado cuando un documento trata de una única entidad, por ejemplo, un documento que brinda una opinión sobre un producto determinado.
- Oración: Determina el sentimiento de todas y cada una de las oraciones dentro de un documento (Mittal and Patidar, 2019). Este nivel se usa a menudo cuando el mismo documento contiene opiniones sobre varias entidades. La oración *'El trabajo es arduo y constante'* expresa un sentimiento un tanto negativo, mientras que *'pero la suma de nuevos aliados y el compromiso creciente de más ciudadanos renuevan nuestras energías'* expresa un sentimiento positivo, al igual que las siguientes oraciones del documento como se muestra en la Tabla 2.

Tweet (Documento)	El trabajo es arduo y constante pero la suma de nuevos aliados y el compromiso creciente de más ciudadanos renuevan nuestras energías. Gracias #HéroesDeLaSalud. Sigamos trabajando. https://t.co/OODYSvUNMa
Oración negativa	El trabajo es arduo y constante
Oración positiva	pero la suma de nuevos aliados y el compromiso creciente de más ciudadanos renuevan nuestras energias. Gracias #HeroesDeLaSalud
Oración neutro	Sigamos trabajando

Tabla 2: *Ejemplo de análisis textual en oraciones*
Fuente: *Elaboración propia*

- Entidad y Aspecto: también llamado análisis a nivel de características, se centra en la opinión expresada, independientemente de los constructores utilizados para expresarla. En este caso, el objetivo de la opinión puede ser una entidad, o alguno de sus aspectos. A este nivel se puede encontrar que le gusta o le disgusta a la gente, lo que en otros niveles como el de documento u oración no es posible (Mittal and Patidar, 2019). Este es el nivel de análisis más complejo y detallado.

2.3. Metodología para la clasificación de *tweets*

El análisis de sentimiento enfocado en *tweets* requiere una serie de pasos para lograr resultados satisfactorios (Zahra et al., 2018). Esta metodología se divide en los pasos que se muestran en la Figura 3.

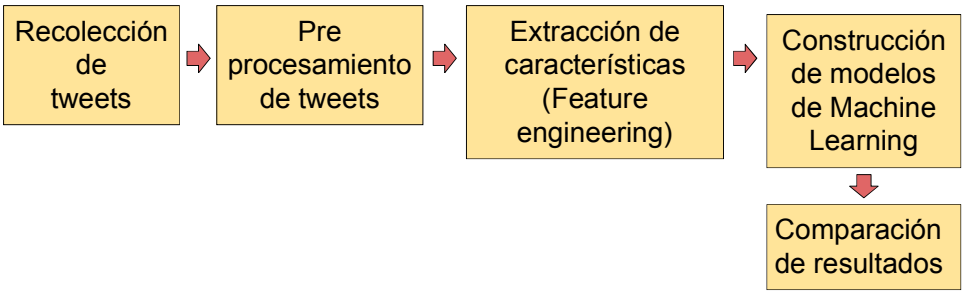

Figura 3: *Metodología de clasificación de tweets utilizando algoritmos de aprendizaje automático*
Fuente: *(Zahra et al., 2018)*

- Recolección de *tweets*: Estos datos pueden ser obtenidos mediante las APIs de Twitter u otras herramientas de *web scrapping*. En esta etapa también se debe considerar el etiquetado de datos si se trabaja con modelos supervisados, dicho etiquetado puede ser realizado manualmente o de forma automática.
- pre-procesamiento de *tweets*: Esta etapa es una de las tareas más importantes y exhaustivas, incluye la limpieza de datos para obtener un *dataset* 'limpio' manteniendo la integridad de los datos originales. Se pueden seguir las estrategias de reducción de dimensiones, donde se detectan y eliminan atributos poco importantes o redundantes, etc. Debido a que los *tweets* reales tienden a tener mucho ruido, ser incompletos e inconsistentes, que resultan ser atributos innecesarios para el análisis. En esta tarea se identifican, corrigen y remueven datos erróneos e inconsistentes, con el objetivo de mejorar la calidad de los *tweets* y los resultados que serán obtenidos a partir de estos. (Hernández et al., 2008).

  Las características irrelevantes para el análisis más comunes de los *tweets* es la presencia de direcciones de correo electrónico, números de celular, símbolos y montos de dinero, porcentajes, horas y fechas. También se observa la presencia de *hashtags*, menciones a otros usuarios, emojis y URLs. Estos elementos son identificados y remplazados por etiquetas según al grupo que pertenezcan. (Pota et al., 2021)

  Algunos ejemplos de *tweets* en crudo son:

  - *@MelissaPeschier @Minsa_Peru A esos 33 mil detenidos, el gobierno que no se haga problema. Incineren a todos ellos, en remplazo de los que tienen el covid-19*
  - *Se publicó la Resolución Ministerial que aprueba el Documento Técnico de Prevención y Atención de personas afectadas por #COVID19 en el #Perú. Se estipula lo siguiente: - El COVID-19 es una enfermedad de notificación obligatoria para todos los establecimientos del país. https://t.co/Qjrzl0MSlV*

- Extracción de características: Como consecuencia de las técnicas de preprocesamiento, las características deben ser extraídas antes de que pueda comenzar la clasificación. En esta etapa, el texto del *tweet* normalizado se transforma en un vector de funciones que alimenta el clasificador de aprendizaje automático (Cerón-Guzmán and León-Guzmán, 2016).
- Construcción de modelos de Machine Learning: En esta etapa, el sistema de análisis de sentimientos clasifica un *tweet* determinado como positivo, negativo o neutral. Es importante señalar que la tarea de clasificación es multiclase y además se trabaja con modelos supervisados, por lo que asigna una sola etiqueta al *tweet*. Cada modelo es entrenado, validado y testeado con el conjunto de datos establecido.
- Comparación de resultados: Se interpretan los resultados obtenidos y de ser necesario se ejecutan los pasos anteriores. Esta etapa también puede incluir la visualización de resultados obtenidos.

2.4. Herramientas de extracción de *tweets*

- **API de Twitter:** El API de Twitter ofrece un acceso exclusivo y avanzado a la información de Twitter, como *tweets*, mensajes directos, seguidores, usuarios, entre otros; esto dependerá al nivel de acceso dado por Twitter al programador (Trupthi et al., 2017). Existen tres niveles de acceso libre:
  1. Essential, su acceso se limita a la versión standar de API v1.1 que nos permite recuperar *tweets* con 7 dias de antigüedad.
  2. Elevated, ofrece algunas características adicionales a la anterior como acceso a la versión premium del API v1.1, y el API enterprise que recuperan *tweets* con 30 días de antigüedad
  3. Academic Research, en la cual se puede obtener más cantidad de datos por consulta y endpoints avanzados; para acceder a estos niveles hay que aplicar a través de Twitter developer y esperar la calificación. Cuenta con una documentación[5] en la que se encuentra tutórales, ejemplo de casos de uso, guías de integración para facilitar y una comunidad de desarrolladores.
- **Tweepy** [6]**:** Es una librería para Python que nos ofrece una fácil forma de conexión a las APIs de Twitter, cuenta con métodos y clases para el proceso de autentificación y peticiones al API, también hay clases modelos para manejar la información recuperada como *tweets* o usuarios. Es compatible con las dos versiones del API disponibles (API v1.1 y API v2) y para cada una de ellas existen clases que nos permite filtrar y mapear *tweets* del search API o streaming API, todo esto esta sujeto al nivel de acceso dado por el API de Twitter.
- **RTweet:** Disponible para Python como una librería, es la implementación de llamadas diseñadas para extraer y ordenar datos de Twitter a través de sus APIs. Las funciones realizan peticiones GET y POST, asi mismo convierten

[5]https://developer.twitter.com/en/docs/twitter-api
[6]https://docs.tweepy.org/en/stable/streaming.html

los objetos de respuesta en estructuras de datos más fáciles de utilizar. Las funciones más relevantes son las diseñadas para devolver datos de *tweets* de búsquedas, streaming y líneas de tiempo e identificaciones de usuario de listas de amigos y seguidores. Es necesario realizar una autenticación con una cuenta de Twitter antes de hacer uso de la herramienta, ya que RTweet[7] trabaja estrictamente acorde a los términos de Twitter's developer.

- **Netlytic [8]:** Es una herramienta en la nube que analiza textos a partir de publicaciones de redes sociales de acceso público y permite resumir textos de forma automatizada. Se conecta a las APIs de redes sociales como Twitter, YouTube y RSS feeds, para recopilar publicaciones compartidas de forma publica. También puede analizar un conjunto de datos existente a través de la carga de archivos CSV o Google Sheet.
- **Crowdbreaks:** Desarrollado por el Laboratorio de Epidemiología Digital de la EPFL en Suiza. Crowdbreaks[9] esta diseñado para rastrear tendencias en tiempo real en diferentes países, se enfoca en temas relacionados a la salud y enfermedades. Recopila *tweets* aplicando filtros por palabras claves relacionadas al tema de salud y mediante técnicas de PLN. El sistema intenta filtrar los *tweets* más relevantes, lo que implica que entre más datos estén etiquetados, se mejora el contenido y la detección de *tweets* relevantes.
- **Twint:** Es una herramienta avanzada de web scrapping para Twitter escrita en Python, permite obtener *tweets* de los perfiles de Twitter sin el uso de los APIs de Twitter y de forma anónima. Esta herramienta hace consultas a los operadores de búsqueda de la misma plataforma para obtener los datos deseados. Las consultas se pueden realizar por usuarios específicos, *tweets* relacionados con ciertos temas, *hashtags* y tendencias; así mismo se puede rastrear los seguidores de un usuario, los *tweets* que le han gustado a un usuario y a quienes sigue, todo esto sin ninguna autenticación (Twintproject, 2018). El código fuente se encuentra disponible en GitHub[10].

2.5. Herramientas de etiquetado de *tweets*

- **TextBlob:** Es una librería para Python de fácil uso, que ofrece un API para acceder a sus métodos y realizar tareas de PLN (Gujjar and HR, 2021). Se utiliza para asignar puntajes de polaridad y subjetividad a los datos. La propiedad *sentiment* analiza un texto y devuelve una tupla basado en la polaridad y subjetividad, donde la polaridad es un dato de tipo `float` con valor en un rango de [-1.0, 1.0]; y la subjetividad también es de tipo `float` en un rango de [0.0, 1.0] donde 0 significa que el texto es muy objetivo y 1 que es muy subjetivo (Ahuja and Dubey, 2017).
- **Bidirectional Encoder Representations from Transformers (BERT):** Es un modelo que utiliza una estructura de transformador multicabezal, desarrollado por el equipo de IA de lenguaje de Google, el cual esta pre-

[7]https://cran.r-project.org/web/packages/rtweet/readme/README.html
[8]https://netlytic.org/home/
[9]https://www.crowdbreaks.org/en/about
[10]https://github.com/twintproject/twint

entrenado con un gran corpus de diferentes fuentes. BERT ha logrado mejores resultados en tareas de PLN mejorando el estado de arte en este área (Liu et al., 2019). Para usar el modelo BERT es necesario preparar los datos, esto consiste en una normalización de los *tweets* agregando tokens especiales al comienzo y final de cada uno de estos, el token[SEP] se agrega al final y [CLS] al inicio, ya que el modelo BERT cuenta con algunas restricciones: cada *tweet* debe tener la misma longitud, por lo cual los *tweets* deben ser rellenados o truncados para que todos cuenten con una longitud uniforme, y la longitud máxima de los *tweets* debe ser de 512 tokens; el token especial para rellenar las oraciones es [PAD] finalmente para utilizar BERT para la clasificación de texto se ajusta el modelo creando una capa única para el análisis de sentimientos, en esta capa de deserción, se aplica la función Softmax y a las salidas del modelo BERT para ayudar a prevenir el overfitting (López Condori and Gonzales Saji, 2021).

- **PySentimiento** PySentimiento[11] es un conjunto de herramientas de Python multilingüe para el análisis de sentimientos y otras tareas de PLN social. Esta librería de código abierto trae modelos de última generación para español e inglés, lo que permite a los investigadores acceder fácilmente a estas técnicas (Pérez et al., 2021). Este modelos fue entreando con dos conjuntos de datos: *TASS 2020 Task 1* y *SemEval 2017 Task 4 Subtask 1.*

2.6. Procesamiento de Lenguaje Natural

El análisis de sentimientos en texto utiliza el poder computacional para identificar y comprender el sentimiento expresado en un texto. El Procesamiento de Lenguaje Natural se encarga de la tarea de encontrar el significado semántico del texto, por lo tanto, analiza la información del texto. En la actualidad existen varias técnicas de PLN que son utilizadas para ayudar al problema de la clasificación de sentimientos (Kanakaraj and Guddeti, 2015).

2.6.1. Tokenización

En los estudios de PLN, es convencional concentrarse en el análisis puro mientras se dan por sentadas las unidades básicas, es decir, las palabras. Estas unidades básicas claramente segregadas, es imposible realizar ningún análisis o generación (Webster and Kit, 1992).

El primer paso es la segmentación de un documento dado en oraciones y palabras. Se suele usar el espacio en blanco como delimitador para la segmentación, pero esto puede ser un poco complicado para ciertos idiomas, si bien parece una tarea fácil, la tokenización implica lidiar con el reconocimiento de abreviaciones, etiquetas especificas, entre otros y decidir si separar o no un token en dos palabras, en donde también influye en esta decisión el idioma del documento (Rajput, 2020).

En la práctica la tokenización debe ejecutarse antes que cualquier otro procesamiento del lenguaje. Por lo tanto, el método estándar para la tokenización es utilizar algoritmos deterministas basados en expresiones regulares compiladas en autómatas de estado finito muy eficientes; asi pueden

[11] https://github.com/pysentimiento/pysentimiento

lidiar con las ambiguedades que surgen, como el hecho de que los guiones o abreviaciones deban ser tokenizado o no (Keselj, 2009).

Un concepto que es bastante utilizando entre los expertos del PLN es 'Expresión regular', la cual indica el formato de la cadena de caracteres ha analizar debe tener, por ejemplo una cadena de caracteres que solo puede contener letras mayúsculas se especificará como [A-Z], mientras que una cadena de caracteres que contiene números sera especificada como [0-9] (Rajput, 2020).

2.6.2. Estandarización: radicales y lemas

Es la tarea de poner en un formato standar las palabras/tokens, se elige una sola forma para palbras con multiples formas como *jaja* y *jajajaja*. (Keselj, 2009) Algunos aspectos de la estandarización ya se han mencionado como parte de la tokenización, por ejemplo, la transformación de mayúsculas en minúsculas, con excepción de los acrónimos o nombres propios. Otras acciones de estandarización tienen como objetivo representar un grupo de palabras a través de un solo término. Para ello se considera Stemming y Lemmatization, siendo Stemming un método que nos ayuda a reducir la longitud del texto y consume menos recursos y tiempo que la Lemmatizacion.

Stemming es una de las técnicas utilizadas en los sistemas de recuperación de información para asegurarse de que las variantes de las palabras no se dejen de lado cuando se recupera el texto. El proceso se utiliza para eliminar sufijos derivativos y flexiones; es decir, sufijos que cambian la forma de las palabras y sus funciones gramaticales, para que las variantes de las palabras se puedan fusionar en las mismas raíces, reduciendo el tamaño del diccionario del documento, esto de debe a que la raíz se puede utilizar para representar todas las variantes de termino usado (Balakrishnan and Lloyd-Yemoh, 2014).

2.6.3. TF-IDF

De las siglas en ingles Term frequency-Inverse document frequency se utilizo como algoritmo de referencia para las tareas de clasificación. El algoritmo comprueba con que frecuencia de las palabras de la consulta de búsqueda en un documento; a mayor número de veces que aparece una palabra de consulta en un documento, este se vuelve más relevante con respecto a la consulta de búsqueda.

Donde el termino tf_{td} representa la frecuencia con la que aparece el termino de consulta t en el documento d. df describe la frecuencia del documento en relación con el número de documentos que contiene el termino de consulta. La frecuencia inversa del documento (idf) representa la relevancia que tiene el termino de consulta en relación con todos los documentos pertenecientes a la colección (Balakrishnan and Lloyd-Yemoh, 2014).

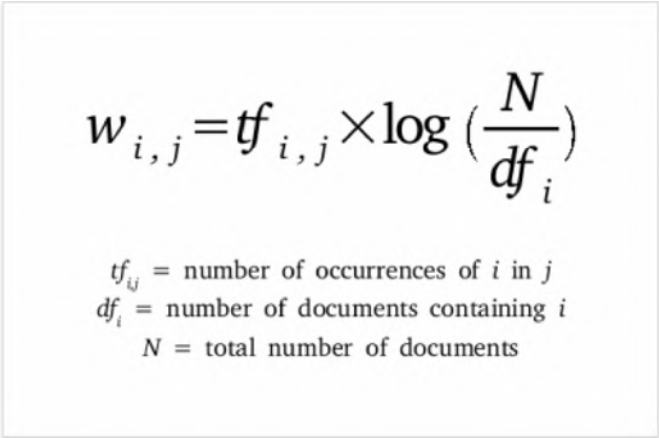

Figura 4: *Formula para calcular TF-IDF*
Fuente: *(Lok, 2017)*

2.6.4. Similitud del coseno

La similitud del coseno es una métrica usada en la recuperación de información y estudios relacionados. Modela un texto como un vector de términos, donde las dimensiones del vector se refieren a los términos disponibles en el documento. Mientras que la similitud entre dos textos deriva del valor del coseno entre los vectores de términos de dichos textos. Sin embargo, la similitud del coseno todavía no puede manejar perfectamente el significado semántico del texto (Rahutomo et al., 2012).

Su cálculo es muy eficiente, especialmente para vectores dispersos, ya que solo se deben considerar las dimensiones distintas de cero. Como componente fundamental, la similitud de coseno se ha aplicado en la resolución de diferentes problemas de minería de texto, como clasificación de texto, resumen de texto, recuperación de información, respuesta a preguntas, etc. Aunque es popular, la similitud del coseno tiene algunos problemas, comenzando con algunas muestras sintéticas que poseen demasiadas características de valores más altos y muchas de ellas no son relevantes. Por lo tanto, se propone una métrica de similitud de coseno ponderada por distancia. Extensos experimentos sobre clasificación de texto muestran la efectividad de la métrica propuesta (Li and Han, 2013).

El proceso de comparación es posible cuando el vector de la consulta y el vector del documento tienen términos coincidentes. Una de las claves del modelo deL espacio vectorial es precisamente la posibilidad de determinar el ángulo que forman los vectores del documento y de la consulta que se está comparando como se muestra en la Figura 5.

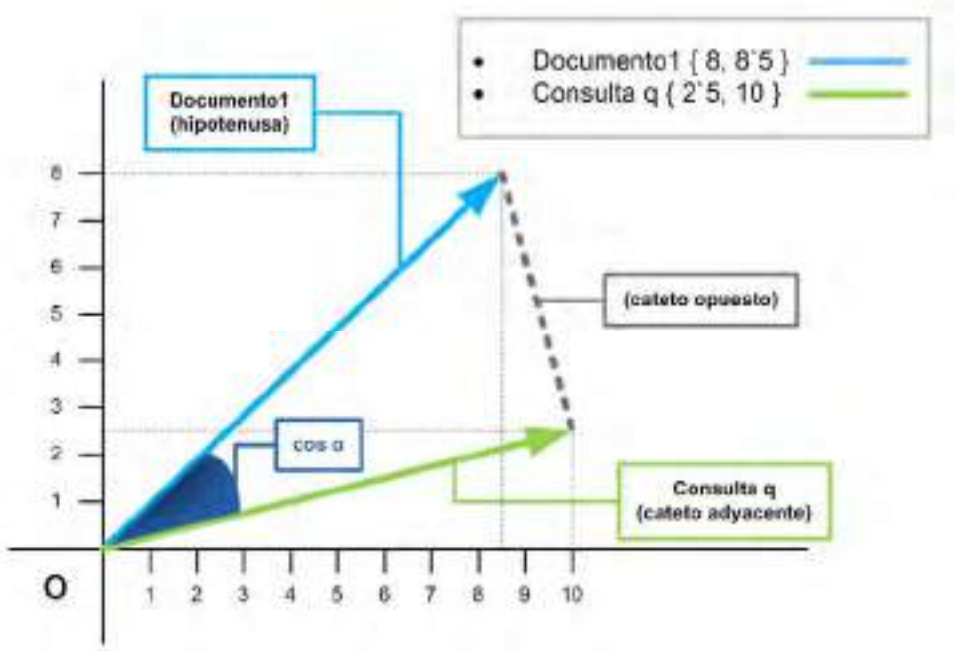

Figura 5: *Ejemplo de visualización de vectores en el plano 2D*
Fuente: *(Blázquez Ochando, 2012)*

Es posible medir cuál es la desviación de un documento con respecto a una consulta, por el número de grados del ángulo que forman. Esto es posible porque crean una estructura triangular a la que se aplica el cálculo del ángulo que forma la hipotenusa (en este caso el vector del documento1) y el adyacente (el vector q de la consulta dada por el usuario) que resulta ser el coseno del triángulo. En el caso de la Figura 5, se comprueba visualmente cierta distancia del vector de la consulta con respecto al documento1; cuando ambos vectores se muestran tan próximos como para superponerse, implicará que el ángulo que forman será menor y que su nivel de coincidencia será superior. De hecho, un coseno de 0^{o} implicaría una similaridad máxima. La Figura 6 muestra la formula a emplear para calcular la similitud.

$$\mathrm{SimCos}(d_{(d)},q) = \frac{\sum_{n=1} \left(P_{(n,d)} \times P_{(n,q)}\right)}{\sqrt{\sum_{n=1} \left(P_{(n,d)}\right)^2 \times \sum_{n=1} \left(P_{(n,q)}\right)^2}}$$

Figura 6: *Formula para hallar la similitud de dos vectores mediante el coseno*
Fuente: *(Blázquez Ochando, 2012)*

2.7. Stratified sampling

En este proceso se divide el *dataset* en distintos grupos que contengan elementos similares entre si, respecto a características definidas previamente, estas características tienen que ser de relevancia para el análisis a realizar(Parsons, 2014). Posteriormente se seleccionan un número de elementos proporcional al tamaño de cada grupo para formar parte del *dataset* de entrenamiento. De esta manera se verifica que el conjunto de entrenamiento represente de la mejor manera al total de los datos, eliminando el sesgo de muestreo (Jing et al., 2015). En algunos casos este proceso puede resultar dificultoso, al momento de clasificar cada elemento del *dataset* en grupos, ya que pueden existir elementos que pueden pertenecer a varios grupos a la vez. En la Figura 7 se muestra un ejemplo de la proporcionalidad de clases que se debe cumplir. Por ejemplo si se tiene 60 *tweets* positivos y 40 negativos, trabajando con una proporción de 70 % para el train y 30 % para el test, correspondería 42 *tweets* positivos y 18 negativos en el train, y 21 positivos y 19 negativos en el test.

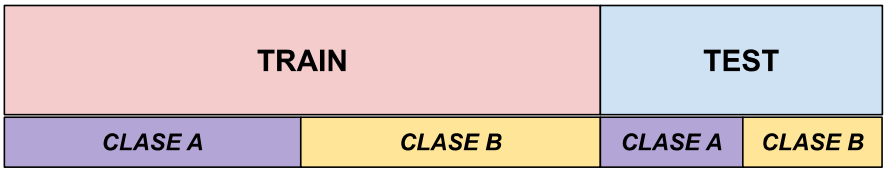

Figura 7: *Estratificación de dataset*
Fuente: Elaboración propia

2.8. Cross validation

Es un método de remuestreo de datos, utilizados para estimar el verdadero error de predicción de los modelos y ajustar los parámetros del modelo para evitar el *overfitting* (Berrar, 2019). Ayuda a garantizar que los resultados obtenidos sean independientes de la partición del *dataset* de entrenamiento y prueba. Ya que en los modelos de *Machine Learning* la clasificación puede depender de la división de datos de entrenamiento y prueba. Existen varios tipos de Cross validation que son: *Single hold-out random subsampling*, *k-fold random subsampling*, *k-fold cross-validation*, *eave-one-out cross-validation*. Cada una de estas variantes realiza la división del *dataset* de entrenamiento de diferente forma, por un número de iteraciones anteriormente definidas, se realiza el análisis para cada iteración y finalmente el resultado es la media aritmética de los valores obtenidos en cada iteración.(Browne, 2000)

2.9. Modelos de clasificación

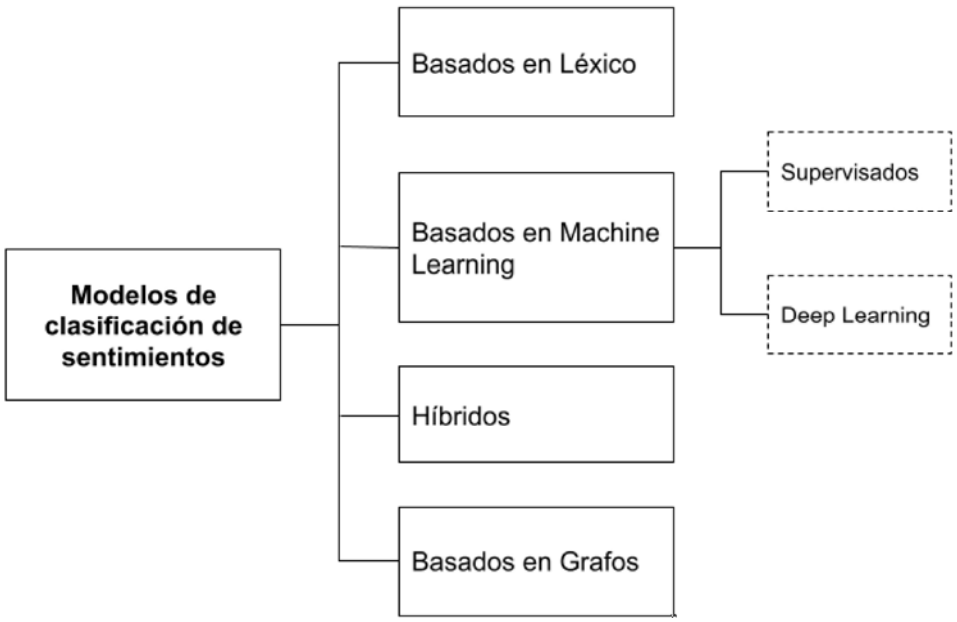

Figura 8: *Clasificación de modelos de análisis de sentimientos según Giachanou*
Fuente: *Elaboración propia*

2.9.1. Basados en léxico

Este tipo de modelos emplean una lista adicional, en la cual se encuentran los principales términos que incluyen un *score* que refleja el sentimiento de cada uno ellos. Basado en ello se obtiene el *score* final de cada expresión analizada (oración o documento). La mayor ventaja que posee este tipo de enfoque es que no necesita entrenamiento (Giachanou and Crestani, 2016), sin embargo no es escalable debido a que se limita a solo llevar en consideración los términos que existen en la lista. Esta metodología ha sido empleada muy poco a diferencia de otros métodos para el análisis de sentimiento en Twitter debido a que el vocabulario que se emplea en las redes sociales es coloquial y además existen elementos como *hashtags* o emoticones que necesitan ser considerados. Algunos trabajos que emplean este enfoque son:

- *SentiStrength* (Thelwall et al., 2012): Este es un algoritmo que resulto ser muy efectivo para la identificación de sentimientos en redes sociales, ya que lidia con el lenguaje informal y también con elementos que no son texto. Sus autores también extendieron este algoritmo para evaluar diferentes idiomas.
- *Emotional Signals for unsupervised Sentiment Analysis (ESSA)* (Hu et al., 2013): Se toma en cuenta el score basado en el sentimiento y en la correlación de los términos, posteriormente se utiliza una matriz de factorizacíón para hallar el sentimiento final de cada termino.
- *SentiCircles* (Saif et al., 2016): A través de este algoritmo los *scores* de los términos en cuestión se van actualizando cada vez que estos vuelven a aparecer según el contexto de análisis.

2.9.2. Basados en *Machine Learning*

Estos modelos utilizan *features* que son extraídos de los *tweets*, a partir de ellos un modelo de clasificación deben ser capaz de predecir el sentimiento (*target*). Estos modelos pueden ser clasificados en:

- **Supervisados**

  1. Naive Bayes: Este es un algoritmo probabilístico simple y muy eficiente en la clasificación de textos en general. Se basa en la aplicación del teorema de Bayes con la premisa de total independencia entre variables (Becker and Tumitan, 2013), asignando la clase más probable a un ejemplo dado descrito por su vector de características. El aprendizaje de tales clasificadores se puede simplificar enormemente asumiendo que las características son de clase dada independiente. Aunque esta suposición de independencia en la practica es pobre, a menudo el clasificador presenta una competitividad muy aceptable ante los clasificadores más sofisticados (Rish et al., 2001).

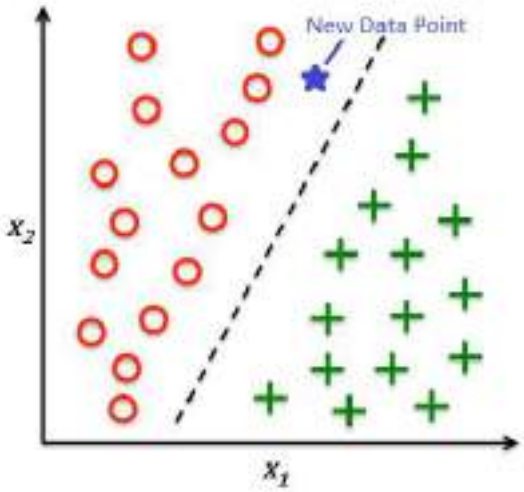

Figura 9: *Representación de clasificación mediante Naive Bayes*
Fuente: *(McCann and Lowe, 2012)*

  2. Super Vector Machine: Este algoritmo es utilizado tanto para clasificación como para regresión. Un modelo SVM es la representación de conjuntos de entrenamiento como puntos en el espacio, mapeados de tal manera que los datos de cada categoría estén separados por el mayor posible espacio entre ellos. Luego se construye un hiperplano que divida el espacio en dos dimensiones. La separación entre las categorías se realiza mediante el hiperplano que maximiza la distancia entre los puntos más cercanos entre cualquier clase (Becker and Tumitan, 2013).
  3. Random Forest: Este algoritmo ensamblado ejecuta un determinado número de arboles de decisiones en lugar de uno solo. Para clasificar un nuevo objeto basado en atributos, cada árbol de decisión da una clasificación y finalmente la decisión con mayor votos es la predicción del algoritmo.

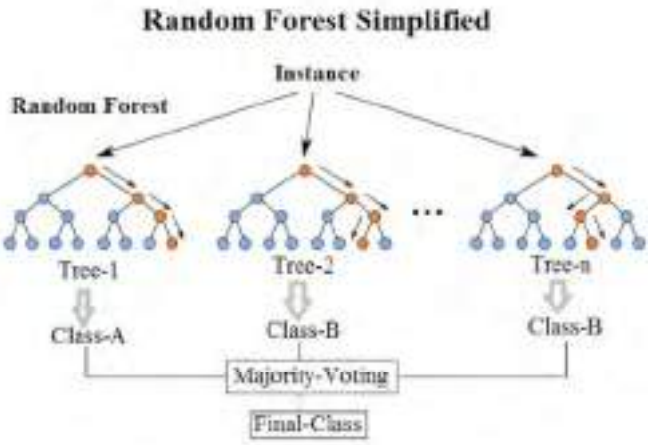

Figura 10: *Representación del algoritmo de Random Forest*
Fuente: *(Koehrsen, 2020)*

- **No supervisados** Los modelos no supervisados tienen una característica importante, ya que no necesitan de datos etiquetados y también nos liberan de realizar *feature engineering*, facilitando el proceso del aprendizaje automático. Estos modelos son aplicados con éxito en los campos de: procesamiento de lenguaje natural, visión por computadora y reconocimiento de voz (Usama et al., 2019).

  1. *K-Means Clustering*: El agrupamiento de *K-means* distribuye `m` observaciones en `n` grupos, donde una observación pertenece al grupo más cercano, este se calcula utilizando la media del conjunto. Es un enfoque simple pero muy usado para la clasificación.
  2. Clasificador *Fuzzy*: Realiza una calorificación suave de datos de forma eficiente mediante una lógica difusa y reglas difusas. A diferencia de otros clasificadores que establecen un análisis nítido, la clasificación difusa nos permite interpretar la muestra de datos de forma más sensible. Los árboles de decisión difusos y las reglas `if-then` son ejemplos de clasificadores difusos eficientes, transparentes y fáciles de interpretar (Krömer et al., 2011).

2.9.3. Basados en *Deep Learning*

Estos modelos utilizan redes neuronales por su alta capacidad de abstracción como por ejemplo el contexto de una oración. A diferencia de los algoritmos supervisados estos son más rápidos y más eficientes (Giachanou and Crestani, 2016). Para este tipo modelos se trabaja con una representación de los términos denominada *words embedings* que servirán como entrada para una red neuronal.

- Multilayer perceptron: Entre sus principales aplicaciones están: el reconocimiento de patrones, de voz y problemas de clasificación. La definición de una arquitectura adecuada es primordial para la efectividad del modelo; ya que una falta de conexiones entre las capas puede causar que el modelo no pueda resolver el problema y un exceso de conexiones

provocara *overfitting* en los datos de entrenamiento, especialmente cuando se usa un gran número de capas en el modelo. En este modelo las neuronas están organizadas en capas y se conectan con otras neuronas de una capa superior (Ramchoun et al., 2016)

- Redes neuronales recurrentes: Para este tipo de red, las neuronas tienen la posibilidad de realizar conexiones de realimentación, ya sea entre neuronas de una misma capa o de diferentes capas. La realimentación permite que las redes neuronales recurrentes, tengan memoria.
- Redes neuronales convolucionales: Este tipo de redes neuronales son capaces de aprender la estructura jerárquica relevante de palabras, oraciones, párrafos, y más. Este enfoque se centra poder aprender a abstraer los detalles más sobresalientes de los *features* de los términos.

2.9.4. Basados en grafo

Últimamente se ha optado por el uso de etiquetas como un método que puede reducir la demanda de los datos anotados. Basado en ellos se propone utilizar el gráfico social de Twitter bajo el supuesto de que las personas se influyen entre sí. La propagación de etiquetas es un método semi supervisado en el que las etiquetas se distribuyen a los nodos utilizando los gráficos de conexión. Usuarios, *tweets*, unigramas, bigramas, *hashtags* y emoticonos se utilizaron como nodos para la construcción del grafo. Posteriormente se realiza una propagación de cada uno de los nodos para obtener la predicción del sentimiento de un determinado texto.

2.10. Métricas de evaluación

La preparación de los datos y el entrenamiento de un modelo son importantes en el proceso de *Machine Learning*, sin embargo es importante medir el rendimiento de este modelo entrenado (Carvalho et al., 2019). Además se debe conocer la capacidad de generalización sobre los datos no vistos para considerarlo como adaptable a cualquier escenario. Una evaluación adecuada del modelo utilizando diferentes métricas permite comprobar la capacidad de generalización para cumplir el objetivo al analizar datos no vistos y evitar que estos solo memoricen y aprendan (Zhou et al., 2021).

- Matriz de confusión Una matriz de confusión es una representación matricial de los resultados de las predicciones de cualquier prueba binaria que se utiliza a menudo para describir el rendimiento del modelo de clasificación sobre un conjunto de datos de prueba cuyos valores reales se conocen (Krstinić et al., 2020). La Figura 11 se muestra un ejemplo.

  Los términos a considerar son:

  1. Verdadero Positivo (VP): cuando la clase real del punto de datos era 1 (Verdadero) y la predicha es también 1 (Verdadero)
  2. Verdadero Negativo (VN): cuando la clase real del punto de datos fue 0 (Falso) y el pronosticado también es 0 (Falso).
  3. Falso Positivo (FP): cuando la clase real del punto de datos era 0 (Falso) y el pronosticado es 1 (Verdadero).

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 11: *Estructura de la matriz de confusión*
Fuente: *Elaboración propia*

4. Falso Negativo (FN): Cuando la clase real del punto de datos era 1 (Verdadero) y el valor predicho es 0 (Falso).

- **Exactitud:** La exactitud es una medida de validez y la métrica más simple y más utilizada para medir el rendimiento de un clasificador. La exactitud permiten evaluar en qué medida los datos realmente miden lo que se supone que deben medir o en qué medida los resultados de una medición son consistentes con el verdadero estado del fenómeno medido. Sin embargo, la exactitud no siempre es una buena métrica, especialmente cuando los datos están desequilibrados (Juba and Le, 2019).

  Se puede definir con los valores obtenidos a partir de la matriz de confusión de la siguiente forma:

  $$\text{Exactitud} = (VP + VN) \,/\, (VP + VN + FP + FN)$$

- **Precisión:** Para cualquier conjunto recuperado dado, la precisión es el número de elementos relevantes recuperados como proporción del número de elementos recuperados. La precisión es, por lo tanto, una medida de pureza en el rendimiento de la exactitud, es decir una medida de la eficacia en la exclusión de elementos no relevantes del conjunto recuperado(Buckland and Gey, 1994).

  Se puede definir con los valores obtenidos a partir de la matriz de confusión de la siguiente forma:

  $$\text{Precisión} = VP \,/\, (VP + FP)$$

  La precisión debe ser óptima para cada clase establecida contenida en el *dataset*, sobre todo si se poseen clases no balanceadas.

- **Exhaustividad:** Para cualquier conjunto recuperado dado, exhaustividad es el número de elementos relevantes recuperados como proporción de todos los elementos relevantes. Exhaustividad es una medida de la eficacia al incluir elementos relevantes en el conjunto recuperado (Buckland and Gey, 1994).

  Se puede definir con los valores obtenidos a partir de la matriz de confusión de la siguiente forma (Tatbul et al., 2018):

  $$\text{Exhaustividad} = VP \,/\, (VP + FN)$$

- **F1-score:** Diversas aplicaciones en las que tanto la exhaustividad como la precisión son importantes. Por lo tanto, es natural pensar en una forma de combinar estos dos en una sola métrica. Una métrica popular que combina precisión y exhaustividad se llama F1-score, que es la media armónica entre ambas, definida como:

F1-score= 2(Precisión*Exhaustividad)/(Precisión+Exhaustividad)

- **F1-Score binario:** Con referencia a la matriz de confusión, dado que precisión y exhaustividad no consideran los elementos VN (Grandini et al., 2020), el F1-Score binario se calcula de la siguiente manera:

  Precisión = 20 / 30 = 0,66

  Exhaustividad = 20 / 25 = 0,80

  F1-score = 2*(0,66*0,80)/(0,66+0,80)=0,72

  F1-score para el caso binario tiene en cuenta tanto la precisión como la exhaustividad. Gracias a estas métricas, se puede confiar en que F1-Score detectará los puntos débiles del algoritmo de predicción, si existe alguno de esos puntos. En el ejemplo, la puntuación de 0,72 demuestra una capacidad de modelo bastante buena para predecir la clase correcta.

En resumen, la exactitud es la métrica más fácil de calcular con menos complejidad, sin embargo la exactitud tiene limitaciones en los procesos de evaluación y discriminación. Una de las principales limitaciones de la exactitud es que produce valores menos distintivos y menos discriminables cuando se poseen clases no balanceadas en el *dataset*. Bajo este escenario, F1-score califica como un buen discriminador para problemas de clasificación binaria o clasificación multiclase (Joshi, 2002). Otro factor importante durante el entrenamiento conduce a poseer datos equilibrados para discriminar y seleccionar la solución óptima durante el entrenamiento de clasificación, la compensación significativa entre clases es esencial para garantizar que cada clase esté representada por su prototipo representativo (Hossin and Sulaiman, 2015).

Capítulo 3

Método Propuesto

Esta propuesta emplea una metodología ***descriptiva***, porque se recopilo información tomando una muestra representativa de cada escenario (político y pandemia), ***cuantitativa***, porque se explora el *dataset* y se identifica características previas al análisis de sentimiento, y ***experimental***, porque se manipulo parámetros para mejorar el rendimiento de cada modelo de *Machine Learning* (Hernández-Sampieri and Mendoza, 2020).

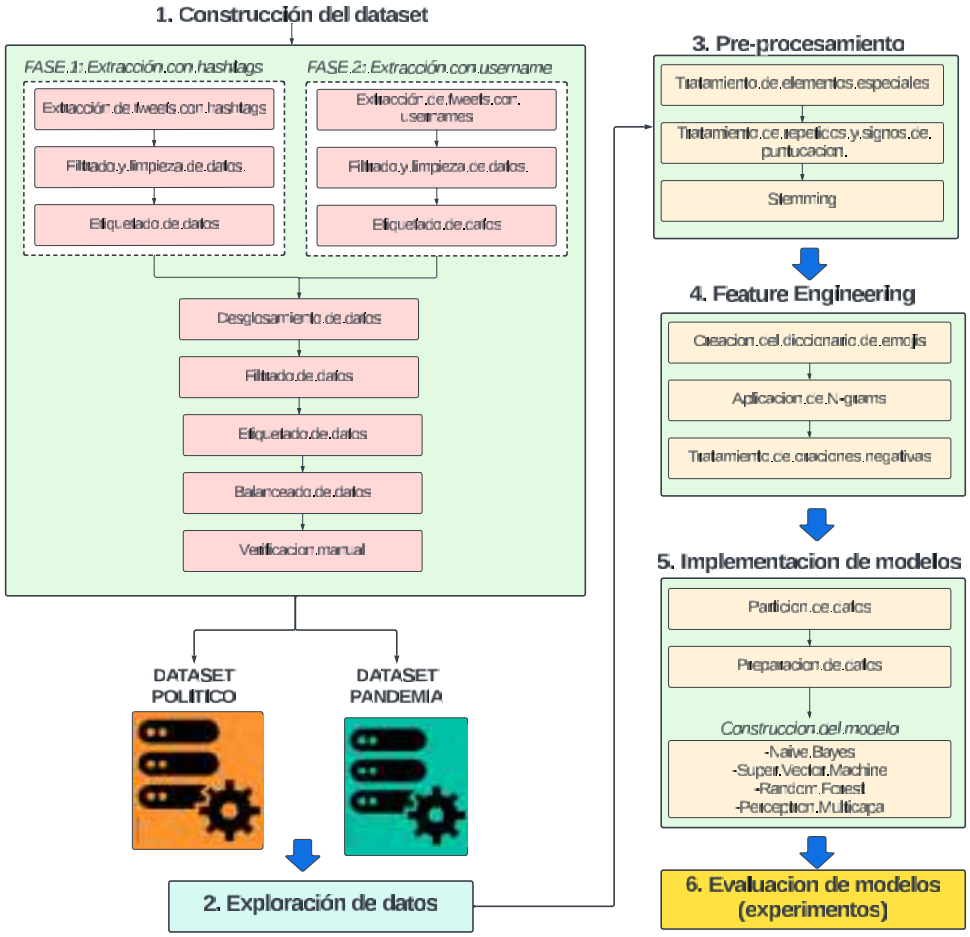

Figura 12: *Pasos seguidos durante la ejecución de la metodología*
Fuente: *Elaboración propia*

La Figura 12 muestra gráficamente los pasos seguidos de acuerdo a la metodología general propuesta en el capitulo 1. Se expone la secuencia de etapas que se siguieron con el fin de construir un *dataset* desde cero. El planteamiento de la metodología se basa en el trabajo de Giachanou, A., & Crestani, F (Giachanou and Crestani, 2016), de este modo, el enfoque principal fue la construcción del *dataset* mediante la extracción de *tweets* de Twitter y el etiquetado de *tweets*. Se enfatiza que se construyo el *dataset* desde cero, es decir, no se importa datos desde un repositorio.

3.1. Construcción del *dataset*

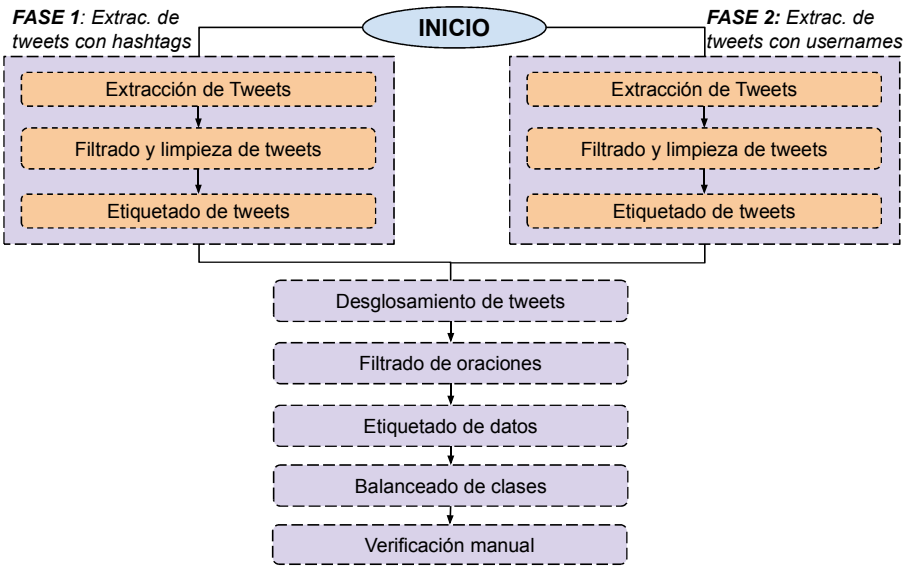

Figura 13: *Flujo de trabajo para la construcción del dataset*
Fuente: *Elaboración propia*

La Figura 13 muestra el flujo de trabajo que se siguió para esta primera etapa. Esta etapa sera dividida en dos fases: (1) Construcción del *dataset* realizando *web scraping* en Twitter mediante *hashtags* y (2) Construcción del *dataset* realizando *web scraping* en Twitter mediante *usernames*. La lista de *hashtags* y *usernames* se obtuvo mediante la búsqueda de los temas en tendencias de Twitter en Perú a través del portal de *getdaytrends* [12], a partir de esta información se extrajo los *hashtag* y *usernames* verificados de interés de forma manual. A continuación se describe el proceso que se sigue para cada fase.

3.1.1. FASE 1: Construcción del *dataset* con *hashtags*

- **Extracción de *tweets* mediante *hashtags*:** La Tabla ??. muestra la distribución de los *hashtags* obtenidos previamente. La configuración que se realizo en la herramienta ***Twint*** para realizar el *web scraping* en Twitter se muestra en la Figura 14.

[12]https://getdaytrends.com/peru/5/

	Político	***Pandemia***
número de hashtags	98 hashtags	115 hashtags
Hashtags considerados	#KeikoFujimori, #PedroCastillo, #FuerzaPopular, #PeruLibre, #EleecionesPeru2021, etc.	#pandemia, #vacunacion, #Covid19, #cuarentena, #UsaMascarilla, etc.

Tabla 3: *Distribución de hashtags para cada escenario*
Fuente: *Elaboración propia*

Los parámetros descritos a continuación, fueron configurados para la extracción de *tweets*:

1. Se establece el parámetro *Search* con los *hashtags* propuestos.
2. Se establece el parámetro *Language* con 'es', lo que significa la recuperación de *tweets* solamente en el idioma español.
3. Se establece el parámetro *Since* y *Until* entre enero del 2020 a agosto del 2022.
4. Se establece el parámetro *Pandas* en *true* para obtener un archivo tipo CSV como salida.

```
def search_tweets(i, since, until):
    tag = Buscar_Palabra(i,'Peru')
    c = twint.Config()
1   c.Search = tag
2   c.Language = 'es'
3   c.Since = since
    c.Until = until
    c.Limit = 10000
4   c.Pandas = True
    c.Hide_Output= True
    twint.run.Search(c)
    df_twint = twint.storage.panda.Tweets_df
    return df_twint
```

Figura 14: *Configuración de parámetros en* ***Twint*** *con hashtags*
Fuente: *Elaboración propia*

Como resultado del *web scraping* se extrajeron **2343 *tweets*** para el escenario político y **2777 *tweets*** para el escenario pandemia.

- **Filtrado y limpieza de datos:** Se efectúa la eliminación de *tweets* duplicados mediante el calculo de la similitud de coseno, así como la selección de aquellos *tweets* que contengan más de tres palabras. Los *tweets* pertenecientes a otros países fueron eliminados manualmente.

  Similitud de coseno: La Figura 15, muestra el vocabulario recuperado a partir de las vectorización de las oraciones por medio *TfidfVectorizer*, sin considerar los *stopwords*.

```
from sklearn.feature_extraction.text import TfidfVectorizer
oracion_0 = '¡Seguimos avanzando!'
oracion_1 = '¡Seguimos creciendo y avanzando.'
oracion_2 = '¡Seguimos avanzando con fe y optimismo por un Perú mejor!'

vectorizer = TfidfVectorizer(stop_words = stopwords.words('spanish'))
X = vectorizer.fit_transform([oracion_0,oracion_1, oracion_2])
#print(vectorizer.vocabulary_)
print('VOCABULARIO:', vectorizer.get_feature_names())
```

```
VOCABULARIO: ['avanzando', 'creciendo', 'fe', 'mejor', 'optimismo', 'perú', 'seguimos']
```

Figura 15: *Vocabulario establecido*
Fuente: *Elaboración propia*

En la Figura 16 se observa los vectores de cada oración obtenidos por *TfidfVectorizer*. Para el primer vector, que pertenece a la primera oración, existen posiciones que tienen valor de cero, las cuales corresponden a las palabras del vocabulario que no aparecen en las otras oraciones.

```
print('Similitud de vocabulario entre oraciones')
#print(X)
print(X.toarray())
```

```
Similitud de vocabulario entre oraciones
[[0.70710678 0.         0.         0.         0.         0.
  0.70710678]
 [0.45329466 0.76749457 0.         0.         0.         0.
  0.45329466]
 [0.27249889 0.         0.46138073 0.46138073 0.46138073 0.46138073
  0.27249889]]
```

Figura 16: *Vectores obtenidos por TfidfVectorizer*
Fuente: *Elaboración propia*

Usando el producto punto es posible encontrar el ángulo entre dos vectores, este es el concepto conocido como similitud de coseno. Teniendo dos vectores y calculando el ángulo entre ellos, es posible medir que tan cerca están, y por tanto permite determinar que tan parecidos son dos textos. Un ángulo de cero significa que los textos son exactamente iguales, por eso la diagonal siempre tiene el valor de 1. En la Figura 17 se calcula la similitud entre los vectores de las oraciones por medio del coseno mediante *linear_kernel*.

```
cosine_sim = linear_kernel(X, X)
print('Calculo de matriz de similitud mediante coseno')
print(cosine_sim)
```

```
Calculo de matriz de similitud mediante coseno
[[1.         0.64105545 0.38537163]
 [0.64105545 1.         0.24704458]
 [0.38537163 0.24704458 1.        ]]
```

Figura 17: *Calculo de similitud entre vectores por medio de lineal_kernel*
Fuente: *Elaboración propia*

Finalmente, se obtiene **1953 *tweets*** para el escenario político y **2151 *tweets*** para el escenario pandemia.

- **Etiquetado de *tweets*:** Para el etiquetado, se considero como primera opción a *TextBlob*[13], una herramienta para el análisis de *tweets* en ingles, por lo cual se requirió traducir los *tweets* de español a ingles y finalmente obtener los *scores* en función de polaridad y subjetividad. Sin embargo, la mayoría de *tweets* fueron clasificados como neutros.

  Ante esta dificultad, se opto por experimentar con Pysentimiento[14], una herramienta que nos permite efectuar un etiquetado automático para clasificar los *tweets* en positivo, negativo o neutro, para ello se considera la categoría que posee el *score* más alto. En la Figura 18 y 19 se observa que la mayoría de *tweets*, en ambos escenarios, poseen el *score* más alto en el sentimiento neutro y la cantidad de *tweets* obtenidos para el sentimiento positivo es muy bajo. La Tabla 4 muestra un resumen de las categorías que posee cada *dataset* de acuerdo a su escenario.

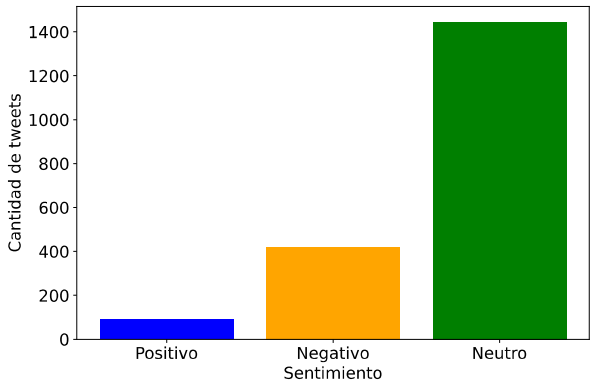

Figura 18: *Distribución de clases en dataset de escenario político*
Fuente: *Elaboración propia*

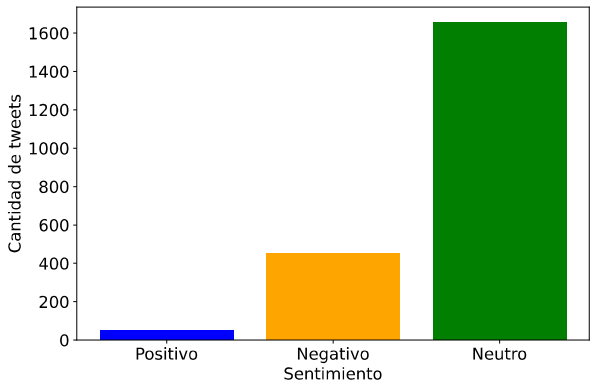

Figura 19: *Distribución de clases en dataset de escenario pandemia*
Fuente: *Elaboración propia*

[13]https://pypi.org/project/textblob/0.9.0/
[14]https://github.com/pysentimiento/pysentimiento

	Político	**Pandemia**
Positivo	90	50
Negativo	419	448
Neutro	1444	1653
Total *tweets*	1953	2151

Tabla 4: *Resumen de la distribución de tweets para cada escenario*
Fuente: *Elaboración propia*

Concluida esta primera fase, se observa que los resultados no fueron los esperados debido a:

- Presencia de ruido en los datos, tales como la recuperación de *tweets* fuera de contexto o uso de vocabulario inusual.
- Distribución de clases no proporcionadas, es decir el *dataset* no posee una cantidad balanceada de cada uno de los tres los sentimientos considerados, lo que perjudica en el proceso de aprendizaje de modelos de *Machine Learning*
- Contenido de más de una oración en un solo *tweet*.

3.1.2. FASE 2: Construcción del *dataset* con *usernames*

- **Extracción de *tweets* mediante usernames:** La Tabla 5 muestra la distribución de los *usernames* de cuentas verificadas que fueron seleccionados para la extracción, este proceso se explica anteriormente. Para el escenario político fueron considerados los candidatos de las elecciones presidenciales, ex-presidentes e instituciones del Estado. Mientras que para el escenario pandemia fueron considerados ex-ministros de salud, médicos, opinólogos e instituciones de salud.

	Político	**Pandemia**
número de usernames	14 usernames	14 usernames
Usernames considerados	@PedroCastilloTe, @KeikoFujimori, @MartinVizcarraC, @DanielUrresti1, @rlopezaliaga1, @yonhy_lescano, @CesarAcunaP, @HDeSotoPeru, @George_Forsyth, @Vero_Mendoza_F, @FSagasti, @MerinoDeLama, @ONPE_oficial, @JNE_Peru.	@Minsa_Peru, @ougarteu, @victorzamora, @drhuerta, @EsSaludPeru, @CMP_PERU, @SuSaludPeru, @SISPeruOficial, @HCevallosFlores, @ErnesBustamante, @maguina_ciro, @A_Aguinaga, @CayetanoHeredia, @EdMalagaTrillo.

Tabla 5: *Distribución de usernames para cada escenario*
Fuente: *Elaboración propia*

La Figura 20 muestra la configuración de ***Twint***, donde apenas necesita ser modificado el *Search* por el *Username*.

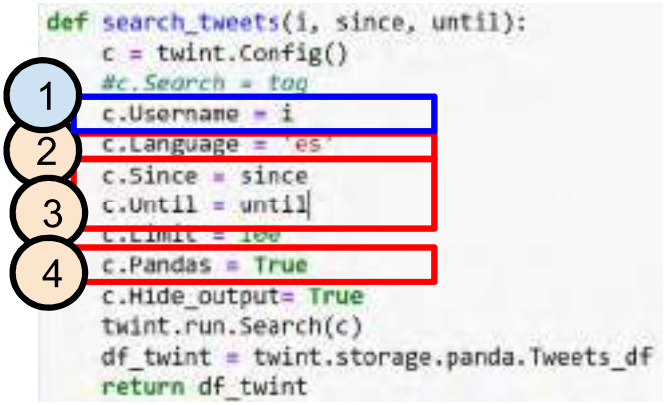

Figura 20: *Configuración de parámetros en* ***Twint*** *con username*
Fuente: *Elaboración propia*

Como resultado se obtiene **7346** ***tweets*** para el escenario político y **10525** ***tweets*** para el escenario pandemia.

- **Filtrado y limpieza de datos:** Se efectuó una limpieza para el nuevo *dataset* generado, donde la eliminación de *tweets* duplicados fue realizada mediante el calculo de la similitud de coseno y se selecciono aquellos *tweets* que contenían más de tres palabras. A partir de ello, se obtuvo **6795** ***tweets*** para el *dataset* político y **8878** ***tweets*** para el *dataset* pandemia.
- **Etiquetado de** ***tweets*****:** Se realiza el etiquetado de *tweets* en positivo, negativo y neutro con Pysentimiento. La Figura 21 muestra la distribución de *tweets* positivos, negativos y neutros respecto a cada *username* para el escenario político. Se observa una mayor proporción referida a los *tweets* de sentimiento neutro, seguida de los *tweets* negativos y finalmente los *tweets* positivos.

  La Figura 22 muestra la distribución de *tweets* positivos, negativos y neutros respecto a cada *username* para el escenario pandemia. Se observa que existió una mayor proporción referida a los *tweets* de sentimiento neutro, seguida de los *tweets* negativos y finalmente los *tweets* positivos.

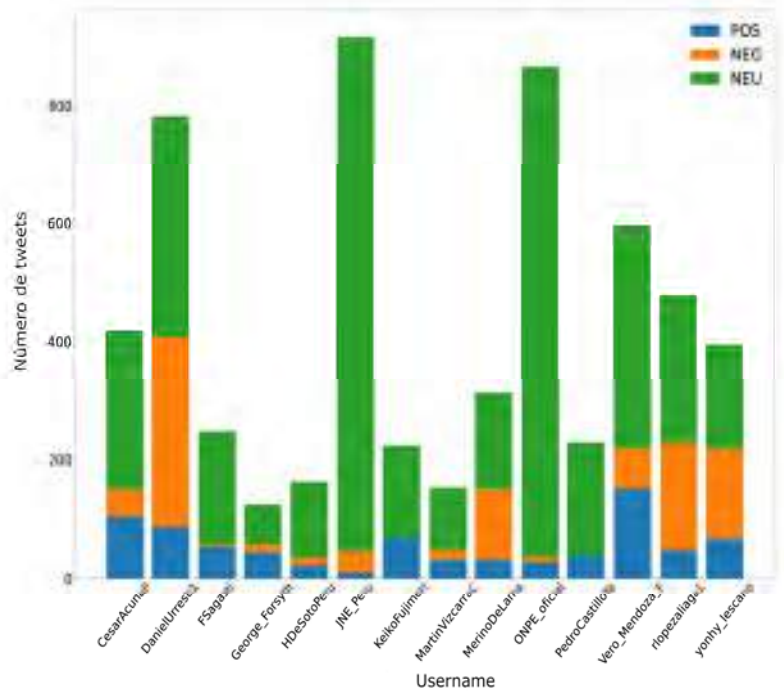

Figura 21: *Distribución de tweets por usernames de acuerdo a la categoría POS, NEG y NEU para el dataset político*
Fuente: *Elaboración propia*

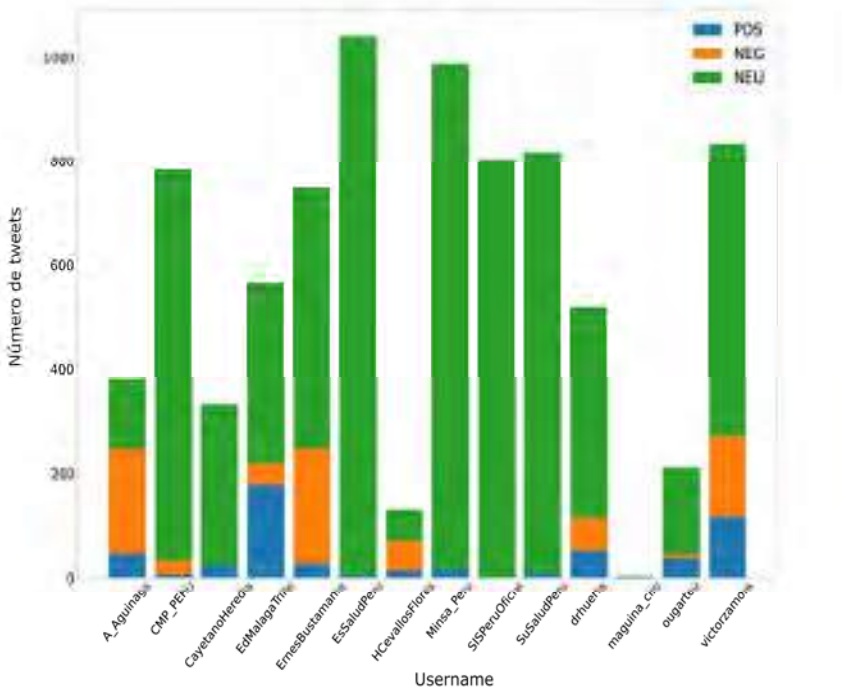

Figura 22: *Distribución de tweets por usernames de acuerdo a la categoría POS, NEG y NEU para el dataset pandemia*
Fuente: *Elaboración propia*

Concluida esta segunda fase, los resultados obtenidos poseen:

- Menos ruido en los datos, debido a que se cuenta con *tweets* de cuentas verificadas.
- Mayor número de *tweets* positivos y negativos, en comparación a los resultados obtenidos durante la FASE 1.

3.1.3. Desglosamiento de *tweets*

Como resultado de FASE 1 y FASE 2, el *dataset* político solo contiene aquellos *tweets* recuperados con *usernames*, mientras que el *dataset* pandemia contiene aquellos *tweets* recuperado con *usernames* y *hashtags*. De este modo, el *dataset* político contiene opiniones de cuentas verificadas que permite un mejor análisis semántico del texto, a diferencia de *dataset* pandemia, que incluye *tweets* recuperados con *hashtags*, que podrían presentar expresiones coloquiales y errores textuales, pero son considerados para incrementar los *tweets* de la clase positiva. En este punto, se noto que un solo *tweet* podía contener dos o más oraciones, por lo tanto estos fueron desglosados mediante la función *sent_tokenize* incluida en la librería *NLTK.tokenize*[15]. Por ejemplo, el *tweet 'Yo no me corro al debate @KeikoFujimori. Yo le propuse un debate urgente, como parte de la agenda popular y usted aceptó. No mienta. El pueblo es mi prioridad de vida y nada se interpondra a escucharlo en las calles. Nos vemos el sabado 1 de Mayo, Dia Internacional del Trabajo. https://t.co/5GzSGgGeBz.'* contiene 6 oraciones, por lo que fue dividido en 6 oraciones independientes. Asimismo se aplico la función *word_tokenize* incluida en la librería *NLTK.tokenize*[16] para una mejor filtrado de datos. En resumen, se obtuvo **10665 *tweets*** para el *dataset* político y **15958 *tweets*** para el escenario pandemia.

3.1.4. Filtrado de oraciones

A continuación se describen las condiciones que se siguieron para descartar oraciones que podrían generar ruido en nuestro *dataset*:

1. Aquellas oraciones que posean menos de 4 palabras y que inicien con 'https' o '(', debido a que el desglosamiento genera oraciones que contiene solamente links o que expresiones sin mayor información entre paréntesis.
2. Aquellas oraciones que posean menos de 9 palabras que contengan *stopwords* que incluyen palabras como *'vía', 'abrazo', 'día', 'sra', 'PD', etc.*

En resumen, la cantidad de *tweets* se redujo a **10424 *tweets*** para el *dataset* político y **15226 *tweets*** para el escenario pandemia.

[15]https://www.nltk.org/api/nltk.tokenize.html
[16]https://www.nltk.org/api/nltk.tokenize.html

3.1.5. Etiquetado de *tweets*

Debido al desglosamiento de *tweets*, es necesario volver a realizar la fase de etiquetado, para ello se considera la categoría que posea el *score* más alto. En la Tabla 6 se muestra la distribución de las clases analizando las oraciones obtenidas de los *tweets* independientemente. En ambos *dataset* predominan la clase de *tweets* neutro, seguido de la clase negativa y finalmente la clase positiva, sin embargo aun los *datasets* no poseen clases balanceadas.

En la Tabla 6 se muestra la nueva distribución

	Político	**Pandemia**
Positivo	1509	1109
Negativo	3558	3739
Neutro	5357	10378
Total *tweets*	10424	15226

Tabla 6: *Resumen de la distribución de oraciones para cada escenario*
Fuente: *Elaboración propia*

3.1.6. Balanceado de clases

El trabajo de la construcción del *dataset* también incluye el balanceado de clases, para ello se aplicaron ciertos criterios que permitieron filtrar las oraciones más relevantes y seleccionar el grupo de oraciones que pasaran por una verificación manual. Durante esta etapa se utiliza los *scores* obtenidos mediante ***Pysentimiento*** en el paso anterior. A continuación se describen los criterios de filtrado:

1. Aquellas oraciones que poseían 0 *likes* eran descartados, debido a que los likes son un indicador de relevancia dentro de una Twitter.
2. Aquellas oraciones que poseían menos de 4 palabras eran descartadas, porque se observo que la mayoría no poseían información con sentido relevante para el análisis.
3. Para el escenario político, solo fueron considerados aquellas oraciones que poseían un *score* mayor de 0.90 en la categoría neutro y negativo.
4. Para el escenario pandemia, solo fueron considerados aquellas oraciones que poseían un *score* mayor de 0.95 en la categoría neutro y negativo.
5. Tanto para el escenario político y pandemia, las oraciones que poseían un *score* entre 0,50 y 0,60 en la categoría neutro fueron destinados para la verificación manual.

El motivo del criterio 3 y 4 radico en la importancia de contar con *tweets* que son altamente perceptibles como negativos o neutros, lo cual se reflejo en el valor del *score*. En tanto, para el criterio 5, las oraciones con *score* ambivalente entre dos categorías no reflejan exactitud, por lo cual una verificación manual convendría en esta situación.

La Tabla 7 muestra la distribución de las clases para la *dataset* de cada escenario: **6453** ***tweets*** para el escenario político y **4342** ***tweets*** para el escenario pandemia. En comparación con la Tabla 6, el escenario político

se redujo en un 30 %, mientras que el escenario pandemia se redujo casi en un 65 %, esto estuvo motivado porque también fueron incluidos *tweets* de opiniones personales que muchas veces contienen opiniones inexactas o sin sentido, en comparación con las opiniones vertidas por personajes de cuentas verificadas.

	Político	**Político revisión**	**Pandemia**	**Pandemia revisión**
Positivo	1470	179	1010	158
Negativo	2347	218	1567	260
Neutro	2636	404	1765	488
Total *tweets*	6453	801	4342	906

Tabla 7: *Resumen de la distribución de oraciones para la versión final de los datasets y para el grupo de verificación manual*

Fuente: *Elaboración propia*

Finalmente se obtuvo **6453 *tweets*** corresponden al *dataset* político y **4342 *tweets*** corresponden al *dataset* pandemia, ambos *datasets* poseen clases balanceadas como se muestra en la Figura 23 y 24l, esto permitirá lograr un mejor proceso de aprendizaje en el entrenamiento de modelos.

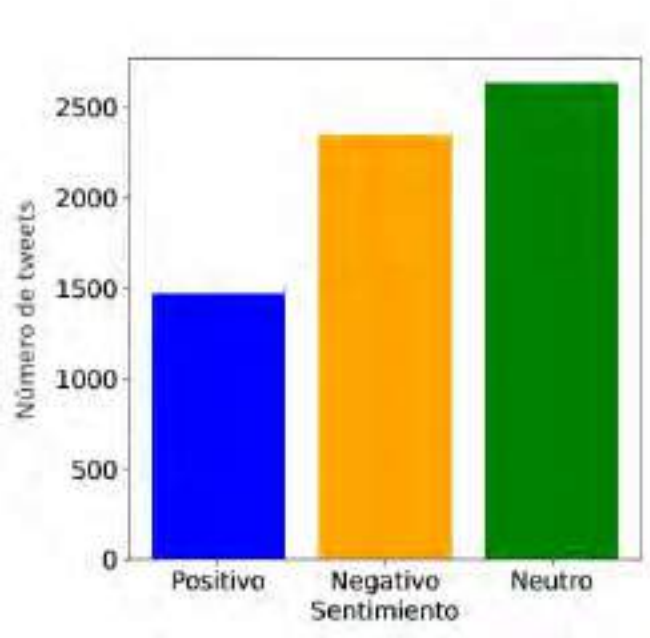

Figura 23: *Distribución final de clases en dataset de escenario político*

Fuente: *Elaboración propia*

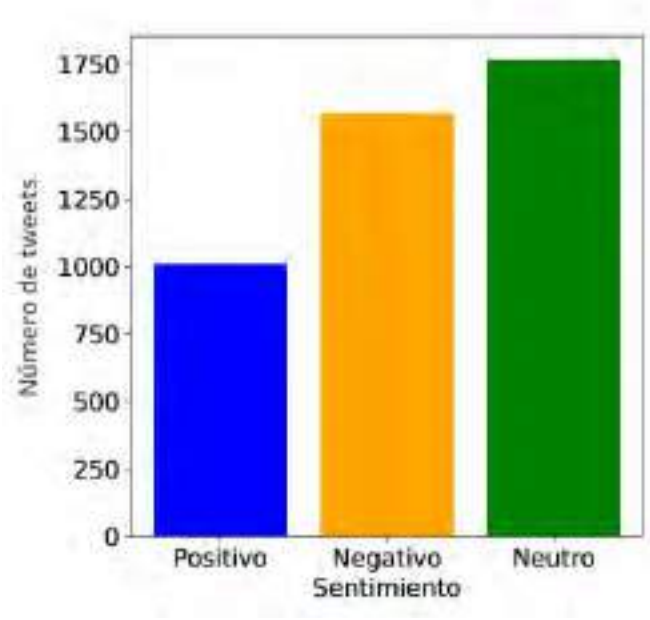

Figura 24: *Distribución de clases en dataset de escenario pandemia*
Fuente: *Elaboración propia*

3.1.7. Verificación Manual

Durante esta etapa se busco comprobar el etiquetado establecido (***Pysentimiento***), con el propósito de corregir posibles errores y verificar el etiquetado. Para este fin, se seleccionaron 801 *tweets* para el escenario político y 906 para el escenario pandemia. Por medio de dos archivos en excel, tres etiquetadores efectuaron la revisión del etiquetado destinando 2 horas durante 31 días, es decir aproximadamente 2 minutos para el etiquetado por cada *tweet*.

En la Figura 25 se describe el esquema que se sigue para la verificación manual. Se observa que cada *tweet* recibe un target entre POS, NEG o NEU, este es definido como el sentimiento. Mientras que la relevancia es un binario que indica con 1 si es irrelevante o 0 si no lo es. Los motivos para considerar a un *tweet* como irrelevante fueron por oración sin sentido o simplemente porque no correspondía al contexto en cuestión. Los tres etiquetadores evaluaron cada *tweet* y posteriormente se obtiene el *target* final y la relevancia final por medio de una comparación de las respuestas generadas por cada etiquetador.

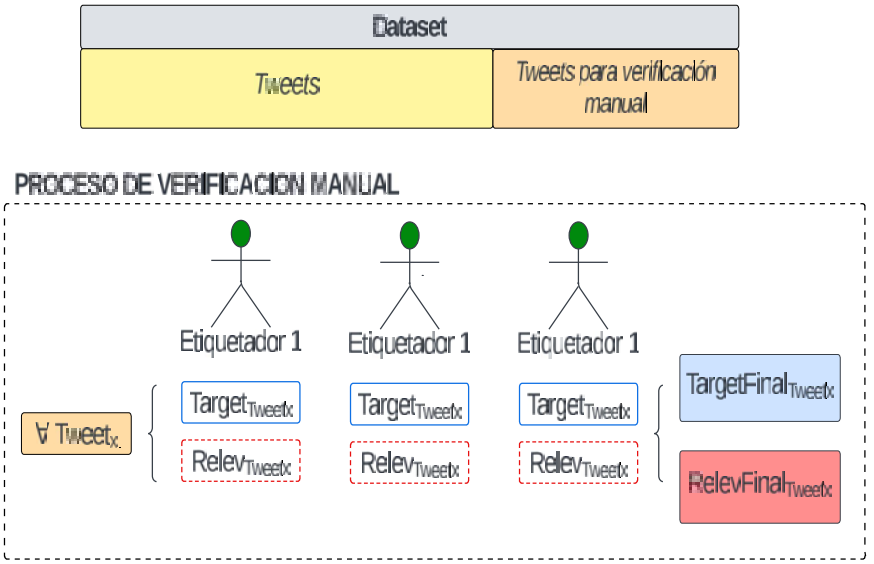

Figura 25: *Esquema a seguir para la verificación de tweets*
Fuente: *Elaboración propia*

Para evitar empates dentro del calculo del *target* final y/o la relevancia final, se escogió un número impar de etiquetadores, en este caso 3. En el caso de la relevancia se utilizo la siguiente condición: si más de dos etiquetadores consideraron un *tweet* como irrelevante, se procedió a descartar el *tweet*. Si este no era el caso, se siguen las siguientes condiciones para la obtención del target final basado únicamente en la comparación de los *target* generados por cada etiquetador:

- Si de las tres etiquetas obtenidas, basta que el número de ocurrencias de una de las categorías supere 2, el *target* final correspondió a la categoría con mayor numero de ocurrencias.
- Si de las tres etiquetas obtenidas, basta que el número de ocurrencias de una de las categorías es igual a 1, entonces el *tweet* no posee un sentimiento claro por lo que es descartado.

En la Tabla 8 se referencia ejemplos basados en las condiciones presentadas anteriormente:

		Etiquetador1	**Etiquetador2**	**Etiquetador3**	***Target** final*
Ejemplo 1	$Tweet_1$	*POS*	*POS*	*POS*	*POS*
Ejemplo 2	$Tweet_2$	*NEG*	*NEG*	*POS*	*NEG*
Ejemplo 3	$Tweet_3$	*NEG*	*POS*	*NEU*	-

Tabla 8: *Tabla de casos para obtener el target final*
Fuente: *Elaboración propia*

Como resultado de este proceso, se obtuvo:

- Para el *dataset* político, se eliminaron 49 *tweets* por ser irrelevantes o por no poseer un sentimiento claro, y se modificó el *target* de 330 *tweets*.
- Para el *dataset* pandemia, se eliminaron 91 *tweets* por ser irrelevantes o por no poseer un sentimiento claro, y se modificó el *target* de 362 *tweets*.

La Tabla 9 muestra la nueva distribución de *tweets* que pertenecen al grupo de verificación manual:

	Político revisión	**Político verificado**	**Pandemia revisión**	**Pandemia verificado**
Positivo	179	**293**	158	**337**
Negativo	218	**156**	260	**218**
Neutro	404	**303**	488	**260**
Total *tweets*	801	**752**	906	**815**

Tabla 9: *Tabla de distribución final de tweets para el grupo de verificación manual*
Fuente: *Elaboración propia*

Al termino de la FASE 2, por el lado del escenario político se obtiene el *dataset* en función de la extracción mediante *usernames*. Para el escenario pandemia se opta por incluir el resultado de la extracción mediante *hashtags* y *usernames*. Se enfatiza que cada escenario posee un *dataset* independiente como resultado del proceso de extracción de *tweets*. La distribución final se observa en la Tabla 10

	Político	**Pandemia**
Positivo	1763	1347
Negativo	2503	1785
Neutro	2939	2025
Total *tweets*	7205	5157

Tabla 10: *Tabla de distribución final de tweets para cada escenario*
Fuente: *Elaboración propia*

3.2. Exploración del *dataset*

Esta etapa permite analizar y visualizar los datos para tener una mejor perspectiva, de ese modo se identifico características importantes en el *dataset* como la verificación de correlación de los datos o la identificación de anomalías (Munappy et al., 2019).

Para el *dataset* político, la Figura 26 muestra que la mayor distribución se concentra entre 1 a 3 oraciones en cada *tweet*. Mientras que la Figura 27 muestra que la mayor distribución se concentra entre 5 a 10 palabras en cada uno de los *tweets*.

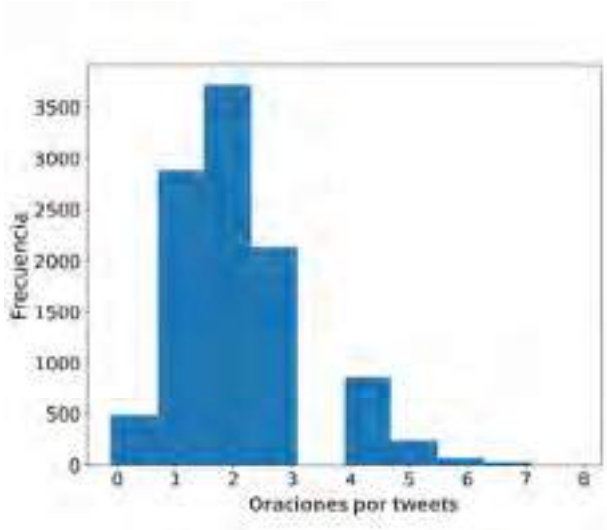

Figura 26: *Distribución de oraciones en los tweets para el escenario político*
Fuente: *Elaboración propia*

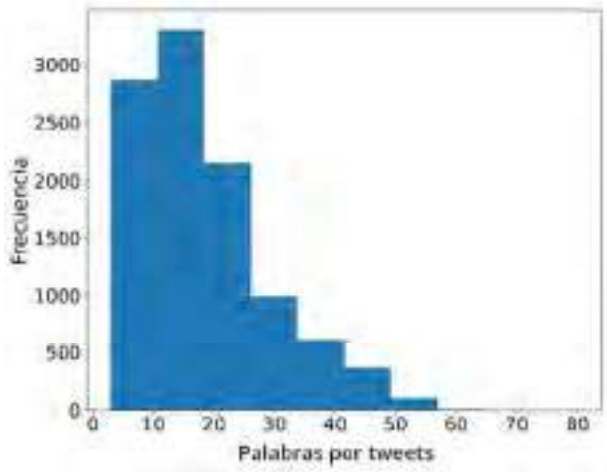

Figura 27: *Distribución de palabras en los tweets para el escenario político*
Fuente: *Elaboración propia*

Para el *dataset* pandemia, la Figura 28 muestra que la mayor distribución se concentra entre 1 a 3 oraciones en cada *tweet*. Mientras que la Figura 29 muestra que la mayor distribución se concentra entre 5 a 25 palabras en cada uno de los *tweets*.

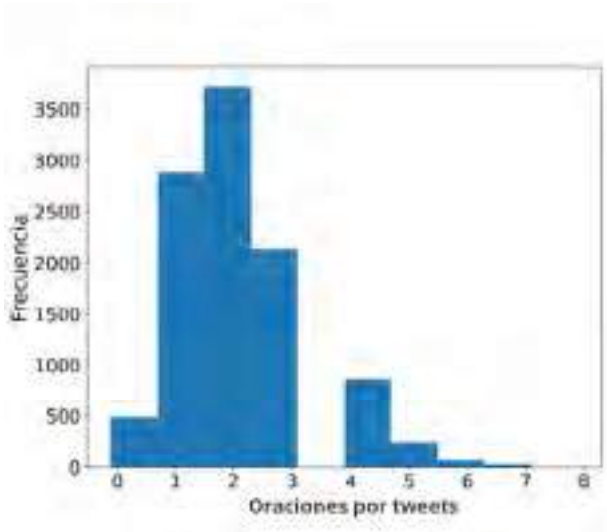

Figura 28: *Distribución de oraciones en los tweets para el escenario pandemia*
Fuente: *Elaboración propia*

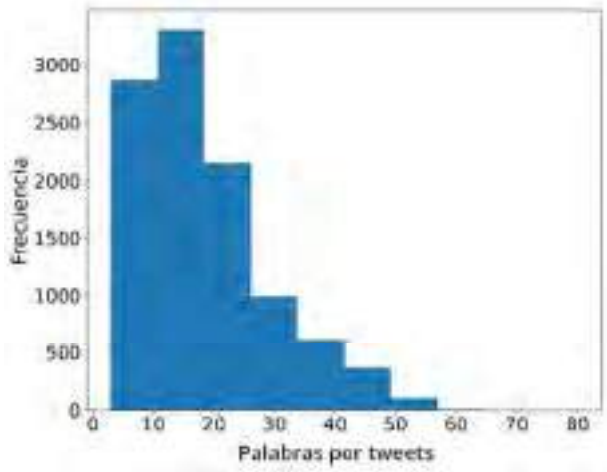

Figura 29: *Distribución de palabras en los tweets para el escenario pandemia*
Fuente: *Elaboración propia*

También se realizo un análisis introspectivo en la distribución de las palabras en los *tweets* en crudo mediante un cuadro de frecuencia. Un *tweet* crudo conserva el estado inicial de cuando es publicado en twitter, este no tiene ningún cambio o edición y puede contener errores ortográficos, minúsculas, mayúsculas, y en su mayoría parecencia de URLs, *hashtags*, *usernames*, emojis, signos de puntuación, números y correos electrónicos. Los *tweets* crudos son datos de entrada complejos para un modelo de clasificación para se analizados directamente, por este motivo se opta por realizar un pre-procesamiento que hace que los *tweets* sean datos más simples sin afectar su interpretabilidad, se esta manera también se logra mejoras los resultados obtenidos con los modelos de clasificación (Denny and Spirling, 2018). Además el pre-procesamiento de datos se puede realizar por las siguientes razones: (Famili et al., 1997)

- Resolver problemas de datos que puedan impedir realizar cualquier tipo de análisis de los datos
- Comprender la naturaleza de los datos y realizar un análisis de datos más significativo
- Extraer conocimiento más significativo de un conjunto dado de datos.

En la Figura 30, se observan que existen elementos como `peru` y `peru.` que deberían ser considerados como uno solo, sin embargo la presencia del carácter '.' incurre en que la segunda palabra sea considerada como diferente. En la Figura 31, se observa que existe el carácter 'd' que es considerado incorrectamente como palabra. Estos problemas fueron propios del análisis del texto en crudo, gracias a este análisis se puede visar una solución mediante el uso de técnicas de PLN.

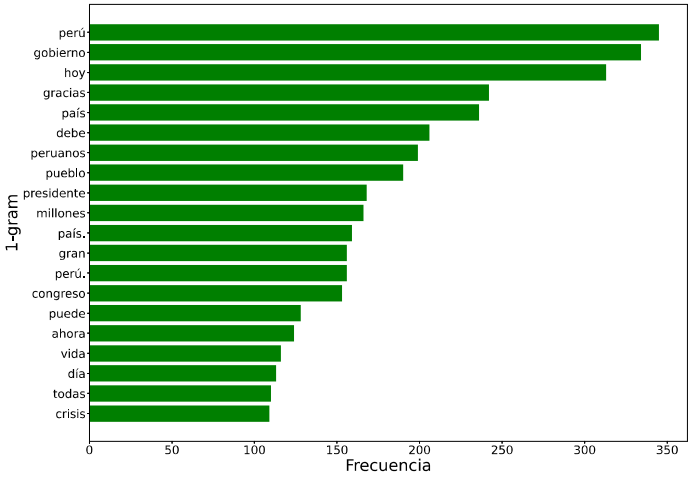

Figura 30: *Frecuencia de palabras para dataset político*
Fuente: *Elaboración propia*

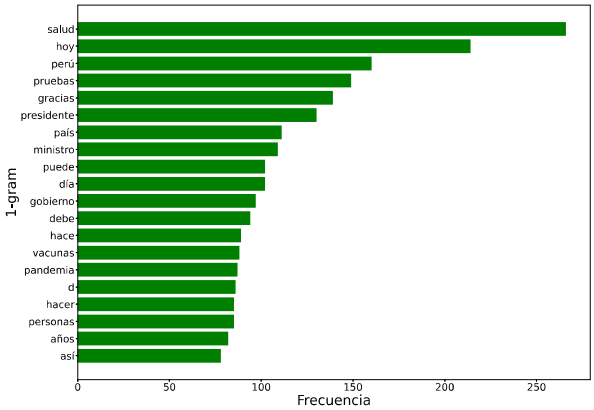

Figura 31: *Frecuencia de palabras para dataset pandemia*
Fuente: *Elaboración propia*

En segundo lugar, se realizo un análisis de los bi-grams con mayor frecuencia. En la Figura 32, se observa que `#elecciones2021 |` o ':' son grams que no deberían ser considerados debido a que son *hashtags* y signos de puntuación. En la Figura 33 se observa que `covid- .` contiene un signo de puntuación, mientras que `ministro @victorzamora:` contiene un *hashtag*. Se observa que los *usernames* y *hashtags* están siendo considerados dentro del análisis de texto.

En tercer lugar, se realizo un análisis a los tri-grams con mayor frecuencia. En la Figura 34 y Figura 35 se observa que *usernames*, *hashtags* y emoticones están siendo considerados dentro del análisis de texto.

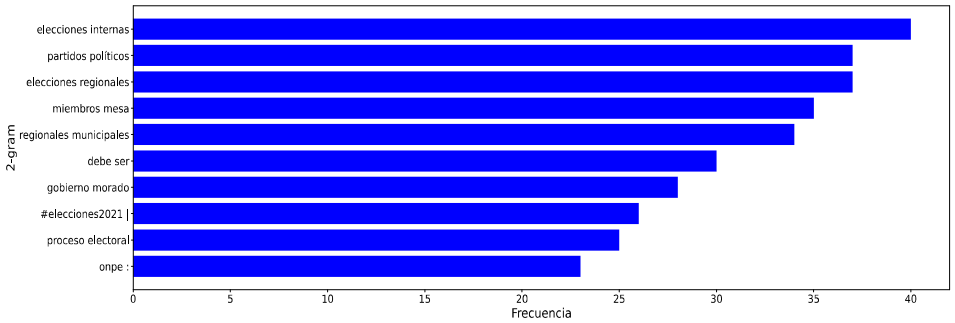

Figura 32: *Frecuencia de bi-grams para dataset político*
Fuente: *Elaboración propia*

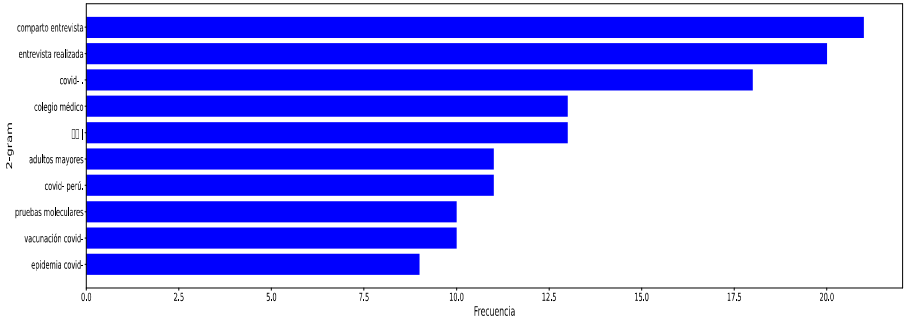

Figura 33: *Frecuencia de bi-grams para dataset pandemia*
Fuente: *Elaboración propia*

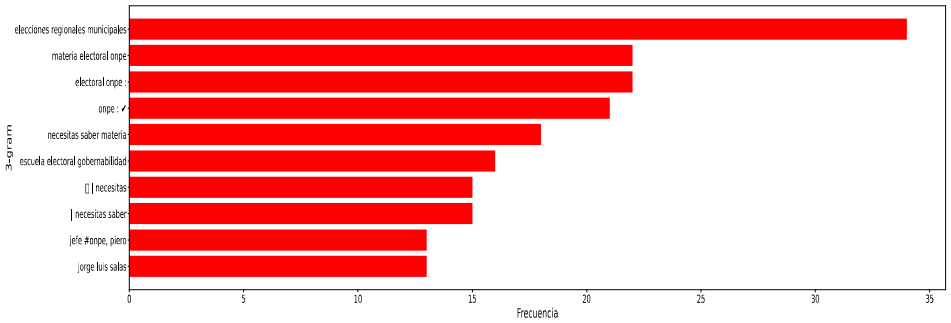

Figura 34: *Frecuencia de tri-grams para dataset político*
Fuente: *Elaboración propia*

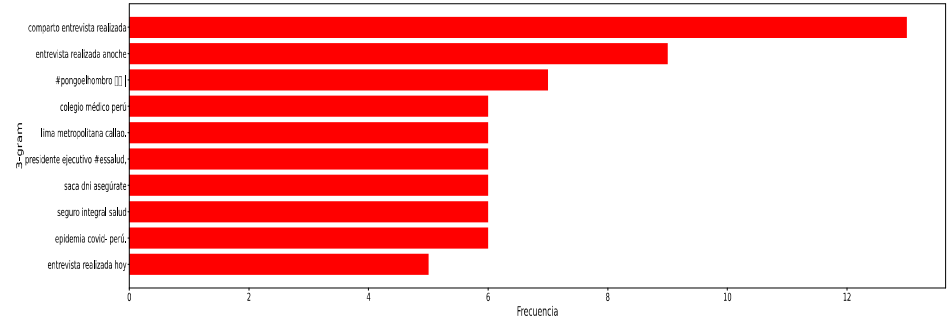

Figura 35: *Frecuencia de tri-grams para dataset pandemia*
Fuente: *Elaboración propia*

Por ultimo se realizo una exploración de palabras frecuentes por medio de la herramienta *WordCloud*[17], para ello se realizo previamente una limpieza del texto, ya que como se vio anteriormente existen elementos que causaron ruido dentro del análisis de texto. En la Figura 36 y 37 se muestran las palabras con mayor frecuencia sin considerar emoticones, urls, *usernames* y *hashtags*. Se observa que palabras como `Perú`, `país` y `Hoy` visualmente son palabras con mayor frecuencia para ambos escenarios.

Figura 36: *WordCloud para dataset político*
Fuente: *Elaboración propia*

Figura 37: *WordCloud para dataset pandemia*
Fuente: *Elaboración propia*

[17]https://pypi.org/project/wordcloud/

Para el escenario político se contó con un vocabulario de 13236 palabras, de las cuales 634 son *stopwords* o palabras vacías que no expresan ningún sentimiento, donde son incluidas palabras como *que, con, por, para, del, los, las, una, más, etc.* Dentro de este escenario, la siguiente lista muestra ciertas palabras para cada categoría con su respectiva frecuencia:

- **Positivo:** Perú: 73, país: 63, Hoy: 49, pueblo: 44, peruano: 33, gobierno: 26, junto: 24, gran: 23, familia: 23, trabajo: 22, compromiso: 21, siempre: 21, esperanza: 20, toda: 19, mejor: 19, año: 19, apoyo: 18, vida: 18, lucha: 18, fuerza: 17, Vamos: 16, equipo: 16, día: 14, saludo: 14, desarrollo: 14, gente: 14, esfuerzo: 14, democracia: 13, patria: 12, salud: 12, política: 12, presidente: 12, cada: 12, momento: 12, hacer: 12, gracia: 12, cambio: 11, propuesta: 11, nuevo: 11, importante: 11, seguir: 11, tiempo: 11, lograr: 11, Congreso: 11, derecho: 11, ahora: 11, nueva: 10
- **Negativo:** gobierno: 220, Perú: 168, país: 150, peruano: 125, Congreso: 115, debe: 107, ahora: 91, presidente: 90, hoy: 90, pueblo: 89, gente: 68, poder: 64, corrupción: 63, ser: 63, ley: 62, ministro: 62, Vizcarra: 58, sino: 58, AFP: 58, política: 56, muerte: 56, pandemia: 56, hace: 54, crisis: 54, derecho: 52, político: 52, trabajadores: 51, tiempo: 50, partido: 48, abuso: 48, persona: 48, Basta: 47, candidato: 47, vez: 47, mientras: 46, empresa: 46, quieren: 43, medio: 43, dinero: 43, pueden: 42, problema: 42, salud: 40, nunca: 40
- **Neutro:** JNE: 220, hoy: 139, Perú: 134, país: 119, peruano: 92, gobierno: 85, presidente: 75, organizaciones: 75, candidato: 73, día: 70, Congreso: 65, política: 63, democracia: 58, debe: 57, ERM2022: 56, toda: 55, ONPE: 52, Elecciones: 51, voto: 50, ley: 50, electoral: 47, miembros mesa: 46, derecho: 44, ciudadano: 44, año: 44, proceso: 43, propuesta: 43, persona: 43, Pleno: 40, caso: 40, medida: 39, deben: 39, momento: 38, electorales: 38, partido: 35, virtual: 35

Para el escenario pandemia se contó con un vocabulario de 12543 palabras, de las cuales 397 son *stopwords* o palabras vacías que no expresan ningún sentimiento como se menciono para el escenario anterior. Dentro de este escenario, la siguiente lista muestra ciertas palabras para cada categoría con su respectiva frecuencia:

- **Positivo:** Perú: 266, salud: 252, COVID19: 202, atención: 144, pandemia: 139, país: 139, año: 125, vacunación: 108, trabajo: 105, vacuna: 105, COVID: 105, lucha: 103, bien: 94, esfuerzo: 91, compromiso: 91, vida: 89, peruano: 86, importante: 83, Hospital: 83, toda: 78, mejor: 75, equipo: 75, persona: 75, paciente: 75, siempre: 72, Vamos: 69, población: 69, Excelente: 66, presidente: 66, campaña: 64, gracias: 64, nuevo: 64, asegurado: 64, gobierno: 61, Feliz: 61, Nacional: 58, personal: 58, gran: 55, Felicitaciones: 55, EsSalud: 55, ciencia: 55, buena: 53, reconocimiento: 50, labor: 50.
- **Negativo:** Perú: 149, pandemia: 80, gobierno: 69, COVID: 67, país: 67, salud: 61, vacuna: 55, ministro: 49, coronavirus: 47, año: 46, persona: 45, muerte: 43, COVID19: 43, caso: 39, problema: 38, presidente: 36, peruano: 36, ser: 34, ahora: 34, menos: 31, política: 30, mal: 27, SaludMental: 27, virus: 26, vida: 26, Congreso: 24, prueba: 24, mismo: 23, mil: 22, gente: 22, corrupción: 22, falta: 22, compra: 22, pobre: 21, hecho: 21, peor: 21, pueblo: 21, crisis: 21.

- **Neutro:** Perú: 350, vacuna: 287, COVID: 178, salud: 174, vacunación: 162, caso: 147, COVID19: 120, persona: 109, año: 108, Minsa: 101, coronavirus: 88, atención: 86, puede: 84, Lima: 81, tema: 79, dosis: 79, días: 75, pandemia: 75, semana: 75, ser: 75, estudio: 70, debe: 68, SIS: 68, médico: 66, información: 66, país: 63, prueba: 63, nuevo: 57, población: 56, virus: 56, Nacional: 54, grupo: 54, datos: 54, paciente: 54, interno: 54, hacer: 52, Presidente: 50, hora: 50, proceso: 48, países: 48, pregunta: 48, mascarilla: 48.

En general para ambos escenarios, se observa que una palabra pueden corresponder a más de una categoría, ya que diversos tópicos como Perú, gobierno o pandemia pudieron haber pertenecido a diferentes tipos de opiniones.

3.3. pre-procesamiento

De la Sección 3.1, se obtiene dos *datasets* con *tweets* crudos que podrían contener elementos que generen ruido en el análisis de texto. Esta etapa se encarga de la transformación de algunos elementos especiales que deben ser incluidos en el análisis, la Figura 38 muestra el flujo de trabajo para esta etapa, donde se incluyen los siguientes pasos:

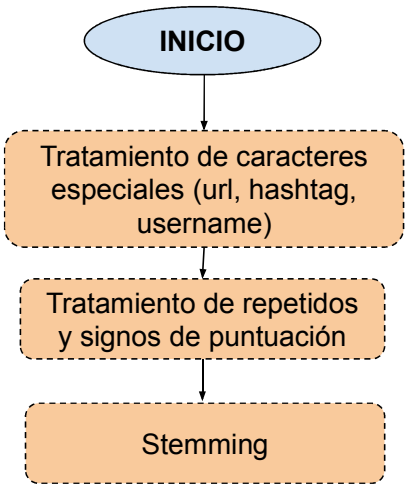

Figura 38: *Flujo de trabajo para el pre-procesamiento de tweets*
Fuente: *Elaboración propia*

Estos *tweets* crudos contienen elementos como urls de imágenes, vídeos, sitios web, entre otros (Figura 39), lo cuales pueden alterar el análisis de texto. Por otro lado, también incluyen elementos especiales que necesitan ser analizados independientemente, como: *usernames* (Figura 40), *hashtag* (Figura 41) y emojis (Figura 52) sin identificar y signos de puntuación (Figura 43).

76 Ahora sabemos por qué la respuesta contra el COVID-19 en Junin ha sido una desgracia. La irresponsabilidad de personajes como Vladimir Cerron hacen que miles sufran ¿Cuando se pronunciara @Minsa_Peru o se sancionará este tipo de declaraciones nefastas? https://t.co/kcezOQKpHS

Figura 39: *Ejemplo de tweet que contiene url*
Fuente: *Elaboración propia*

592 @drhuerta @Minsa_Peru @MartinVizcarraC @pcmperu @SuSaludPeru @EsSaludPeru Doctor Huerta cómo la Ministra de Salud @EHinostrozaP va a afrontar el #Covid19 si actualmente en el #Minsa y @EsSaludPeru a las #enfermeras les limitan los insumos básicos como mascarillas, gorros y guantes? @NursingNow2020

Figura 40: *Ejemplo de tweet que contiene username*
Fuente: *Elaboración propia*

132 La Iglesia en el #Peru sigue ayudando a los hermanos más necesitados, esta vez en los andes de #Ayacucho. A pesar de los meses de #pandemia todos debemos seguir apoyando a la Iglesia pues nos garantiza que nuestra ayuda llegará a quienes lo necesitan. https://t.co/8KR3f8sNGj

Figura 41: *Ejemplo de tweet que contiene hashtag*
Fuente: *Elaboración propia*

3168 @FenixconL A Uds por el apoyo! Buen finde!

Figura 42: *Ejemplo de tweet que contiene emojis*
Fuente: *Elaboración propia*

1376 ¡¡Sálvese quien pueda!! Que el mensaje [illegible] lo peor, ¡¡¡ siquiera nos acompañan orientándonos sobre qué hacer!!! ¿Cambiamos de mascarillas? ¿Recomendaciones para el auto cuarentena? ¿Lo hago si algo positivo? ¿Cómo uso el pulsioxímetro? Una mezcla de indolencia e incapacidad.

Figura 43: *Ejemplo de tweet que contiene signos de puntuación*
Fuente: *Elaboración propia*

3.3.1. Tratamiento de elementos especiales

En esta etapa se implementa funciones para identificar *hashtags*, *usernames*, url's y signos de puntuación, y luego reemplazar estos elementos especiales por etiquetas para ser analizados junto al texto. Primero se realiza la normalización de la oración, es decir se uniformiza todas las palabras en mayúsculas a minúsculas

con la función *lower()*, tal como se muestra en la Figura 44. Para realizar esta tarea se utiliza el modulo *Regular Expression Operations (re)* [18], incluida en Python para el análisis de texto. A través de operaciones de coincidencia y sustitución entre cadenas de caracteres, resultó muy útil para identificar los *hashtags*, urls y *usernames* (#, https://, @) y remplazarlos por las etiquetas correspondientes como se muestra en la Tabla 11

Figura 44: *Ejemplo de tweet normalizado*
Fuente: *Elaboración propia*

Caracter	Etiqueta
#	__HASH_
@	__user_
https://url	__URL_
?	__PUNC_QUES
!	__PUNC_EXCL
...	__PUNC_ELLP

Tabla 11: *Tabla de caracteres reemplazados*
Fuente: *Elaboración propia*

Se observa, que el símbolo '#' en el *hashtag* fue remplazado por el identificador `__HASH_` (Figura 45), '@' en el *username* fue remplazados por `__user_` (Figura 46) y el url, que normalmente es identificado por iniciar con 'https://', fue remplazado por la etiqueta `__URL_` (Figura 47). Las palabras especiales como los *hashtags* y *usernames* se guardaron en mayúsculas usando la función *upper* para estandarizarlas. Este proceso fue ejecutado para cada *tweet* sin alterar el texto original.

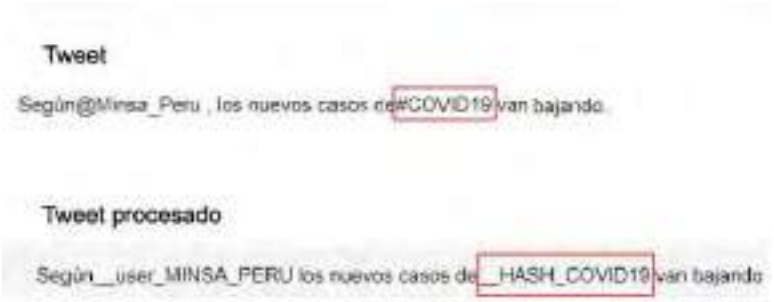

Figura 45: *Tweet con hashtag identificado*
Fuente: *Elaboración propia*

[18]https://docs.python.org/3/library/re.html

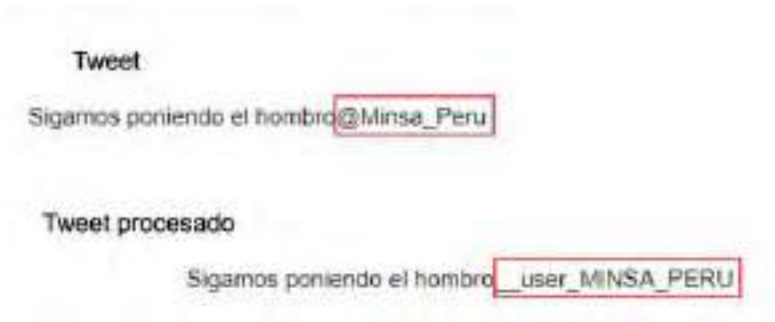

Figura 46: *Tweet con username identificado*
Fuente: *Elaboración propia*

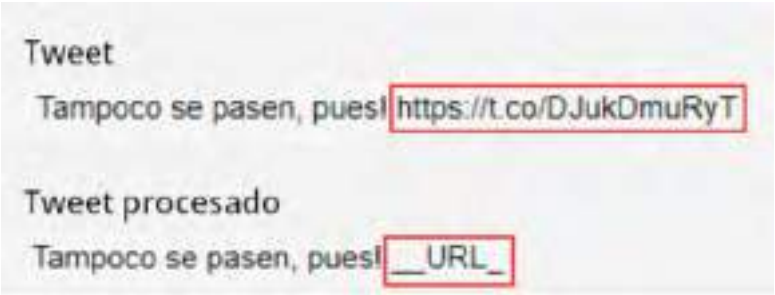

Figura 47: *Tweet con Url identificado*
Fuente: *Elaboración propia*

3.3.2. Tratamiento de signos de puntuación y repeticiones

En caso de los signos de puntuación se usa `__PUNC_EXCL` para los signos de exclamación, `__PUNC_QUES` para los signos de interrogación y `__PUNC_ELLP` para los puntos suspensivos. Por ejemplo, en la Figura 48, se observa un caso donde se ejecuta el reemplazo de los signos de exclamación e interrogación.

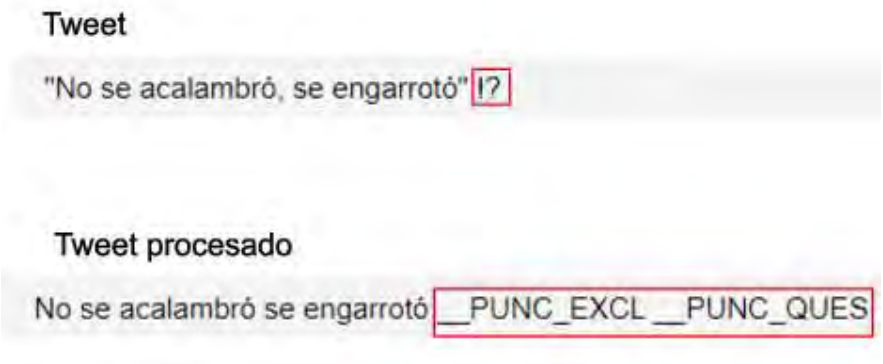

Figura 48: *Tweet con signos de puntuación identificados*
Fuente: *Elaboración propia*

Adicionalmente se incluyo la implementación de una función para la corrección de letras repetidas en una palabra, por ejemplo `leeeento` y obtener la forma correcta de la palabra `lento`. Mediante las funciones que ofrece el modulo *re* para

procesamiento de texto. Se identifica un conjunto de caracteres iguales dentro de una palabra y se considero solo la primera aparición del caracter, ignorando los demás. En la Figura 49 se muestra un ejemplo para la corrección de elementos repetidos.

```
rpt_regex = re.compile(r"(.)\1{1,}", re.IGNORECASE);
def rpt_repl(match):
    return match.group(1)+match.group(1)

# Test
re.sub(rpt_regex, rpt_repl, "Identificaaaando letras reeeeepetiiiiidaaaaaas" )

'Identificando letras repetidas'
```

Figura 49: *Ejemplo de corrección de letras repetidas en una palabra*
Fuente: *Elaboración propia*

3.3.3. Stemming

Durante la etapa de *stemming* se identifican y conservan solo las raíces de las palabras. Se utiliza el algoritmo ***SnowballStemmer***[19] disponible para ***Python***, que permite el trabajo con texto en español. Este algoritmo esta desarrollado en ***Snowball***, un lenguaje (Porter, 2001). En este proceso se siguen los siguientes pasos:

- Se utiliza la función *lower* para convertir el texto restante, que no tenga una etiqueta de *hashtag*, *username* y demás, en minúsculas.
- Se eliminan las palabras con longitud menor o igual a 3 para eliminar los stopwords, palabras que no aportan significado a la oración, como artículos y preposiciones. Con el fin de realizar el análisis de las negaciones se conservan algunas palabras que pueden ser consideradas como *stopwords* como son: 'no', 'ni' o 'si' entre otras.
- Finalmente se aplica la función *stem* para obtener la raíz de cada palabra.

Al aplicar las funciones desarrolladas se obtienen el siguiente resultado:

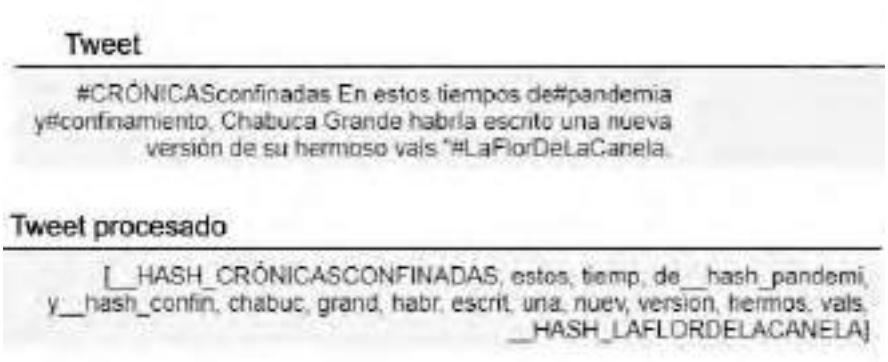

Figura 50: *Tweet despues de aplicar Stemming*
Fuente: *Elaboración propia*

[19]https://pypi.org/project/snowballstemmer/

Se opto por no realizar la lematización, por ser un proceso que consume el tiempo, en comparación a *Stemming* que puede reconocer cuando dos palabras tienen la misma raíz, esto nos ayuda a reducir el número de elementos que conforman el texto. En el caso de nombres propios no es relevante que estos sean recortados por el algoritmo, siempre y cuando cada una de las apariciones de nombres propio sean recortados de la misma forma para ser reconocido por los modelos de clasificación.

3.4. Feature Engineering

En esta etapa se busco mejorar la calidad del análisis del texto por medio de la inclusión de nuevos atributos que permitan extraer mayor información para la clasificación. La Figura 51 muestra el flujo de trabajo para esta etapa, donde se incluyen los siguientes pasos:

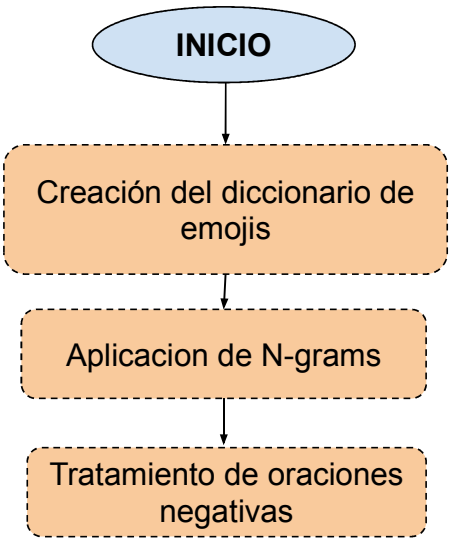

Figura 51: *Flujo de trabajo para feature Engineering*
Fuente: *Elaboración propia*

3.4.1. Creación del diccionario de emojis

Los emojis fueron considerados para el análisis de sentimientos mediante la construcción de un diccionario que los clasifica según el sentimiento que exprese. A partir de un repositorio que almacena el UNICODE[20] de cada emoji, se realiza la clasificación en 80 categorías como *face-positive, face-neutral, face-negative, face-sick, monkey-face, person, person-role, person-fantasy, family, body, emotion, clothing, animal-mammal, animal-bird, food-fruit, place-map, time, money, mail, writing, office, flag, etc.* Este archivo contiene la categoría del emoji, el UNICODE y la descripción o nombre corto.

[20]http://www.unicode.org/emoji/charts/full-emoji-list.html

Sentimiento	Código
Felicidad	U+1F601
Risa	U+1F923
Amor	U+2764
Guiño	U+1F609
Desaprobación	U+2639
Llanto	U+1F622

Tabla 12: *Tabla de los códigos por sentimiento*
Fuente: *Elaboración propia*

Posteriormente los emoticones son identificados y reemplazados por el *UNICODE* correspondiente, como se muestra en la Figura 52. Seguidamente, se recupera la categoría de cada emoji y emoticon identificado en un *tweet*, para ello se establece columnas extra en el *dataset* como se observa en la Figura 53.

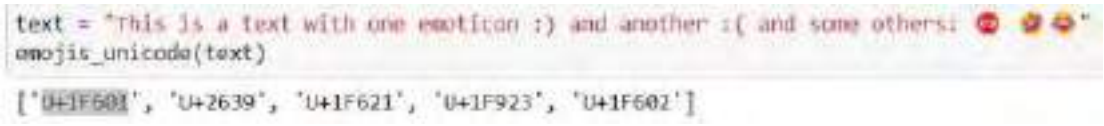

Figura 52: *Tweet con emojis identificados*
Fuente: *Elaboración propia*

Figura 53: *Dataset con la columna Emoji*
Fuente: *Elaboración propia*

3.4.2. Aplicación de N-grams

Se aplica N-grams para convertir el texto a un formato estructurado, ya que la secuencia de las palabras es relevante para el análisis de sentimientos. Durante este proceso se obtiene las listas de uni_grams, bi_grams y tri_grams descartando las repeticiones como se observa del ejemplo en la Figura 54, del cual se obtiene el resultado mostrado en la Figura 55. Para obtener los N-grams se utiliza la librería *NLTK* [21] que incluye la funciones bigrams y trigrams. También se utiliza la función *nltk.FreqDist* que nos retorna la frecuencia de los N-grams en la oración.

[21] https://www.nltk.org/

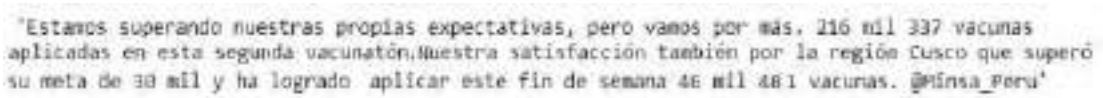

Figura 54: *Tweet antes de aplicar N-grams*
Fuente: *Elaboración propia*

Figura 55: *Ngrams resultantes*
Fuente: *Elaboración propia*

3.4.3. Tratamiento de oraciones negativas

Para determinar las oraciones negativas se hace uso del modulo ***re***, mediante la cual se identifican las palabras: nunca, no, nada, ningún, ninguno, ninguna y tampoco. Cuando una de estas palabras es identificada en la oración, estas son etiquetadas con el valor 1 y las palabras siguientes son etiquetadas con valores decrecientes a 1. Este recorrido se hace de derecha a izquierda y viceversa para cada *tweet*. Por ejemplo para la oración *'Este tweet no es positivo'* el resultado esperado seria la Figura 56.

```
{'neg_l(Este)': 0.0,
 'neg_l(tweet)': 0.0,
 'neg_l(no)': 1.0,
 'neg_l(es)': 0.9,
 'neg_l(positivo)': 0.8,
 'neg_r(Este)': 0.8,
 'neg_r(tweet)': 0.9,
 'neg_r(no)': 1.0,
 'neg_r(es)': 0.0,
 'neg_r(positivo)': 0.0}
```

Figura 56: *Ejemplo 1 de negación*
Fuente: *Elaboración propia*

Para la oración *'No hay duda de que es el mejor'* el resultado esperado seria la Figura 57.

```
{'neg_l(no)': 1.0,
 'neg_l(hay)': 0.9,
 'neg_l(duda)': 0.8,
 'neg_l(de)': 0.7000000000000001,
 'neg_l(que)': 0.6000000000000001,
 'neg_l(es)': 0.5000000000000001,
 'neg_l(el)': 0.40000000000000013,
 'neg_l(mejor)': 0.30000000000000016,
 'neg_r(no)': 1.0,
 'neg_r(hay)': 0.0,
 'neg_r(duda)': 0.0,
 'neg_r(de)': 0.0,
 'neg_r(que)': 0.0,
 'neg_r(es)': 0.0,
 'neg_r(el)': 0.0,
 'neg_r(mejor)': 0.0}
```

Figura 57: *Ejemplo 2 de negación*
Fuente: *Elaboración propia*

3.5. Implementación de modelos

Para la implementación de los modelos se utiliza los *datasets* creados en la Sección 3.1, se enfatiza que no se incluye evaluación la calidad de los datos debido a que no es parte de los objetivos de este trabajo. En la figura 58 se describe los pasos a seguir para la etapa de evaluación de los modelos seleccionados para este trabajo.

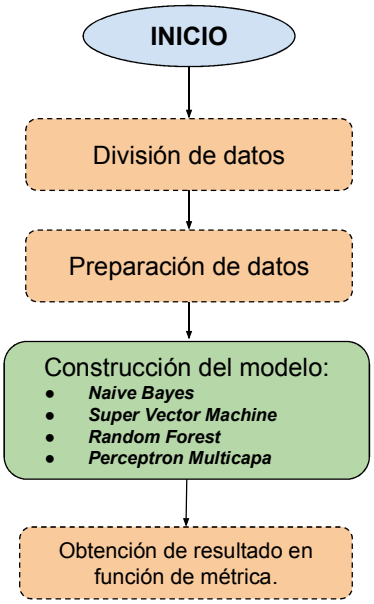

Figura 58: *Diagrama de flujo para la evaluación de modelos*
Fuente: *Elaboración propia*

3.5.1. Partición del *dataset*

Cada modelo necesita ser entrenado y probado con los datos, para ello el *dataset* debe ser dividido en tres grupos: entrenamiento, validación y *test*. Como primer paso, el *dataset* fue particionado bajo la siguiente proporción: 60 % para entrenamiento, 10 % para la validación y 30 % para el testeo. En la Tabla 13 se muestra la distribución solo de *tweets* positivos y negativos para cada grupo, posteriormente también se realizará el análisis incluyendo la categoría de *tweets* neutros.

	TRAIN		**TEST**	
	Positivo	*Negativo*	*Positivo*	*Negativo*
POLÍTICO	1024 (69,66 %)	1647 (70,17 %)	446 (30,34 %)	700 (29,83 %)
PANDEMIA	723 (71,58 %)	1080 (68.92 %)	287 (28,42 %)	487 (31,08 %)

Tabla 13: *Tabla de distribución de tweets para dataset*
Fuente: *Elaboración propia*

3.5.2. Preparación de datos

Todas las funciones definidas en la Sección 3.3 y 3.4 se aplican a cada *tweet*, esto incluye la identificación de *usernames*, *hashtag*, url, signos de puntuación, *steamming*, la generación de los n-grams, negaciones y emojis. Para facilitar este proceso se define la función *extract_features* que recibe como entrada un *tweet* y tiene como resultado una lista de los features extraídos de *tweet* después de aplicar la limpieza como se muestra en la Figura 59.

```
Entrada
('Muy buen articulo.', 'POS')
Salida
({'has(muy)': 1, 'has(buen)': 1, 'has(articul)': 1, 'has(muy,buen)': 1,
'has(buen,articul)': 1, 'has(muy,buen,articul)': 1, 'neg_l(muy)': 0.0,
'neg_l(buen)': 0.0, 'neg_l(articul)': 0.0, 'neg_r(muy)': 0.0,
'neg_r(buen)': 0.0, 'neg_r(articul)': 0.0, 'emoji_(no_emojis)':
'no_emojis'), 'POS')
```

Figura 59: *Input del clasificador*
Fuente: *Elaboración propia*

La función *nltk.classify.apply_features* disponible en la librería *NLTK* es aplicada a cada porcion de entrenamiento, validación y testeo de cada *dataset*. Esta función tiene como datos de entrada una función y un *dataset*; el proceso que se realizó sera aplicar la función *extract_features* a cada elemento del *dataset*, para cada *dataset* anteriormente mencionados; así se obtiene los *datasets* procesados para ser analizados mediante los modelos de clasificación.

3.5.3. Construcción del modelo

Para la evaluación de los *datasets* mediante cada uno de los modelos, se utilizo la librería *NLTK.classify*[22] que incluye los modelos más relevantes para la clasificación de texto. Fueron evaluados cuatro modelos de ML: Naive Bayes, Super Vector Machine, Random Forest y Multilayer perceptron. Cada uno es evaluado mediante cuatro métricas: exactitud, precisión, exhaustividad y F1-score.

Debido a que se posee un *dataset* con las clases balanceadas, se opta por considerar la exactitud para la validación, porque es un análisis a primera vista del conjunto reducido de datos. Los datos de *test* son evaluados mediante la exactitud, precisión, exhaustividad y F1-score, debido a que estos datos deben ser analizados con mayor detalle.

3.5.3.1. Naive Bayes

El algoritmo de NB [23] es instanciado con los parámetros por defecto que trae la librería. A partir de ellos se procede al entrenamiento, validación y *test*. Para conocer a detalle como fueron clasificados los *tweets* de cada categoría, la Figura 60 y 61 muestran la matriz de confusión de cada experimento. Se observa que en ambos *datasets* existe aproximadamente un 10 % de datos que fueron clasificados erróneamente tanto en la categoría de positivos como negativos.

[22] `https://www.nltk.org/api/nltk.classify.html`
[23] `https://www.nltk.org/_modules/nltk/classify/naivebayes.html#NaiveBayesClassifier`

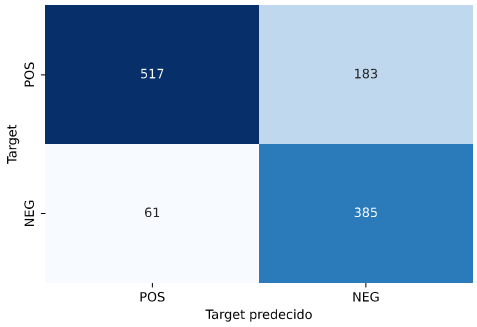

Figura 60: *Matriz de confusión para el dataset político con NB*
Fuente: *Elaboración propia*

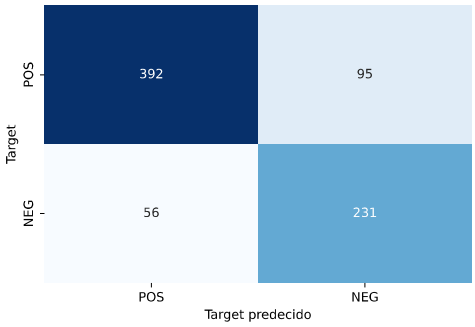

Figura 61: *Matriz de confusión para el dataset pandemia con NB*
Fuente: *Elaboración propia*

Asimismo, la Tabla 14 muestra los resultados para las cuatros métricas propuestas. Se observa que la exactitud obtenida durante la validación es superior a la obtenida durante el *test*, lo que podría estar motivado porque el *train* podría contener instancias de un solo tipo de categoría, ya sea positivo o negativo, logrando que la generalización sea alta al momento de validar. Sin embargo, el conjunto de *test* contendría datos de otras categorías que no fue analizada durante el entrenamiento, lo que genera una exactitud significativamente baja.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
Político	0,887	0,787	0,678	0,863	0,759
Pandemia	0,789	0,805	0,709	0,805	0,754

Tabla 14: *Tabla de resultados basado en las métricas para Naive Bayes*
Fuente: *Elaboración propia*

3.5.3.2. Super Vector Machine

El algoritmo de SVM [24] es instanciado con los parámetros por defecto que trae la librería adicionando el parámetro *LinearSVC()* utilizado para problemas de clasificación. A partir de ello, se procede al entrenamiento, validación y *test*. Para conocer a detalle como fueron clasificados los *tweets* de cada categoría, la Figura 62 y 63 muestran la matriz de confusión de cada experimento. Se observa que en ambos *datasets* existe aproximadamente un 10 % que fueron clasificados erróneamente tanto en la categoría de positivos o negativos.

[24] `https://github.com/scikit-learn/scikit-learn/blob/364c77e04/sklearn/svm/_classes.py#L14`

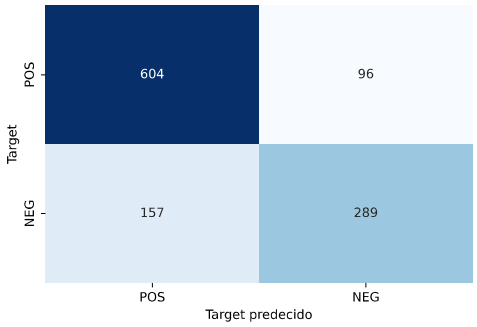

Figura 62: *Matriz de confusión para el dataset político para SVM*
Fuente: *Elaboración propia*

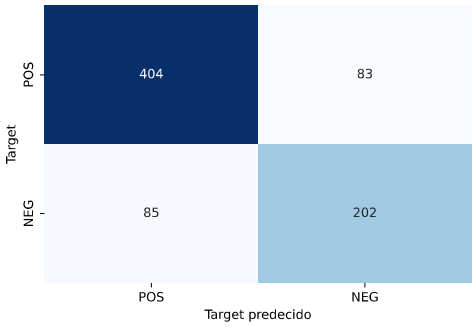

Figura 63: *Matriz de confusión para el dataset pandemia para SVM*
Fuente: *Elaboración propia*

Asimismo, la Tabla 15 muestra los resultados para las cuatros métricas propuestas. Se observa que la exactitud obtenida durante la validación es superior a la obtenida durante el *test*, lo cual podría ser motivado por el desbalance de clases en el *train* y el *test*. En tanto, la exactitud del *test* supera significativamente al obtenido con NB.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
Político	0,869	0,779	0,751	0,648	0,696
Pandemia	0,868	0,783	0,709	0,704	0,706

Tabla 15: *Tabla de resultados basado en las métricas para Super Vector Machine*
Fuente: *Elaboración propia*

3.5.3.3. Random Forest

El algoritmo de RF [25] es instanciado con los parámetros por defecto que trae la librería. A partir de ellos, se procede al entrenamiento, validación y *test*. Para conocer a detalle como fueron clasificados los *tweets* de cada categoría, la Figura 64 y 65 muestran la matriz de confusión de cada experimento. Se observa que en ambos *datasets* existe aproximadamente un 10 % que fueron clasificados erróneamente tanto en la categoría de positivos o negativos.

[25] `https://www.nltk.org/_modules/nltk/classify/decisiontree.html#DecisionTreeClassifier`

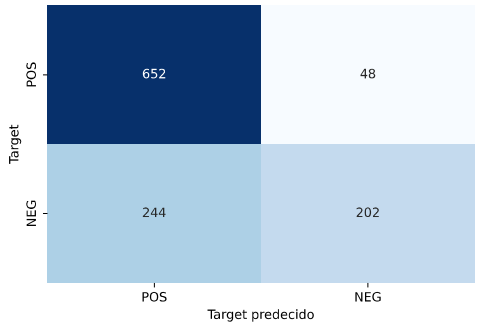

Figura 64: *Matriz de confusión para el dataset político para RF*
Fuente: *Elaboración propia*

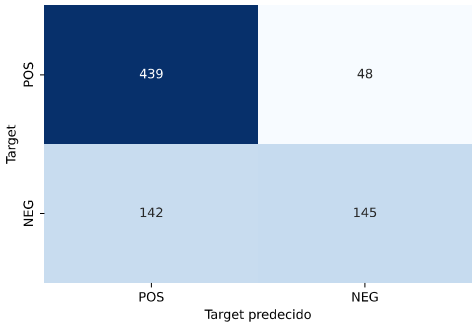

Figura 65: *Matriz de confusión para el dataset pandemia para RF*
Fuente: *Elaboración propia*

Asimismo, la Tabla 16 muestra los resultados para las cuatros métricas propuestas. Se observa que la exactitud obtenida durante la validación es superior a la obtenida durante el *test*, este resultado es similar a lo obtenido con NB y podría ser motivado por el desbalance de clases en el *train* y el *test*. En tanto, la exactitud del *test* para este modelo fue menos que los obtenidos para NB y SVM.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
Político	0,866	0,745	0,808	0,453	0,581
Pandemia	0,841	0,755	0,751	0,505	0,604

Tabla 16: *Tabla de resultados basado en las métricas para Random Forest*
Fuente: *Elaboración propia*

3.5.3.4. Perceptrón Multicapa

El algoritmo de MLP [26] es instanciado con los parámetros que muestra en la Figura 66. A partir de ellos, se procede al entrenamiento, validación y *test*. Para conocer a detalle como fueron clasificados los *tweets* de cada categoría, la Figura 67 y 68 muestran la matriz de confusión de cada experimento. Se observa que en ambos *datasets* existe aproximadamente un 10 % que fueron clasificados erróneamente tanto en la categoría de positivos o negativos.

```
perceptron_classifier = nltk.classify.SklearnClassifier(MLPClassifier(activation='relu', solver='adam', alpha=1e-5,
                        hidden_layer_sizes=(50,50), random_state=1,
                        learning_rate_init=0.001,max_iter=500,early_stopping=True))
```

Figura 66: *Configuración de parámetros para MLP*
Fuente: *Elaboración propia*

[26] https://github.com/scikit-learn/scikit-learn/blob/364c77e04/sklearn/neural_network/_multilayer_perceptron.py#L761

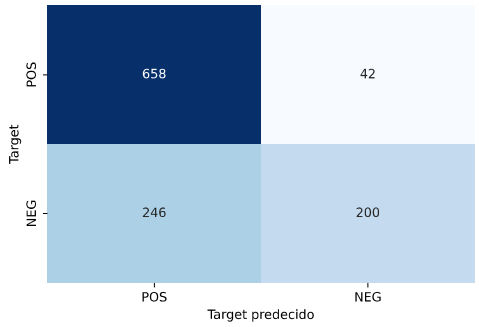

Figura 67: *Matriz de confusión para el dataset político para MLP*
Fuente: *Elaboración propia*

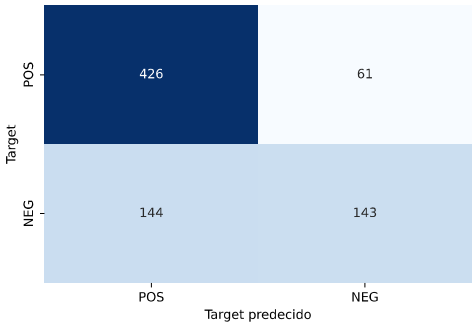

Figura 68: *Matriz de confusión para el dataset pandemia para MLP*
Fuente: *Elaboración propia*

Asimismo, la Tabla 17 muestra los resultados para las cuatros métricas propuestas. Se observa que la exactitud obtenida durante la validación es superior a la obtenida durante el *test*, este resultado es similar a lo obtenido con NB y podría ser motivado por el desbalance de clases en el *train* y el *test*. En tanto, la exactitud obtenida con MLP fue la mejor obtenida en comparación a los otros modelos con ambos *datasets*.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
Político	0,859	0,749	0,826	0,448	0,581
Pandemia	0,833	0,735	0,701	0,498	0,582

Tabla 17: *Tabla de resultados basado en las métricas para Multilayer perceptron*
Fuente: *Elaboración propia*

Capítulo 4

Desarrollo de los experimentos

4.1. Especificaciones Técnicas

Para el desarrollo de esta tesis se uso el siguiente hardware y software que son detallados a continuación:

- **Recursos Hardware**
  1. Notebook: VivoBook_ASUS Laptop, Procesador Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz (8 CPUs), 2.0GHz, 8 GB, 256GB SSD, NVIDIA GeForce MX110.
  2. Notebook: Rog Zephyrus M15 ASUS Laptop. Procesador Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 2.59 GHz, 16.0 GB, 500GB SSD, NVIDIA GeForce GTX 1660Ti.
  3. Server: ASUS_TEK, Procesador Intel Xeon Gold 6134 3700MHz, 342.99 GB, 2.05 TiB, NVIDIA GP100/PCIe/SSE2 vs 4.6.0 NVIDIA 470.82.00.
- **Recursos Software**
  1. **Sistema Operativo:** Windows 10 x64 y Ubuntu 20.04.3
  2. **Lenguaje de programación:** Python 3.10.4
  3. **Sistema de gestion de paquetes:** Anaconda 2021.11
  4. **Librerías:** Numpy (1.16.4), scipy (1.2.2), matplotlib, aiohttp (3.8.1), Twint, tensorflow (1.14.0), tensorflow-gpu (1.14.0), keras (2.1.5), pysentimiento (0.3.2), sklearn,nltk(3.6.5) cudatoolkit (11.3.1),
  5. **Sistema de control de versiones:** GitHub

4.2. Descripción de los experimentos

En la Sección 3.1 fueron obtenidos los primeros resultados para cada modelo seleccionado, sin embargo se considero optimizarlos mediante una serie de experimentos presentados en la Figura 69. Los parámetros para cada experimento son escogidos basados en el experimento previo. Los resultados finales fueron obtenidos del Experimento 4, que dio lugar a la comparación entre la estratificación o *cross validation*.

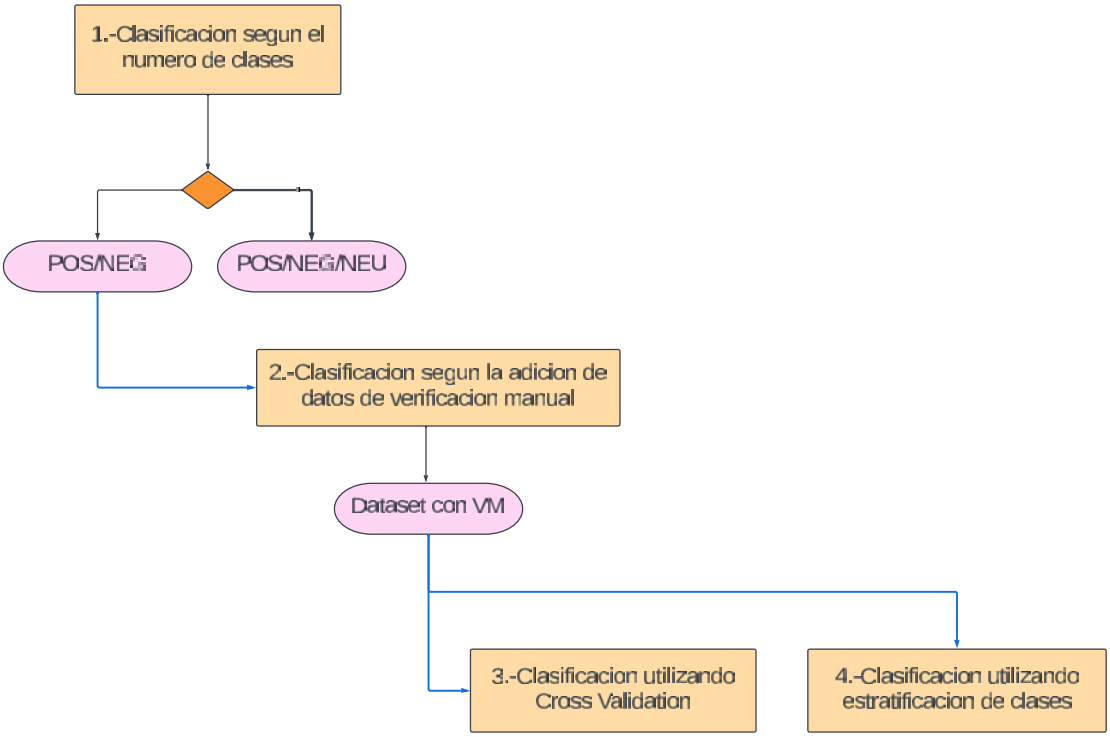

Figura 69: *Flujo de trabajo para la fase de experimentos*
Fuente: *Elaboración propia*

4.3. Experimento 01: Clasificación según el número de clases

Este experimento busca hacer una comparación en función del número de clases: POS/ NEG vs POS/ NEG/NEU). Este experimento esta sujeto a los siguientes parámetros:

- **Objetivo:** Comparar los resultados con dos y tres clases.
- **Clases:** POS/NEG vs POS/NEG/ NEU
- **Proporción train/test:** 80/20.
- **Estratificación:** NO.
- **Modelos:** NB, SVM, RF y MLP.
- **Datasets:** Político y pandemia.
- **Métricas:** Exactitud, precisión, exhaustividad y F1-score.

A continuación se describen los resultados obtenidos para cada escenario de acuerdo al número de clases:

- *Para político:* Las Tablas 18 y 19 muestran las métricas obtenidas para cada experimento utilizando dos y tres clases respectivamente. En general, se observa que para el experimento que considera POS/NEG/NEU, los valores de las métricas obtenidas son menores a los obtenidos con POS/NEG, lo cual indica que los modelos no alcanzaron a clasificar *tweets* neutros de manera

óptima, esto también se debe a que los *tweets* de tipo neutro presentaron dificultades en la fase de etiquetado durante la construcción del *dataset*.

De la Tabla 18, se observa que la exactitud para la validación son superiores a la exactitud obtenida con datos del test para tres modelos (SVM, RF, MLP), esto corresponde a que las particiones no están proporcionadas de acuerdo al número de instancias. Por otro lado, MLP es el modelo con mayor exactitud (0,805), RF con mayor precisión(0,843) y NB con mayor exhaustividad (0,864) y F1-score (0,765). Mediante estos resultados, se observa que a pesar de que NB no posee el mayor valor de exactitud, el F1-score indica una mejor generalización en términos de clasificación para las clases POS y NEG.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0,782	0,791	0,686	0,864	0,765
SVM	0,822	0,766	0,750	0,608	0,672
RF	0,777	0,749	0,843	0,445	0,583
MLP	0,830	0,805	0,830	0,635	0,719

Tabla 18: *Resultados para el escenario político en base al análisis de las clases POS/NEG*

Fuente: *Elaboración propia*

De la Tabla 19, se observa que la exactitud para la validación son superiores a la exactitud obtenida con datos del test para dos modelos (NB, RF), esto corresponde a que las particiones no están proporcionadas de acuerdo al número de instancias. Por otro lado, MLP es el modelo con mayor exactitud (0,734) y exhaustividad (0.689), y NB con mayor precisión (0,878) y F1-score (0,533). Mediante estos resultados, se observa que a pesar de que NB no posee el mayor valor de exactitud, el F1-score indica una mejor generalización en términos de clasificación para las clases POS/NEG/NEU.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0,585	0,404	0,878	0,404	0,533
SVM	0,588	0,732	0,426	0,632	0,438
RF	0,598	0,440	0,397	0,582	0,312
MLP	0,628	0,734	0,440	0,689	0,459

Tabla 19: *Resultados para el escenario político en base al análisis de las clases POS/NEG/NEU*

Fuente: *Elaboración propia*

- *Para pandemia:* Las Tablas 20 y 21 muestran las métricas obtenidas para cada experimento utilizando dos y tres clases respectivamente. En general, se observa que para el experimento que considera POS/NEG/NEU, los valores de las métricas obtenidas son menores a los obtenidos con POS/NEG, lo que indica que los modelos no alcanzaron a clasificar *tweets* neutros de

manera óptima. De manera similar, al escenario político, esta situación se justifica porque los *tweets* de tipo neutro presentaron dificultades en la fase de etiquetado durante la construcción del *dataset*.

De la Tabla 20, se observa que la exactitud para la validación son superiores a la exactitud obtenida con datos del test para los cuatro modelos, esto corresponde a que las particiones no están proporcionadas de acuerdo al número de instancias. Por otro lado, MLP y NB son los modelos con mayor exactitud (0,789), RF con mayor precisión(0,888) y NB con mayor exhaustividad (0,795) y F1-score (0,770). Mediante estos resultados, se observa que a pesar de que NB no posee el mayor valor de exactitud, el F1-score indica una mejor generalización en términos de clasificación para las clases POS y NEG.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0,845	0,789	0,746	0,795	0,770
SVM	0,784	0,777	0,788	0,681	0,731
RF	0,749	0,717	0,888	0,415	0,565
MLP	0,868	0,789	0,813	0,681	0,741

Tabla 20: *Resultados para el escenario pandemia en base al análisis de las clases POS/NEG*

Fuente: *Elaboración propia*

De la Tabla 19, se observa que MLP es el modelo con mayor exactitud (0,682), precisión (0,695), exhaustividad (0,682)y F1-score (0,686). Mediante estos resultados, se observa que a pesar de que MLP indica una mejor generalización en términos de clasificación para las clases POS/NEG/NEU.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0,653	0,658	0,661	0,658	0,655
SVM	0,644	0,634	0,674	0,634	0,642
RF	0,528	0,529	0,668	0,529	0,532
MLP	0,657	0,682	0,695	0,682	0,686

Tabla 21: *Resultados para el escenario pandemia en base al análisis de las clases POS/NEG/NEU*

Fuente: *Elaboración propia*

En general, los resultados son superiores para el trabajo con las clases POS/NEG para los *datasets* construidos, por consiguiente este parámetro sera utilizado en los próximos experimentos.

4.4. Experimento 02: Clasificación incluyendo datos de la verificación manual

Para este experimento se analizan los *datasets* incluyendo los *tweets* que fueron verificados manualmente como POS/NEG.

- **Objetivo:** Verificar el impacto de incluir *tweets* verificados manualmente.
- **Clases:** POS/NEG
- **Prop. train/test:** 80/20.
- **Estratificación:** NO.
- **Modelos:** NB,SVM,RF,MLP.
- **Datasets:** Político y pandemia (incluyendo datos de la verificación manual).
- **Métricas:** Exactitud, precisión, exhaustividad y F1-score.

A continuación se describen los resultados para cada escenario:

- *Para político:* Se observa que los resultados superan los obtenidos en la Tabla 18. En tanto MLP es el modelo que obtiene la mayor exactitud (0,878) en la validación, mientras que para los datos de test, MLP obtiene la mayor exactitud (0,815), RF obtiene la mayor precisión (0,874) y NB obtiene la mayor exhaustividad (0,872) y F1-score (0,818). También se enfatiza que NB no posee la mayor exactitud, sin embargo el valor de F1-score indica una mejor generalización para clasificar *tweets* positivos y negativos, a diferencia de MLP que posee una mayor exactitud, sin embargo el F1-score no supera a la de NB.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0.841	0.809	0.771	0.872	0.818
SVM	0.859	0.773	0.833	0.673	0.745
RF	0.813	0.734	0.874	0.539	0.667
MLP	0.878	0.815	0.849	0.759	0.802

Tabla 22: *Resultados obtenidos en el dataset político con VM considerando dos clases*
Fuente: *Elaboración propia*

- *Para pandemia:* Se observa que los resultados superan los obtenidos en la Tabla 20. En tanto MLP es el modelo que obtiene la mayor exactitud (0,860) en la validación, mientras que para los datos de test, NB obtiene la mayor exactitud (0,800), exhaustividad (0,839) y F1-score (0,815), y RF obtiene la mayor precisión (0,942). Para este escenario, NB parece ser el mejor modelo para la clasificación de *tweets* hasta el momento.

	Validación	Test			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0.860	0.800	0.791	0.839	0.815
SVM	0.799	0.798	0.861	0.732	0.791
RF	0.764	0.735	0.942	0.526	0.675
MLP	0.831	0.789	0.911	0.663	0.767

Tabla 23: *Resultados obtenidos en el dataset pandemia con VM considerando dos clases*
Fuente: *Elaboración propia*

En general, los resultados son superiores para el trabajo cuando se incluyen los datos que fueron verificados manualmente en cada uno de los *datasets* construidos, por consiguiente este parámetro sera utilizado en los próximos experimentos.

4.5. Experimento 03: Clasificación utilizando *Cross Validation*

Para este experimento se incluyo la técnica de *cross validation*, que sirve para evaluar varios modelos de *Machine Learning* mediante el entrenamiento de diferentes subconjuntos de *train* y *test*, como se muestra en la Figura 70.

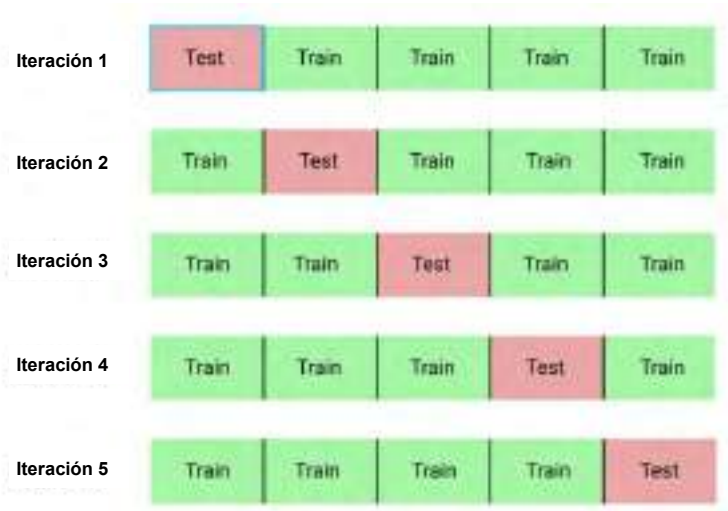

Figura 70: *Estratificación de dataset con 5-folds*
Fuente: *Elaboración propia*

Los resultados serán comparados con las Tablas 22 y 23, considerando los siguientes parámetros:

- **Objetivo:** Verificar el impacto de la técnica de *cross validation* en cada modelo.
- **Clases:** POS/NEG.
- **Modelos:** NB, SVM, RF, MLP.
- ***Datasets*:** Político y Pandemia.

- **Métricas:** Exactitud, precisión, exhaustividad y F1-score.
- ***Folds*:** 20 *folds*.

A continuación se describen los resultados obtenidos para cada escenario:

- *Para político:* Los resultados que se visualizan en la Tabla 26 son comparados con los resultados obtenidos en la Tabla 24. Para la validación, superan a los obtenidos anteriormente. Para el test, se muestra que los valores de exactitud mejoran al utilizar la técnica de *cross validation*, sin embargo para el resto de métricas ocurre lo contrario, ya que solo algunos modelos presentan mejoras. Esto se debe a que los modelos no consiguen clasificar nuevas instancias del conjunto de *train*, ya que el entrenamiento no se estaría realizando con la suficiente variedad de datos para ambas clases.

  Se observa que para la validación, que MLP posee la mayor exactitud (0,873), mientras que para el test, NB posee la mayor exactitud (0,870), RF la mayor precisión (0,815), NB la mayor exhaustividad (0,882) y F1-score (0,813). Una vez más se comprueba que NB es el mejor modelo de clasificación para este escenario.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0.781	0,870	0.763	0.882	0.813
SVM	0.848	0.842	0.772	0.754	0.759
RF	0.823	0.818	0.815	0.625	0.704
MLP	0.873	0.864	0.802	0.790	0.793

Tabla 24: *Resultados del análisis con 20-folds para el dataset político*
Fuente: *Elaboración propia*

- *Para pandemia:* Los resultados que se visualizan en la Tabla 27 son comparados con los resultados obtenidos en la Tabla 25. Para la validación, superan a los obtenidos anteriormente. Para el test, se muestra que una mejora significativa para los valores de las métricas en casi todos los modelos al utilizar la técnica de *cross validation*.

  Se observa que para la validación, MLP posee la mayor exactitud (0,867), mientras que para el test, se observa que MLP posee la mayor exactitud (0,840), RF la mayor precisión (0,814), NB la mayor exhaustividad (0,823) y F1-score (0,771). Bajo estos resultados, tanto NB como MLP podrían ser considerando como óptimos clasificadores para este escenario.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0.835	0,831	0.735	0.823	0.771
SVM	0.830	0.827	0.777	0.733	0.750
RF	0.814	0.796	0.814	0.586	0.674
MLP	0.867	0.840	0.799	0.732	0.759

Tabla 25: *Resultados 20-folds para el dataset pandemia*
Fuente: *Elaboración propia*

4.6. Experimento 04: Clasificación utilizando estratificación de clases

Para este experimento fue configurado el parámetro `stratify` de la función `train_test_split` para obtener el *train* y *test* con el mismo número de instancias para cada sentimiento.

Partiendo de los resultados obtenidos del experimento de la Sección 4.3 y 4.4, este experimento aplicara la estratificación de clases para las clases POS/NEG con los *datasets* que incluyen datos de verificación manual. Se consideran los siguientes parámetros:

- **Objetivo:** Verificar el impacto en los resultados al usar estratificación de clases.
- **Clases:** POS/NEG.
- **Prop. train/test** 80/20.
- **Estratificación:** SI.
- **Modelos:** NB, SVM, RF y MLP.
- ***Datasets*:** Político y Pandemia
- **Métricas:** Exactitud, precisión, exhaustividad y F1-score.

A continuación se describen los resultados obtenidos con estratificación:

- *Para político:* Los resultados que se visualizan en la Tabla 26 son comparados con los resultados obtenidos en la Tabla 24. Para la validación, los valores de exactitud superan a los obtenidos anteriormente, al contrario de las métricas obtenidas por los datos del test, que muestran una mejora significativa. Esto se debe a que los datos de *train* y *test* son estratificados, es decir, las instancias de cada clase son distribuidas proporcionalmente en cada conjunto.

  Se observa que para la validación, MLP es el modelo con mayor exactitud (0,859), mientras que para los datos del test, NB posee la mayor exactitud (0,884), exhaustividad (0,898) y F1-score (0,865), y MLP la mayor precisión (0,905). Nuevamente, NB es el mejor modelo de clasificación para este escenario.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0.848	0.884	0.834	0.898	0.865
SVM	0.831	0.847	0.851	0.762	0.804
RF	0.806	0.804	0.884	0.606	0.719
MLP	0.859	0.877	0.905	0.785	0.841

Tabla 26: *Resultados obtenidos para el dataset con VM considerando estratificación*
Fuente: *Elaboración propia*

- *Para pandemia:* Los resultados que se visualizan en la Tabla 27 son comparados con los resultados obtenidos en la Tabla 25. Para la validación, los valores de exactitud superan a los obtenidos anteriormente para todos los modelos, al contrario de las métricas obtenidas por los datos del test, en las que la exactitud y F1-score muestran una mejoras significativa. Esto se debe a que los datos de *train* y *test* son estratificados, es decir, las instancias de cada clase son distribuidas proporcionalmente en cada conjunto.

  Se observa que para la validación, NB es el modelo con mayor exactitud (0,861), mientras que para los datos del test, SVM posee la mayor exactitud (0,850) y F1-score (0,817), posee la mayor exhaustividad (0,844), y RF posee la mayor precisión (0,906). Para este escenario SVM es elegido como el mejor clasificador para este escenario.

	Validación	**Test**			
	Exactitud	Exactitud	Precisión	Exhaustividad	F1-score
NB	0.861	0.818	0.760	0.844	0.800
SVM	0.812	0.850	0.861	0.778	0.817
RF	0.822	0.805	0.906	0.611	0.730
MLP	0.844	0.847	0.845	0.788	0.816

Tabla 27: *Resultados obtenidos para el dataset pandemia con VM considerando estratificación*

Fuente: *Elaboración propia*

Finalmente los resultados demuestran que cada modelo refleja la importancia de trabajar con un *train* y *test* estratificado.

Capítulo 5

Resultados e interpretaciones

5.1. Experimento 01: Clasificación según el número de clases

Los valores de las cuatro métricas obtenidas por todos los modelos al trabajar con las clases POS, NEG y NEU no supera los resultados obtenidos por POS y NEG. Esto se debe a que los *tweets* etiquetados como 'neutro' de acuerdo al score (Pysentimiento) presentaron inconsistencias, de este modo seleccionar el score con mayor valor no es un buen indicador. Además en su mayoría, los *tweets* de tipo 'neutro' de esta categoría suelen ser ambiguos y pueden compartir el vocabulario de las clases POS Y NEG. Finalmente para este primer experimento, se observa que para POS/NEG los mejores clasificadores para el escenario político podrían ser: (1) NB con 79,1 % de exactitud y 76,5 % de F1-score, o (2) MLP con 80,5 % de exactitud y 71,9 % de F1-score. Para el escenario pandemia, el mejor clasificador seria NB con 78,9 % de exactitud y 77 % de F1-score.

De este experimento se infiere que los *tweets* de categoría NEU perjudican los resultados, por lo tanto es recomendable trabajar con POS y NEG. También se infiere que durante este primer experimento, NB podría ser considerado el mejor modelo para ambos escenarios.

5.2. Experimento 02: Clasificación incluyendo datos de verificación manual

Los resultados para las cuatro métricas de este experimento reflejan mejora cuando los datos de verificación manual son incluidos para ambos escenarios, por lo que se evidencia que ningún modelo de etiquetado automático es perfecto y que es necesario contar con una fase de verificación.

Para el escenario político, se evidencia que MLP es el modelo con mayor número de instancias clasificadas correctamente, sin embargo NB es el modelo que posee mayor acierto al clasificar cada instancia en su categoría. Para el escenario pandemia, se evidencia que NB es el modelo con mayor número de instancias clasificadas correctamente y también posee mayor acierto al clasificar cada instancia en su categoría correspondiente

De este experimento se infiere que al construir un *dataset*, que emplee

etiquetado automático, requiere de una fase de verificación manual. También se infiere que la exactitud no siempre refleja que la clasificación se este efectuando correctamente, por lo cual es necesario observar las otras métricas que pueden dar otra perspectiva de como se clasifican las instancias en cada categoría.

5.3. Experimento 03: Clasificación utilizando *Cross Validation*

Los resultados de las cuatro métricas mejoran cuando se utiliza *Cross Validation* para obtener las instancias del entrenamiento y del test. En ambos escenarios la exactitud obtenida de la validación es mayor a la exactitud obtenida para el test, debido a que el test contendría instancias nuevas que los modelos no consiguen clasificar correctamente.

Para el escenario político, se evidencia que NB es el modelo con mayor número de instancias clasificadas correctamente y también posee mayor acierto al clasificar cada instancia en su categoría correspondiente. Para el escenario pandemia, se evidencia que MLP es el modelo con mayor número de instancias clasificadas correctamente, sin embargo NB es el modelo que posee mayor acierto al clasificar cada instancia en su categoría correspondiente.

De este experimento se infiere que crear diversos modelos con diferentes conjuntos de datos en *Cross Validation* mejora los resultados. También es importante observar las otras métricas que pueden dar otra perspectiva de como se clasifican las instancias en cada categoría. También se infiere nuevamente que la exactitud no siempre refleja que la clasificación se este efectuando correctamente, por lo cual es necesario observar las otras métricas, que dan otra perspectiva de como se clasifican las instancias en cada categoría.

5.4. Experimento 04: Clasificación según la estratificación de clases

Los resultados de las cuatro métricas mejoran cuando se utiliza la estratificación para el conjunto de datos de entrenamiento y de test. En ambos escenarios, la mayoría de modelos obtienen un mayor valor de exactitud para el test, debido a que la estratificación de clases permite a los modelos aprender correctamente durante el entrenamiento, disminuyendo las instancias desconocidas durante test.

Para el escenario político, se evidencia que NB es el modelo con mayor número de instancias clasificadas correctamente y también posee mayor acierto al clasificar cada instancia en su categoría correspondiente. Para el escenario pandemia, se evidencia que SVM es el modelo con mayor número de instancias clasificadas correctamente y también posee mayor acierto al clasificar cada instancia en su categoría correspondiente.

De este experimento se infiere que crear diversos modelos con los conjuntos de entrenamiento y test estratificados mejora los resultados. Finalmente se infiere para el escenario político, NB es el mejor clasificador, mientras que para el escenario pandemia, SVM es considerado el mejor clasificador.

Conclusiones

1. Las conclusión en base al objetivo general es:
   Se logro implementar dos *datasets* con clases balanceadas que contienen *tweets* Positivo, Negativo y Neutro. Estos contienen 7205 *tweets* para el escenario político y 5157 para el escenario pandemia con la distribución que se visualiza en la Tabla 10.
2. Las conclusiones con base al primer objetivo especifico son:
   La metodología propuesta para la construcción del *dataset* incluye la extracción de *tweets* mediante Twint, el filtrado y la limpieza de datos, el etiquetado automático de *tweets* mediante ***Pysentimiento***, el balanceo de clases y finalmente la verificación manual de *tweets*, como se describe en la Sección 3.1.
3. Las conclusiones con base al segundo objetivo especifico son:
   Se utilizo una metodología general para el análisis de sentimientos. Esta metodología contempla todos los pasos necesarios para este análisis, desde el pre-procesamiento como se describe en la Sección 3.3, la transformación de datos (*feature engineering*) como se describe en la Sección 3.4, la construcción y evaluación de modelos de clasificación como se describe en la Sección 3.5.
4. Las conclusiones con base al tercer objetivo especifico son:
   Se aplico la tokenizacion durante la limpieza de datos y exploración de datos, *Stemming* para la etapa de pre-procesamiento y N-grams para la etapa de *Feature Engineering.*
5. La conclusión con base al cuarto objetivo especifico es:
   Se ejecuto cuatro modelos de clasificación: Naive Bayes, Super Vector Machine, Random Forest y Multilayer perceptron, basado en la obtención del mejor conjunto de entrenamiento. De acuerdo a los resultados, se concluye que NB es el mejor clasificador con 88,4 % de exactitud, 83,4 % de precisión, 89,8 % de exhaustividad y 86,5 % de F1-score como se muestra en la Tabla 26, mientras que para el escenario pandemia SVM es considerado como el mejor clasificador con 85 % de exactitud, 86,1 % de precisión, 77,8 % de exhaustividad y 81,7 % de F1-score como se muestra en la Tabla 27.
6. La conclusión con base al quinto objetivo especifico es: logro evaluar el desempeño de los modelos de aprendizaje supervisado para el análisis de sentimientos con *datasets* etiquetados mediante ***Pysentimiento***, donde el análisis de las clases positiva y negativa mostraron mejores resultados, a diferencia de la clase neutra.
7. La conclusión con base al sexto objetivo especifico es:
   Se determino los mejores criterios para obtener un conjunto de datos que

provea mayor conocimiento a los modelos de clasificación durante el entrenamiento: utilizar dos clases (POS/NEG) como se muestra en el Experimento 4.3 y la estratificación de clases como se muestra en el Experimento 4.6.

8. La conclusión con base al séptimo objetivo especifico es:
   Se analizó que los datos con verificación manual mejora los resultados para la construcción de los modelos para ambos escenarios como se muestra en el Experimento 4.4, de este modo se concluye siempre es necesario que el *dataset* pase por una fase de verificación manual para comprobar el etiquetado.

Recomendaciones

Las recomendaciones son:

1. Contar con una lista extensa de hashtag y usernames para realizar web scraping en Twitter, así se conseguirá mayor número de tweets recuperados, y después de una depuración de repetidos e irrelevantes, aun quedara una cantidad aceptable de tweets para el dataset.
2. De ser el caso de usar el API de Twitter para la obtención de datos, escoger temas relevantes en el momento actual, para obtener mayor cantidad de tweets relacionados al tema elegido.
3. Definir de antemano el nivel de análisis textual que se considerara para el etiquetado de los tweets (Documento, oración o entidad y aspecto).
4. Verificar que el dataset contenga cantidades proporcionadas de las clases que se quieran analizar.
5. Verificar las características de los datos obtenidos para elegir las técnicas de pre procesamiento que se utilizaran acorde a las necesidades observadas.

Trabajos Futuros

Se deja como trabajo futuro:

1. Utilizar métricas que permitan evaluar la pureza de los datos en el dataset.
2. Utilizar otras herramientas de etiquetado automático que puedan establecer más de tres categorías para el análisis de sentimientos, por ejemplo por medio de la clusterizacion.
3. Utilizar otra herramienta de etiquetado automático que tenga mejor rendimiento al etiquetar tweets neutros.
4. Evaluar otros modelos de Machine Learning o Deep Learning que podrían optimizar aun más los resultados.
5. Implementar el prototipo de un modelo para la generación automática de texto basado en los datasets implementados.

Bibliografía

Ahuja, S. and Dubey, G. (2017). Clustering and sentiment analysis on twitter data. In *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*, pages 1–5. IEEE.

Antonakaki, D., Fragopoulou, P., and Ioannidis, S. (2021). A survey of twitter research: Data model, graph structure, sentiment analysis and attacks. *Expert Systems with Applications*, 164:114006.

Balakrishnan, V. and Lloyd-Yemoh, E. (2014). Stemming and lemmatization: A comparison of retrieval performances.

Becker, K. and Tumitan, D. (2013). Introdução à mineração de opiniões: Conceitos, aplicações e desafios. *Simpósio brasileiro de banco de dados*, 75.

Berrar, D. (2019). Cross-validation.

Browne, M. W. (2000). Cross-validation methods. *Journal of mathematical psychology*, 44(1):108–132.

Buckland, M. and Gey, F. (1994). The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19.

Cambria, E., Das, D., Bandyopadhyay, S., Feraco, A., et al. (2017). *A practical guide to sentiment analysis*. Springer.

Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832.

Cerón-Guzmán, J. A. and León-Guzmán, E. (2016). A sentiment analysis system of spanish tweets and its application in colombia 2014 presidential election. In *2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (socialcom), sustainable computing and communications (sustaincom)(BDCloud-socialcom-sustaincom)*, pages 250–257. IEEE.

Chaudhuri, A. (2006). *Emotion and reason in consumer behavior*. Routledge.

Dangi, D., Dixit, D. K., and Bhagat, A. (2022). Sentiment analysis of covid-19 social media data through machine learning. *Multimedia Tools and Applications*, 81(29):42261–42283.

Denny, M. J. and Spirling, A. (2018). Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*, 26(2):168–189.

Famili, A., Shen, W.-M., Weber, R., and Simoudis, E. (1997). Data preprocessing and intelligent data analysis. *Intelligent data analysis*, 1(1):3–23.

Giachanou, A. and Crestani, F. (2016). Like it or not: A survey of twitter sentiment analysis methods. *ACM Computing Surveys (CSUR)*, 49(2):1–41.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Grandini, M., Bagli, E., and Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.

Greenwade, G. D. (1993). The Comprehensive Tex Archive Network (CTAN). *TUGBoat*, 14(3):342–351.

Gujjar, J. P. and HR, P. K. (2021). Sentiment analysis: Textblob for decision making. *Int. J. Sci. Res. Eng. Trends*, 7(2):1097–1099.

Hernández, C., Rodríguez, J. E. R., et al. (2008). Preprocesamiento de datos estructurados. *Revista vínculos*, 4(2):27–48.

Hernández-Sampieri, R. and Mendoza, C. (2020). *Metodología de la investigación: las rutas cuantitativa, cualitativa y mixta*. Mcgraw-hill.

Hossin, M. and Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1.

Hu, X., Tang, J., Gao, H., and Liu, H. (2013). Unsupervised sentiment analysis with emotional signals. In *Proceedings of the 22nd international conference on World Wide Web*, pages 607–618.

Jing, L., Tian, K., and Huang, J. Z. (2015). Stratified feature sampling method for ensemble clustering of high dimensional data. *Pattern Recognition*, 48(11):3688–3702.

Joshi, M. V. (2002). On evaluating performance of classifiers for rare classes. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 641–644. IEEE.

Juba, B. and Le, H. S. (2019). Precision-recall versus accuracy and the role of large data sets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4039–4048.

Kanakaraj, M. and Guddeti, R. M. R. (2015). Nlp based sentiment analysis on twitter data using ensemble classifiers. In *2015 3Rd international conference on signal processing, communication and networking (ICSCN)*, pages 1–5. IEEE.

Kausar, M. A., Soosaimanickam, A., and Nasar, M. (2021). Public sentiment analysis on twitter data during covid-19 outbreak. *International Journal of Advanced Computer Science and Applications*, 12(2).

Keselj, V. (2009). Speech and language processing daniel jurafsky and james h. martin (stanford university and university of colorado at boulder) pearson prentice hall, 2009, xxxi+ 988 pp; hardbound, isbn 978-0-13-187321-6.

Koehrsen, W. (2020). Random forest simple explanation.

Krömer, P., Platoš, J., Snášel, V., and Abraham, A. (2011). Fuzzy classification by evolutionary algorithms. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pages 313–318. IEEE.

Krstinić, D., Braović, M., Šerić, L., and Božić-Štulić, D. (2020). Multi-label classifier performance evaluation with confusion matrix. *Comput Sci Inf Technol*, 10:1–14.

Li, B. and Han, L. (2013). Distance weighted cosine similarity measure for text classification. In *International conference on intelligent data engineering and automated learning*, pages 611–618. Springer.

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Liu, P., Li, W., and Zou, L. (2019). Nuli at semeval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *SemEval@ NAACL-HLT*, pages 87–91.

Lok, E. J. (2017). Episode 1: Using tf-idf to identify the signal from the noise.

López Condori, J. J. and Gonzales Saji, F. O. (2021). Análisis de sentimiento de comentarios en español en google play store usando bert. *Ingeniare. Revista chilena de ingeniería*, 29(3):557–563.

McCann, S. and Lowe, D. G. (2012). Local naive bayes nearest neighbor for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3650–3656. IEEE.

Mittal, A. and Patidar, S. (2019). Sentiment analysis on twitter data: A survey. In *Proceedings of the 2019 7th International Conference on Computer and Communications Management*, pages 91–95.

Mujahid, M., Lee, E., Rustam, F., Washington, P. B., Ullah, S., Reshi, A. A., and Ashraf, I. (2021). Sentiment analysis and topic modeling on tweets about online education during covid-19. *Applied Sciences*, 11(18):8438.

Müller, M., Salathé, M., and Kummervold, P. E. (2020). Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. *arXiv preprint arXiv:2005.07503*.

Munappy, A., Bosch, J., Olsson, H. H., Arpteg, A., and Brinne, B. (2019). Data management challenges for deep learning. In *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 140–147. IEEE.

Nakov, P., Rosenthal, S., Kiritchenko, S., Mohammad, S. M., Kozareva, Z., Ritter, A., Stoyanov, V., and Zhu, X. (2016). Developing a successful semeval task in sentiment analysis of twitter and other social media texts. *Language Resources and Evaluation*, 50(1):35–65.

O'Glasser, A. Y., Jaffe, R. C., and Brooks, M. (2020). To tweet or not to tweet, that is the question. In *Seminars in nephrology*, volume 40, pages 249–263. Elsevier.

Parsons, V. L. (2014). Stratified sampling. *Wiley StatsRef: Statistics Reference Online*, pages 1–11.

Pérez, J. M., Giudici, J. C., and Luque, F. (2021). pysentimiento: A python toolkit for sentiment analysis and socialnlp tasks. *arXiv preprint arXiv:2106.09462*.

Porter, M. F. (2001). Snowball: A language for stemming algorithms.

Pota, M., Ventura, M., Fujita, H., and Esposito, M. (2021). Multilingual evaluation of pre-processing for bert-based sentiment analysis of tweets. *Expert Systems with Applications*, 181:115119.

Rahutomo, F., Kitasuka, T., and Aritsugi, M. (2012). Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1.

Rajput, A. (2020). Natural language processing, sentiment analysis, and clinical analytics. In *Innovation in Health Informatics*, pages 79–97. Elsevier.

Ramchoun, H., Ghanou, Y., Ettaouil, M., and Janati Idrissi, M. A. (2016). Multilayer perceptron: Architecture optimization and training.

Rish, I. et al. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

Saif, H., He, Y., Fernandez, M., and Alani, H. (2016). Contextual semantics for sentiment analysis of twitter. *Information Processing & Management*, 52(1):5–19.

Sandoval-Almazán, R. and Valle-Cruz, D. (2016). Understanding network links in twitter: A mexican case study. In *Proceedings of the 17th International Digital Government Research Conference on Digital Government Research*, pages 122–128.

Swaminathan, R. T., Balaji, V., and Subramanian, S. Chennai floods 2021: Sentiment analysis of twitter data using tweepy and textblob.

Tatbul, N., Lee, T. J., Zdonik, S., Alam, M., and Gottschlich, J. (2018). Precision and recall for time series. *Advances in neural information processing systems*, 31.

Tejani, A. S., Ng, Y. S., Xi, Y., Fielding, J. R., Browning, T. G., and Rayan, J. C. (2022). Performance of multiple pretrained bert models to automate and accelerate data annotation for large datasets. *Radiology: Artificial Intelligence*, page e220007.

Thelwall, M., Buckley, K., and Paltoglou, G. (2012). Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, 63(1):163–173.

Trupthi, M., Pabboju, S., and Narasimha, G. (2017). Sentiment analysis on twitter using streaming api. In *2017 IEEE 7th International Advance Computing Conference (IACC)*, pages 915–919. IEEE.

Twintproject (2018). Twint - twitter intelligence tool.

Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K.-L. A., Elkhatib, Y., Hussain, A., and Al-Fuqaha, A. (2019). Unsupervised machine learning for networking: Techniques, applications and research challenges. *IEEE access*, 7:65579–65615.

Van Atteveldt, W., Van der Velden, M. A., and Boukes, M. (2021). The validity of sentiment analysis: Comparing manual annotation, crowd-coding, dictionary approaches, and machine learning algorithms. *Communication Methods and Measures*, 15(2):121–140.

Webster, J. J. and Kit, C. (1992). Tokenization as the initial phase in nlp. In *COLING 1992 volume 4: The 14th international conference on computational linguistics*.

Zahra, K., Azam, F., Butt, W. H., and Ilyas, F. (2018). A framework for user characterization based on tweets using machine learning algorithms. In *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*, pages 11–16.

Zhou, J., Gandomi, A. H., Chen, F., and Holzinger, A. (2021). Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5):593.

Anexo

- **Dataset entreglable:** https://drive.google.com/drive/folders/1dAFPmnew2r16WRmr1y7PAZHbmi0ahqWf?usp=share_link
- **Código de implementación:** https://github.com/ClaudiaSM-12/Extraction_with_Twint.git
- **Carpeta de verificación manual de tweets:** https://drive.google.com/drive/folders/1GePo7JVNDgZQERGAoHIXIVCFLre7_3Oc?usp=sharing

*En caso de no poder acceder a los archivos, contactar directamente con las autoras (131094@unsaac.edu.pe / 130734@unsaac.edu.pe)