ANÁLISIS DE POLARIDAD DE TWEETS SOBRE CONTEXTO POLITICO COLOMBIANO USANDO TECNICAS DE APRENDIZAJE NO SUPERVISADO

Francisco Pizarro Rivera

Nota de Aceptación

Certificamos que el presente Trabajo de Grado Satisface, en alcances y calidad, todos los requisitos que demanda un Trabajo de Grado de Maestría.

MARÍA CONSTANZA PABÓN

Directora

HERNÁN DARÍO VARGAS
Jurado

GERARDO SARRIA
Jurado

Aprobado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana Cali, para optar el título de Magister en Ciencia de Datos

HERNÁN CAMILO ROCHA NIÑO Ph. D.
Decano Facultad de Ingeniería y Ciencias

JUAN CARLOS MARTÍNEZ ARIAS
Director Posgrados de Ingeniería y Ciencias

Santiago de Cali y 04/03/2024

Acta de Correcciones al Documento de Trabajo de Grado

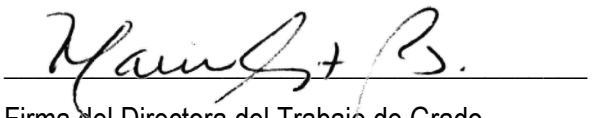
Santiago de Cali, 04/03/2024

Autor: Francisco Pizarro Rivera

Título del Trabajo de Grado: "ANÁLISIS DE POLARIDAD DE TWEETS SOBRE CONTEXTO POLITICO COLOMBIANO USANDO TECNICAS DE APRENDIZAJE NO SUPERVISADO"

Director: María Constanza Pabón

Como indica el artículo 2.13 de las Directrices para Trabajo de Grado de Maestría, he verificado que el estudiante indicado arriba ha implementado todas las correcciones que los Jurados del Proyecto de Trabajo de Grado definieron que se efectuaran, como consta en el Acta de Evaluación correspondiente.



Firma del Directora del Trabajo de Grado

Santiago de Cali, 15 de enero del 2024

Doctora

Gloría Inés Álvarez V.

Directora Maestría en Ciencia de Datos

Facultad de Ingeniería y Ciencias

Pontificia Universidad Javeriana de Cali

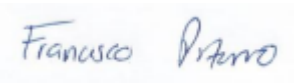
Asunto: Presentación para evaluación del proyecto aplicado

Cordial Saludo,

Con el fin de cumplir con los requisitos exigidos por la Universidad para optar por el título de Magíster en Ciencia de Datos, nos permitimos presentar a su consideración el proyecto denominado "Análisis de polaridad de tweets sobre contexto político colombiano usando técnicas de aprendizaje no supervisado" el cual fue realizado por el (los) estudiante (s) Francisco Pizarro Rivera con cedula 1018432075 perteneciente a la Maestría en Ciencia de Datos, bajo la dirección de María Constanza Pabón

El suscrito director del Proyecto Aplicado autoriza para que se proceda a hacer la evaluación de este proyecto, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado y sustentado oficialmente.

Atentamente,



_____Firma Estudiante(s)_____

Firma Director

Francisco Pizarro Rivera

C.C. 1018432075 de Bogotá

María Constanza Pabón

C.C. 34.559.226 de Popayán (C)

Documentación anexa:

Resumen del Proyecto Aplicado en formato digital (máximo 1 página).

Una copia digital (PDF) del documento del proyecto aplicado

FICHA RESUMEN

PROYECTO APLICADO – MAESTRÍA EN CIENCIA DE DATOS

TÍTULO: ANÁLISIS DE POLARIDAD DE TWEETS SOBRE CONTEXTO POLITICO COLOMBIANO USANDO TECNICAS DE APRENDIZAJE NO SUPERVISADO

1. ÁREA DE TRABAJO: Análisis de Lenguaje Natural
2. TIPO DE PROYECTO (Aplicado, Innovación, Investigación): Aplicado
3. ESTUDIANTE(S): Francisco Pizarro Rivera
4. CORREO ELECTRÓNICO: franciscopizarro9090@javerianacali.edu.co
5. DIRECCIÓN Y TELEFONO: Av Calle 127 # 17ª-64, Apto 511, Bogotá- Colombia
6. DIRECTOR: María Constanza Pabón
7. VINCULACIÓN DEL DIRECTOR:
8. CORREO ELECTRÓNICO DEL DIRECTOR: mcpabon@javerianacali.edu.co
9. CO-DIRECTOR (Si aplica): NA
10. GRUPO O EMPRESA QUE LO AVALA (Si aplica): NA
11. OTROS GRUPOS O EMPRESAS: NA
12. PALABRAS CLAVE (al menos 5): Tweets, No Supervisado, NLP, Tf-idf, Orientación Semántica, Machine Learning, VADER
13. FECHA DE INICIO: 1/08/2022
14. FECHA DE FINALIZACIÓN: 15/01/2024
15. RESUMEN:

El análisis de polaridad u orientación semántica es una de las ramas del ***Natural Language Processing*** que ha tenido más crecimiento en el última década, con amplias aplicaciones a nivel académico y comercial. En este proyecto de grado se realizó una exploración sobre la aplicación de modelos de Machine Learning de carácter Auto Supervisado y No Supervisado para realizar el análisis de polaridad en tweets escritos por los

usuarios de la red social X específicamente escritos sobre el contexto político colombiano. Se exploró el uso de un enfoque con modelos híbridos, en los cuales se hace un preproceso de pseudo etiquetado por medio de un modelo basado en lexicones (modelo VADER) para luego entrenar modelos supervisados como SVM, Logistic Regression y Multinomial Naive Bayes. El segundo enfoque constó de usar el modelo No Supervisado de K-Means, obteniendo un performance superior en la ejecución del modelo hibrido. Este trabajo tiene también por output la exportación a modo de prototipo del modelo con mejor performance y su vectorizador entrenado con el vocabulario de los 4.830 tweets recolectados de manera manual para ser desplegado en posibles ambientes de producción para el desarrollo de herramientas de análisis de orientación semántica aplicada a textos de redes sociales, pero en específico a tweets relacionados con el contexto político colombiano.

ANÁLISIS DE POLARIDAD DE TWEETS SOBRE CONTEXTO POLITICO COLOMBIANO USANDO TECNICAS DE APRENDIZAJE NO SUPERVISADO

Francisco Pizarro Rivera

Código Estudiante: . 8974858

CC 1018432075

Proyecto Aplicado para optar al título de

Magister en Ciencia de Datos

Director(a)

María Constanza Pabón

FACULTAD DE INGENIERÍA Y CIENCIAS

MAESTRÍA EN CIENCIA DE DATOS

SANTIAGO DE CALI, ENERO 15 DE 2024

Contenido

Introducción

El análisis de polaridad u orientación semántica es una de las ramas del ***Natural Language Processing*** que ha tenido más crecimiento en el última década, con amplias aplicaciones a nivel académico y comercial. En este proyecto de grado se realizó una exploración sobre la aplicación de modelos de Machine Learning de carácter Auto Supervisado y No Supervisado para realizar el análisis de polaridad en tweets escritos por los usuarios de la red social X específicamente escritos sobre el contexto político colombiano. Se exploró el uso de un enfoque con modelos híbridos, en los cuales se hace un preproceso de pseudo etiquetado por medio de un modelo basado en lexicones (modelo VADER) para luego entrenar modelos supervisados como SVM, Logistic Regression y Multinomial Naive Bayes.

El segundo enfoque constó de usar el modelo No Supervisado de K-Means, obteniendo un performance superior en la ejecución del modelo hibrido. Este trabajo tiene también por output la exportación a modo de prototipo del modelo con mejor performance y su vectorizador entrenado con el vocabulario de los 4.830 tweets recolectados de manera manual para ser desplegado en posibles ambientes de producción para el desarrollo de herramientas de análisis de orientación semántica aplicada a textos de redes sociales, pero en específico a tweets relacionados con el contexto político colombiano.

1. Definición del Problema

1.1. Planteamiento del Problema

Twitter cómo red social relevante en el contexto político:

Twitter es una red social de origen estadounidense, fundada en el año 2006, que en los últimos años se ha mostrado cómo una de las plataformas con mayor penetración en los usuarios para expresar sus opiniones y posturas sobre diversos temas, llegando a la cifra de 338 millones de usuarios a nivel mundial, generando en promedio 750 millones de "tweets" al día, y posicionándose cómo la séptima red social favorita por los usuarios según el estudio de permeabilidad digital realizado en el 2022 por We Are Social [1].

Sumado a esto, esta red social ha tomado mucha relevancia en su uso e impacto en el ámbito político en muchos países; dicha relevancia se debe a varios factores: en primer lugar, a que los diferentes actores políticos (entidades gubernamentales, personalidades políticas, lideres de opinión, etc.) la utilizan cómo uno de sus canales principales para emitir anuncios a la opinión pública, opiniones sobre temas y coyunturas de temas políticos, en segundo lugar, a que el 93% de sus usuarios tienen más de 18 años [2], lo cual la configura como una red social orientada en cierto modo a un sector adulto de la población, en tercer lugar, a que es una red social que está orientada a la gestión y monitoreo de tendencias en su tráfico de contenido, lo cual permite que los usuarios puedan tener "acceso a información en tiempo real y generada directamente por sus protagonistas" [3], y por último, a que Twitter permite la interacción directa entre estos actores políticos y los ciudadanos por medio de "tweets", "retweets" y comentarios.

Herramientas para el análisis de texto:

Actualmente la ciencia de datos provee un conjunto de herramientas y técnicas que pueden ser aplicadas al análisis de textos, entre ellas sobresale el NLP (Natural Language Processing), el cual integra técnicas de lingüística computacional, Machine Learning y aprendizaje profundo para permitir la interpretación, manipulación y comprensión del lenguaje natural humano, proveyendo así la capacidad de procesar información proveniente de fuentes no estructuradas cómo textos, audio y video, y que aplicadas en específico a los textos (libros, PDFs, correos electrónicos, mensajes en redes sociales, etc.) [4], presentan una gran oportunidad para extraer información valiosa que se puede utilizar en el ámbito comercial y académico, por ejemplo, detección de patrones de consumo en usuarios, creación de chat-bots, análisis de registros médicos, etc.

Motivación del trabajo de grado:

Teniendo en cuenta la relevancia que tiene la red social X (antes Twitter) en el encausamiento y convergencia de interacciones directas entre los actores políticos y los demás actores de la sociedad, la aplicación de técnicas de Machine Learning con un enfoque No supervisado para el análisis de polaridad se podrían utilizar en el desarrollo de soluciones para el análisis de lenguaje natural con aplicaciones comerciales.

Por ejemplo, futuras implementaciones más robustas del prototipo que se desarrolló en este trabajo de grado se podrían alimentar en tiempo real por medio de una API a la red social X para hacer extracción masiva de tweets sobre un tópico puntual con el objetivo de determinar niveles de favorabilidad y des favorabilidad en la precepción de la sociedad; Este tipo de soluciones podrían ser útiles como input para analistas políticos, jefes de campaña y a la opinión pública en general en tiempos de evento electorales o de toma de decisiones políticas coyunturales.

1.2. Formulación del Problema

La descripción anterior deriva en las siguientes preguntas: ¿Puede un modelo de aprendizaje no supervisado ser efectivo en el proceso de clasificación de polaridad para tweets en el contexto político colombiano?, ¿Qué trabajos anteriores sobre análisis de polaridad en textos se han desarrollado basados en modelos de aprendizaje No Supervisado?, ¿Qué técnicas de preprocesamiento y transformación de datos se deben aplicar para posteriormente implementar modelos de aprendizaje No Supervisado en tweets sobre el contexto político colombiano?, ¿Se puede hacer una implementación efectiva de un modelo de aprendizaje No Supervisado para el análisis de polaridad en tweets sobre el contexto político colombiano?, ¿Cuál es el desempeño que podría alcanzar un modelo de aprendizaje No Supervisado para análisis de polaridad aplicado tweets relacionados con el contexto político colombiano?, y por último, ¿Qué herramientas se podrían utilizar para generar un prototipo usable por parte de un usuario final para para la asignación de polaridades en tweets sobre sobre temas de política en el contexto colombiano?

2. Objetivos del Proyecto

1.3. Objetivo General

Desarrollar un modelo de aprendizaje no supervisado para la clasificación de polaridad (análisis de sentimientos) para tweets, y evaluar su efectividad aplicada a tweets relacionados con el contexto político colombiano.

1.4. Objetivos específicos

1. Realizar una revisión de herramientas y recursos disponibles para el análisis de polaridad en tweets en español mediante técnicas de aprendizaje no supervisado.
2. Explorar y seleccionar las técnicas no supervisadas de ciencia de datos que serán utilizadas para el análisis de polaridad para tweets.
3. Implementar un modelo no supervisado para análisis de polaridad en tweets.
4. Realizar una evaluación de la efectividad en el proceso de clasificación de polaridad ejecutado por medio del modelo a desarrollar sobre un corpus de tweets relacionados con el contexto político colombiano previamente etiquetados de manera manual.
5. Realizar el despliegue de un prototipo que implemente el modelo seleccionado para la interacción con usuario final para el análisis de polaridad para tweets extraídos de Twitter dando por parámetro un tweet extraído por el usuario.

3. Metodología

La metodología que se utilizó para poder desarrollar los objetivos del proyecto de grado fue la siguiente: Primero, se empezó por la construcción de un corpus de datos con 4.830 tweets en español relacionados con el contexto político colombiano mediante web scrapping manual, seguido de un proceso de etiquetado manual de polaridad por parte de un experto que fue contratado para la ejecución de esta tarea. El objetivo del proceso de etiquetado fue el de tener una referencia en términos de polaridad generada por un humano para posteriormente comparar el desempeño de los modelos generados.

Adicionalmente, con la supervisión de la directora de grado, se decidió ampliar la exploración de los modelos de aprendizaje No Supervisado para la ejecución de tareas de análisis de sentimiento, agregando al estudio la aplicación de modelos Auto Supervisados, los cuales, como se describirá más adelante, son modelos que al igual que los modelos No Supervisados, no necesitan tener datos etiquetados a priori para desarrollar los procesos de aprendizaje de máquina. Esta decisión se tomó con base en la relevancia y pertinencia encontrada sobre este tipo de modelos en el proceso de investigación bibliográfica que se realizó sobre métodos de análisis de sentimiento a datos no etiquetados, aumentando así el campo de acción y de investigación de este trabajo de grado ya que también se incluyó un contraste de los desempeños de estos dos tipos de modelos (No Supervisados y Auto Supervisados) en la ejecución de tareas de análisis de polaridad (análisis de sentimiento) en textos cortos sobre la temática especifica del contexto político colombiano.

Posteriormente, se procedió a realizar una etapa de preprocesamiento de los datos, iniciando con un proceso traducción de los tweets al idioma inglés, para posteriormente proseguir con un proceso de limpieza de datos.

Una vez obtuvo un corpus de datos apto para alimentar modelos No Supervisados y Auto

supervisados, se procedieron a crear y evaluar varios modelos de ambas naturalezas, empezando con 3 modelos Auto Supervisados, para luego entrenar 2 modelos No supervisados.

Por último, se procedió a evaluar el desempeño de estos 5 modelos en contraste con las etiquetas asignadas por el experto con el objetivo de seleccionar el mejor modelo e implementarlo en un prototipo funcional que pudiera ser utilizado por un usuario final para realizar tareas de análisis de sentimiento a tweets ingresados de manera manual.

4. Marco Teórico y Antecedentes

4.1. Marco Teórico

Aprendizaje Auto Supervisado (*Self-Supervised Learning*):

Los métodos de aprendizaje Auto Supervisado son una mezcla de la implementación de un modelo No Supervisado y un modelo Supervisado, con el objetivo de sortear algunos obstáculos derivados de la naturaleza de cierto tipos de problemas a la hora de resolverlos con técnicas de Machine Learning, en especial, problemas con la consecución de datos etiquetados debido al costo económico o logístico necesario para poder tener un set de datos lo suficientemente robusto como para entrenar un modelo de aprendizaje supervisado [5].

La primera etapa de estos modelos mixtos consta de una tarea de *clustering* ejecutada por un modelo No Supervisado, en el cual el modelo va a encontrar patrones en los datos de las observaciones sin etiquetas y las asignará a alguna de las clases (salida del modelo). Una vez se tengan estas ***pseudo etiquetas***, se procede a una validación de las asignaciones, usualmente por un experto, con el objetivo de verificar la eficacia de la tarea de asignación automática, para una posterior fase de entrenamiento de un modelo supervisado con estos datos pseudo etiquetados con el objetivo de poder etiquetar finalmente nuevas observaciones de la misma naturaleza [5].

Aprendizaje No Supervisado:

El aprendizaje no supervisado es una de las estrategias que puede usar el Machine Learning para realizar los procesos de aprendizaje. En este tipo de modelos no se cuenta con un corpus de datos etiquetados en su variable de salida (variable objetivo) para el entrenamiento del modelo, sino que es el modelo el encargado de realizar agrupaciones de manera autónoma según la identificación de patrones en los datos. Las aplicaciones más comunes para la implementación de aprendizaje no supervisado son los problemas de *clustering* [6], en los cuales el modelo identifica los patrones

discriminatorios presentes en los datos y establece la cantidad de clases en las cuales serán divididas cada una de las observaciones del dataset que se usó para el entrenamiento.

A pesar que la mayoría de los modelos usados en el gran campo del Machine Learning pertenecen a la sub rama de Aprendizaje Supervisado, en la práctica, una ingente cantidad datos no se encuentran etiquetados, ya que cómo se mostró anteriormente, la asignación de etiquetas reviste un esfuerzo significativo en términos de tiempo, recurso humano y dinero que hace que la naturaleza de algunos problemas no presente una situación costo-efectiva para la implementación de soluciones basadas en modelos de Aprendizaje Supervisado. Es en estos casos en los cuales las técnicas de Aprendizaje No Supervisado se erigen cómo una alternativa factible y efectiva para solucionar este nicho de problemas [7].

Por ejemplo, Gerón (2019) muestra en caso hipotético que si se tiene por objetivo implementar una solución basada en Machine Learning para identificar defectos de fabrica en un producto a la salida de una línea de ensamble mediante el análisis de imágenes captadas en tiempo real se entraría en la disyuntiva entre escoger la implementación de un modelo Supervisado versus un modelo No Supervisado. La selección del modelo deberá tener en cuenta las siguientes preguntas*:* ¿Cuántos tipos de defectos se pueden presentar en el producto?, ¿Cuál es el costo de contratar un experto que identifique y etiquete las imágenes?, ¿Con que frecuencia se cambian o actualizan las características físicas del producto?; Teniendo estas preguntas en mente, si por ejemplo se determina que las características físicas del producto se actualizan constantemente, esto derivaría en que los defectos de fábrica también van a cambiar al mismo ritmo, siendo así, se necesitaría actualizar el etiquetado de las muestras en sintonía con la frecuencia de estos cambios y esto revestiría un gran esfuerzo en términos de tiempo y costos. Es en estos casos donde una solución basada en modelos de Aprendizaje No Supervisado podrían ser aplicados, ya que sería el modelo el encargado de detectar y actualizar esas características marginales que clasificarían cuando un producto presenta un desperfecto [7].

Por otro lado, según Géron (2019), Las principales aplicaciones de ***Clustering*** son:

- Solución de problemas de análisis de datos con los cuales no se tiene familiaridad, de tal manera que el científico de datos pueda hacer una segmentación inicial de los datos, encontrando así agrupaciones de observaciones con variables similares y de esta manera poder focalizar sus análisis sobre estas agrupaciones [7].
- Detección de anomalías, mediante el cual el modelo identifica que observaciones presentan un bajo nivel de ***afinidad*** con el resto de las características de los *clusters*, encontrando así observaciones atípicas que pueden ser de utilidad en problemas de control de calidad o de identificación de fraude [7].
- Para segmentación de clientes, mediante el cual el modelo hace agrupaciones de clientes según los valores de sus variables. Este tipo de aplicaciones es usado en perfilamiento de clientes con el objetivo de diseñar productos o estrategias de marketing focales [7].
- Para reducción de dimensionalidad, mediante el cual el modelo arroja por salida un vector con los valores de afinidad que tiene cada una de las observaciones del set de datos de entrenamiento con cada uno de los *clusters* detectados [7]. Esto lleva a que, por ejemplo, si se tiene una base de datos con 1.000 variables y 1.000.000 de observaciones (sistema 1.000.000 X 1.000), mediante la generación de ***j*** clusters, se pase un sistema de 1.000.000 X ***j,*** reduciendo así la complejidad del modelo y la demanda de recurso computacional para llegar a una solución.

Del mismo modo, Kubat (2017), subraya que las aplicaciones más comunes para los modelos de ***Clustering*** son:

- Estimación de valores perdidos. Por ejemplo, si se tiene una base de datos en la cual se han encontrado varios *clusters*, y se ha determinado que en cierto cluster hay una relación directamente proporcional entre la variable ***x*** y la variable ***y***, de tal manera que, si ***x*** tiende a ser “pequeña”, entonces ***y*** tenderá a ser también pequeña. Es por medio de esta relación

que se pudo extraer por medio de la identificación de los clusters que, si en dicho dataset hay un registro con datos faltantes, por ejemplo, en ***y***, se puede establecer un valor probable de ***y*** si se conoce el valor de ***x***, ya que se esperaría que para que una observación pertenezca al cluster correspondiente ser debería guardar la relación evidenciada entre ***x*** y ***y.***

- Reducción del dimensiones en otros modelos. Kubat (2017) muestra cómo la implementación de soluciones de *Clustering* pueden servir para reducir la complejidad computacional de otros modelos, por ejemplo, menciona aplicaciones previas de clustering para implementaciones posteriores de Clasificadores Bayesianos y Redes de Funciones de Base Radial (RBF), ya que un modelo de clustering puede segmentar los registros de una base de datos y de esta manera el científico de datos puede dividir el dataset en N clusters y así centrar sus análisis por separado [8].
- Identificación de clusters para luego implementar un modelo de Aprendizaje Supervisado. En este caso se evidencia uno de los obstáculos más típicos en aplicaciones de modelos de Machine Learning frente a la consecución de datos etiquetados, y es por medio de la implementación de tareas de *clustering* mediante la cual se puede tener una primera aproximación de generación de datos etiquetados (preferiblemente si son posteriormente validados por un experto) para una posterior implementación de una solución basada en aprendizaje Supervisado [8].

Se subraya que resulta interesante que tanto Géron (2019) cómo Kubat (2017) no incluyen entre los usos más comunes para los modelos de ***Clustering*** tareas de análisis de texto, sin embargo, cómo se muestra en el capítulo 5, en la revisión bibliográfica sobre trabajos anteriores de aplicaciones de modelos no supervisados para el análisis de polaridad en textos se encontraron algunos ejemplos de aplicación de este tipo de modelos.

En el contexto de este proyecto, se explorará el modelo de K-Means para las tareas de asignación

de polaridad con el objetivo de explorar la eficacia de esta familia de modelos a la hora de clasificar los textos de los tweets según su significado, sin que se le muestre a priori una clasificación previa. Esta tarea presenta gran relevancia ya que la inmensa mayoría de modelos para el análisis de polaridad en textos están basado en soluciones de Aprendizaje Supervisado.

K-Medias y fue creado por Stuart Lloyd mientras trabajó en la empresa Bell Labs durante los años 50, es por esto por lo que este algoritmo se conoce cómo K-Means o cómo Lloyd-Forgy [7]. El algoritmo consiste en primero crear ***k*** clusters iniciales, donde el número de clusters será un hiperparametro del modelo y será establecido empíricamente por el científico de datos, de tal manera que cada una de las observaciones del corpus es asignada a una de las ***k*** agrupaciones. Una vez que se haya realizado la asignación inicial a cada uno de los registros del dataset, el algoritmo procede a calcular los centroides, los cuales se podrían definir cómo puntos calculados (no son necesariamente puntos idénticos a alguna de las observaciones de la base de datos) por medio de alguna de las fórmulas de distancia, entre las cuales se encuentran las distancias ***Euclidiana***, ***Manhattan, Minkowski*** o ***Mahalanobis*** entre otras. Paso seguido, se calculan las distancias entre cada una de las observaciones y todos los centroides calculados inicialmente de los diferentes *Clusters*, en este momento, el algoritmo determinará si existe una distancia menor alterna entre la observación analizada y alguno de los centroides de los clusters que sean diferentes al cluster asignado inicialmente, de ser así, el algoritmo le asignará a esta observación el cluster cuya distancia con dicho centroide se minimice [8]. Si por el contrario, el algoritmo determina que la distancia mínima entre la observación se minimiza con respecto al centroide del cluster asignado inicialmente, no hará nada al respecto y seguirá con los mismos pasos aplicados a la siguiente observación del dataset.

Una vez que una observación haya sido “relocalizada”, se deben actualizar los cálculos de los centroides del cluster que “perdió” dicha observación y del cluster que “gano” la asignación de la observación en curso; esta actualización se debe realizar para cada una de las observaciones del

dataset [8], lo cual hace que los modelo de K-Means sean intensivos en consumo de recursos computacionales si se cuenta con base de datos de gran tamaño, sin embargo, presenta varias ventajas en la simplicidad de sus pasos.

La finalización del algoritmo se alcanza una vez que se ha determinado que cada una de las observación del corpus de datos ha sido asignada a el cluster que asegura que la distancia entre dicha observación sea la mínima en comparación con los demás centroides.

Kubat (2017) explica el algoritmo de K-Means de la siguiente manera [8, p. 277]:

1. Creación de $\boldsymbol{k}$ clusters.
2. Asignación aleatoria de cada una de las observaciones $X_i \in \boldsymbol{x}$ a uno de los clusters
3. Cálculo de los centroides $C_i \in C$ de cada uno de los clusters, es importante subrayar que se debe seleccionar alguna de las métricas de distancia teniendo en cuenta la naturaleza de los datos; por ejemplo, ¿las variables son de naturaleza continua o de naturaleza discreta?, ¿Qué tipo de distancia es mejor en términos de costo computacional?
4. Tomar una observación X_i y calcular las distancias contra cada uno de los centroides, tomando a C_j cómo el centroide asignado a esta observación en el momento de la inicialización.
5. Si se determina que la menor de las distancias $\mathrm{Min}[d(X_i, C_i)]$ se da en C_j (centroide del cluster asignado inicialmente), no se hace nada, de lo contrario, se debe "desasignar" la observación X_i al cluster inicial, y se le debe asignar al cluster que cumpla con la función de minimización de distancias.
6. Recalcular los centroides para los clusters que se hayan actualizado (cluster desasignado y cluster asignado).

7. **Criterio de parada:** Cuando se determine que no se pueden modificar los clusters para ninguna de las observaciones.

Al analizar los pasos del anterior algoritmo, nos damos cuenta de que el objetivo de K-Means es agrupar cada una de las observaciones de tal manera que se maximicen las distancias entre observaciones de diferentes clusters y como consecuencia, se minimicen las distancias entre observaciones pertenecientes al mismo cluster. A continuación se muestra una representación gráfica del algoritmo de K-Means tomada de Gerón (2019) [7].

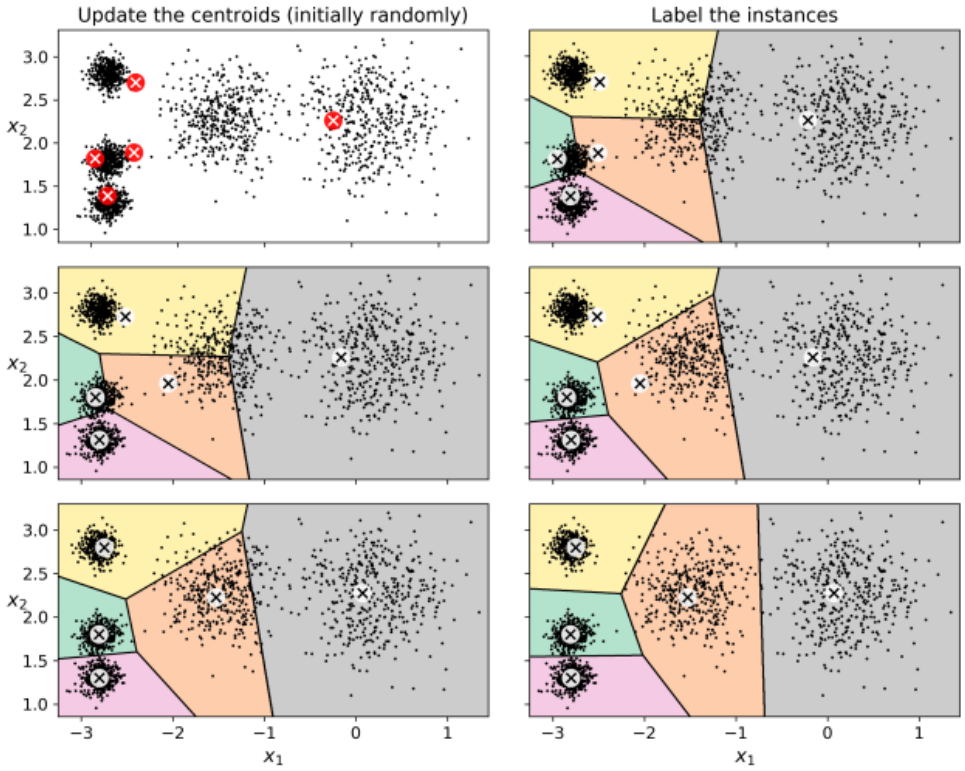


Ilustración 1 Algoritmo de K-Mean, tomado de Gerón (2019)

Es importante resaltar que el algoritmo de K-Means funciona con datos numéricos, para lo cual se debe realizar un preprocesamiento de los textos de los tweets para ser adecuados a la hora de

alimentar estos modelos.

Derivado del proceso de agrupación de datos sale a relucir una importante pregunta: ¿Cómo se determina la cantidad de ***Clusters*** que debe tener en cuenta el modelo?, Kubat (2017) expresa que en problemas donde los datos se pueden graficar hasta en 3 dimensiones (solo 3 variables), el cerebro humano es capaz de identificar con relativa facilidad hasta 4 ***clusters*** [8], sin embargo, muchas de las aplicaciones reales de las tareas de ***Clustering*** revisten problemas de varias decenas de variables, lo cual hace imposible una representación gráfica del comportamiento de los datos. Es por eso por lo que en muchos problemas de clasificación con modelos No Supervisados se pueden empezar por preguntarle a un experto u usuario familiarizado con la problemática a solucionar *¿Cuántos clusters se esperaría evidenciar en el corpus de datos?*, mientras que en otros problemas se espera que sea el mismo modelo el que arroje el número de clusters que mejor explican el comportamiento de los datos de entrenamiento [8].

Análisis de polaridad en textos:

Se entiende por análisis de sentimiento o polaridad un problema de clasificación [9], mediante el cual se desea realizar la identificación y determinación del sentimiento (positivo, negativo o neutro) realizado por medio de un actor, en este caso un usuario de una red social, sobre un tópico especifico, en este caso, un tema político del contexto político colombiano. El hecho de que la naturaleza del problema de análisis de polaridad se pueda configurar cómo un problema de clasificación permite que la utilización de técnicas de aprendizaje no supervisado sea pertinente en este caso.

API (Application Programming Interfaces):

Es un conjunto de protocolos e instrucciones mediante los cuales dos aplicaciones de software pueden intercambiar información e interactuar entre ellas [10]. En el contexto de la realización del proyecto, se exploró el uso de una API para la extracción de tweets desde la red social Twitter para

la conformación del corpus etiquetado manualmente para la posterior evaluación de la efectividad del modelo de clasificación de polaridad a desarrollar.

Vectorización de textos

Debido a que una amplia gama de algoritmos de Machine Learning solo se pueden entrenar con bases de datos donde los valores de la variables sean de naturaleza numérica, en el caso del NLP se utiliza la técnica de vectorización, en la cual se dividen los textos a nivel de n-gramas, donde un n-grama es la combinación de n palabras adyacentes en un texto, para luego transformar los textos en vectores numéricos de frecuencias donde se plasman las apariciones de cada uno de estos n-gramas en cada observación [5].

Frecuencia de Término-frecuencia inversa de documento (Tf-idf)

Este algoritmo permite ir un paso más allá que el de la vectorización, ya que en este caso se busca no solo identificar la aparición o no de los n-gramas en cada uno de los textos que conforman el corpus, si no de poder filtrar las palabras que representan información útil para discriminar las características de dichos textos [5], en nuestro caso, se busca identificar en cada texto las palabras que dan mayor nivel de polaridad en el análisis de sentimiento. Esta técnica de vectorización se escoge ya que ha mostrado ser de utilidad en la transformación de textos en vectores numéricos para aplicaciones de modelos de vectores espaciales (VSMs) [11].

Natural Language Processing (NLP):

Uno de los puntos de inflexión en la evolución de la especie humana fue la aparición del lenguaje hablado; esto se evidencia en que esta capacidad es una de las características más primarias que nos diferencian del resto de seres vivos que habitan este planeta; sin embargo, la aparición del lenguaje escrito podría describirse cómo un salto cuántico en el desarrollo de la especie humana. Mediante esta herramienta el ser humano fue capaz, no solo de poder transmitir conceptos e ideas

a nuestros semejantes por medio de la codificación y decodificación de estos en palabras, frases, proposiciones y demás estructuras lingüísticas, si no de poder perpetuar en el tiempo sus hitos históricos.

Los diferentes lenguajes escritos que existen están en constante movimiento; es más, podría afirmarse que son organismos que van transformándose en yuxtaposición con la evolución del ser humano y van asimilando en sus estructuras los diferentes cambios de tiempo y lugar que han acompañado la historia humana. Este dinamismo continuo se diferencia de los lenguajes de programación, los cuales son primero diseñados partiendo de un conjunto de reglas establecidas que, cómo expresa Collet (2021) "buscan describir que declaraciones podrías hacer en dicho lenguaje y que quieren decir estas" [12, p. 309], en este sentido, los lenguajes de programación tienen un ciclo de vida en los cuales primero se establecen las reglas semánticas de lenguaje, y luego el lenguaje está listo para ser utilizado por el usuario final; sin embargo, en el caso del lenguaje humano (de ahora en adelante llamado Lenguaje Natural), el Genesis del ciclo de vida empezó primero con el uso del lenguaje por parte de las diferentes sociedades y culturas, para posteriormente haber dado cabida a la estructuración de las reglas, convenciones y parámetros que rigen cada uno de los idiomas.

Adicionalmente, el campo de análisis de lenguaje natural en texto ha mostrado un gran interés por parte de diferentes sectores, entre ellos los investigadores, lingüistas, profesionales en marketing, empresas de Retail e E-commerce entre otros, debido a que la gran mayoría del conocimiento humano ha sido consignado en textos a través de la historia, llegando a un punto de convergencia de que casi todo nuestros conocimientos hayan sido consignados en la internet, la cual en su mayoría está en forma de textos. Es por esto qué la explotación de esta disciplina cada día ha revestido una mayor importancia y potencial para todas estas partes interesadas.

El interés en el desarrollo de programas computacionales para el análisis del lenguaje humano tuvo

uno de sus primeros “sprints” durante la década de los 60’s, durante la cual se desarrollaron algoritmos que pretendían describir de manera determinística las reglas del lenguaje inglés para crear aplicaciones muy básicas de “chat bots” con los cuales un usuario podría interactuar con un set de palabras y secuencias previamente parametrizadas en el programa [12]. Un ejemplo de estos esfuerzos primigenios fue el sistema ELIZA, el cual fue desarrollado en el MIT (Massachusetts Institute of Technology) por el laboratorio de inteligencia Artificial a cargo de Joseph Weizenbaum en el año 1966. Este prototipo tenía por objetivo simular ser una psicóloga que pudiera interactuar con un usuario, y se podría afirmar que fue una implementación con un gran performance para las limitaciones tecnológicas y teóricas de la época, ya que en palabras del propio Weizenbaum “fue difícil convencer a algunos de los usuarios que ELIZA no era humana” [13]

Sin embargo, la tarea de analizar el lenguaje natural había demostrado ser una quimera esquiva de alcanzar, ya que cómo se expuso anteriormente, una de las características principales del lenguaje natural es que este no es estático ni monolítico; todo lo contrario, el lenguaje natural es un organismo casi vivo, que no solo evoluciona según el tiempo y el contexto, si no que está plagado de interpretaciones derivadas de componentes difíciles de parametrizar cómo el sarcasmo, el doble sentido, etc. Estos componentes latentes en el lenguaje natural, que aparentemente solo podían ser decodificados por otro humano, fue una de las grandes dificultades a las cuales se enfrentaron esos primeros esfuerzos de analizar la forma en que nos comunicamos.

Cómo expone Collet (2021), uno de los puntos de inflexión en el análisis de lenguaje natural fue el cambio de enfoque en los inicios de los años 90’s, al pasar de “¿cómo puedo establecer y modelar todas las reglas del lenguaje” a, “cómo, según un corpus de ejemplos de lenguaje natural, puedo inferir dichas reglas dentro de algún tipo de espacio de reglas sin tener que establecerlas a priori?” [12]

Teniendo en cuenta este cambio de paradigma que se presentó a finales del siglo pasado, los

modelos computacionales dejaron de tener por objetivo modelar matemáticamente todas las reglas del lenguaje natural y así buscar una manera de "entender el lenguaje" [12], y trascendieron a , mediante grandes cantidades de datos conformados por textos, proveer a las computadoras la capacidad de poder responder a preguntas prácticas que pudieran dar insights sobre los datos dados de entrenamiento tales como: ¿Cuál es el tópico del texto?, ¿Este texto tiene un sentimiento positivo o negativo?, ¿Cuál debería ser la próxima palabra a esta oración incompleta?, etc. [12]. De esta manera la disciplina del Machine Learning se ha convertido en el estado del arte para la ejecución de tareas de NLP en la actualidad.

Stemming:

Este proceso tiene por objetivo estandarizar palabras con la misma raíz, mapeando así palabras como ***andar, andando,*** con su raíz ***anda***. Este proceso se utiliza para facilitar la normalización de los datasets de datos, y así, obtener un corpus optimizado para las posteriores fases de construcción de los modelos de análisis de texto. [14]

Tokenization:

Proceso de preprocesamiento de datos para el análisis de textos mediante el cual se dividen los textos en fragmentos denominados "tokens" por medio de criterios y reglas establecidas cómo por ejemplo los signos de puntuación, patrones de mayúsculas y minúsculas, etc. El proceso de la Tokenización es uno de los pasos más importantes en el análisis de textos, ya que por medio de este proceso se "limpian" los caracteres que no serán utilizados en el análisis por NLP, cómo por ejemplo los signos de puntuación, caracteres especiales, contracciones, etc. [14].

Tweet:

Texto de máximo 280 caracteres que incluyendo caracteres especiales (*¡"#$%&/()=?*), signos de puntuación y letras, componen un "post" realizado por un usuario en la red social de Twitter, estos

"post" también pueden incluir audios, enlaces a artículos externos a la app, videos e imágenes. En el contexto del proyecto, solo se analizarán las componente de textos de los tweets.

Support Vector Machines:

Como se mostrará más adelante, este modelo fue utilizado en la fase de entrenamiento de datos pseudo etiquetados para el entrenamiento con modelo supervisado en el enfoque ***Auto Supervisado.***

Los SVM son modelos supervisados de Machine Learning que son capaces de hacer procesos de clasificación en datos linealmente y no linealmente separables, y son capaces de hacer clasificación binaria y de multi clase mediante la introducción de una zona de decisión de clasificación llamada "margen", el cual es definido como la distancia entre el hiperplano de separación y las observaciones del dataset más próximas a dicha frontera de decisión, llamadas ***vectores de soporte*** [5].

Para datos linealmente separables primero se hace el cálculo de las líneas del margen, las cuales se basan en las observaciones periféricas de las clases, de ahí el nombre de ***Support Vector Machines***, por ende, la adición de más observaciones que no estén en el área periférica de las fronteras de decisión no afectarán el cálculo de dichas fronteras [7]. Ya que se pueden implementar infinitas rectas de separación de datos linealmente separables, el modelo de SVM tiene por objetivo encontrar la mejor recta o hiperplano que maximiza la región de margen, llevando así a un problema de optimización cuadrática teniendo por raciocinio que al tener un margen de clasificación más amplio se tiende a disminuir los errores de clasificación [5].

Ya que el establecimiento de los vectores de soporte (líneas de frontera de la zona de margen) dependen de las observaciones más periféricas de las clases, dichos vectores son muy sensibles a

la aparición de ***outliers*** (observaciones con valores atípicos según su clase etiquetada), haciendo la zona de margen cambie drásticamente sus dimensiones y el modelo puede disminuir su capacidad de clasificación al disminuirse el margen de decisión. Para evitar esto, se introduce el hiperparametro ***C***, el cual tiene por objetivo tener un buen balance entre mantener un buen margen de clasificación mientras se limitan los errores de clasificación (***violaciones de margen***). Cuando este hiperparámetro es bajo (tendiendo a 0), se tiende a ampliar el "ancho" del margen, llevando así a aumento de las violaciones de clasificación, caso contrario, si ***C*** tiende a crecer, el margen de decisión se disminuye y con menores violaciones de margen, pudiendo llevar a problemas de sobreajuste [7].

Regresión logística:

Como se mostrará más adelante, este modelo fue utilizado en la fase de entrenamiento de datos pseudo etiquetados para el entrenamiento con modelo supervisado en el enfoque ***Auto Supervisado.***

Los modelos de regresión logística son modelos aprendizaje supervisado que tienes por objetivo realizar el proceso de clasificación mediante el cálculo de la probabilidad de que una observación pertenezca a una clase dado el valor de sus variables predictoras.

Con base a Raschka (2019), este cálculo se realiza mediante la computación de los valores de las variables predictoras ***x*** y de los parámetros ***w*** que el modelo "aprende", llevando así al cálculo de la función de entrada, para posteriormente alimentar la función de activación ***sigmoide*** (***σ)***,la cual es una función doblemente asintótica cuando el parámetro ***t*** tiende al ***infinito***, aproximándose a **1**, y cuando ***t*** tiende a ***-infinito*** tendiendo a 0 [7].

$$\sigma = \frac{1}{1 + e^{-t}}$$

Por último, si la función de activación sobrepasa el umbral, se presenta el caso ***positivo*** de clasificación.

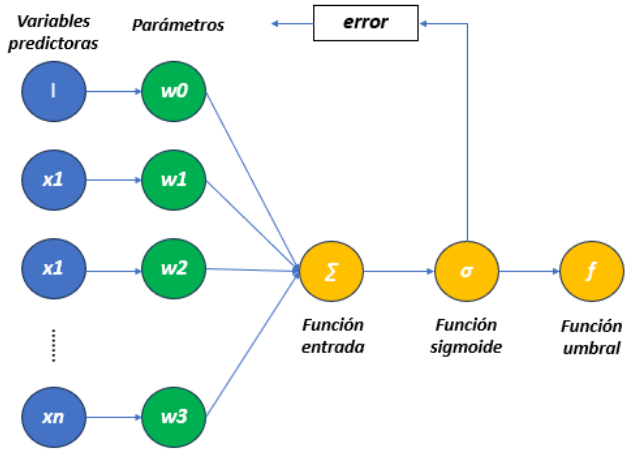


Ilustración 2 Modelo de regresión logística, basado en Raschka (2019)

Cómo se muestra en el capítulo de implementación del modelo de regresión logística, este modelo se puede ampliar para casos de clasificación multiclases por medio de la regresión ***Softmax***, el cual es una combinación de múltiples clasificadores binarios [7].

Multinomial Naive Bayes:

Los modelos de clasificación por ***NB*** se basan en el teorema de Bayes, el cual busca establecer la probabilidad de que un evento ocurra dada (salida del modelo) la ocurrencia de otros eventos relacionados (valores de la variable predictora) [15], de tal manera que se pueda determinar la probabilidad de que un nuevo registro pertenezca a una de las posibles categorías en la variable de salida; Sin embargo, se tiene el problema de que para clasificar un nuevo registro con el teorema de Bayes se necesitaría que los valores de sus variable predictoras fueran exactamente iguales a las

variables predictoras de un registro clasificado en la etapa de entrenamiento, lo cual es poco probable cuando el número de variables predictoras es muy grande. Por ejemplo, Bruce (2022) muestra que el hecho de ingresar una sola variable adicional a un modelo basado totalmente en el teorema de Bayes puede disminuir la probabilidad de clasificación en un factor de ***5X*** [15].

$$P(Y = i \mid X_1, X_2, \dots X_n) = \frac{P(Y = i) * P(X_1, X_2, \dots X_n | Y = i)}{P(Y = 0) * P(X_1, X_2, \dots X_n | Y = 0) * P(Y = 1) * P(X_1, X_2, \dots X_n | Y = 1) * \dots .}$$

Para evitar este problema, el modelo de ***Naive Bayes*** revisa las relaciones entre los valores de estas variables predictoras ***Xi*** y la variable objetivo ***Yi***, calculando la probabilidad previa de cada una de las clases (***Positivo, Negativo*** y ***Neutral***), la cual se establece mediante el análisis de las frecuencias de las categorías en el set de datos de entrenamiento, y toma en cuenta las probabilidades individuales de que cada palabra esté relacionada con la categoría de clasificación [4]. Esto se logra mediante la implementación de la premisa de independencia entre los valores de las variables predictoras ***Xi*** [15].

$$P(Y = i \mid X_1, X_2, \dots X_n) = \frac{P(Y = i) * P(X_1 | Y = i) \dots . P(Y = i) * P(X_n | Y = i)}{P(Y = 0) * P(X_1 | Y = 0) \dots P(X_n | Y = 0) + P(Y = 1) * P(X_1 | Y = 1) \dots P(X_n | Y = 1)}$$

En el contexto de clasificación de polaridad para textos, la inclusión de la premisa de independencia permite que la probabilidad de que una palabra ***Xi*** aparezca en un tweet, dada una polaridad, no dependa de la aparición o no de las demás palabras [4].

4.2. Antecedentes

En la revisión bibliográfica realizada se encuentra una gran cantidad de estudios, recursos, papers

académicos y trabajos de grado relacionados con el análisis de textos mediante el uso de técnicas supervisadas y no supervisadas, resaltando los siguientes trabajos por su cercanía con este proyecto de grado.

TASS:

Cómo se describió anteriormente, uno de los recursos más relevantes en el área de análisis de polaridad aplicados en específico a estructuras de texto extraídas de Twitter es el TASS. Esta organización provee una gran variedad de herramientas y ejercicios relacionados con este tópico, por ejemplo, tienen set de datos etiquetados con polaridad que pueden ser usados por terceros para el entrenamiento de modelos de análisis de texto, tienen un workshop el cual funciona a modo de "campeonato" para que los grupos inscritos en cada una de sus versiones puedan competir en tareas de análisis de tweets [16], sin embargo, el acceso a los lexicones generados en el marco de esta organización tienen restricciones para su uso, por lo cual se decidió explorar otros lexicones para las tareas posteriores de análisis de textos.

Comparación del uso de Lexicones para el análisis de sentimiento en Twitter:

En el año 2016, Juárez y Calvo hicieron una comparación del impacto del uso de dos Lexicones del idioma español para la detección de polaridad de tweets en español, utilizando técnicas no supervisadas [9], llevando así a demostrar que los resultados en la efectividad de clasificación de tweets según su polaridad está estrechamente relacionados con la estructura de los lexicones (tamaño, palabras contenidas, modismos, expresiones coloquiales, etc.), lo cual muestra que la efectividad en la clasificación de detección de sentimiento para un texto es susceptible a mejoras [9] si se incluye en su análisis las palabras pertenecientes, no solo al idioma, si no al contexto especifico, por ejemplo, un análisis de polaridad de tweets en español basado en un modelo entrenado con expresiones lingüísticas utilizadas en España, no va a ser tan efectivo en la clasificación de polaridad si se aplica a tweets generados en España, que a tweets generados en

Argentina, ya que cada país tiene palabras, que a pesar de ser del mismo idioma, pueden tener un significado de sentimiento diferente.

Análisis Auto supervisado de sentimiento textos sobre narrativa del conflicto colombiano

En 2022 Rendón, Gil, Páez y Rivera, realizaron un modelo Auto supervisado entrenado en un corpus de textos extraídos de artículos producidos por diferentes universidades colombianas sobre el conflicto armado en Colombia, el cual fue traducido previamente al idioma inglés para alimentar posteriormente el modelo de análisis de polaridad basado en Lexicones (Lexicon Based Model) ***LR-Sentia*** con el objetivo generar pesudo etiquetas, entrenando después 4 modelos de aprendizaje supervisados (Support Vector Machine, Logistic Regression, Naive Bayes, sobre un subcorpus de textos considerados con alta polaridad. Este modelo hibrido fue comparado con modelos basados en Lexicones, mostrando un desempeño superior a estos, llegando a un ***accuracy*** del 75%, el cual según sus conclusiones presenta una buena solución para análisis de polaridades en textos sin etiquetar [17].

Inducción de Lexicones por métodos semi supervisados de grafos

En 2009, Rao y Ravichandran realizaron un estudio sobre el enriquecimiento de un lexicon al agregarle más palabras mediante un modelo semi supervisado de propagación de etiquetas (**Label Propagation**) mediante grafos, en donde cada nodo del grafo correspondía a una palabra a la cual se le determinaría la polaridad. La idea del modelo era tener un conjunto de palabras previamente etiquetadas (semillas) y otro conjunto de palabras si etiquetar (objetivos), posteriormente se procedió a relacionar el universo de nodos mediante los enlaces entre las diferentes palabras haciendo que dichos enlaces codificaran las similitudes y relaciones entre las palabras conexas, para determinar estas similitudes, se usó la función de ***synonymy*** y ***hypernymy*** de WordNet. El proceso de propagación se realizó de manera iterativa etiquetando a cado nodo objetivo según los valores de polaridad de los nodos conexos a él. Al evaluar este método de propagación de etiquetas

en 3 corpus de datos con palabras previamente etiquetadas (francés e hindi), tomando en cada caso una partición del 50% para training y 50% para validación en test, se obtuvieron resultados de F1-Score del 93% para hindi, 82,4% para francés [18].

5. Objetivo específico 1: Referenciación bibliográfica sobre modelos supervisados y no supervisados para análisis de polaridad en texto

Teniendo en cuenta la breve descripción en el marco teórico sobre la naturaleza de los modelos de aprendizaje de máquina basados en soluciones de aprendizaje supervisado y no supervisado, se realizará una revisión del estado del arte de 2 enfoques en específico aplicables al análisis de sentimientos para textos cortos, uno será un enfoque basado en un modelo No Supervisado y el otro será un modelo Auto supervisado (***Self-Supervised***). El objetivo de esta sección es ahondar sobre las bases teóricas y prácticas de estos modelos con miras a lograr un entendimiento que permita responder preguntas cómo: ¿Qué tipo de preprocesamiento de datos se debe aplicar al corpus de tweets para entrenar el modelo?, ¿Cuáles son los outputs del modelo?, ¿cómo se interpretan dichas salidas?, ¿Qué limitaciones tiene el modelo?

De estos dos conjuntos de modelos a revisar a profundidad el modelo No Supervisado será ***K-Means***, mientras que se explorará en el enfoque Auto Supervisado una combinación de LBS (***Lexicon Based Model***) para la generación de pseudo etiquetas para un subconjunto de los tweets para luego entrenar un modelo Supervisado con el objetivo de poder etiquetar el restante del corpus de datos. Esta exploración bibliográfica y de antecedentes tiene el objetivo de cimentar el posterior entrenamiento de los 2 modelos con el set de datos recopilado, queriendo así dotar al proyecto aplicado de las herramientas de contraste necesarias para evaluar la idoneidad de los modelos No Supervisados en comparación con los modelos Auto Supervisados sobre los cuales se han estructurado con mayor frecuencia las soluciones de análisis de polaridad.

5.1. Modelos de Aprendizaje No Supervisados

En este contexto de modelos No Supervisados para el análisis de polaridad en textos se resalta el trabajo de Alshari & Azma (2017), en el cual exploraron la utilización de un lexicon con 4.783 palabras con polaridad negativa y 2.005 con polaridad positiva, vectorizando estos dos tipos de palabras mediante la técnica de ***Word2Vec***, el cual transforma cada palabra con un vector numérico que modela las relaciones de esta con otras palabras de las frases con el objetivo de generar 2 centroides, un positivo y uno negativo para comprar con estos dos conjuntos de referencia a los textos analizados y así poder identificar la polaridad con la cual se presentaba mayor afinidad, para esto se utilizó la técnica de comparación por medio de la ***similitud del coseno***, la cual permite comparar dos vectores según sus orientaciones y magnitudes [19].

Otro enfoque de aprendizaje No Supervisado para el análisis de textos se muestra en el estudio de Mass & Daly (2011), el cual es un modelo probabilístico en el que se calcula la matriz de ***"la función de energía***", la cual representa las frecuencias de aparición las palabras que componen el vocabulario en cada uno de los documentos del corpus, de tal manera que cada una de las columnas ***w*** de la matriz es la representación vectorial de cada palabra. Inicialmente esta aproximación tiene un buen desempeño en el análisis de similaridades semánticas, ya que cada vector ***w*** se puede comparar los otros vectores y mediante similitud de coseno se pueden agrupar, evidenciando que estas agrupaciones mostraban similitudes en los tópicos de cada documento, sin embargo, Mass & Daly muestran que la expansión del modelo de vectores de palabas se pudo ampliar al análisis de polaridad mediante el uso de un segundo modelo con polaridades etiquetadas, aproximándose más a un modelo auto supervisado [11].

Por último, aunque no fue un trabajo aplicado al análisis de polaridad, Cutting et al (1992) desarrollaron un algoritmo de clustering para textos con aplicaciones en la detección de tópicos mediante el análisis de frecuencias de palabras clave llamado el Scatter/Gather. En este trabajo

mostraron como con un proceso de vectorización por Tf-idf podían capturar las frecuencias de palabras clave con alta capacidad de diferenciar los textos que las componían para luego entrar un modelo de clustering con K-Means. En este caso, crearon varios modelos de clustering seleccionando aleatoriamente varias observaciones como centroides y cambiando los valores del hiperparametro K, para luego mediante la ***similitud de coseno*** encontrar las similitudes con textos de referencia vectorizados [20].

5.2. Lexicons Based Models (LBM)

El sustrato sobre el que se cimentan los Lexicons Based Models (LBM) son los ***Lexicones de Sentimientos (Sentiment Lexicons)*** , los cuales son listas de palabras que están etiquetadas y validados de manera manual (generalmente) con una orientación semántica que depende de la naturaleza de la palabra. Estos grupos de palabras y expresiones se constituyen según el idioma objetivo y pueden ser tan específicos en el conjunto de palabras como se requiera, por ejemplo, ***VADER*** es un Lexicon Based Model creado por Hutto y Gilbert para el Georgia Institute of Technology enfocado enfáticamente el análisis sentimiento de textos cortos extraídos de redes sociales [21].

Existen diferentes tipos de Lexicones de sentimientos, entre ellos resaltan los ***Lexicones de Orientación Semántica***, cuyo rango de polaridad pueden ser ***Positivo*** y ***Negativo*** mediante un score numérico (por ejemplo-1;1), algunos ejemplos de estos tipos de Lexicones son LIWC, GI y Hu-Liu04 [21]. Hutto y Gilbert (2014) explican que el desarrollo de estos tipos de Lexicones es intensivo en tiempo y en recursos económicos, sin embargo, se ha mostrado mediante evidencia que los LBM basado en este tipo de Lexicones tienen desempeños comparables con modelos de Machine Learning Supervisado en tareas de análisis de sentimiento.

Por otro lado, los ***Lexicones Intensidad (Valence-Based)*** no solo buscan identificar la orientación

semántica de las palabras, sino, identificar la intensidad de dicha polaridad mediante un score numérico continuo con el objetivo de poder hacer análisis de contraste en sentimientos, por ejemplo, mediante este tipo de lexicones se pueden desarrollar modelos que identifiquen variaciones de favorabilidad y des favorabilidad de textos sobre alguna temática, por ejemplo, un producto o un servicio [22]. Los resultados obtenidos por este enfoque de clustering con centroides inicializados fue aplicado al set de datos de *Movie Review Dataset*, y al comparar el performance del modelo en contraste con un modelo de Logistic Regression y otro de Support Vector Machine mostró una mejora del 12.3% frente a la regresión logística y de un 23.3% en contraste con SMV [19].

Teniendo como base los lexicones, los modelos basados en Lexicones (LBM) son modelos que, mediante reglas algorítmicas, por ejemplo, al analizar las valencias de las diferentes palabras por medio de métodos de propagación, asignan una polaridad a un texto al comparar características semánticas, contextuales y de valencia aplicadas a las palabras que conforman dicho texto. Estos modelos tienen algunas ventajas sobre modelos de Machine Learning Supervisados, resaltando que en primer lugar no se necesita contar con un corpus de textos previamente etiquetados para el desarrollo de tareas de análisis de sentimiento con una efectividad competitiva, y en segundo lugar, la ejecución de este tipo de modelos no consumen tanto tiempo ni recursos computacionales [21].

En el desarrollo de este proyecto de grado, se utilizó el LBM de VADER, tanto para realizar el proceso de Pseudo etiquetado de tweets con alto grados de polaridad para servir de insumo para el posterior entrenamiento del modelo de Machine Learning Supervisado (modelo hibrido), cómo para la creación de centroides con palabras que revisten altos puntajes de polaridad para la inicialización de los modelos de K-Means.

5.3. Modelos Auto Supervisados

En el campo del análisis de sentimientos esta técnica se ha aplicado con un eje central situado en que en una primera etapa se realiza la generación de ***pseudo etiquetas*** a los textos del corpus de datos gracias a un LBM (***Lexicon Based Model***) por medio de un proceso de análisis de sentimiento en el cual se utiliza un Lexicon, cómo por ejemplo LIWC, GI, Hu-Liu04 [21], con el objetivo de hacer el contraste de la polaridad que contiene cada una de las palabras que componen el texto contra los valores de ***polaridad*** asignada en el Lexicon a dichas palabras (si las contiene, claro está) [21]. Esta asignación de polaridad arrojada por el LBM puede ser de diferente índole si el Lexicon usado es de ***Orientación semántica*** o de ***Intensidad de sentimientos***.

Una vez se tienen la asignación de polaridades dadas por el Lexicon, se toman los textos con las asignaciones de polaridad más altas (las más positivas y las más negativas) con el objetivo de servir de sustrato para el posterior entrenamiento de un modelo de aprendizaje supervisado, por ejemplo, SMV, LR entre otros. [22] con estos datos etiquetados.

El último paso, consta de utilizar un modelo supervisado para etiquetar los textos que presentaron un desempeño por debajo del lumbral de alta polaridad [22]. Esta metodología presenta varias ventajas.

- En primer lugar, no requiere textos etiquetados previamente [21].
- Los tiempos de procesamiento de asignación de ***pseudo etiquetas*** son computacionalmente económicas sin necesidad de sacrificar *accuracy*, ya que Hutto et al (2014) muestra que con un "equipo de cómputo de características normales (3 GHz de procesador y 6GB de RAM) un corpus que toma unas fracciones de segundo para analizarse con el LBM VADER pueden tomar varias horas para procesarse" con modelos más

complejos de ejecución 100% supervisada, por ejemplo, Support Vector Machines [21, p. 224].

- Buen desempeño en tareas de clasificación de texto, por ejemplo, Rendón y Gil (2022), muestran cómo el resultado de un modelo hibrido ***Self-Supervised*** alcanzo en promedio un desempeño del 75% en asignación de polaridad (se utilizaron varios modelos supervisados en la segunda etapa, entre ellos RL, SVM y NB) [17].

Sin embargo, también presentan algunas desventajas y restricciones:

- Al igual que los modelos basados exclusivamente en LBM, el enfoque de aprendizaje auto supervisado presenta los problemas con el análisis de vocabulario especifico que no esté presente en el Lexicon, por lo cual estos modelos se deben estar actualizando con relativa frecuencia para no perder substancialmente su eficacia [22]. En el caso de este proyecto de grado, este es uno de los puntos álgidos, ya que el lenguaje típico del contexto político colombiano es muy específico y puede variar ampliamente según variables sociales, geográficas y culturales dentro de nuestro mismo país.
- Están restringidos a utilizar Lexicones preexistentes, teniendo en cuenta la restricción subyacente de tiempo y recursos económicos para desarrollar un Lexicon específico desde ceros [21].
- Puede haber deficiencias en la identificación de mensajes no estructurales en el lenguaje, cómo el sarcasmo o el doble sentido, cuya presencia puede ser cotidiana en los tweets, y más específicamente, en un contexto tan puntual como el ámbito político colombiano.

6. Objetivo específico 2: Explorar y seleccionar las técnicas no supervisadas de ciencia de datos que serán utilizadas para el análisis de polaridad para tweets.

Cómo se mostró previamente en la metodología, este trabajo de grado pasó por una serie de etapas a nivel del ciclo natural de un proyecto de Ciencia de Datos. En esta sección se abarcarán las fases de análisis exploratorio del corpus de datos recolectado, luego, se procede a mostrar los resultados de la fase de preprocesamiento de datos, la cual se hizo con el objetivo de transformar las variables del corpus de datos para que fueran óptimas para el procesamiento de los diferentes modelos a evaluar.

6.1. Construcción del corpus de datos

El primer paso para poder implementar los 2 modelos propuestos y luego contrastar su comportamiento fue la construcción del dataset o corpus de datos. Para esta tarea se evaluaron varias posibilidades para la extracción de tweets relacionados en específico con el contexto político colombiano. Para esto, con el asesoramiento de la directora del Proyecto de grado, se puso un número objetivo de 5.000 tweets para poder contener un corpus de datos que tuviera la oportunidad de ofrecer un sustrato de entrenamiento suficiente para los modelos a explorar.

6.1.1. Tecnologías de extracción analizadas

En primer lugar, se exploró la posibilidad de extraer los tweets por medio de la ***API v1.1*** de Twitter (ahora X) en la cual se tenía una versión gratuita para la extracción de máximo 1.000 tweets cada 24 horas [23], para tal objetivo se creó una cuenta de desarrollador en Twitter y se generaron las llaves de acceso para la API (***Keys***). Posteriormente, se creó un script con el objetivo de poder acceder a los tweets dando por parámetro el ***Hashtag,*** sin embargo, al ejecutarse el script Twitter

expone que es necesario el acceso por medio de la ***API v2***, la cual necesita una credencial pagada de 100 USD/mes para poder descargar tweets por parte de la API.

```python
import credentials
import tweepy

## Parametrizacion de credenciales
##def get_auth():
auth = tweepy.OAuthHandler(credentials.API_KEY, credentials.API_SECRET_KEY)
auth.set_access_token(credentials.ACCESS_TOKEN, credentials.ACCESS_TOKEN_SECRET)

# creacion de objeto API
api = tweepy.API(auth)

public_tweets = api.home_timeline()
for tweet in public_tweets:
    print(f'{tweet.user.screen_name}:\n{tweet.text}\n{"*"*60}')

id = None
counter = 0
while counter <= 3000:
    tweets = api.search(q="Petro", lang ="es", tweet_mode="extended", max_id= id)
    id = tweets.id

    #Agregar los tweets a archivo para posterior lectura
    for tweet in tweets:
        if tweet.full_text.startswith('RT'):
            counter += 1
            continue
        f = open('./onepiece.txt', 'a', encoding='utf-8')
        f.write(tweet.full_text + '\n')
        f.close
        counter += 1
    id = tweet.id
    print(counter)
```

Ilustración 3 Script para extracción de Tweets mediante API v1.1

```
tweepy.errors.Forbidden: 403 Forbidden
453 - You currently have access to a subset of Twitter API v2 endpoints and limited v1.1 endpoints (e.g. media post, oauth) only. If you need access to this endpoint, you may need a different access level. You can learn more here: ht
tps://developer.twitter.com/en/portal/product
```

Ilustración 4 Mensaje de error al intentar extraer tweets por API v1.1

Al tener en cuenta este impedimento, se procedió a explorar otras alternativas para la construcción del corpus de datos, siguiendo con la técnica de web scraping, mediante la cual se extrae información de la web por medio de la obtención del contenido HTML para filtrarla y almacenarla de manera automática por medio de un Script de la librería ***BeautifulSoup;*** Esta es una librería dentro de Python que permite hacer WebScraping sobre cualquier URL. La opción no fue viable ya que Twitter tiene una protección anti-scraping, lo cual impidió extraer la información mediante este medio. Se intentó hacer el procedimiento con un script que renderizara la respuesta de Twitter, obteniendo los siguientes resultados.

```python
import requests
from bs4 import BeautifulSoup

url = 'https://twitter.com/infopresidencia'

try:
  response = requests.get(url)
  print(response.text)
  soup = BeautifulSoup(response.text, 'html.parser')
  print(soup('body.div'))
except Exception as e:
  print(repr(e))
```

Ilustración 5 Script sin renderizado de la pagina

La ejecución del código fuente, mostraba el HTML de la página web de Twitter solicitando que se realice la activación del JavaScript.

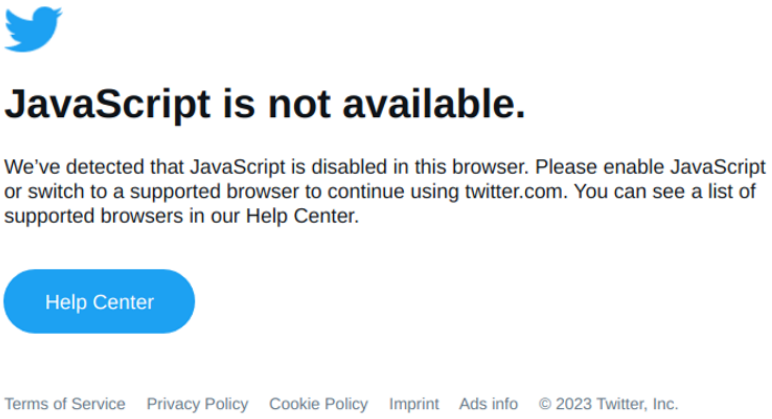


Ilustración 6 Respuesta a petición por parte de Twitter (X)

Cómo otra opción, se procedió a utilizar la versión gratuita de la herramienta **Octoparse**, la cual permite diseñar y ejecutar tareas de **Scraping** sin necesidad de diseñar los scripts, sino mediante la selección de los campos que se quieren capturar por medio de una interfaz gráfica, permitiendo extraer los componentes de la página web objetivo y descargarlos en un fichero .csv.

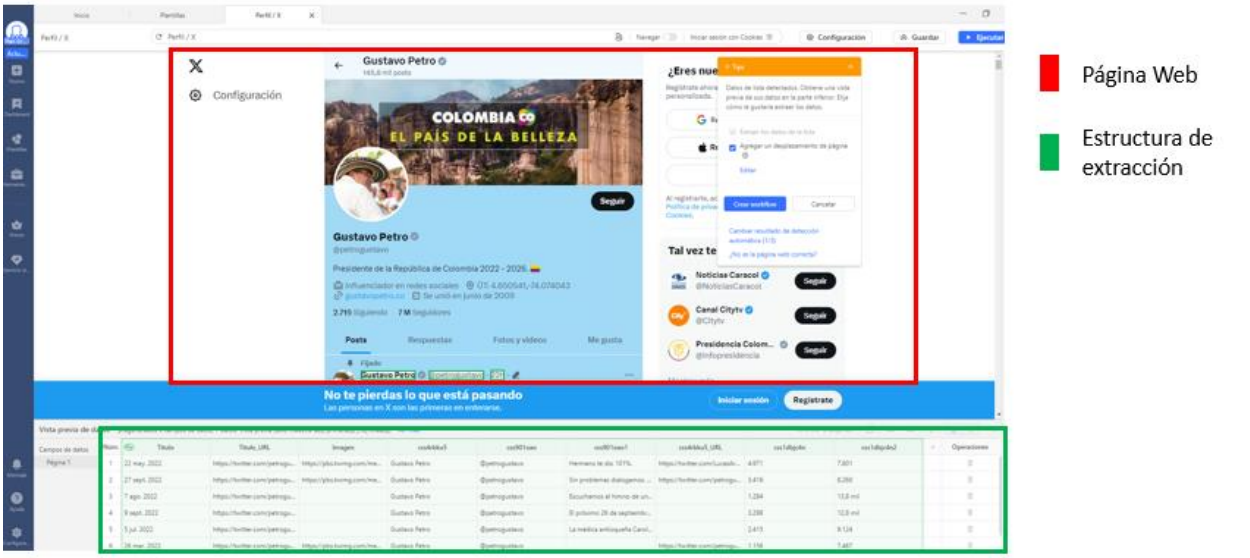
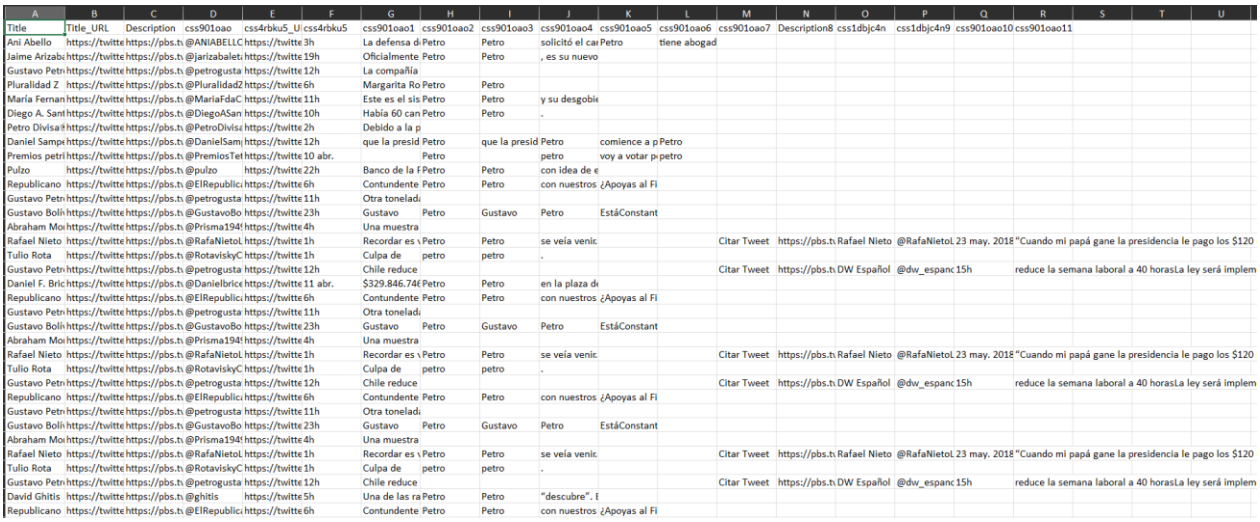


Ilustración 7 Ejemplo de extracción de Tweets por Octoparse

Sin embargo, la exportación de estos datos en archivos .CSV no son consistentes en la estructura de los campos en cada uno de los tweets extraídos, separando desde el origen de la tarea de web scrapping secciones de los textos por diferentes “separadores”. Esto impidió que se pudieran consolidar los tweets a extraer en una sola base de datos con campos homogéneos.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
Title	Title_URL	Description	css901oao	css4rbku5_U	css4rbku5	css901oao1	css901oao2	css901oao3	css901oao4	css901oao5	css901oao6	css901oao7	Description8	css1dbjc4n	css1dbjc4n9	css901oao10	css901oao11			
Ani Abello	https://twitte	https://pbs.t	@ANIABELLC	https://twitte	3h	La defensa d	Petro	Petro	solicitó el cas	Petro	tiene abogad									
Jaime Arizabi	https://twitte	https://pbs.t	@jarizabalet	https://twitte	19h	Oficialmente	Petro	Petro	, es su nuevo											
Gustavo Petr	https://twitte	https://pbs.t	@petrogusta	https://twitte	12h	La compañía														
Pluralidad Z	https://twitte	https://pbs.t	@Pluralidad2	https://twitte	6h	Margarita Ro	Petro	Petro												
María Fernan	https://twitte	https://pbs.t	@MariaFdaC	https://twitte	11h	Este es el sis	Petro	Petro	y su desgobie											
Diego A. Sant	https://twitte	https://pbs.t	@DiegoASan	https://twitte	10h	Había 60 can	Petro	Petro	.											
Petro Divisa	https://twitte	https://pbs.t	@PetroDivisa	https://twitte	2h	Debido a la p														
Daniel Samp	https://twitte	https://pbs.t	@DanielSam	https://twitte	12h	que la presid	Petro	que la presid	Petro	comience a p	Petro									
Premios petri	https://twitte	https://pbs.t	@PremiosTet	https://twitte	10 abr.		Petro		petro	voy a votar p	petro									
Pulzo	https://twitte	https://pbs.t	@pulzo	https://twitte	22h	Banco de la f	Petro	Petro	con idea de e											
Republicano	https://twitte	https://pbs.t	@ElRepublic	https://twitte	6h	Contundente	Petro	Petro	con nuestros	¿Apoyas al Fi										
Gustavo Petr	https://twitte	https://pbs.t	@petrogusta	https://twitte	11h	Otra tonelad														
Gustavo Bolí	https://twitte	https://pbs.t	@GustavoBo	https://twitte	23h	Gustavo	Petro	Gustavo	Petro	EstáConstant										
Abraham Mo	https://twitte	https://pbs.t	@Prisma194	https://twitte	4h	Una muestra														
Rafael Nieto	https://twitte	https://pbs.t	@RafaNietoL	https://twitte	1h	Recordar es v	Petro	Petro	se veía venir.			Citar Tweet	https://pbs.t	Rafael Nieto	@RafaNietoL	23 may. 2018	"Cuando mi papá gane la presidencia le pago los $120			
Tulio Rota	https://twitte	https://pbs.t	@RotaviskyC	https://twitte	1h	Culpa de	petro	petro	.											
Gustavo Petr	https://twitte	https://pbs.t	@petrogusta	https://twitte	12h	Chile reduce						Citar Tweet	https://pbs.t	DW Español	@dw_espanc	15h	reduce la semana laboral a 40 horasLa ley será implem			
Daniel F. Bric	https://twitte	https://pbs.t	@Danielbric	https://twitte	11 abr.	$329.846.74(	Petro	Petro	en la plaza de											
Republicano	https://twitte	https://pbs.t	@ElRepublic	https://twitte	6h	Contundente	Petro	Petro	con nuestros	¿Apoyas al Fi										
Gustavo Petr	https://twitte	https://pbs.t	@petrogusta	https://twitte	11h	Otra tonelad														
Gustavo Bolí	https://twitte	https://pbs.t	@GustavoBo	https://twitte	23h	Gustavo	Petro	Gustavo	Petro	EstáConstant										
Abraham Mo	https://twitte	https://pbs.t	@Prisma194	https://twitte	4h	Una muestra														
Rafael Nieto	https://twitte	https://pbs.t	@RafaNietoL	https://twitte	1h	Recordar es v	Petro	Petro	se veía venir.			Citar Tweet	https://pbs.t	Rafael Nieto	@RafaNietoL	23 may. 2018	"Cuando mi papá gane la presidencia le pago los $120			
Tulio Rota	https://twitte	https://pbs.t	@RotaviskyC	https://twitte	1h	Culpa de	petro	petro	.											
Gustavo Petr	https://twitte	https://pbs.t	@petrogusta	https://twitte	12h	Chile reduce						Citar Tweet	https://pbs.t	DW Español	@dw_espanc	15h	reduce la semana laboral a 40 horasLa ley será implem			
Republicano	https://twitte	https://pbs.t	@ElRepublic	https://twitte	6h	Contundente	Petro	Petro	con nuestros	¿Apoyas al Fi										
Gustavo Petr	https://twitte	https://pbs.t	@petrogusta	https://twitte	11h	Otra tonelad														
Gustavo Bolí	https://twitte	https://pbs.t	@GustavoBo	https://twitte	23h	Gustavo	Petro	Gustavo	Petro	EstáConstant										
Abraham Mo	https://twitte	https://pbs.t	@Prisma194	https://twitte	4h	Una muestra														
Rafael Nieto	https://twitte	https://pbs.t	@RafaNietoL	https://twitte	1h	Recordar es v	Petro	Petro	se veía venir.			Citar Tweet	https://pbs.t	Rafael Nieto	@RafaNietoL	23 may. 2018	"Cuando mi papá gane la presidencia le pago los $120			
Tulio Rota	https://twitte	https://pbs.t	@RotaviskyC	https://twitte	1h	Culpa de	petro	petro	.											
Gustavo Petr	https://twitte	https://pbs.t	@petrogusta	https://twitte	12h	Chile reduce						Citar Tweet	https://pbs.t	DW Español	@dw_espanc	15h	reduce la semana laboral a 40 horasLa ley será implem			
David Ghitis	https://twitte	https://pbs.t	@ghitis	https://twitte	5h	Una de las ra	Petro	Petro	"descubre". E											
Republicano	https://twitte	https://pbs.t	@ElRepublic	https://twitte	6h	Contundente	Petro	Petro	con nuestros	¿Apoyas al Fi										

Ilustración 8 Ejemplo de archivo exportado desde Octoparse con heterogeneidad de campos

Por último, se procedió a hacer una extracción manual de los tweets desde la página de Twitter, para lo cual se contrató a un tercero para extraer de manera canónica los tweets buscando por *Hashtag* a: #Petro¸ #Uribe, #IVANDUQUE, #FRANCIAMARQUEZ, #CABAL y #LAURASANABRIA.

6.1.2. Corpus de datos extraído

Teniendo en cuenta las restricciones de presupuesto y restricciones tecnológicas para la extracción automática de tweets, se decidió hacer una extracción por medio de técnica de web scrapping manual, en la cual se exploró la página de Twitter y se obtuvieron las variables de FECHA, TWEET y HASHTAG. De este manera se pudieron extraer los siguientes tweets:

Hashtag	Tweet
Cabal	833
Francia Marquez	833
Ivan Duque	833
Petro	833
Uribe	833
laura sarabia	658

Ilustración 9 Distribución de datos por Hashtag

	A	B	C
	Tweet	Fecha	Hashtag
1	Pregunta: ¿La @FiscaliaCol entregó el informe de avan	30/05/23	laura sarabia
2	Wow....que supernoticia....aunque está más coherente	30/05/23	laura sarabia
3	Viejo, lo dantesco es que la inútil de Laura Sarabia, cor	30/05/23	laura sarabia
4	El polémico polígrafo de Laura Sarabia | #WALLYOPINA	30/05/23	laura sarabia
5	Señor presidente @petrogustavo cómo le explica al mu	30/05/23	laura sarabia
6	La Jefe de Gabinete, Laura Sarabia, tiene un salario de	29/05/23	laura sarabia
7	Es increíble que en la @WRadioColombia no hablen un	29/05/23	laura sarabia
8	¿Quién dio la orden de sacar al periodista de @Revista	30/05/23	laura sarabia
9	Atacar y silenciar a la prensa como parte de una sola e	30/05/23	laura sarabia
10	Laura Sarabia y el reclamo a su niñera óleo sobre lienz	30/05/23	laura sarabia
11	Lo ocurrido con Marelbys Meza, exniñera del hijo de La	30/05/23	laura sarabia
12	La señora Marelbys Meza la exniñera que trabajó con L	30/05/23	laura sarabia
13	Los zurdos son realmente patéticos y envidiosos, como	30/05/23	laura sarabia
14	Petro citó a @AABenedetti a rendir cuentas por tema d	30/05/23	laura sarabia
15	A ésta imagen de Petro, Benedetti, Barreras, les falta L	30/05/23	laura sarabia
16	🔴URGENTE: La Fiscalia Allana en este momento LA	30/05/23	laura sarabia
17	El escándalo al que Semana le dio portada va quedando	30/05/23	laura sarabia
18	Esto es el colmo de la hipocresía; hablan de dignificaci	30/05/23	laura sarabia
19	LAURA SARABIA @laurisarabia LA QUE MUCHOS DE	30/05/23	laura sarabia
20	El hampa que gobierna a Colombia. @laurisarabia con	30/05/23	laura sarabia
21	Mi gente, pillen como algunos medios organizaron la es	27/05/23	laura sarabia
22	Es asombroso como Laura Sarabia pasó de ser una fur	30/05/23	laura sarabia
23	Pero qué maravilla, ¿ahora estamos ante un complot, s	30/05/23	laura sarabia
24	#LOÚLTIMO | Agentes del CTI, tras entrar a la Casa de	30/05/23	laura sarabia
25	.....el altanero fiscal Barbosa da protección a mujer por	30/05/23	laura sarabia
26	Petro, le tiene prohibido a la @WRadioColombia y a su	30/05/23	laura sarabia
27	¡Nojoda loco! que descaro tan hijueputa.Hasta la procur	30/05/23	laura sarabia
28	Ella es Marelbys Meza, exniñera del hijo de Laura Sara	30/05/23	laura sarabia
29	A Laura Sarabia la defienden:Gustavo PetroCielo Rusin	30/05/23	laura sarabia
30	¿Un complot? ¿O sea que destaparon todo lo ocurrido	30/05/23	laura sarabia
31	La joven Laura Sarabia, 29 años, pasó de tener un patri	27/05/23	laura sarabia

Ilustración 10 Ejemplo de tweets extraído manualmente

El corpus de datos se guardó en un archivo de Excel y unificó todas las búsquedas por hashtags dadas por parámetro, buscando tener una cantidad similar de tweets para cada hashtag.

a. 833 con hastag **Petro**
b. 833 con hastag **Uribe**
c. 833 con hastag **Francia Márquez**
d. 833 con hastag **María Fernanda Cabal**
e. 833 con hastag **Iván Duque**
f. 658 con hastag **Laura Sarabia**

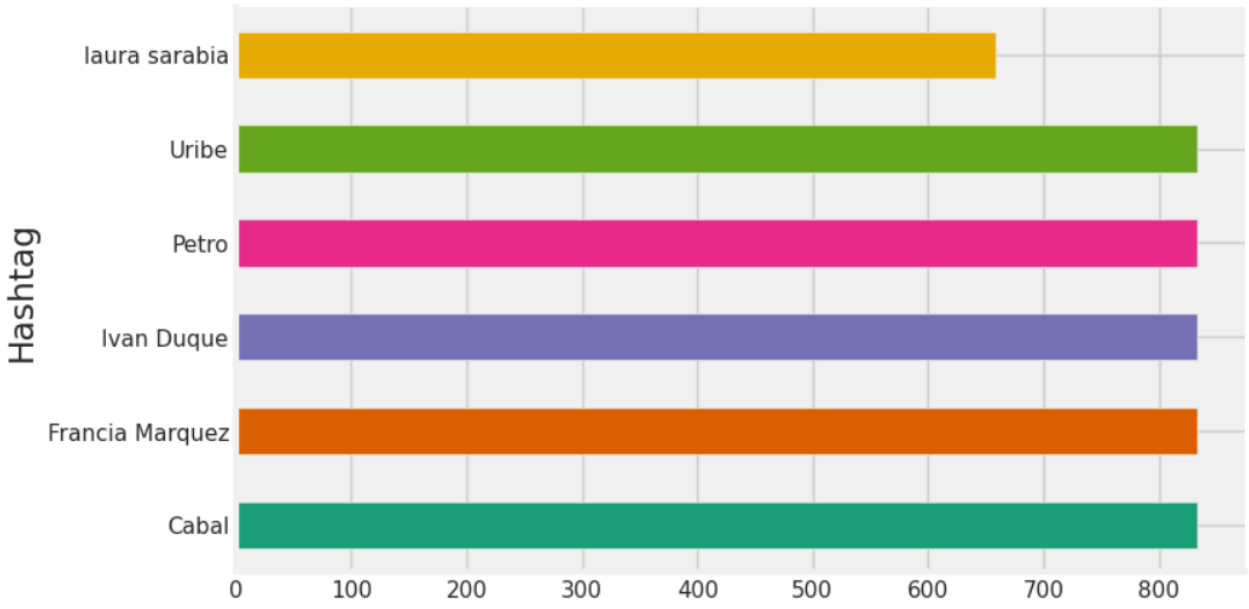


Ilustración 11 Distribución de Tweets extraídos

Cada uno de estos conjuntos de tweets fue ejecutado por una tarea de extracción manual independiente y fue guardada en un archivo .csv, para luego ser unificado en un solo fichero de datos con los 4.823 tweets.

El objetivo de estos hashtags fue tener una muestra de tweets de reconocidos representantes de las dos tendencias políticas que están actualmente en orillas diametralmente opuestas en el contexto político del país, esto con el objetivo de no sesgar ni el análisis ni los resultados con ningún tinte de politización, si no de ver el comportamiento de la polaridad de los tweets hechos por los

usuarios de Twitter (X) sobre estos personajes de la vida pública colombiana.

Posteriormente, se procedió a realizar un proceso de etiquetado manual por parte de un tercero con el objetivo de no interferir en la asignación de polaridad, dando las siguientes etiquetas posibles a los tweets: ***Positivo*** en el caso en que el tweet este expresando una posición positiva o a favor del sujeto (hashtag), ***Negativo*** en el caso en que el tweet este expresando una posición negativa o en contra del sujeto (hashtag) y ***Neutral*** en caso de que el tweet no tenga una postura clara frente al sujeto (hashtag). Este proceso de etiquetado se realiza con el objetivo principal de poder dar un punto de referencia para contrastar el resultado del proceso de etiquetado del método de ***Self Supervised*** y el análisis derivado del proceso de clustering del modelo ***No Supervisado*** contra la identificación de polaridad por una fuente humana.

Cómo resultado del ejercicio de extracción de tweets y del proceso de etiquetado manual realizado por externo se obtiene la siguiente estructura del corpus de datos:

```
data_raw
```

	Tweet	Fecha	Hashtag	Polaridad
0	Oposición: hay que multiplicar por 5 las UCI\n...	2023-05-12	Petro	Negativo
1	Acabamos de radicar la denuncia contra el seño...	2023-05-12	Petro	Positivo
2	A Uribe le dicen paraco\nPetro ES un terrorist...	2023-05-11	Petro	Negativo
3	Petro incendia el país en su época de guerrill...	2023-05-11	Petro	Negativo
4	No pos #CronicasChairas\nGustavo Petro: “Si lo...	2023-05-11	Petro	Negativo
...	...	...	...	...
4818	Gracias Laura Sarabia. Sin tu actuar estos tit...	2023-06-01	laura sarabia	Negativo
4819	Te respeto y te admiro Presidente \n@petrogust...	2023-06-01	laura sarabia	Negativo
4820	Exniñera de Laura Sarabia habría sido intercep...	2023-06-01	laura sarabia	Negativo
4821	Urgente. Armando Benedetti CONFESÓ prácticamen...	2023-06-01	laura sarabia	Neutro
4822	Chuzadas ordenadas por Laura Sarabia?	2023-06-01	laura sarabia	Negativo

4823 rows × 4 columns

Ilustración 12 Data sin procesar extraída de Octoparse y etiquetada manualmente

6.2. Análisis exploratorio

En primer lugar, los campos extraídos de los tweets son los siguientes:

- [Tweet]: Es el texto del tweets extraído, este campo puede contener componentes tanto alfanuméricos (A-Z; 0-9) y no alfanuméricos (“#$%&/?, etc.)
- [Fecha]: Fecha de publicación de los tweets
- [Hashtag]: Hashtag de búsqueda de los tweets, hace referencia al sujeto etiquetado en los tweets, en este caso, al actor político de estudio.
- [Polaridad]: Etiqueta de polaridad asignada manualmente por un externo.

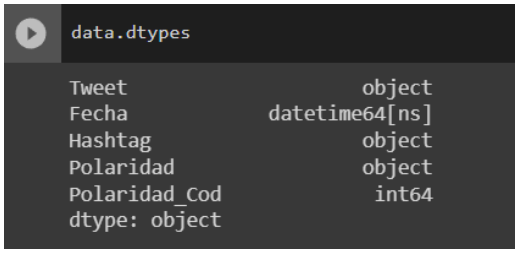


Ilustración 13 Tipo de datos de campos

En primer lugar, se especifica que la ***variable objetivo*** va a ser la etiqueta de polaridad, cual podrá tener los 3 estados expuestos anteriormente (*Positivo, Negativo y Neutral)*, por lo tanto, las otras variables se podrán clasificar como ***variables predictoras***, sin embargo, esto no quiere decir que en la construcción de los modelos se vayan a tomar todas las variables predictoras para el entrenamiento de estos, ya que la única variable predictora será la del texto del tweet transformado por un proceso de vectorización.

Al analizar la distribución de la variable [Polaridad], se puede ver que el set de datos esta desbalanceado, ya que un 64% de las observaciones tienen una polaridad ***Negativa***, el 19.6% tienen una polaridad ***Neutral*** y el 16.5% están etiquetados con polaridad ***Positiva***

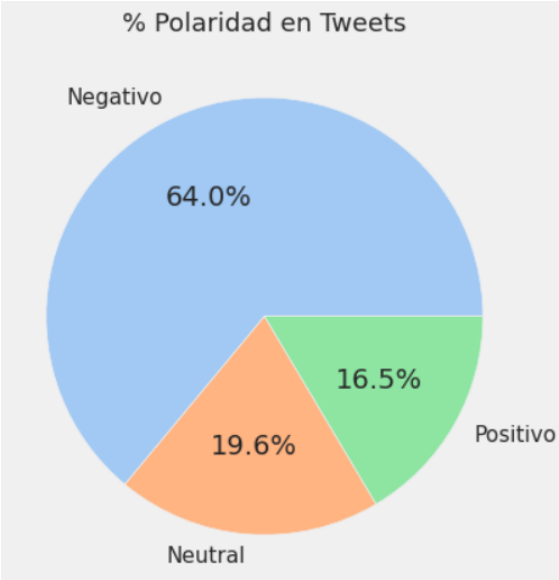


Ilustración 14 Distribución porcentual de polaridades de Tweets

Polaridad	
Negativo	3085
Neutro	944
Positivo	794

Ilustración 15 Distribución total de polaridad de Tweets

A continuación, se muestran las fechas en las cuales fueron publicados los tweets recolectados, la mayor cantidad de tweets tienen fechas entre el 8 de mayo hasta el 1 de junio del 2023, esto se debe al periodo en la cual se realizaron las tareas de extracción manual de los tweets.

Ilustración 16 Fechas de los tweets extraídos

Así mismo, si analizamos el comportamiento de las polaridades segmentadas por los actores políticos se puede ver el siguiente comportamiento:

Hashtag	Polaridad	
Cabal	Negativo	597
	Neutro	146
	Positivo	90
Francia Marquez	Negativo	448
	Neutro	164
	Positivo	221
Ivan Duque	Negativo	531
	Neutro	149
	Positivo	153
Petro	Negativo	512
	Neutro	157
	Positivo	164
Uribe	Negativo	629
	Neutro	81
	Positivo	123
laura sarabia	Negativo	368
	Neutro	247
	Positivo	43

Name: Polaridad, dtype: int64

Ilustración 17 Distribución por Hashtag/Polaridad

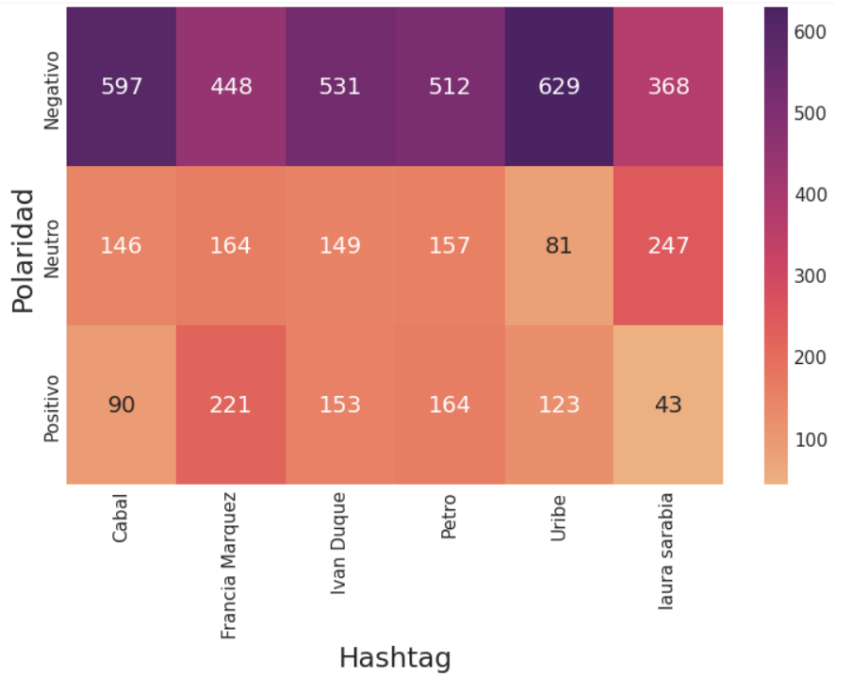


Ilustración 18 Heatmap de polaridad vs Hashtag

Se puede ver que los tweets con el hashtag ***"Cabal"*** y ***"Uribe"*** presentaron la mayor tasa de tweets ***Negativos***, con una proporción del 72% para el hashtag ***"Cabal"*** y del 76% para el hashtag ***"Uribe"***, por otra parte, los hashtags con menor tasa de tweets negativos fueron ***"Francia Márquez"*** y ***"Laura Sarabia"***, con un respectivo 54% y 56% de tweets con polaridad negativa.

Adicionalmente, se puede ver que en todos los hashtags las proporciones de tweets con polaridad negativa fue mayoritaria. Este comportamiento va a ser de importancia a la hora de generar el dataset de entrenamiento tanto para los modelos Auto Supervisados como para los modelos No Supervisados.

Por otro lado, al separar los tweets según la etiqueta asignada manualmente se puede visualizar por medio del grafico de ***Nube de Palabras*** según su etiqueta asignada de manera manual se obtienen los siguientes resultados (traducidas previamente al idioma inglés):

Al graficar las palabras más frecuentes con etiqueta asignada ***Positiva*** se puede ver que la palabra **“Petro”** es la de mayor tamaño, seguido de la palabra “**Uribe”**.

Ilustración 19 Nube palabras con etiqueta Positiva

Al graficar las palabras más frecuentes con etiqueta asignada ***Neutral*** se puede ver que la palabra **“Laura Sarabia”** es la de mayor tamaño, seguido de la palabra “**Petro”**.

Ilustración 20 Nube de palabras para etiqueta Neutral

Por último, al graficar las palabras más frecuentes con etiqueta asignada ***Negativa*** se puede ver que la palabra **"Uribe"** es la de mayor tamaño, seguido de la palabra "**Petro**".

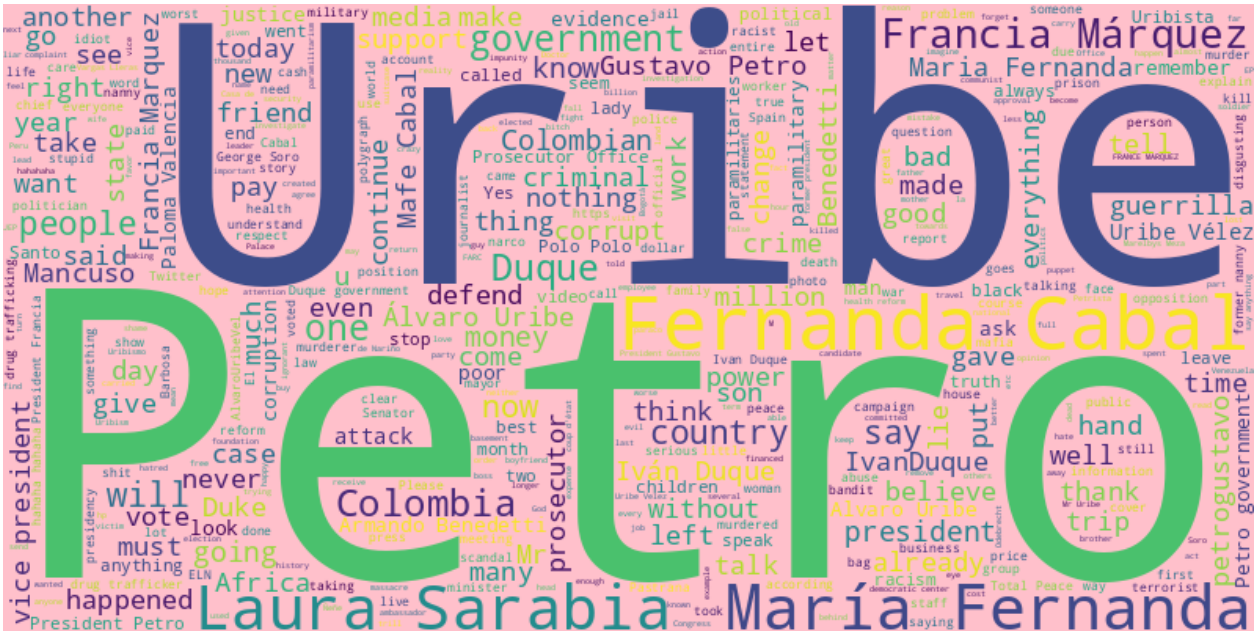


Ilustración 21 Nube de palabras para etiqueta Negativa

Este tipo de análisis nos permite ver que las categorías ***Positiva*** y ***Negativa*** tienen en sus términos más frecuentes tweets sobre los políticos **Petro** y **Uribe**¸ los cuales se posicionan en el contexto político colombiano en orillas diametralmente opuestas.

6.3. Preprocesamiento del corpus para ML

Los experimentos que se realizaron en este trabajo de grado estuvieron enmarcados en dos enfoques, uno Auto Supervisado y uno No Supervisado, pero antes se hizo necesario implementar una serie de pasos de preprocesamiento de los datos para hacerlos aptos para alimentar los modelos de cada uno de estos enfoques; para tal motivo, se procedió a aplicar la siguiente secuencia de pasos:

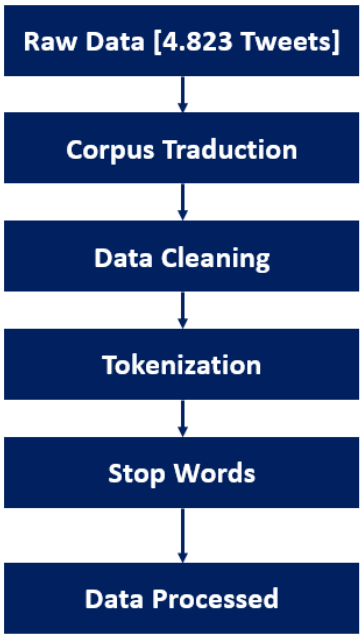


Ilustración 22 Fases de preprocesamiento de data para los modelos a implementar, producción propia

6.3.1. Traducción del corpus de tweets al idioma inglés

El primer proceso aplicado al set de datos de tweets fue la traducción del corpus de tweets al idioma inglés, esto se realizó con el objetivo de poder utilizar el LBM de VADER, ya que este modelo está diseñado para el análisis de sentimiento de textos de Social Media en inglés.

La decisión de realizar la traducción del corpus de datos al idioma inglés se basó en que la gran mayoría de lexicones de acceso libre están en el idioma inglés; esto se puede evidenciar en que en la fase de revisión bibliográfica solo se encontró un Lexicon en español, el cual es el Lexicon de la organización TASS [16], y para acceder a este Lexicon se debe hacer una solicitud formal a la entidad con tiempos de respuesta que podrían ser holgados y no estaban alineados con el cronograma del proyecto. Por otro lado, Rendon y Gil (2020) mostraron que al utilizar un proceso previo de traducción de textos universitarios en la narrativa del conflicto colombiano se pueden usar métodos basados en lexicones en inglés con buenos resultados.

Para la traducción de los tweets se utilizó la herramienta *Google Translate*, en la cual se cargó el archivo Excel con los tweets en español y se obtuvo un archivo con los tweets traducidos.

Ilustración 23 Resultado de proceso de traducción de Tweets

6.3.2. Limpieza de textos

En primer lugar, se realizó una fase de limpieza de datos mediante el uso de ***Regular Expressions (RE)***, mediante el cual se ejecutaron los siguientes tratamientos:

- Conversión de todos los caracteres a minúsculas (*Lower Case*).
- Filtrado de caracteres no alfanuméricos, quitando caracteres no pertenecientes al estándar AISC-II.
- Eliminación de saltos de línea (\n) dentro de un tweet.

```
# Limpieza de datos quitando signos de puntuación, caracteres no alfanumericos y combiernto todo a lower case
lower_case = lambda x: re.sub('[%s]' % re.escape(string.punctuation), ' ', x.lower())
alpha = lambda x: re.sub('\w*\d\w*', ' ', x)
space = lambda x: re.sub('\n',' ', x)

dataEN['Tweet_Procesado'] = dataEN.clean_tweet.map(alpha).map(lower_case)
dataEN['Tweet_Procesado'] = dataEN.clean_tweet.map(space)
dataEN
```

Ilustración 24 Chunk de código para limpieza de corpus

	Tweet	Fecha	Hashtag	Polaridad	Polaridad_Cod	Tweet_Procesado
0	Oposición: hay que multiplicar por 5 las UCI\n...	2023-05-12	Petro	Negativo	0	oposición hay que multiplicar por las uci\n...
1	Acabamos de radicar la denuncia contra el seño...	2023-05-12	Petro	Positivo	1	acabamos de radicar la denuncia contra el seño...
2	A Uribe le dicen paraco\nPetro ES un terrorist...	2023-05-11	Petro	Negativo	0	a uribe le dicen paraco\npetro es un terrorist...
3	Petro incendia el país en su época de guerrill...	2023-05-11	Petro	Negativo	0	petro incendia el país en su época de guerrill...
4	No pos #CronicasChairas\nGustavo Petro: “Si lo...	2023-05-11	Petro	Negativo	0	no pos cronicaschairas\ngustavo petro “si lo...
...	...	...	...	...	...	...
4818	Gracias Laura Sarabia. Sin tu actuar estos tit...	2023-06-01	laura sarabia	Negativo	0	gracias laura sarabia sin tu actuar estos tit...
4819	Te respeto y te admiro Presidente \n@petrogust...	2023-06-01	laura sarabia	Negativo	0	te respeto y te admiro presidente \n petrogust...

Ilustración 25 Resultado de limpieza de Lower Case y filtrado de No alfanuméricos

6.3.3. Tokenización y normalización de texto

Cómo se mostró en el marco teórico, el proceso de Tokenización es de vital importancia en el NLP ya que el modelo Supervisado en el enfoque hibrido que se va a utilizar en este proyecto requiere una transformación de los textos a formas vectoriales numéricas para su entrenamiento, y previo a esta transformación se necesita transformar los tweets en un arreglo de palabras separadas para su posterior análisis de relevancia mediante ***Frecuencia de término-frecuencia inversa (tf-idf).*** Con el objetivo de evaluar varias alternativas de Tokenización se utilizó la técnica del algoritmo de Porter y el de Lematización.

6.3.3.1. Stemmer de Porter

Uno de los procesos de Tokenización que se utilizaron fue el algoritmo de Porter, en el cual se busca

separar las palabras que componen los tweets y transformarlas a sus formas declinadas enfocándose en la eliminación de los sufijos y prefijos; mediante este proceso se puede transformar cada palabra a su forma raíz (stem) [4] y así se pueden mapear palabras con diferentes conjugaciones de manera que converjan a su forma básica. Por ejemplo, al aplicar el algoritmo de Porter a la palabra *"corriendo"* se obtendría su raíz, la cual es *"corr"* [5].

Para aplicar este algoritmo se utilizó el algoritmo de Porter de la librería NLTK de Python:

```
#Utilizacipon de Stemmimer de Porter de NLTK

from nltk.stem.porter import PorterStemmer

porter = PorterStemmer()

def tokenizer_porter(text):
    return [porter.stem(palabra) for palabra in text.split()]

dataEN["Tweet_Tokenizado"] = dataEN["Tweet"].apply(tokenizer_porter)
dataEN.head()
```

Ilustración 26 Chunk de código de Algoritmo de Porter

El resultado del proceso de Tokenización por medio del algoritmo de Porter da un arreglo con las palabras transformadas a sus formas declinadas.

	Tweet	Tweet_Tokenizado
0	Opposition: the ICUs must be multiplied by 5 G...	[opposition:, the, icu, must, be, multipli, by...
1	We have just filed a complaint against Mr. Joh...	[we, have, just, file, a, complaint, against, ...
2	They call Uribe paraco Petro IS a guerrilla te...	[they, call, urib, paraco, petro, is, a, guerr...

Ilustración 27 Resultado de Tokenización

6.3.3.2. Tokenización por Lematización

Por otro lado, el proceso de lematización se enfoca en la obtención de la forma canónica o lema de cada una de las palabras, en la cual se toma en cuenta su estructura gramatical, significado y conjugaciones [4]. Por ejemplo, para la misma palabra "*corriendo"* se obtendría el lema "*Correr"*. El proceso de lematización es computacionalmente más complejo que el algoritmo de Porter, sin embargo, al llevar las palabas a su forma canónica se considera que es un proceso más preciso y que evita algunos errores derivados del cálculo de raíces de declinación en Porter.

Para el proceso de Lematización se utilizó la función de lematización de WordNet, ya que como resalta Bird et al. (2009) la cual "*es recomendada para tareas en las cuales se desea compilar vocabularios de textos en una lista valida de lemas*".

```python
#Importación de libreria nltk
import nltk
nltk.download('wordnet')

#Creación de tokenizador
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
#creación de lemmatizador
lemmatizer = nltk.stem.WordNetLemmatizer()

def lemmatize_text(text):
    return [lemmatizer.lemmatize(w) for w in w_tokenizer.tokenize(text)]

#creación de columnas lemmatizada, en la cual se identifica
dataEN['Tweet_Lemmatizado'] = dataEN["Tweet_Procesado"].apply(lemmatize_text)
```

Ilustración 28 Chunk de código Función de lematización

Del mismo modo, el resultado del proceso de Lematización es un vector con las palabras de cada tweet en su forma canónica.

	Tweet	Tweet_Lemmatizado
4818	Thank you Laura Sarabia. Without your actions ...	[Thank, you, Laura, Sarabia., Without, your, a...
4819	I respect and admire you President @petrogusta...	[I, respect, and, admire, you, President, @pet...
4820	Laura Sarabia's former nanny would have been i...	[Laura, Sarabia's, former, nanny, would, have,...
4821	Urgent. Armando Benedetti practically CONFESSE...	[Urgent., Armando, Benedetti, practically, CON...

Ilustración 29 Resultado de proceso de Lematización

Cómo se muestra más adelante, tanto la técnica de Stemmer de Porter, cómo la técnica de lematización se aplicaron a los respectivos sets de datos en el proceso de entrenamiento, seleccionando por medio de una búsqueda de grilla cuál de las dos maneras de vectorizar fue la más adecuada según las métricas de desempeño.

6.3.4. Filtrado de Stop Words

Una vez se tienen los tweets extraídos tokenizados (por Porter y por Lematización), se procedió a realizar un filtrado de las palabras que no aportan valor diferencial a la polaridad por medio de la técnica de StopWords, el cual es un listado de palabras comunes, tales como "*is, a, an, the, they, to*". Para este paso de preprocesamiento se utilizó el conjunto de palabas de StopWords de la librería NLTK en el idioma inglés.

```
[32] nltk.download("stopwords")

     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Unzipping corpora/stopwords.zip.
     True

[33] from nltk.corpus import stopwords

     stop = stopwords.words('english')
```

Ilustración 30 conjunto de palabras vacías

7. Objetivo específico 3 “Implementar un modelo no supervisado para análisis de polaridad en tweets.

Cómo se describió anteriormente, alineándose con los objetivos propuestos en el anteproyecto, se decidió seleccionar dos enfoques para la tarea de análisis de polaridad en tweets del contexto político colombiano, en primer lugar, se realizó el modelo ***Self Supervised*** y en segundo lugar un modelo ***Unsupervised***, la inclusión del enfoque Auto Supervisado se hizo para enriquecer el análisis de la efectividad del enfoque No Supervisado y poder posteriormente contrastar sus respectivas efectivades contra la tarea de etiquetado manual realizada por el tercero.

7.1. Arquitectura de los modelos

A continuación, se muestra la arquitectura general de los dos modelos que se desarrollaron para la tarea de análisis de polaridad en tweets en el contexto político colombiano, subrayando que en ninguno de estos modelos se utilizó la variable de etiqueta de polaridad asignada de manera manual, ya que esta solo se utilizó en la fase final para realizar el contraste de efectividad de cada uno de los dos modelos contra una tarea de etiquetado humano.

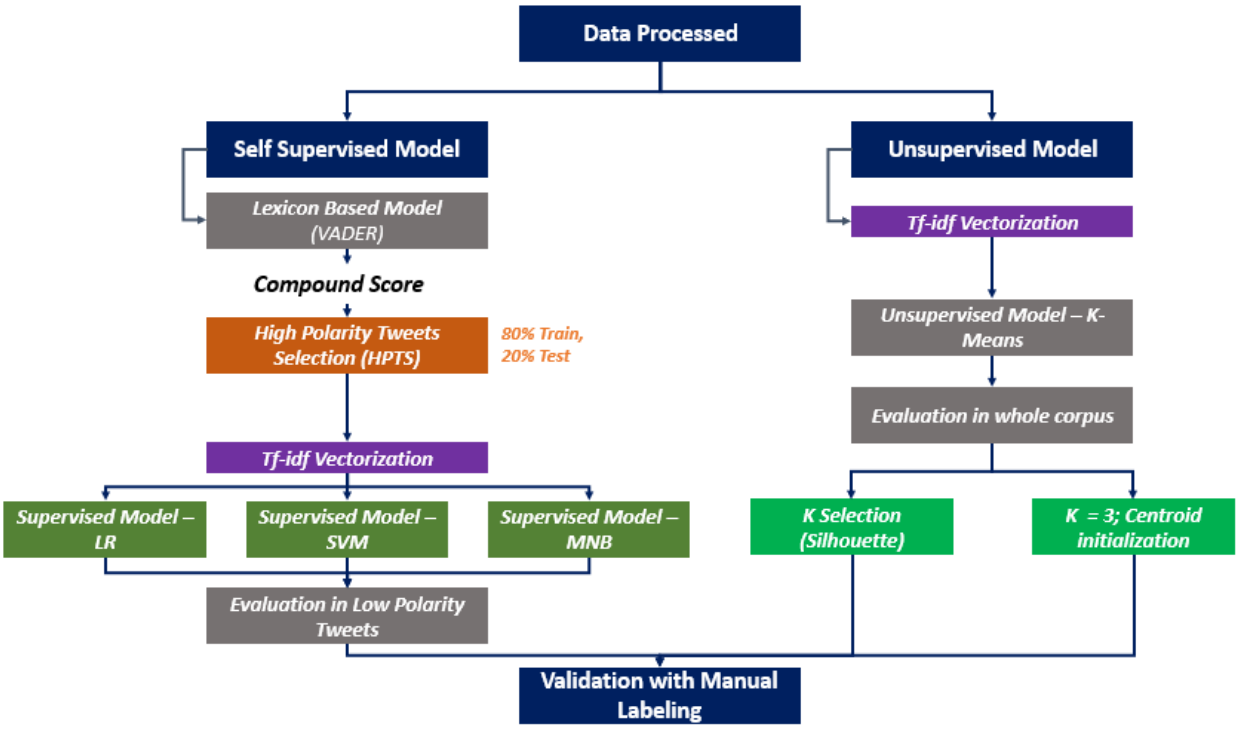


Ilustración 31 Arquitectura general de los modelos implementados

Una vez que se procedió con la etapa de preprocesamiento de los datos, se crearon dos modelos de Machine Learning, el primer modelo fue un modelo híbrido de aprendizaje Auto-Supervisado, el cual consta de un primer submodelo de análisis de polaridad basado en lexicones (LBM) el cual utilizó el modelo VADER [21], esta selección se hace ya que cómo se mostró anteriormente, a pesar de ser un modelo basado en un Lexicon construido en el idioma inglés, está especialmente diseñado para analizar la orientación semántica de pequeños textos específicamente derivados de las interacciones en redes sociales, por lo cual reviste una ventaja en la tarea del análisis de polaridad para tweets, para luego hacer una selección de tweets con altas polaridades semánticas y posteriormente dividir este sub corpus en dos conjuntos, un conjunto con el 80% de los tweets catalogados como con "***alta polaridad***" para realizar el entrenamiento de tres modelos de aprendizaje supervisado: ***Regresión Logística (LR)***, ***Support Vector Machines (SVM)*** y ***Multinomial Naive Bayes (MNB)*** con previa aplicación de vectorización por ***Tf-idf*** y luego usar el 20% restante de tweets con alta polaridad para validar la efectividad de los modelos correspondientes. La

selección de estos 3 tipos de modelos se basa en los estudios previos de Eissa M. Alshari et al (2017) y de Rendón-Cardona & Páez-Valdez (2022), en los cuales tanto los modelos de regresión logística, como los de SVM mostraron un buen desempeño en la asignación de orientación semántica a textos.

En estos 3 modelos supervisados se hizo una selección de hiperparámetros tuneados por medio de búsqueda en grilla para encontrar los mejores desempeños de cada modelo en el set de datos de training. Una vez encontradas las mejores combinaciones de hiperparametros en cada uno de los 3 modelos, se procedió entrenar estos 3 modelos tuneados con el 80% de los tweets con alta polaridad, luego, se procedió a hacer la validación con el 20% restante de los tweets de alta polaridad, para finalmente terminar con el etiquetado de los tweets del subconjunto de baja polaridad y así hacer el análisis de desempeño con el objetivo de escoger el mejor modelo según la métrica del f1-Score.

El segundo enfoque, el modelo No Supervisado se hizo un preprocesamiento de los tweets por medio de un análisis de frecuencia de termino-frecuencia inversa *Tf-idf*, para luego seguir dos estrategias en el entrenamiento de un modelo de clustering por K-Means aplicadas a la totalidad de los tweets. En la primer estrategia se realizó un proceso de clustering ejecutando el algoritmo de K-Means de manera iterativa para K en el rango [3,15], generando 12 posibles modelos, los cuales se compararon para la mejor selección del K que presentara la mejor segregación de los datos vectorizados. Por otro lado, en la segunda estrategia se procedió a ejecutar un modelo de clustering dando por parámetro K=3 (número de clases conocidas según la polaridad), inicializando los centroides con vectores artificiales generados con las palabras categorizadas como positivas, negativas y neutras extraídas del Lexicon sobre el cual se basó el LBM de VADER, comparando al final el desempeño de las dos estrategias contra las etiquetas asignadas manualmente.

Por último, se hizo un contraste de los resultados obtenido en el mejor modelo del enfoque híbrido (Self Supervised) y el mejor modelo del enfoque No supervisado.

7.2. Lexicon Based Model VADER

Cómo se expuso en la arquitectura de los modelos, el modelo de análisis de sentimientos basado en Lexicones **VADER** se utilizó en el enfoque ***Self Supervised*** para realizar un **pseudo etiquetado** con el objetivo de encontrar los tweets con mayores polaridades según los valores calculados por VADER, el cual corresponde al ***Compound Score***.

El algoritmo de VADER está basado en un Lexicon en el cual se evaluaron 9.000 características lexicales (candidatos) que se identificaron como de gran interés en el uso de publicaciones y textos cortos en Social Media, esta evaluación se realizó por medio de la evaluación individual de un conjunto de evaluadores humanos seleccionados por medio de la plataforma de micro tareas **Amazon Mechanical Turk**, mediante los cuales se le podía asignar un número entre -4 (muy negativa) hasta 4 (muy positiva), con la posibilidad de evaluar en 0 la característica léxica en caso que fuera neutral [21].

Al aplicar el LBS de VADER sobre un texto, el algoritmo retorna un diccionario con un score por cada una de las polaridades posibles (**Positivo**, **Negativo** y **Neutral**), adicionalmente, el modelo calcula un **Compound Score**, el cual incorpora cada uno de los scores individuales y tiene un rango entre -1 (muy negativo) hasta +1 (muy positivo), este score se calcula como el valor normalizado de la sumatoria de las valencias de las palabras y expresiones que componen la cadena de texto [21]. Los rangos del **Compound Score** son:

Compound Score (CS)	*Desde*	*Hasta*
Positivo	0.05	1
Neutral	-0.05	0.05
Negativo	-0.05	-1

Tabla 1 Threshold de variable [Compound] para análisis semántico por LBM VADER

7.3. Modelo de Aprendizaje Auto Supervisado (SSM)

El enfoque de modelo ***Self Supervised*** inició con el procesamiento de los tweets traducidos y preprocesados para luego alimentar al Lexicon Based Model **VADER** para la asignación de **pseudo etiquetas** de orientación semántica, para posteriormente filtrar los tweets cuyo score (***Compound Score)*** sobrepaso los umbrales de decisión (***thresholds***), en otras palabras, los tweets que presentaron las polaridades más altas calculadas por **VADER** ya sean polaridades positivas, negativas o neutras. Posteriormente, se seleccionaron de manera aleatoria el 80% de estos tweets con altas polaridades para entrenar 3 modelos de aprendizaje supervisado, donde se usaron las etiquetas asignadas por VADER como variables de salida, usando así el 20% restante de este subcorpus (***Test***) para el proceso de validación.

7.3.1. Corpus de datos con alta polaridad de VADER

El primer paso fue el de la creación de una función para la obtención de los ***Compound scores (CS)***.

```
[ ]  # Importar SentimentIntensityAnalyzer
     from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
     analyzer = SentimentIntensityAnalyzer()

     # Crear función de Score de sentimientos
     def VADER_Score (text):
         vs = analyzer.polarity_scores(text)
         return vs
```

Ilustración 32 Creación de función de scores de polaridades

Luego se procedió a aplicar la función de cálculo de ***CS*** para el tweet en ingles procesado.

```
# Asignación de diccionario con Score de VADER a cada Tweet en la versión traducida al inglés

dataEN["Score_VADER"] = dataEN["Tweet_Procesado"].apply(VADER_Score)
dataEN.head(50)
```

Ilustración 33 Creación de Scores de Polaridades

De esta manera se obtienen los diccionarios con los scores de polaridad para las 3 orientaciones semánticas.

Tweet_Procesado	Score_VADER
Opposition: the ICUs must be multiplied by 5	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...
We have just filed a complaint against Mr. Joh...	{'neg': 0.051, 'neu': 0.835, 'pos': 0.114, 'co...
"They call Uribe paraco Petro IS a guerrilla t...	{'neg': 0.241, 'neu': 0.663, 'pos': 0.096, 'co...

Ilustración 34 Calculo de Scores de Polaridad

Posteriormente, se creó la variable ***Compound_Score*** en la cual se calculó el ***CS*** para cada uno de los tweets traducidos al inglés.

```
# crearción de variable Compound_Score

dataEN["Compound_Score"] = dataEN.Score_VADER.apply(lambda x: x.get("compound"))
dataEN.head()
```

Obteniendo como resultado un escalar que representa la polaridad de cada uno de los textos.

Tweet_Procesado	Score_VADER	Compound_Score
Opposition: the ICUs must be multiplied by 5	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	0.0000
We have just filed a complaint against Mr. Joh...	{'neg': 0.051, 'neu': 0.835, 'pos': 0.114, 'co...	0.4082
"They call Uribe paraco Petro IS a guerrilla t...	{'neg': 0.241, 'neu': 0.663, 'pos': 0.096, 'co...	-0.8528

Ilustración 35 Resultado de cálculo de CS para cada Tweet

Al analizar la distribución de los ***Compound Scores*** asignados por VADER se puede ver lo siguiente que la media es de-0.098, con desviación estándar de 0.510.

```
count    4823.000000
mean       -0.098904
std         0.510850
min        -0.996400
25%        -0.538100
50%         0.000000
75%         0.318200
max         0.998700
Name: Compound_Score, dtype: float64
```

Ilustración 36 Percentiles y SD del Compound Score asignado por VADER

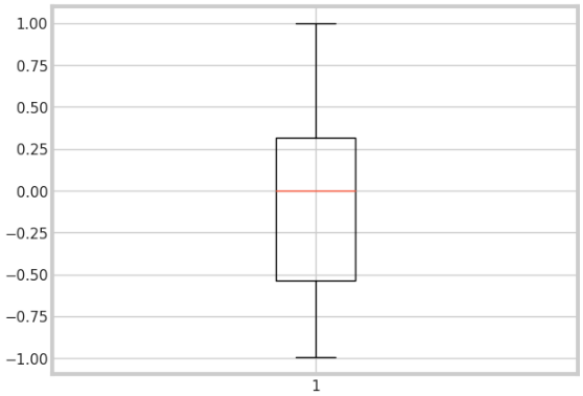


Ilustración 37 Percentiles del Compound Score

Una vez se obtuvieron los ***CS*** para cada uno de los Tweets, se procedió a calcular los ***thresholds*** de selección para tweets con alto grado de polaridad según la ***media (μ)*** y la ***desviación estándar (σ)*** (se aplicó la fórmula de la desviación estándar de la población completa).

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Obteniendo así los rangos según el valor del ***CS*** calculado por VADER:

$$Alta\ Polaridad\ Negativa\ \in [-1;\ \mu-\sigma]$$
$$Alta\ Polaridad\ Positiva\ \in [\mu+\sigma;1]$$
$$Polaridad\ Neutra\ \in [0-\mu;\ 0+\mu]$$

CS	*Desde*	*Hasta*	*Uso*
Altamente Positivo	0.42	1	Entrenamiento Modelo Supervisado ***(80% del subcorpus)***
Positivo Ambiguo	0.09	0.42	Etiquetado por Modelo Supervisado
Neutral	-0.09	0.09	Entrenamiento Modelo Supervisado ***(80% del subcorpus)***
Negativo Ambiguo	-0.6	-0.09	Etiquetado por Modelo Supervisado
Altamente Negativo	-0.6	-1	Entrenamiento Modelo Supervisado ***(80% del subcorpus)***

Ilustración 38 Rangos de selección Tweets para score asignado por VADER

A continuación, se muestra el código para implementar el filtrado de tweets con alto grado de polaridad calculada por VARDER.

```
#Asignación de threshold a tweets según clasificación VADER

def threshold_conditions(x):
    if x >= 0.42: return "Selected_Pos"
    elif x >=-0.09 and x <=0.09: return "Selected_Neu"
    elif x <=-0.6: return "Selected_Neg"
    else:          return "Not_Selected"

func = np.vectorize(threshold_conditions)
dataEN["Selection"] = func(dataEN.Compound_Score)

dataEN
```

Ilustración 39 Función de umbrales para tweets con alta polaridad

Cómo resultado de la aplicación de estos filtros se obtuvo dos subconjuntos, el primer subconjunto se llamó ***Corpus_Pseudo***, el cual contiene los tweets que cumplen con los criterios de alta polaridad, este subconjunto de datos está compuesto por ***3.015 tweets***, correspondiendo al 62% del corpus total, distribuidos de la siguiente manera:

```
Selection
Selected_Neg    1002
Selected_Neu    1049
Selected_Pos     964
```

Ilustración 40 Tweets con alta polaridad por VADER seleccionados

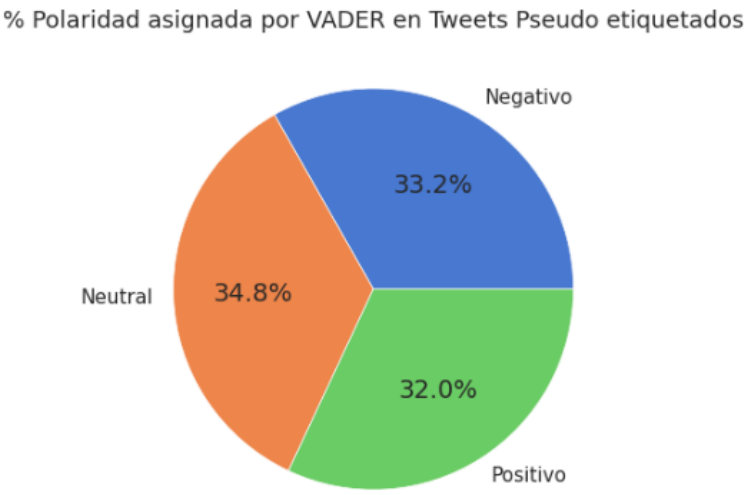


Ilustración 41 Distribución de Pseudo etiquetas asignadas por VADER en subconjunto de tweets con alta polaridad

Como se puede ver, las clases de tweets con altas polaridades están balanceadas, por lo cual no es necesaria la implementación de alguna técnica de balanceo de datos.

7.3.2. Corpus de datos con baja Polaridad de VADER

Posterior al proceso de filtrado de tweets con altas polaridades asignadas por VADER, se obtuvo el subconjunto de tweets con baja polaridad, el cual se utilizó en las siguientes fases para ser usado en un experimento adicional con el objetivo de probar la efectividad de los modelos híbridos en tweets con baja polaridad. Este corpus tiene 1.808 registros, correspondiendo al 38% del total de tweets recolectados.

```
data_Not_Selected.groupby("Polarity")["Polarity"].count()
```

```
Polarity
Negative    1182
Neutral      383
Positive     243
Name: Polarity, dtype: int64
```

Ilustración 42 Corpus con baja polaridad para etiquetado por modelos supervisados

Al analizar la composición de este set de datos según la polaridad asignada de manera manual se puede ver que el 65,4% estos tweets fueron clasificados inicialmente como ***Negativos***, un 21,2% fue etiquetado como ***Neutral*** y un 13,4% cómo ***Positivos***.

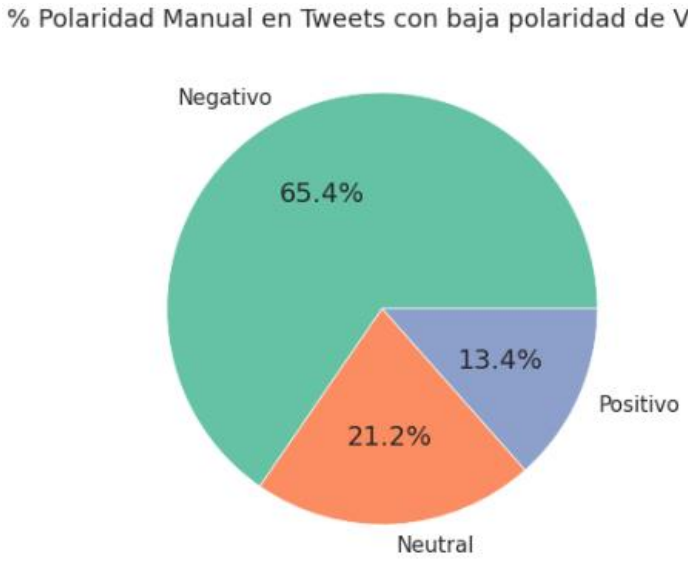


Ilustración 43 Distribución de etiquetas asignadas manualmente

7.3.3. Entrenamiento de Modelos Supervisados

Cómo se describió al inicio de esta sección, se entrenaron 3 modelos supervisados sobre el set de datos seleccionado con el criterio de tweets con alta polaridad. Paso seguido, se realizó la creación de los sets de datos de ***training*** y ***test*** para el set de datos pseudo etiquetados, tomando un 80% de los datos para training y un 20% para el test. Estos dos sets de datos se utilizaron para el entrenamiento y validación de cada uno de los modelos supervisados.

```
#Conformación de set de datos de entrenamiento y validación para tweets con altas polaridades
from sklearn.model_selection import train_test_split

X = Corpus_Pseudo['Tweet_Procesado']
y= Corpus_Pseudo['Pseudo_Polarity']


p_test = 0.2
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=p_test)
```

Ilustración 44 Creación de set de test y training para corpus pseudo etiquetado

Paso seguido, se procedió a crear un objeto de Termino de Frecuencia Inversa de Frecuencia de Documento (tf-idf) con el objetivo de vectorizar los textos filtrando las palabras que tienden a ser comunes en los diferentes tweets, de tal manera que se busca así disminuir el peso de estas palabras que aportan poca aplicabilidad a orientación semántica de cada uno de los textos [5]. En este caso se utilizó la clase TfidfVectorizer de la librería scikit-learn, la cual tiene por ventaja realizar la tarea de vectorización y aplicación del algoritmo tf-idf de manera conjunta.

```
[ ]  # Creación del tf-idf vectorizer
     tfidf = TfidfVectorizer(strip_accents=None,lowercase=False,preprocessor=None)
```

Ilustración 45 Creación del Tf-Idf

Este proceso de vectorización se utilizó cómo método de transformación de las cadenas de texto de los tweets para alimentar cada uno de los 3 modelos Supervisados.

7.3.3.1. Modelo con Regresión Logística

Como se describió en el marco teórico, los modelos de regresión logística son modelos de clasificación de aprendizaje Supervisado y se basan en la determinación de probabilidad de que una observación pertenezca a una de las clases posibles en el espacio de decisión del modelo. En este caso, se utilizó un modelo de regresión logística multinomial, ya que se tienen 3 clases posibles

[15]. El modelo de regresión logística multinomial o ***Softmax Regresión*** permite tener un modelo de regresión logística sin necesidad de entrenar y combinar varios modelos clasificadores ya que permite que para cada instancia ***X***, dado un número de clases ***k***, la función ***softmax*** estime la probabilidad de que esa instancia pertenezca a cada una de las clases en el espacio de selección mediante la computación de los valores del vector ***Sk(X)***, el cual contiene el conjunto de scores de "afinidad" de dicha instancia con cada una de las clases [7].

```
[158] #Creación del GridSearch para tuneo de hiperparametros

    param_grid = [{'vect__ngram_range':[(1,1)],
                   'vect__stop_words':[stop,None],
                   'vect__tokenizer':[tokenizer_porter,lemmatize_text],
                   'clf__penalty': ['l1','l2','elasticnet'],
                   'clf__C':[1,10,100],
                   "vect__analyzer": ['word', 'char', 'char_wb']
                   }]

    pipeline_RL = Pipeline(steps=[('vect',tfidf),('clf',LogisticRegression(multi_class="multinomial", solver="lbfgs",random_state=0))])

    gridS_RL = GridSearchCV(pipeline_RL,param_grid,scoring="f1_weighted",cv=3, verbose=1,n_jobs=-1)
```

Ilustración 46 Grilla de hiperparametros para entrenar un modelo de Regresión logística, basado en Raschka & Mirjalili 2019

En esta grilla de selección de hiperparametros se resalta que se utiliza tanto el método de Tokenización de Porter, cómo el método de Lemmatization, esta comparación de métodos de Tokenización es un aporte de este trabajo de grado, ya que en el ejemplo del libro citado solo se estaba contrastando el método de Porter contra un proceso simple de separación de cadenas de texto (Split básico). Adicionalmente, se usó el tipo de modelo "multinomial" para usar la función ***Softmax*** y el algoritmo de optimización del problema fue el solver por default para la función Softmax, el cual es "***lbfgs***".

Por otro lado, se utiliza una comparación de métodos de regularización ***L1, L2*** y ***Elasticnet*** para evitar el sobreajuste en el proceso de aprendizaje automático, mediante el cual se incluyen funciones de restricción sobre la complejidad del modelo llevando a que los valores de los parámetros estimados por el modelo supervisado no diverjan sin control mediante la "*adición a la*

función de costo del modelo de un costo asociado a tener valores altos en la parámetros " [12]. La diferencia entre el método de regularización L1, L2 y Elasticnet radica en que en la regularización L1 se añade un costo que es proporcional al valor absoluto de los valores de los parámetros, la L2 es un costo proporcional al cuadrado de los parámetros del modelo, mientras que Elasticnet combina los dos tipos de penalización (L1 y L2) [12].

Posteriormente, se realizó la creación de un PipeLine para implementar el proceso de vectorización con implementación conjunta del algoritmo ***Tf-idf*** y la aplicación del modelo Supervisado de Regresión Logística**.** Adicionalmente, se seleccionó la métrica de desempeño de ***f1_weighted***, con el objetivo de tener una adecuada selección de hiperparametros teniendo en cuenta que se le quiere dar la misma importancia al correcto etiquetado de las 3 categorías**.** Por último, se utilizó un ***Cross Validation*** con 3 particiones, de tal manera que el set de datos de training se divide en 3 partes, donde dos se toman para entrenar el modelo y la restante se toma para la validación, realizando el proceso 3 veces hasta iterar la validación con cada una de estas particiones [8].

```
pipeline_tfidf = Pipeline(steps=[('vect',tfidf),('clf',LogisticRegression(random_state=0))])

gridS_tfidf = GridSearchCV(pipeline_tfidf,param_grid,scoring="f1_weighted",cv=3, verbose=1,n_jobs=-1)
```

Ilustración 47 Creación de grilla para tuneo de hiperparametros

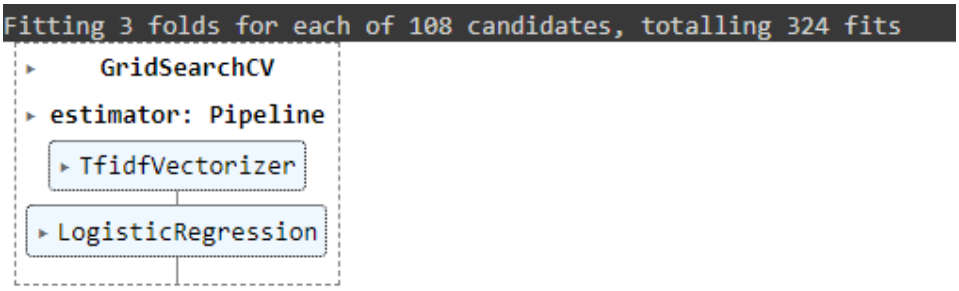


Ilustración 48 Modelo de Machine Learning de Regresión Logística

Al entrenar el modelo de regresión logística con la grilla de hiperparametros en el conjunto de

X_train y *y_train* se obtuvo que la mejor combinación de hiperparametros fue:

- Clf_C:100
- Clf_penalty: L2
- Analizer: “Word”
- stop_words: None
- tokenizer: tokenizer_porter

Una vez se encontraron los hiperparametros tuneados por el GridSearchCV, se procedió a crear un modelo de Regresión Logística con estos hiperparametros para entrenarlo con el set de datos de training (80% de los tweets de alta polaridad).

```
[69] #Replicación del mejor modelo hibrido Logistic Regression
     vectorizerLR = TfidfVectorizer(analyzer='word', tokenizer = tokenizer_porter)
     data_tweets_LR = vectorizerLR.fit_transform(X_train)

     #creación del mejor modelo con los hiperparametros tunueados
     tfidfLR = TfidfVectorizer(ngram_range=(1,1), stop_words=None)
     tfidfLR.fit_transform(X_train)

     pipelineLR = Pipeline([
         ('tfidf', tfidfLR),
         ('svm', LogisticRegression(random_state=0, penalty='l2',C=100, solver="lbfgs"))
     ])
```

Ilustración 49 Creación de modelo de LR con hiperparametros encontrados por GridSearchCV

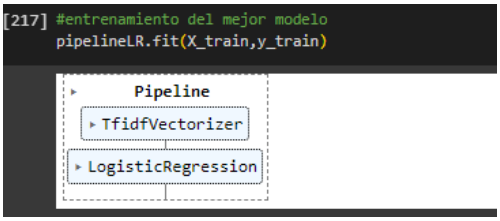


Ilustración 50 Entrenamiento de modelo LR con set de datos de Training

El F1-Score de este modelo con hiperparametros tuneados evaluado en el set de datos de ***test*** fue del 72,3%

```
[239] # F1-Score del modelo tuneado de LR

      f1_score(resultados_Hibrid_LR.cat_real, resultados_Hibrid_LR.cat_predict, average="macro")

      0.7233302878862765
```

Ilustración 51 F1-Score de modelo tuneado LR

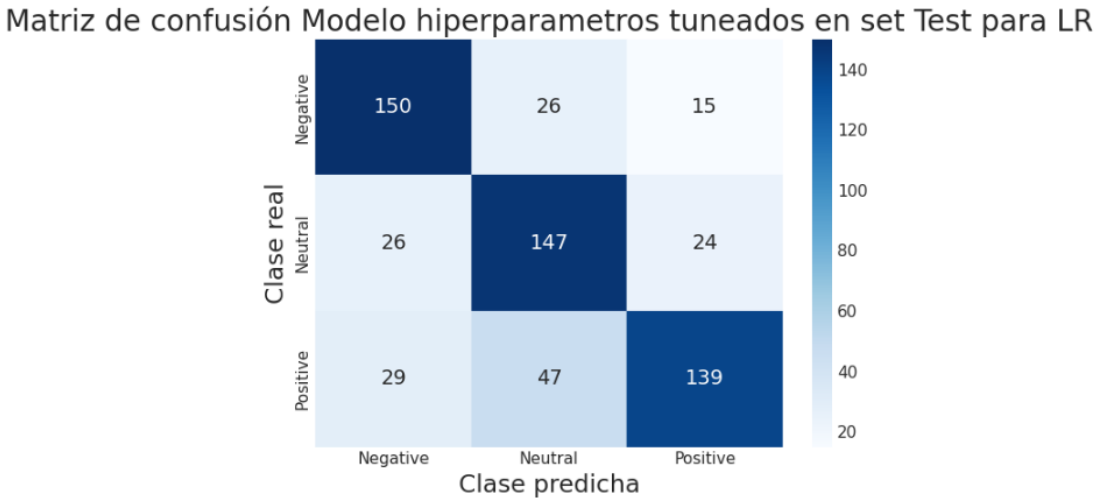


Ilustración 52 Matriz de confusión de modelo tuneado LR en test

A continuación, se muestran los resultados de precisión, sensibilidad y especificidad para las categorías ***Negativa***, ***Neutral*** y ***positiva*** respectivamente para le modelo de Regresión Logística.

```
La precisión de cada clase es: [0.73170732 0.66818182 0.78089888]
La sensibilidad de cada clase es: [0.78534031 0.74619289 0.64651163]
La especificidad de cada clase es: [0.86650485 0.82019704 0.89948454]
El informe de clasificación es:
              precision    recall  f1-score   support

    Negative       0.73      0.79      0.76       191
     Neutral       0.67      0.75      0.71       197
    Positive       0.78      0.65      0.71       215

    accuracy                           0.72       603
   macro avg       0.73      0.73      0.72       603
weighted avg       0.73      0.72      0.72       603
```

Ilustración 53 Métricas de desempeño modelo tuneado LR en test

Se puede ver que el modelo con hiperparametros tuneados validado con el set de datos de ***test*** tiene un F1-Score del 76% para la clase ***Negativa***, del 71% para ***Neutral*** y del 71% para la clase

positiva.

7.3.3.2. Modelo con Support Vector Machines

Para el entrenamiento del segundo modelo de Machine Learning Supervisado se procedió a modificar la grilla de parámetros para adecuarla a la estructura de modelos de ***Support Vector Machines.*** Estos modelos son útiles para tareas de clasificación de problemas lineales y no lineales; sin embargo, solo funcionan con datos numéricos, por ende, la importancia de realizar un procesamiento previo de vectorización.

En el caso de clasificación de polaridad, nos encontramos con la incógnita de si tendremos un hiperplano en el cual las clases sean linealmente separables, por lo tanto, se seleccionó usar la función kernel ***Radial Basis Function (rbf)*** ya que esta función puede adaptarse a diferentes formas y por ende se puede aproximar a funciones que no son linealmente separables [5].

```
[72] #Creación del GridSearch para tuneo de hiperparametros

     param_grid = [{'vect__ngram_range':[(1,1)],
                    'vect__stop_words':[stop,None],
                    'vect__tokenizer':[tokenizer_porter,lemmatize_text],
                    'clf__gamma': [0.01,0.1,1,10,100],
                    'clf__C': [0.01,0.1,1,10,100],
                    'vect__analyzer': ['word', 'char', 'char_wb'],
                    }]

     pipeline_SVM = Pipeline(steps=[('vect',tfidf),('clf',svm.SVC(kernel="rbf"))])

     gridS_SVM = GridSearchCV(pipeline_SVM,param_grid,scoring="f1_weighted",cv=3, verbose=1,n_jobs=-1)
```

Ilustración 54 Grilla de hiperparametros para entrenar un modelo de Support Vector Machines, basado en Raschka & Mirjalili 2019

En esta grilla de selección de hiperparametros se utilizaron las mismas opciones de ***tokenizador*** y de ***StopWords*** y de ***Analizer,*** incluyendo el parámetro de ***gamma***, el cual define cual va a ser el peso de una observación que es seleccionada como un **Support vector**, por otro lado, el hiperparametro

C define el trade-off entre una clasificación errada y la maximización del margen de clasificación [5].

Adicionalmente, se exploran dos valores para el hiperparámetro ***decision_function_shape***, el cual puede ser *"one-vs-one"* (ovo) o "*one-vs-rest*" (ovr), este hiperparametro tiene la función de determinar la forma de la función de clasificación cuando se utilizan hiperplanos que no son linealmente separables para la clasificación de multiclases. En el caso de ***ovo*** se usa un clasificador binario para cada par de clases de la forma *(n_samples, n_clases*(n_classes-1)/2)*, mientras que en el caso de ***ovr*** se usa un clasificador binario para cada clase en contraste con el resto de las clases [24].

Al entrenar el modelo de ***Support Vector Machine*** con la grilla de hiperparametros en el conjunto de ***X_train*** y ***y_train*** se obtuvo que la mejor combinación de hiperparámetros fue:

- C:100
- decision_function_shape: ovo
- Gamma: 1.0
- Analizer: "Word"
- stop_words: None
- tokenizer: tokenizer_porter

Una vez se encontraron los hiperparametros tuneados por el GridSearchCV, se procedió a crear un modelo de Support Vector Machine con estos hiperparametros para entrenarlo con el set de datos de training (80% de los tweets de alta polaridad), para luego ser validado con el set de datos de test (20%)

```
#Replicación del mejor modelo hibrido Logistic Regression
vectorizerSVM = TfidfVectorizer(analyzer='word', tokenizer = tokenizer_porter)
data_tweets_SVM = vectorizerSVM.fit_transform(X_train)

#creación del mejor modelo con los hiperparametros tunueados
tfidfSVM = TfidfVectorizer(ngram_range=(1,1), stop_words=None)
tfidfSVM.fit_transform(X_train)

pipelineSVM = Pipeline([
    ('tfidf', tfidfLR),
    ('svm', SVC(C=10,gamma=1,decision_function_shape="ovo"))
])
```

Ilustración 55 Creación de modelo de SVM con hiperparametros encontrados por GridSearchCV

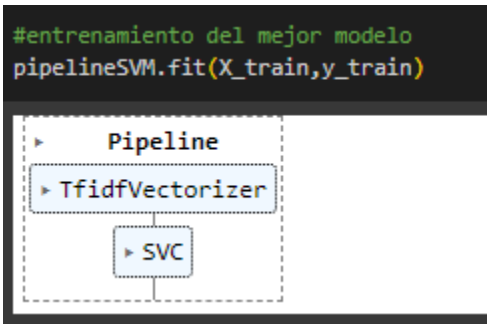


Ilustración 56 Entrenamiento de modelo SVM con set de datos de Training

El F1-Score de este modelo con hiperparametros tuneados evaluado en el set de datos de ***test*** fue del 67.9%

```
# F1-Score del modelo tuneado de LR

f1_score(resultados_Hibrid_SVM.cat_real, resultados_Hibrid_SVM.cat_predict, average="macro")
```

```
0.6796461662515026
```

Ilustración 57 F1-Score de modelo tuneado SVM

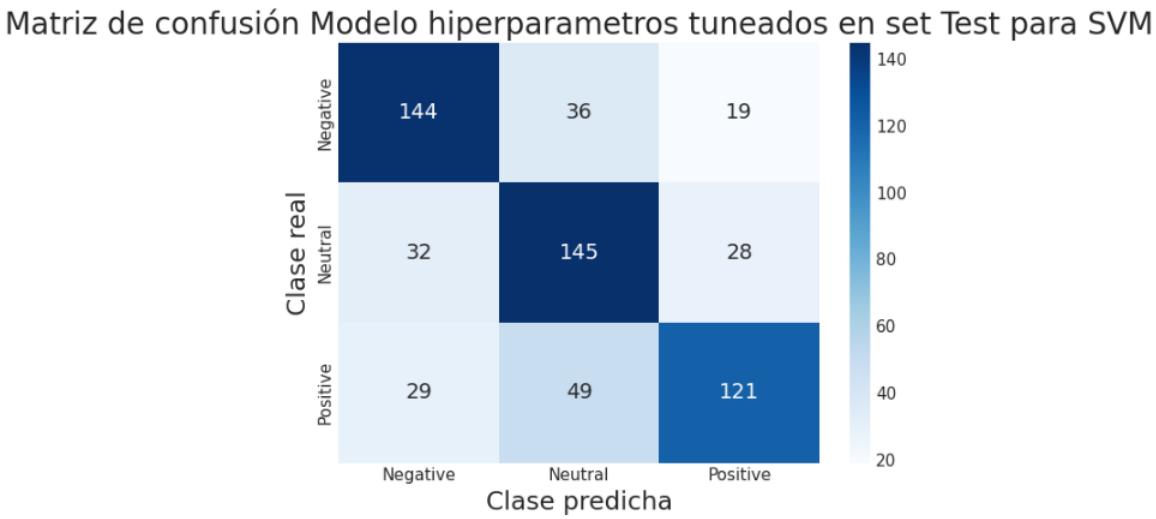


Ilustración 58 Matriz de confusión de modelo tuneado SVM en test

A continuación, se muestran los resultados de precisión, sensibilidad y especificidad para las categorías ***Negativa***, ***Neutral*** y ***positiva*** respectivamente para le modelo de SVM.

```
La precisión de cada clase es: [0.70243902 0.63043478 0.7202381 ]
La sensibilidad de cada clase es: [0.72361809 0.70731707 0.6080402 ]
La especificidad de cada clase es: [0.8490099  0.78643216 0.88366337]
El informe de clasificación es:
              precision    recall  f1-score   support

    Negative       0.70      0.72      0.71       199
     Neutral       0.63      0.71      0.67       205
    Positive       0.72      0.61      0.66       199

    accuracy                           0.68       603
   macro avg       0.68      0.68      0.68       603
weighted avg       0.68      0.68      0.68       603
```

Ilustración 59 Métricas de desempeño modelo tuneado SVM en test

Se puede ver que el modelo con hiperparametros tuneados validado con el set de datos de ***test*** tiene un F1-Score del 71% para la clase ***Negativa***, del 67% para ***Neutral*** y del 66% para la clase positiva.

7.3.3.3. Modelo con Multinomial Naive Bayes (MNB)

Por último, para el entrenamiento del tercer modelo de Machine Learning Supervisado se procedió

a modificar la grilla de parámetros para adecuarla a la estructura de modelos de ***Multinomial Naive Bayes.***

Para la implementación del método de ***MNB*** se modificó nuevamente la grilla de hiperparámetros, incluyendo el hiperparametro ***Alpha***, el cual suaviza el proceso de cálculo de probabilidades para cada tweet para evitar problemas de clasificación.

```
[336] #Creación del GridSearch para tuneo de hiperparametros de Multinomial Naive Bayes

      param_grid = [{'vect__ngram_range':[(1,1)],
                     'vect__stop_words':[stop,None],
                     'vect__tokenizer':[tokenizer_porter,lemmatize_text],
                     'clf__alpha': [0.01, 1, 10],
                     'clf__fit_prior' : [True, False]
                     }]

      pipeline_MNB = Pipeline(steps=[('vect',tfidf),('clf',MultinomialNB())])

      gridS_MNB = GridSearchCV(pipeline_MNB,param_grid,scoring="f1_weighted",cv=3, verbose=1,n_jobs=-1)
```

Ilustración 60 Grilla de hiperparametros para entrenar un modelo de Multinomial Naive Bayes, basado en Raschka & Mirjalili 2019

Al entrenar el modelo de ***MNB*** con la grilla de hiperparametros en el conjunto de ***X_train*** y ***y_train*** se obtuvo que la mejor combinación de hiperparámetros fue:

- alpha:1
- fit_prior: False
- stop_words: None
- Tokenizer: Tokenizer_Porter

Una vez se encontraron los hiperparametros tuneados por el GridSearchCV, se procedió a crear un modelo de Multinomial Naive Bayes con estos hiperparametros para entrenarlo con el set de datos de training (80% de los tweets de alta polaridad), para luego ser validado con el set de

datos de test (20%)

```
[135] #Replicación del mejor modelo hibrido Logistic Regression
      vectorizerMNB = TfidfVectorizer(analyzer='word', tokenizer = tokenizer_porter)
      data_tweets_MNB = vectorizerMNB.fit_transform(X_train)

      #creación del mejor modelo con los hiperparametros tunueados
      tfidfMNB = TfidfVectorizer(ngram_range=(1,1), stop_words=None)
      tfidfMNB.fit_transform(X_train)

      pipelineMNB = Pipeline([
          ('tfidf', tfidfLR),
          ('svm', MultinomialNB(alpha=1,fit_prior=False))
      ])
```

Ilustración 61 Creación de modelo de MNB con hiperparametros encontrados por GridSearchCV

```
#entrenamiento del mejor modelo
pipelineMNB.fit(X_train,y_train)
```

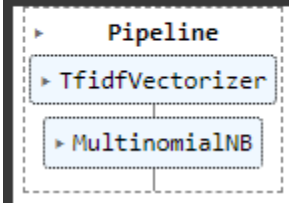


Ilustración 62 Entrenamiento de modelo MNB con set de datos de Training

El F1-Score de este modelo con hiperparametros tuneados evaluado en el set de datos de ***test*** fue del 65.5%

```
[155] # F1-Score del modelo tuneado de LR

      f1_score(resultados_Hibrid_MNB.cat_real, resultados_Hibrid_MNB.cat_predict, average="macro")

      0.6559359029403972
```

Ilustración 63 F1-Score de modelo tuneado MNB

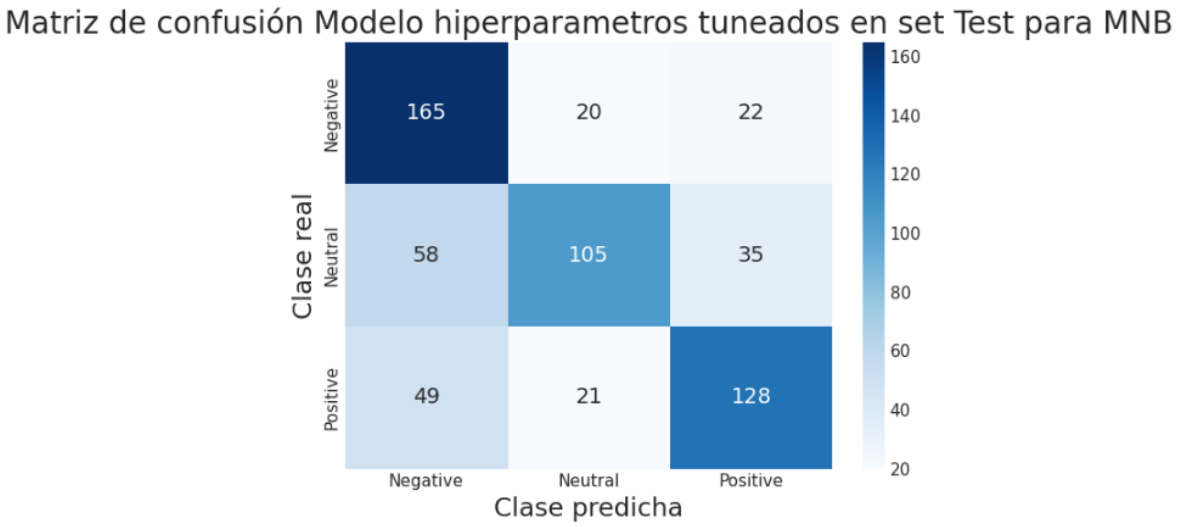


Ilustración 64 Matriz de confusión de modelo tuneado SVM en test

A continuación, se muestran los resultados de precisión, sensibilidad y especificidad para las categorías ***Negativa***, ***Neutral*** y ***positiva*** respectivamente para le modelo de MNB.

```
La precisión de cada clase es: [0.60661765 0.71917808 0.69189189]
La sensibilidad de cada clase es: [0.79710145 0.53030303 0.64646465]
La especificidad de cada clase es: [0.72979798 0.89876543 0.85925926]
El informe de clasificación es:
              precision    recall  f1-score   support

    Negative       0.61      0.80      0.69       207
     Neutral       0.72      0.53      0.61       198
    Positive       0.69      0.65      0.67       198

    accuracy                           0.66       603
   macro avg       0.67      0.66      0.66       603
weighted avg       0.67      0.66      0.66       603
```

Ilustración 65 Métricas de desempeño modelo tuneado MNB en test

Se puede ver que el modelo con hiperparametros tuneados validado con el set de datos de ***test*** tiene un F1-Score del 69% para la clase ***Negativa***, del 61% para ***Neutral*** y del 67% para la clase positiva.

7.3.4. Comparación de performance de modelos híbridos (SSM)

Al realizar el etiquetado de los tweets con baja polaridad con cada uno de los 3 modelos híbridos

se obtuvieron los siguientes resultados de F1-Score:

Modelo	F1 (Low Polarity)
F1_LR	0,38
F1_SVM	0,38
F1_MNB	0,41

Ilustración 66 Resultados de F1-Score para tweets de baja polaridad

Al comparar el desempeño por medio del F1-Score de los 3 modelos híbridos construidos con los hiperparametros tuneados correspondientes en el set de datos de ***test*** (20% de los datos categorizados como con alta polaridad según el score de VADER) en contraste con el F1-Score de los tweets de baja polaridad, subrayando que la etiqueta de referencia en ambos casos fue la etiqueta asignada de manera manual, se pudo ver que los modelos de regresión logística y de Support Vector Machine presentaron el mejor desempeño en el set de datos de altas polaridades con un 74% y 73% respectivamente de F1-Score para ***test*** y un 37% para ***Low Polarity*** , mientas que el modelo de Naive Bayes presentó un menor desempeño en ***test*** con un 67%, pero un mayor desempeño en ***low Polarity*** con un 41%

	Model	F1-Score_LPVader	F1-Score_test
0	F1_LR	0.377075	0.741294
1	F1_SVM	0.378719	0.733906
2	F1_MNB	0.411759	0.679910

Ilustración 67 F1-Score de modelos en Test y Low Polarity sets

Este decaimiento en el performance de los 3 modelos se puede evidenciar de igual forma al analizar las matrices de confusión sobre la aplicación en ***test*** y en ***low Polarity.***

Desempeño LR en tweets test

```
La precisión de cada clase es: [0.74042553 0.74736842 0.7247191 ]
La sensibilidad de cada clase es: [0.82464455 0.67619048 0.70879121]
La especificidad de cada clase es: [0.84438776 0.8778626  0.88361045]
El informe de clasificación es:
              precision    recall  f1-score   support

    Negative       0.74      0.82      0.78       211
     Neutral       0.75      0.68      0.71       210
    Positive       0.72      0.71      0.72       182

    accuracy                           0.74       603
   macro avg       0.74      0.74      0.74       603
weighted avg       0.74      0.74      0.74       603
```

Desempeño LR en tweets Low Polarity

```
La precisión de cada clase es: [0.69486405 0.27858293 0.16      ]
La sensibilidad de cada clase es: [0.3891709  0.45169713 0.34567901]
La especificidad de cada clase es: [0.67731629 0.68561404 0.71821086]
El informe de clasificación es:
              precision    recall  f1-score   support

    Negative       0.69      0.39      0.50      1182
     Neutral       0.28      0.45      0.34       383
    Positive       0.16      0.35      0.22       243

    accuracy                           0.40      1808
   macro avg       0.38      0.40      0.35      1808
weighted avg       0.53      0.40      0.43      1808
```

Ilustración 68 Comparación de medidas de desempeño de modelo LR tuneado en test y low Polarity

Al analizar el desempeño de los otros dos modelo se obtuvieron los mismos comportamientos divergentes en los desempeños de los modelo híbridos al evaluarlos en el set de datos con baja polaridad.

Desempeño SVM en Test

```
La precisión de cada clase es: [0.68016194 0.76243094 0.71428571]
La sensibilidad de cada clase es: [0.79620853 0.65714286 0.68681319]
La especificidad de cada clase es: [0.79846939 0.89058524 0.88123515]
El informe de clasificación es:
              precision    recall  f1-score   support

    Negative       0.68      0.80      0.73       211
     Neutral       0.76      0.66      0.71       210
    Positive       0.71      0.69      0.70       182

    accuracy                           0.71       603
   macro avg       0.72      0.71      0.71       603
weighted avg       0.72      0.71      0.71       603
```

Desempeño SVM en tweets Low Polarity

```
La precisión de cada clase es: [0.7170088  0.33385335 0.16701031]
La sensibilidad de cada clase es: [0.41370558 0.55874674 0.33333333]
La especificidad de cada clase es: [0.69169329 0.70035088 0.74185304]
El informe de clasificación es:
              precision    recall  f1-score   support

    Negative       0.72      0.41      0.52      1182
     Neutral       0.33      0.56      0.42       383
    Positive       0.17      0.33      0.22       243

    accuracy                           0.43      1808
   macro avg       0.41      0.44      0.39      1808
weighted avg       0.56      0.43      0.46      1808
```

Ilustración 69 Comparación de medidas de desempeño de modelo SVM tuneado en test y Low Polarity

Desempeño MNB en test

```
La precisión de cada clase es: [0.61971831 0.78861789 0.66326531]
La sensibilidad de cada clase es: [0.83412322 0.46190476 0.71428571]
La especificidad de cada clase es: [0.7244898  0.93384224 0.8432304 ]
El informe de clasificación es:
              precision    recall  f1-score   support

    Negative       0.62      0.83      0.71       211
     Neutral       0.79      0.46      0.58       210
    Positive       0.66      0.71      0.69       182

    accuracy                           0.67       603
   macro avg       0.69      0.67      0.66       603
weighted avg       0.69      0.67      0.66       603
```

Desempeño MNB en tweets Low Polarity

```
La precisión de cada clase es: [0.71380846 0.45355191 0.17463235]
La sensibilidad de cada clase es: [0.54230118 0.43342037 0.3909465 ]
La especificidad de cada clase es: [0.58945687 0.85964912 0.71309904]
El informe de clasificación es:
              precision    recall  f1-score   support

    Negative       0.71      0.54      0.62      1182
     Neutral       0.45      0.43      0.44       383
    Positive       0.17      0.39      0.24       243

    accuracy                           0.50      1808
   macro avg       0.45      0.46      0.43      1808
weighted avg       0.59      0.50      0.53      1808
```

Ilustración 70 Comparación de medidas de desempeño de modelo MNB tuneado en test y Low Polarity

Analizando este decaimiento en el desempeño de los 3 modelos híbridos al evaluarlos en el set de datos de tweets catalogados como con baja polaridad por el LBM VADER se podría intuir a que probablemente esta diferencia sea explicada en un primer lugar debido a que los tweets con altas polaridades utilizan una mayor cantidad de palabras con alta calificación de orientación semántica, llevando a que los vectores derivados de aplicar el algoritmo de ***Tf-idf*** tengan diferencias muy marcadas entre las 3 categorías posibles, facilitando así la identificación de patrones más marcados en dichos vectores, y derivado de esto haciendo que la asignación de polaridad por los modelos sea más efectiva.

Al comparar el performance de los modelos al evaluarlos en el set de datos ***test*** del corpus ***Pseudo-Etiquetado*** con altas polaridades en contraste con el set de datos con baja polaridad asignada por VADER se puede ver que el modelo de Regresión Logística y Support Vector Machine presentaron un mejor desempeño en tweets con altas polaridades sobrepasando en un 6% a Naive Bayes, mientras que Naive Bayes los sobrepasó en un 3% a los dos modelos en el etiquetado de tweets con bajas polaridades asignadas por VADER, teniendo esto en cuenta, se decide seleccionar a Logistic Regression como el modelo a implementar en el prototipo por parte del enfoque hibrido, ya que fue el modelo con el mejor performance teniendo en cuenta el F1-Score.

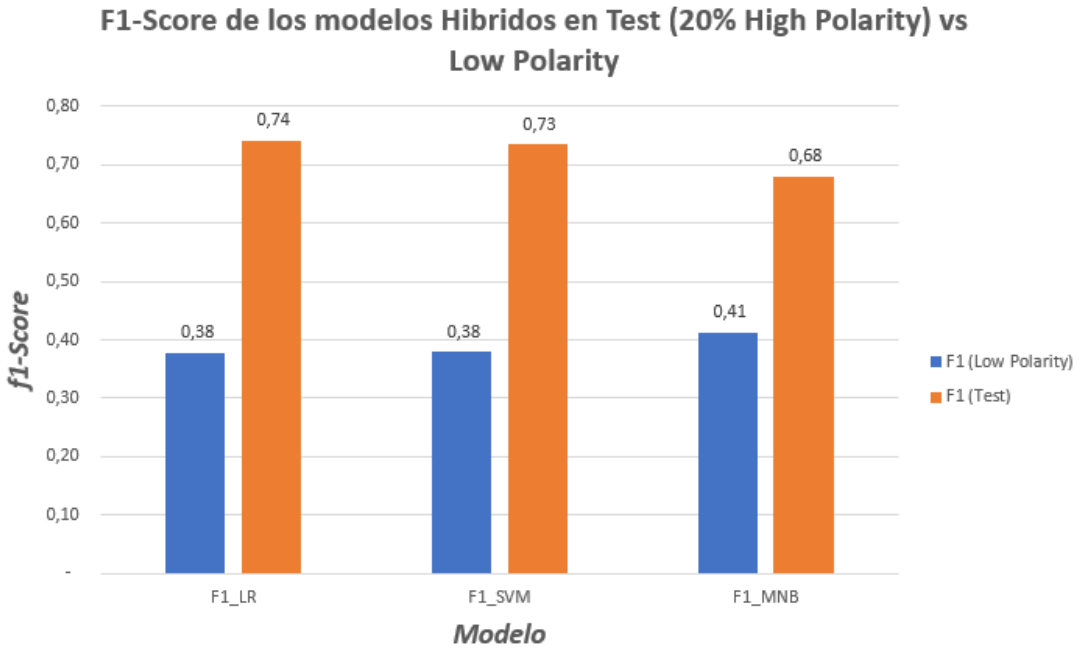


Ilustración 71 Desempeño de F1-Score de modelos en test y Low Polarity

7.4. Modelo No Supervisado

Cómo se expuso con anterioridad, en el enfoque de aprendizaje No Supervisado se abordó con 2 estrategias, la primera de ellas consistió en realizar un proceso iterativo de clustering con diferentes valores para el hiperparametro ***K***, seleccionando el mejor modelo; por otro lado, se utilizó un modelo con ***K=3***, con la particularidad de que se inicializaron sus centroides con 3 vectores canónicos constituidos por palabras con alta polaridad semántica extradíos de un lexicon "***Gold Standard"***, el cual es el lexicon sobre el cual se basó el LBM de VADER.

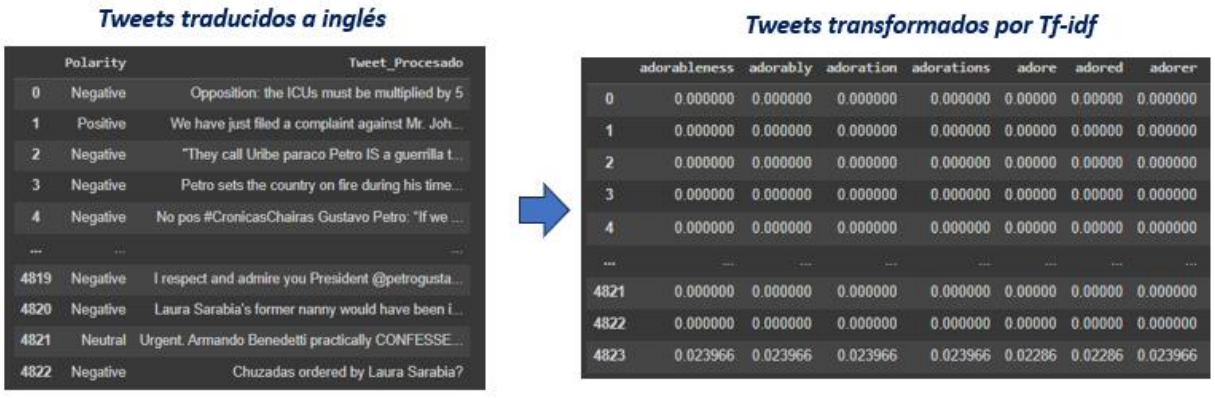
Así mismo. la piedra angular del enfoque No Supervisado para el análisis de orientación semántica es el proceso de vectorización implementando el ***Tf-idf*** con el objetivo de hacer apto el set de datos completo con los tweets traducidos al inglés.

El proceso de vectorización estándar consiste en transformar cadenas de texto en vectores numéricos, de tal manera que se busca primero crear un vocabulario que contenga el universo de

palabras diferentes que componen todas las observaciones en la base de datos, para luego representar a cada uno de los textos con un vector cuyo tamaño es el número de palabras contenidas en dicho vocabulario, y cada posición en el vector hacer referencia a la frecuencia de aparición de cada una de las palabra en el texto correspondiente [5].

Adicionalmente, cómo lo muestra Raschka (2019), el proceso de ***Tf-idf*** busca no solo transformar textos en vectores numéricos que representan las frecuencias de cada palabra en cada cadena de texto. Si no que va un paso más allá, encontrando las palabras que aparecen con mayor frecuencia a través de los diferentes textos, las cuales a pesar de ser las más frecuentes, no proveen mucha información útil que pueda discriminar la caracterización de un texto en comparación con los demás [5]. Por ejemplo, los pronombres "El", "Ella", etc. Pueden aparecer en muchos de los tweets, pero no aportan significativamente en el cálculo de la orientación semántica del texto para diferenciar entre un texto positivo o negativo. Es en este punto donde el algoritmo de ***Tf-idf*** es clave, ya que castiga estas palabras con mayor frecuencia a través de los elementos del corpus.

Esta transformación se puede ver a continuación:



	Polarity	Tweet_Procesado
0	Negative	Opposition: the ICUs must be multiplied by 5
1	Positive	We have just filed a complaint against Mr. Joh...
2	Negative	"They call Uribe paraco Petro IS a guerrilla t...
3	Negative	Petro sets the country on fire during his time...
4	Negative	No pos #CronicasChairas Gustavo Petro: "If we ...
...	...	...
4819	Negative	I respect and admire you President @petrogusta...
4820	Negative	Laura Sarabia's former nanny would have been i...
4821	Neutral	Urgent. Armando Benedetti practically CONFESSE...
4822	Negative	Chuzadas ordered by Laura Sarabia?

	adorableness	adorably	adoration	adorations	adore	adored	adorer
0	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
4	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
...	...	...	...	...	...	...	...
4821	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
4822	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
4823	0.023966	0.023966	0.023966	0.023966	0.02286	0.02286	0.023966

Ilustración 72 Transformación de tweets por vectorización con Tf-idf

7.4.1. Creación de centroides canónicos de Lexicon de VADER

Cómo se explicó en la sección del marco teórico, en la cual se expuso las características de los modelos de K-Means, estos modelos necesitan variables numéricas para realizar el cálculo de los centroides, los cuales son tomados como referencia para realizar las mediciones de las distancias a cada una de las observaciones en el hiperplano caracterizado por la cantidad de variables del sistema. Para este fin, se procedió a construir 3 centroides "canónicos" derivados de las palabras y expresiones con mayor polaridad extraídas del lexicon sobre el cual funciona el ***Lexicon Based Model VADER***.

Cómo lo explica Hutto & Gilbert (2014), el Lexicon sobre el cual está estructurado el LBM de VADER está compuesto por 7.500 características léxicas, cada una con un score o valencia de polaridad que está en el rango desde -4 (muy negativo) hasta 4 (muy positivo). Este Lexicon se podría considerar un ***Gold Standard***, ya que fue construido teniendo en cuenta la evaluación individual para cada una de estas características léxicas por parte de un grupo de evaluadores previamente entrenados para la identificación de polaridad en palabras, expresiones y emoticones en formato de texto (:D, :(, XD, etc.), lo cual lo constituye como una herramienta validada por humanos para la correcta identificación de orientaciones semánticas [21].

Teniendo en cuenta estas características del Lexicon base de VADER, se extrajo el diccionario de este lexicon junto con sus respectivos scores, con el objetivo de dividir las características léxicas en 3 categorías: ***Altamente Positivas***, ***Altamente Negativas*** y ***Neutrales.***

```
sid2 = SentimentIntensityAnalyzer()

# Extraer el lexicon de VADER como un diccionario
vader_lexicon = sid2.lexicon

# Mostrar el lexicon de VADER
print(vader_lexicon)
```

```
ivinely': 2.9, 'diviner': 0.3, 'diviners': 1.2, 'divines': 0.8, 'divinest': 2.7, 'diving': 0.3, 'divining': 0.9, 'divinise': 0.5,
```

Ilustración 73 Extracción de Lexicon de VADER

```
# Clasificación de polaridad para palabras y expresiones de VADER según su Score

df_lexicon["Polarity"] = np.where(df_lexicon["score"]>=1.5,"High_Positive",np.where(df_lexicon["score"]<=-1.5,"High_Negative","High_Neutral"))
df_lexicon
```

Ilustración 74 Limites de decisión para alta polaridad en palabras de Lexicon de VADER

Paso seguido, se crearon 3 cadenas de texto concatenando estas palabras con altos niveles de Score, para posteriormente agregarlas al corpus original de datos y proceder con la vectorización de estas tres cadenas de texto sintéticas. La unión previa de estas cadenas de texto con el vocabulario de VADER a la base de datos de tweets se realiza con el objetivo de asegurar que los vectores con frecuencias inversas de documento subyacentes hayan sido generados con el mismo tamaño que el número de palabras totales que componen el vocabulario de los tweets originales.

```
# Append de textos con altas polaridades generados por VADER a dataKM

dataKM1 = data_KM.append({"Polarity":"Positive","Tweet_Procesado":positive_text},ignore_index=True)
dataKM1
```

Ilustración 75 Ejemplo de inclusión de cadena de texto sintética de orientación positiva de lexicon VADER

```
vectorizer = TfidfVectorizer(strip_accents = 'ascii',lowercase = True, stop_words = 'english')

vector = vectorizer.fit_transform(dataKM1['Tweet_Procesado'])
```

Ilustración 76 Vectorización de cadenas de textos sintéticas de lexicon VADER

Cómo se puede ver a continuación, el resultado de este proceso arrojo la creación de 3 vectores con las frecuencias inversas de documento para los 3 vectores canónicos con altas polaridades extraídos de VADER, se puede ver que por ejemplo el vector correspondiente a la polaridad positiva es la única que presenta frecuencias mayores a 0 para palabras como “adorable” y “adoretion”, lo cual es consistente con lo esperado

Polarity	Tweet_Procesado	adorableness	adorably	adoration	adorations	adore	adored	adorer	adorers	adores	adoring
Positive	(}{') (^-: (^: ((-: (-: (-:0 (-:o (-:O (-:|...	0.023966	0.023966	0.023966	0.023966	0.02286	0.02286	0.023966	0.023966	0.023966	0.023966
Negative	$: %-))^:)-^:):-: }-:< }-:{): }:<){):<	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.000000
Neutral	%) &-: &: (% (* (-% (-* (-:< (-:{ (-: (< (; (...	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.000000

Ilustración 77 Resultado de vectorización de textos con alta polaridad de lexicon de VADER

Estos 3 vectores canónicos de polaridad se utilizarán para dos fines, en primer lugar, para comparar matemáticamente los vectores de los tweets que conforman cada uno de los clusters en la estrategia 1 y así poder interpretar los resultados del proceso de clustering. En segundo lugar, estos centroides se utilizaron para inicializar el modelo de la estrategia 2.

7.4.2. Similitud de vectores

Debido a que se utilizó la transformación de textos a vectores numéricos, se contó con el método de la "***Similitud del Coseno"***, el cual sirve para comparar dos vectores diferentes de 0 dentro del espacio vectorial con dimensión ***n***. Esta selección de método de comparación de vectores también se hizo con base en los buenos resultados evidenciados en el proceso de clustering de Word2Vec [19].

Esta similitud se calcula según la ecuación:

$$cos\theta = \frac{\vec{u} \cdot \vec{v}}{|\overrightarrow{u}| * |\overrightarrow{v}|}$$

Donde ***u*** y ***v*** son vectores con tamaño ***n***, de tal manera que la similitud del coseno es la división del producto escalar (producto punto) de los dos vectores dividido en el producto de sus módulos. Esta similitud del coseno varia en el rango de [-1:1], donde si el valor se acerca a 1, indica que los

vectores son muy similares y son paralelos, mientras si el producto es cercano a -1 indica que los vectores son ortogonales o perpendiculares, por ende, son vectores opuestos [25].

El objetivo de este método fue el de poder contrastar los resultados de los dos procesos de clustering contra los vectores canónicos extraídos del vocabulario del lexicon sobre el cual se basa en LBM de VADER. Para esto se generó una función customizada que recibe por parámetros dos vectores, retornando el valor de similitud del coseno.

```
#función de calculo de similaridad entre vectores unitarizados

def vector_similarity_single(vector,df):
    array = df.to_numpy()
    return [np.dot(row, vector) / (np.linalg.norm(row) * np.linalg.norm(vector)) for row in array]
```

Ilustración 78 Función de similitud de cluster contra vector canónico

7.4.3. Enfoque No Supervisado con iteración de hiperparametro K

En el primer enfoque del modelo de machine Learning No Supervisado se procedió a entrenar diferentes modelos de K-Means con el objetivo de ver el comportamiento de diferentes procesos de ***clustering*** realizados por el algoritmo para finalmente comparar dichas agrupaciones de tweets de manera no supervisada en contraste con la asignación manual de etiquetas. Se procedió a entrenar 12 modelos de K-Means sin inicializar los centroides para los valores de:

$$K \in [3; 15]$$

Paso seguido, se procedió a entrenar cada uno de estos modelos en el corpus de datos global para luego calcular la inercia de cada modelo, la cual corresponde a la media del cuadrado de las distancias entre cada una de las observaciones contra el centroide al cual fue asignada [7].

```
# Creación de modelos K-means con K in [3,15)]
kmeans_per_k = [KMeans(n_clusters=k, random_state=42).fit(df_Kmeans) for k in range (3,15)]
inertias = [model.inertia_ for model in kmeans_per_k]
```

Ilustración 79 Chuck de código para creación de modelos K-Means y cálculo de Inercias, basado en código presentado en la maestría de Ciencia de Datos, Universidad Javeriana de Cali

El objetivo de este proceso fue el de encontrar el K en el cual se presentará una mayor disminución de inercia en un solo paso, esto se puede ver gráficamente al cruzar el comportamiento de la inercia contra el aumento de los clusters en los modelos respectivos [7].

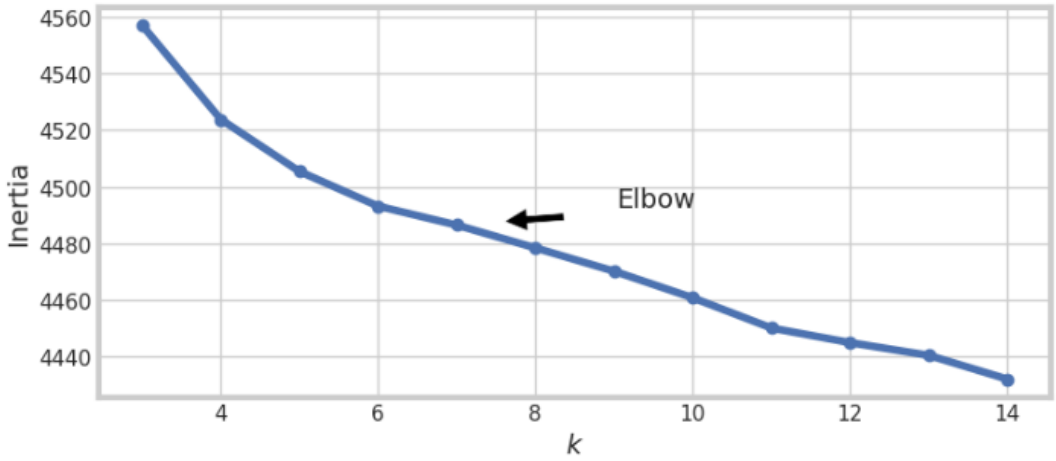


Ilustración 80 Inercia vs K

Se puede ver que hasta K = 7 se presentaron disminuciones significativas en la inercia de los modelos subyacentes, lo cual nos indicios que con 7 clusters los vectores de frecuencia inversa tienden a tener un buen grado de agrupación; sin embargo, como lo muestra Gerón (2019), este método es muy "holístico" y se debe complementar con otra estrategia de selección del hiperparametro ***K***, para tal caso se utilizó el análisis de silueta, en el cual se computa el promedio de los coeficientes de silueta para cada una de las instancias.

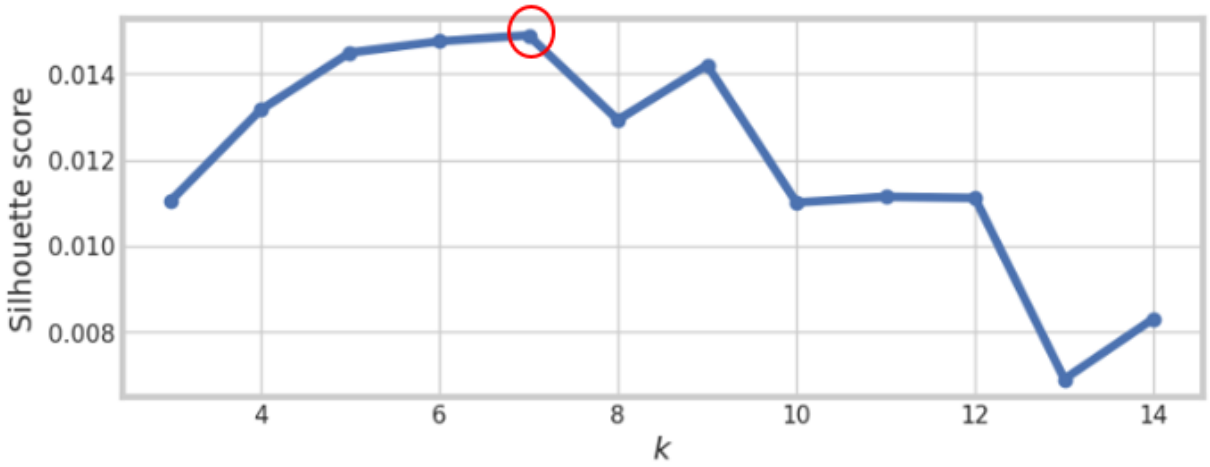


Ilustración 81 Gráfica de Score de Silueta vs K

El coeficiente de silueta es un valor que varía en el rango [-1,1], donde valores cercanos a 1 indican que cada una de las observaciones de la base de datos se encuentra más cercana a su respectivo centroide que a los centroides de los demás clusters, mostrando un correcto desempeño en la tarea de agrupamiento [7]. En este caso se busca el K en el cual se maximice el score de silueta, para el caso de estudio este comportamiento se dio con ***K=7***. Sin embargo, se pudo evidenciar que en este caso el ***Silhoutte Score*** fue de 0.16, mostrando un puntaje muy bajo, lo cual podría indicar que el proceso de clustering no presento fronteras muy claras.

Por último, Gerón (2019) recomienda igualmente utilizar el grafico del coeficiente de silueta para cada una de las instancias para ver cuál es el K que permite una mayor segregación de los datos en sus respectivos clusters. Al graficar este comportamiento se obtuvieron los siguientes resultados.

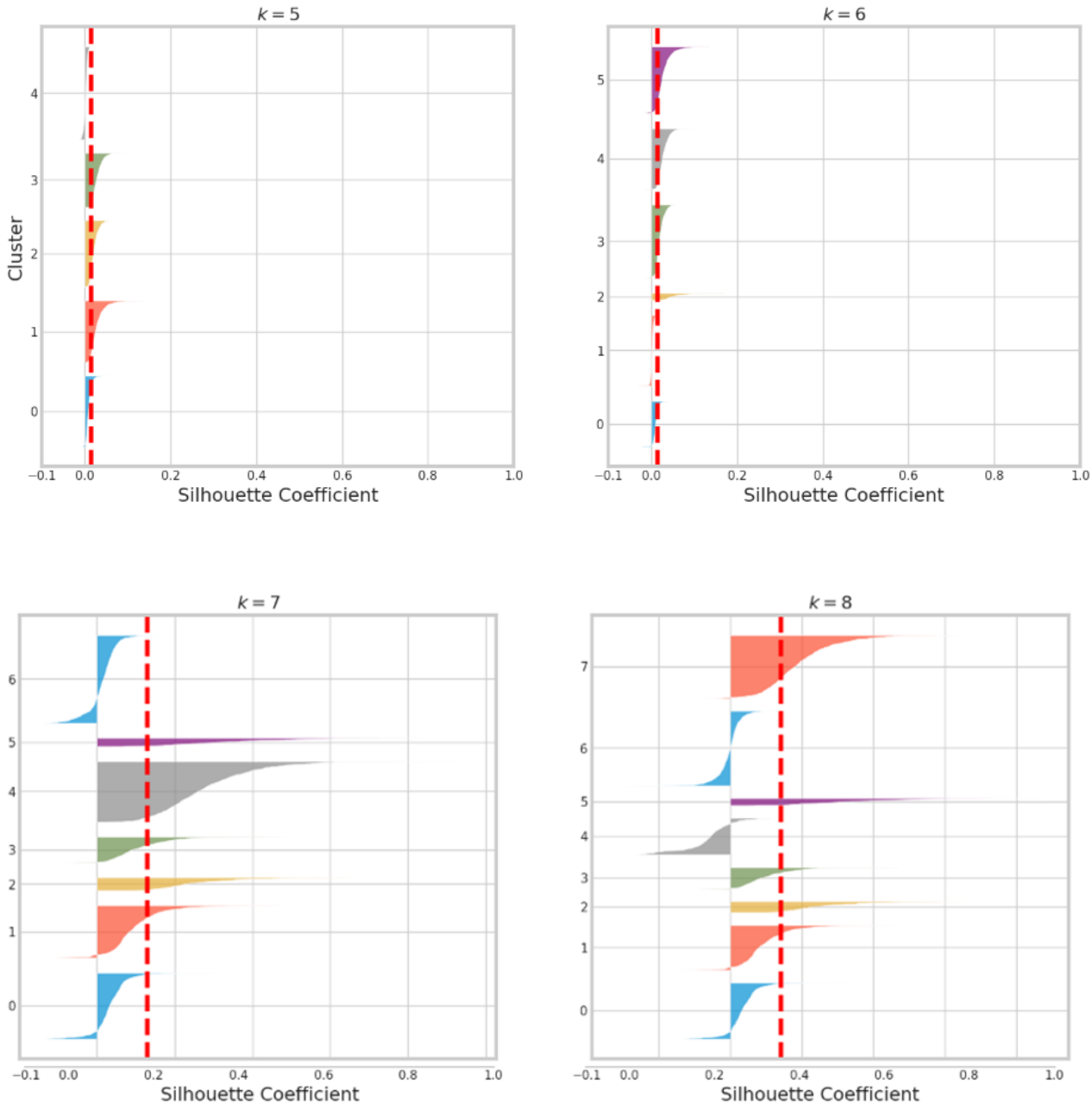


Ilustración 82 Grafico de Score de Silhoutte para diferentes valores de K

Cómo lo explica Gerón (2019), el objetivo de este grafico es encontrar el valor de ***K*** en el cual cada uno de los valores de silueta sobrepasen el valor de silueta de cada modelo, de tal manera que se busque que cada instancia este lo más lejos posible de los otros clusters [7]. En este caso, se analizaron las siluetas de los modelos con los ***K*** con mejor score de silueta (5,6,7 y 8).

El comportamiento con ***K=7*** muestra que todos los clusters sobrepasan la línea roja; sin embargo, a pesar de mostrar el mejor comportamiento de los clusters analizados, las siluetas no son homogéneas y nuevamente se muestra que los valores del score de cada cluster esta alrededor del nivel de mostrando un comportamiento con bajo desempeño en la tarea de clustering.

El resultado del proceso de clustering de los tweets con ***K=7***, fue el siguiente:

cluster	Frecuencia	%
0	868	17.997097
1	720	14.928468
2	977	20.257101
3	735	15.239478
4	809	16.773792
5	71	1.472113
6	643	13.331951

Ilustración 83 Clustering con K-Means, con K=7

Se puede evidenciar que todos los clusters tienen un tamaño relativamente homogéneo, conteniendo cada uno entre un 13% a un 20% de los tweets, sin embargo, el cluster 5 solo tiene el 1.4% de los tweets, correspondiendo a solo 71 observaciones.

Una vez implementado el modelo de K-Means con K=7, se procedió a obtener las etiquetas derivadas del proceso de clustering para cada uno de los tweets vectorizados y a consolidarlos en subgrupos de observaciones segmentados según su cluster respectivo, para posteriormente examinar las comparaciones de cada uno de los tweets vectorizados que componen cada cluster con cada uno vectores canónicos (***Positivo, Negativo*** y ***Neutral***) por medio de la función de similitud de coseno.

Al analizar la composición de los clusters se puede ver que en todos hay presencia de las 3 categorías asignadas por comparación de similitud de coseno entre los tweets contra los vectores canónicos extraídos de VADER, y que en todos la categoría con mayor presencia fue la de polaridad ***Positiva***. Esto quiere decir que se evidencia heterogeneidad en la conformación al interior de los clusters, por lo cual se ve posiblemente afectada la eficacia de este enfoque con clustering con K abierto.

Ilustración 84 Composición de clusters según similitud semántica

Se puede evidenciar que en todas los clusters la orientación semántica dominante es la ***Positiva***, sin embargo, hay tweets con categorías ***Negativa*** y ***Neutral*** presentes con proporciones importantes (por ejemplo, en el cluster 3 es del 37.8%), esto se puede ver en que de los 7 clusters, los clusters 0, 1, 2, 3, 4 y 6 no tienen una polaridad definida y muestran un gran grado de heterogeneidad. Según este enfoque todos los tweets serían catalogados como ***Positivos***, por lo cual se deduce que su performance es muy inferior a los resultados obtenidos en el enfoque hibrido.

7.4.4. Enfoque No Supervisado con inicialización de centroides

La segunda estrategia que se usó en el análisis de textos con el enfoque No Supervisado fue el de realizar un modelo de clustering por K-Means parametrizando a ***K=3***, correspondiendo a la cantidad de categorías de orientación semántica conocida de antemano (**Positiva, Negativa** y **Neutral**), con la particularidad que se inicializó el modelo con los 3 centroides, correspondientes a los vectores generados por el algoritmo de ***Tf-idf*** para cada una de las cadena de texto generada con todas las palabras de alta polaridad según el Lexicon de VADER.

Cómo se explicó al inicio de la sección, se extrajeron estos 3 vectores canónicos derivados del vocabulario del lexicon de VADER y se erigieron como los centroides a utilizar en la inicialización del modelo de K-Means con ***K=3.*** Posteriormente, se creó el modelo de K-Means inicializando los centroides de las 3 categorías con los 3 centroides generados.

```
#grupo de centroides para inicialización
centroids = np.array([vector_pos, vector_neg, vector_neu])

#creación del modelo KMeans con K=3
KMeans_init = KMeans(n_clusters=3, init= centroids, n_init=1)
```

Ilustración 85 Creación de modelo K-Means con K=3 e inicialización con centroides canónicos

Una vez implementado el modelo de clustering, se procedió a agrupar los tweets vectorizados según su cluster asignado, obteniendo los siguientes resultados:

cluster	Frecuencia Cluster
0	1530
1	2638
2	655

Ilustración 86 Clusters por modelo K=3

Al analizar el score de silueta para el caso de K=3 con centroides inicializados se tiene que es de 0.19, el cual es mayor que el score de K=7 (0.14), sin embargo, sigue estando cercano al valor ***0***, por lo cual las distancias entre las observaciones y los centroides de los demás clusters no significativamente inferiores a las distancias entre las observaciones y sus centroides correspondientes.

```
silhouette_scoreK3 = silhouette_score(df_Kmeans_init, KMeans_init.labels_)
silhouette_scoreK3
```

```
0.1964160986776662
```

Ilustración 87 Score de silueta modelo K=3 con centroides inicializados

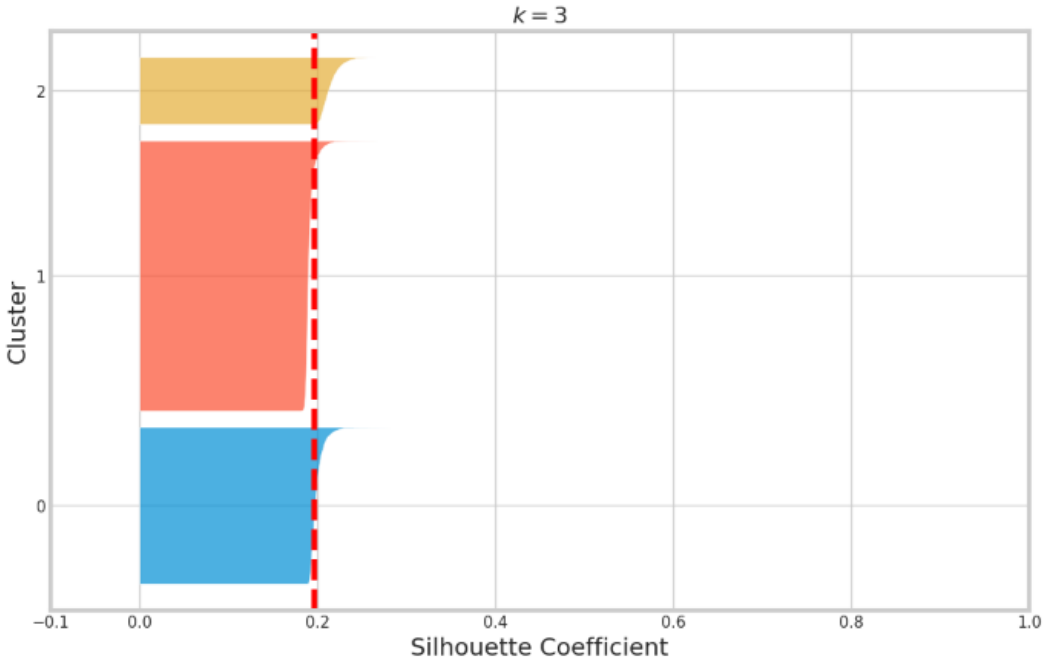


Ilustración 88 Grafico de score de siluetas para K=3 con inicialización de centroides

Paso seguido, se procedió a calcular la similitud de coseno de cada uno de los tweets vectorizados contra cada uno de los vectores canónicos extraídos del lexicon de VADER, seleccionando la polaridad semántica para cada tweet como la polaridad con la cual se tenga el mayor valor de similitud de coseno. Al realizar este análisis se obtuvieron los siguientes resultados:

	cluster	Cluster_VADER	Conteo_Similitud
0	0	Negative	522
1	0	Neutral	309
2	0	Positive	699
3	1	Negative	961
4	1	Neutral	533
5	1	Positive	1144
6	2	Negative	213
7	2	Neutral	185
8	2	Positive	257

Ilustración 89 Composición de clusters según polaridad por similitud de coseno con vectores canónicos

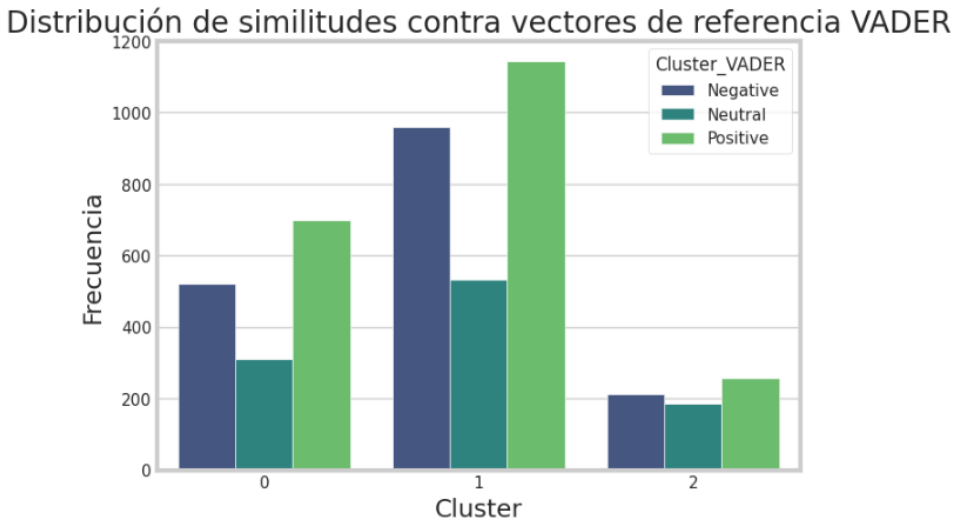


Ilustración 90 Composición de clusters según similitud de coseno

Al analizar los porcentajes según las polaridades asignada a cada tweet por la similitud del coseno se puede ver lo siguiente:

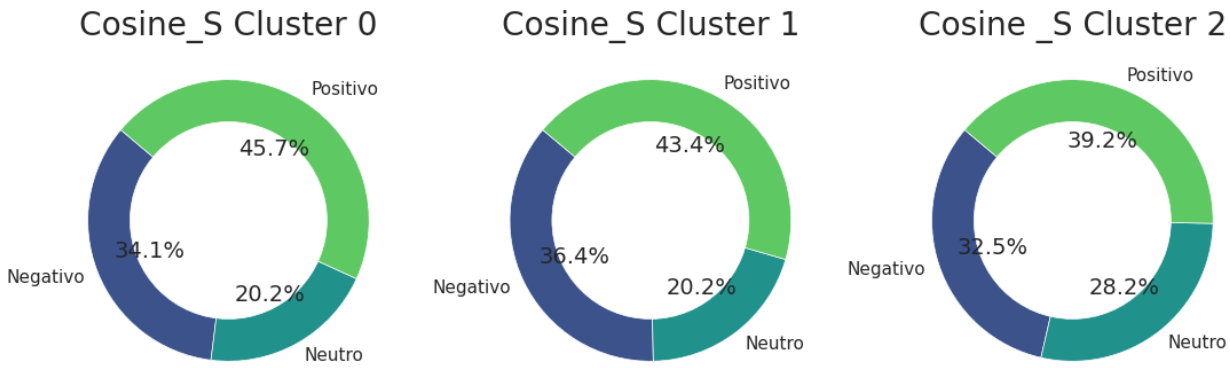


Ilustración 91 Conformación de los 3 Clusters con inicialización de centroides

De igual manera que en la primer estrategia, se puede ver que en los 3 clusters inicializados con los vectores canónicos no tienen una composición claramente homogénea ya que a pesar de que en los 3 clusters están compuestos en su mayoría por tweets con mayor similitud de coseno con el vector positivo, tiene una presencia significativa de tweets con similitud negativa, por ejemplo, en el cluster 0 el 45,7% de los tweets tiene mayor afinidad con el vector canónico positivo, mientras que el 34.1% tiene más afinidad con el vector canónico negativo.

Al igual que en el enfoque de selección del K, este método clustering por medio de K-Means con centroides inicializados con vectores con altas polaridades extraídos desde el Lexicon de VADER no presentó un buen desempeño a la hora de separar los tweets según sus polaridades.

8. Objetivo Especifico 4: Evaluación de la efectividad en el proceso de clasificación de polaridad para selección de Mejor Modelo

Al contrastar el comportamiento del modelo seleccionado con el enfoque Hibrido contra las dos estrategias de modelos No Supervisados se puede ver que el modelo con mejor desempeño fue el modelo con Support Vector Machine debido a que presentan un buen desempeño etiquetando tweets que contienen palabras con alta polaridad, sin embargo, este modelo presenta un desempeño inferior en tweets en el cual la polaridad es más ambigua.

Se tiene por hipótesis que esta debilidad evidenciada en la identificación de orientación semántica en una buena proporción de los tweets que fueron calificados inicialmente por el Lexicon Based Model de VADER puede radicar en varios factores semánticos y orgánicos del léxico utilizado en la escritura de tweets en español, y más aún en un contexto tan puntual como el ámbito de opiniones generadas sobre temas políticos en Colombia. Temas como el sarcasmo, por ejemplo, si un tweet dice “se nota que es todo un genio!!”, onomatopeyas propias de nuestra cultura como “uepa” o “upa”, o palabras que no tienen una buena traducción al inglés pueden generar ruido a la hora de la asignación de las etiquetas.

Por ejemplo, Hutto (2914) mostró cómo algunos de los obstáculos que se evidenciaron con mayor frecuencia en la construcción de VADER fue el de clasificar textos con palabras que no se encontraban dentro del lexicon sobre el cual se cimentó el método VADER, de igual manera el sarcasmo también mostró ser un reto para este LBM, ya que en este tipo de textos se encuentran palabras que de manera unitaria tendrían una polaridad determinada, pero que acompañada de un contexto y dependiendo de su posición en la cadena de texto, un intérprete humano la podría catalogar con polaridad inversa [21].

Sin embargo, queda por fuera de este trabajo de grado analizar el impacto de estas características semánticas propias del léxico colombiano en específico y se deja abierto para futuros trabajos de investigación.

Como conclusión de este capítulo, se selecciona el modelo hibrido de Support Vector Machine entrenado en el set de datos categorizados por el LBM de VADER con alta polaridad para ser utilizado en el desarrollo del prototipo de etiquetado de tweets.

9. Objetivo Especifico 5: Realizar el despliegue de un prototipo que implemente el modelo seleccionado para la interacción con usuario final para el análisis de polaridad para tweets extraídos de Twitter dando por parámetro un tweet extraído por el usuario.

9.1. Exportación del mejor modelo

El primer paso para el desarrollo del prototipo de etiquetado de tweets para uso de usuario final fue el de la exportación del modelo seleccionado. Para esto se usó la librería de Python ***joblib***, la cual permite serializar objetos para luego poder ser importados en diferentes frameworks. Para esto se debió exportar tanto el modelo ganador, en el cual están contenidos los parámetros e hiperparametros del modelo de Logistic Regression entrenado previamente, como el vectorizador *Tf-idf* entrenado con el vocabulario contenido en los 4.823 tweets recolectados de manera manual; Al final, se obtienen dos archivos con formato ***.pkl***, los cuales corresponden a objetos de la clase *joblib.*

```
joblib.dump(pipeline2,"model2.pkl")
joblib.dump(tfidf2,"vectorizer.pkl")

['vectorizer.pkl']
```

Ilustración 92 Exportación de modelo seleccionado

Para importar estos dos archivos se debe usar la misma librería ***joblib*** utilizando la función ***joblib.load("model")*** , la cual permite deserializar los objetos contenidos en los archivos ***.pkl.***

```
import joblib
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()
model = joblib.load("model2.pkl")
vectorizer = joblib.load("vectorizer.pkl")
```

Ilustración 93 Importación de objetos .pkl

Al cargar los dos archivos en un editor de código, y ejecutar el código anterior ya se puede empezar a utilizar el modelo de etiquetado de tweets. Para esto se puede utilizar el método ***.predict***.

```
texto =["its very bad"]

y_prueba = model.predict(texto)
```

Ilustración 94 Uso de modelo para etiquetar tweet

Una vez se exportó el modelo hibrido con el mejor desempeño, se procedió a crear un prototipo de etiquetado de tweets para la interacción básica con el usuario final. Para el desarrollo de este prototipo se utilizó en la parte del ***backend*** el lenguaje de Python con el Framework de Django, acompañado de las librerías de scipy y joblib para la ejecución de los modelos entrenados con formato (.pkl).

9.2. Pila Tecnológica

Una vez se tuvo el modelo exportado, se procedió a la creación de un prototipo que permitiera la interacción con un usuario final para la tarea de etiquetado de textos pequeño, en específico tweets, traducidos al inglés para la ejecución de tareas de asignación de polaridad semántica. Para esto se utilizaron las tecnologías que se describen a continuación.

Tecnología para el Desarrollo FrontEnd

Para el desarrollo Froten, es decir, la parte gráfica del aplicativo, se utilizó el lenguaje de etiquetado HTML, los estilos brindados por CSS y DjangoTemplate, que permite con las bondades de Python estructurar una plantilla a partir de objetos más pequeños.

Base de Datos

Para la Base de Datos se utilizó MySQL, en la cual se almacenan los modelos usados y se persiste la información generada por la polarización del modelo de aprendizaje seleccionado, en este caso, el modelo de ***Regresión Logística.***

Despliegue App

En el despliegue de la aplicación se utilizó la parte gratuita de PythonAnyWhere, ya que permite cierta facilidad para este proceso en aplicativos desarrollados con Frameworks de Python, al ser la capa gratuita posee una capacidad muy limitada en termino de procesamiento de solicitudes, pero permite mostrar el funcionamiento básico de este aplicativo. Para este despliegue se utilizó el siguiente tutorial: ***https://recursospython.com/guias-y-manuales/desplegar-un-proyecto-de-django-en-pythonanywhere/#google_vignette.***

9.3. Arquitectura del prototipo

Este aplicativo se divide en el app ***tweet_behaviour*** que es el generado por el Framework y donde están las configuraciones generales de toda la aplicación, y la app ***main***, en la cual se almacena toda la lógica de lo que es requerido para el aplicativo. Las carpetas ***static*** y ***media***, son carpetas que contienen archivos estáticos y los subidos por el usuario al aplicativo respectivamente.

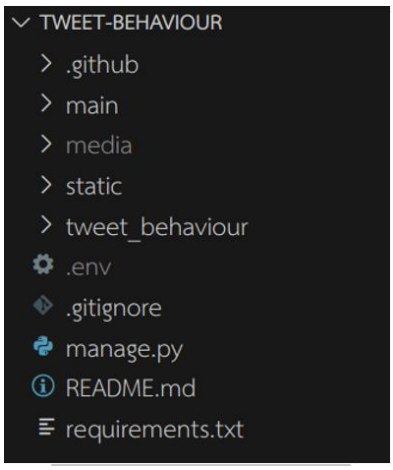


Ilustración 95 Estructura de carpetas del prototipo

Se configura el archivo de las URL (main/urls.py) para que sirva las vistas generadas haciendo un mapeo con la URL, por medio de estas URLs el usuario puede visualizar las diferentes páginas del prototipo con sus respectivas funcionalidades.

```
main > urls.py > ...
1   from django.urls import path
2   from .views import ManagementFormView, ExecutionView,
    LearningModelCreateView
3
4   app_name = 'main'
5
6   urlpatterns = [
7       path('', ManagementFormView.as_view(), name='management'),
8       path('result/<int:pk>', ExecutionView.as_view(),
        name='result'),
9       path('learning-model', LearningModelCreateView.as_view(),
        name='learning-model')
10  ]
```

Ilustración 96 URLs del prototipo para navegación

Las vistas son aquellas que permiten renderizar una plantilla HTML y permite comunicar la información obtenida con el resto del aplicativo.

```python
class ManagementFormView(FormView):
    template_name = 'pages/index.html'
    form_class = ManagementForm
    success_url = reverse_lazy('main:management')

    def form_valid(self, form):
        if form.is_valid():
            words = [[word.strip()] for word in form.cleaned_data.get('words').split
            (',')]
            model = form.cleaned_data.get('learning_model')
            obj_exec = Execution.objects.create(model=model)
            try:
                for word in words:
                    result = execute_model(word, obj_exec.model.model_file)
                    PredictResult.objects.create(text=''.join(word), execution=obj_exec,
                    result=result)
                self.success_url = reverse_lazy('main:result', kwargs={'pk': obj_exec.
                id})
                return super().form_valid(form)
            except Exception as error:
                obj_exec.delete()
        return super().form_invalid(form)
```

Ilustración 97 Vista para renderizado en la página HTML

Gracias a la plataforma de PythonAnyWhere, se creó un nuevo proyecto en el servidor que se dispuso de esa plataforma. Esta selección se hizo ya que tiene una capa gratuita y que la configuración por default permite no tener que realizar ningún trámite correspondiente al desarrollo en nube.

9.4. Vista del prototipo

Para el usuario final, se ingresará al siguiente link: *https://tweetbehaviour.pythonanywhere.com/*

Se observará una interfaz como la siguiente:

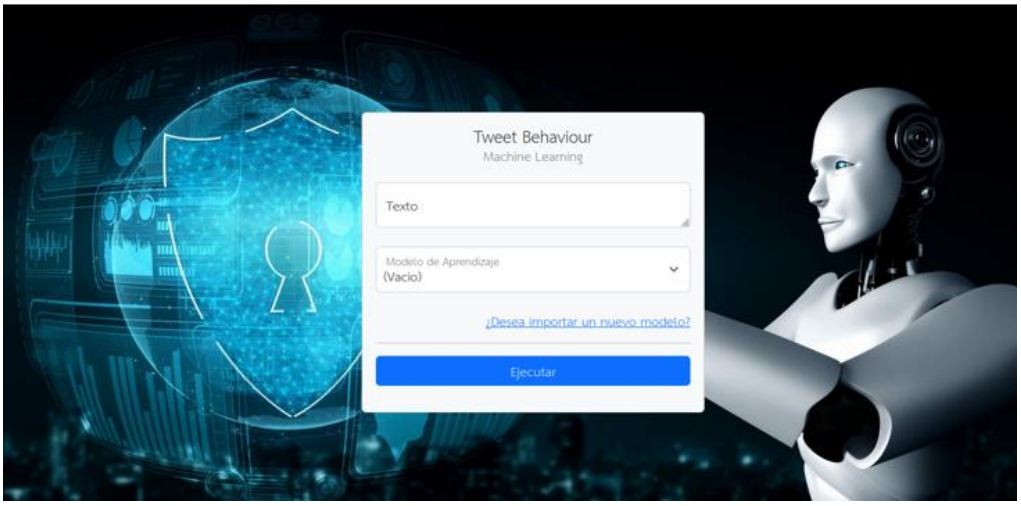


Ilustración 98 Interfaz principal prototipo

En esta interfaz, el usuario deberá ingresar un el texto del tweet extraído desde la aplicación ***X***, previamente traducido al idioma inglés, para luego seleccionar el modelo ***Hibrido***, el cual tiene el modelo de Regresión Logística con lo hiperparametros tuneados.

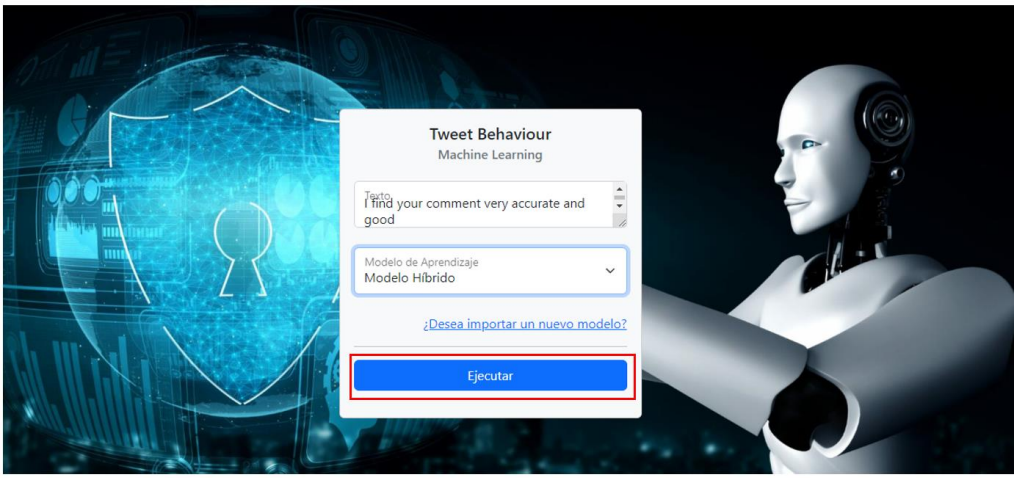


Ilustración 99 Ejemplo de uso de prototipo

Paso seguido, al oprimir el botón ***Ejecutar***, el prototipo ejecutará la tarea de predecir la polaridad del tweet y devolverá la polaridad con un valor discreto cuyo dominio este en el conjunto [-1,0,1], siendo ***1*** correspondiente a una polaridad ***Positiva***, ***0*** a una polaridad ***Neutral*** y ***-1*** a una polaridad ***Negativa.***

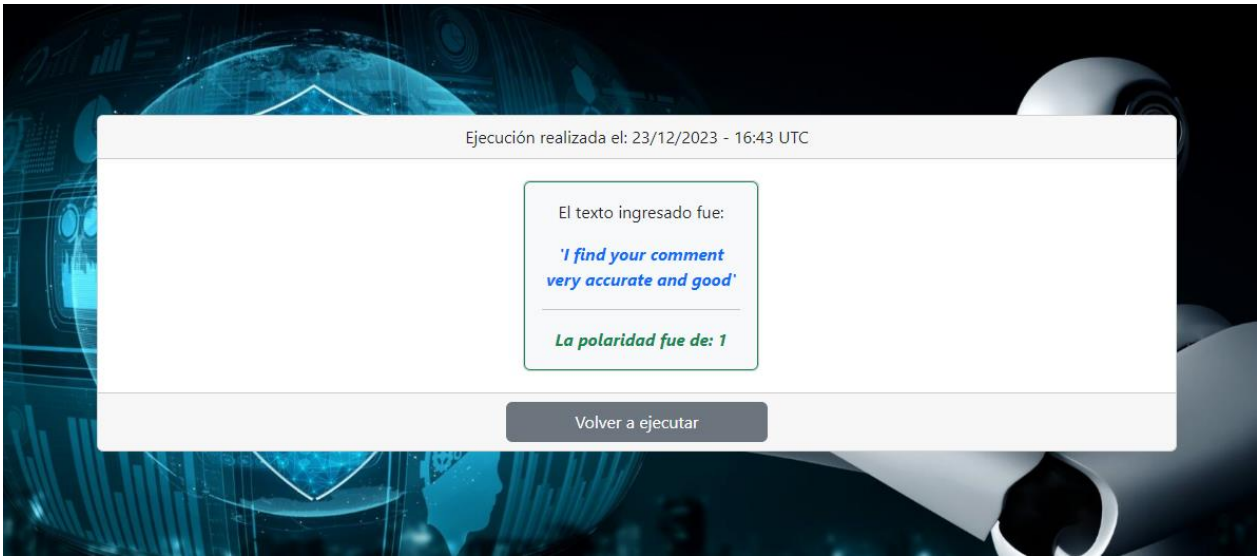


Ilustración 100 Resultado de ejecución de polaridad de texto

10.Conclusiones y Trabajos Futuros

El desarrollo de este trabajo de grado permitió extraer varias conclusiones y enseñanzas, ya que mostró resultados fehacientes de la aplicabilidad de varios enfoques de modelos de Machine Learning al análisis de datos no estructurados derivados de interacciones cotidianas de los ciudadanos en la red social de X (antes Twitter), esto muestra que hay un gran potencial en el desarrollo de herramientas de inteligencia artificial que pueden ser aplicables con buenos resultados en la automatización de este tipo de análisis con posibles usos académicos y comerciales.

10.1. Conclusiones

Al haber desarrollado la serie de pasos y análisis para llegar a la implementación de los 5 modelos realizados en este proyecto de grado se pueden extraer las siguientes conclusiones:

En primer lugar, se pudo evidenciar que según lo visto en la literatura consultada, el análisis de sentimiento en textos de redes sociales ha mostrado un gran crecimiento en la última década, erigiéndose como una de las ramas más activas dentro de la generación de modelos de Machine Learning, esto alimentado sus aplicaciones en el ámbito académico y comercial. Así mismo, las rama general de NLP es una de las piedras angulares para las soluciones de inteligencia artificial, por ejemplo, inteligencias como Chat GPT han mostrado que el análisis semántico y las herramientas generativas de lenguaje son y seguirán siendo en los próximos años uno de los campos con mayor actividad, no solo dentro de esta disciplina, si no de la economía mundial en general.

Sumando a esto, las tareas de análisis de lenguaje natural, entre ellas, los problemas de identificación de orientación semántica aún presentan un obstáculo en su desarrollo por medio de

modelos supervisado debido a las restricciones en la consecución de datos de textos etiquetados para el entrenamiento de estos modelos, por ende, las aproximaciones con enfoques usando modelos de aprendizaje Auto Supervisado son muy útiles para tal fin, ya que como se mostró en este trabajo se pueden utilizar herramientas no supervisadas basadas en lexicones para la generación inicial de pseudo etiquetas para posteriormente entrenar un modelo supervisado que permita etiquetar otros textos.

Al comparar el performance con el F1-Score del enfoque Auto Supervisado contra el enfoque No Supervisado se pudo evidenciar que el mejor modelo del primer enfoque mostró un desempeño significativamente mayor al implementarlo sobre la totalidad del corpus de tweets, llegando a un F1-Score del 54% para la categoría ***Negativa***, 36% ***Neutral*** *y 25%* para la categoría ***Positiva***, contra un muy bajo desempeño en el enfoque No Supervisado del 0% para la categoría ***Negativa***, 0% ***Neutral*** *y 18%* para la categoría ***Positiva.*** Sin embargo, el desempeño del mejor modelo Auto Supervisado fue del 79% para la categoría ***Negativa***, 77% ***Neutral*** *y 90%* para la categoría ***Positiva*** para los tweets que fueron catalogados como con **alta polaridad** por medio del Lexicon Based Model de VADER.

Sumando a la conclusión anterior, y complementándola con lo expuesto por Hutto (2014), la divergencia en la efectividad del enfoque Auto Supervisado en las tareas de asignación de orientación semántica en tweets con altas polaridades en contraste con tweets que se encuentran en zonas más grises en su polaridad se puede deber a varios factores, entre ellos, debido a la presencia de sarcasmo, presencia de onomatopeyas propias del léxico colombiano y de nuestra idiosincrasia, y de palabras y expresiones cotidianas que no tienen una traducción fidedigna al idioma inglés, idioma al cual se necesitó traducir los tweets para poderlos procesar con el LBM de VADER o palabras y expresiones que no aparecen en el lexicon de VADER. Este tipo de análisis semántico más profundo para explicar estas divergencias se escapa al alcance de esto proyecto, dejándolo abierto para futuros trabajos y de esta manera ampliar la investigación sobre métodos

para mejorar el desempeño de futuros modelos implementados para tareas tan específicas como el análisis de tweets en español sobre el contexto político colombiano

Por último, se evidenció que el proceso de clustering utilizando modelos No Supervisados de K-Means no son adecuados para la ejecución de tareas de asignación de polaridad para tweets en español para el contexto específico de este trabajo de grado, esto se puede deber a las restricciones estructurales intrínsecas del modelo de K-Mean, ya que como expone Gerón (2019), estos modelos solo funcionan bien con datos cuyo patrón de dispersión tiende a presentar un grado de homogeneidad en formaciones circulares o esféricas, y probablemente los datos de estos tweets vectorizados no presenten este tipo de simetrías.

10.2. Trabajos Futuros

Cómo trabajos futuros derivados de este proyecto aplicado se deja la exploración detallada de las repercusiones de estructuras semánticas propias del lenguaje colombiano en el marco de las discusiones políticas presentes en los tweets, teniendo en cuenta que este ámbito lingüístico requiere de intervenciones y análisis multidisciplinares que no fueron objeto de este estudio.

Por otro lado, este ejercicio se puede robustecer en el futuro utilizando a varios expertos para la ejecución de las tareas de etiquetado, ya que de esta manera se reduciría los sesgos se subjetividad en el etiquetado al tener varios puntos de vista sobre un mismo tweet.

Así mismo, el prototipo desarrollado en el marco de este trabajo de grado se podría robustecer para poderse conectar mediante una API pagada a la red social *X* u alguna otra, permitiendo así la extracción de comentarios y post en grandes cantidades, pudiéndose convertir en una herramienta útil para análisis de polaridad con varios usos académicos y comerciales.

Por último, se podría enriquecer el ecosistema del análisis de orientación semántica para tweets relacionados con las redes sociales en el contexto político colombiano por medio de la generación de un lexicon en español que contenga las polaridades de las palabras y expresiones típicas de este contexto en específico.

Referencias

[1] We Are Social, «Digital in 2022,» Madrid, 2022.

[2] R. Fernandez, «Previsión del número de usuarios mensuales activos (MAU) de Twitter a nivel mundial desde 2021 hasta 2025,» 11 11 2022. [En línea]. Available: https://es.statista.com/estadisticas/636174/numero-de-usuarios-mensuales-activos-de-twitter-en-el-mundo/. [Último acceso: 13 11 2022].

[3] P. Mansanse, «Twitter es la red donde la información política tiene mayor relevancia,» Blog de Twitter, 2019.

[4] E. K. &. E. L. Steven Bird, Natural Languahe Processing with Python, Sebastopol, CA: O'Reilly, 2009.

[5] V. M. Sebastian Raschka, Python Machine Learning, Aprendizaje automatico y aprendizaje profundo con Python, scikit-learn y Tensor Flow, Marcombo, 2019.

[6] BookDown Org, «Data Sciencie con R».

[7] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow, Sebastopol: O´Reilly, 2019.

[8] M. Kubat, An Introduction to Machine Learning, Coral Gables, Florida: Springer, 2017.

[9] O. J. G. &. H. Calvo, «A Comparison Between Two Spanish Sentiment,» *Springer International Publishing AG,* nº 10.1007/978-3-319-47955-2 11, pp. 127-138, 2016.

[10] Y. Fernández, «Xataka Basics/ API: Qué es y para qué sirve,» 23 08 2019. [En línea]. Available: https://www.xataka.com/basics/api-que-sirve. [Último acceso: 23 12 2022].

[11] R. E. D. P. T. P. D. H. Andrew L. Maas, «Learning Word Vectors for Sentiment Analysis,» *Association for Computational Linguistics,* 2011.

[12] F. CHOLLET, Deep Learning With Python, New York: Manning Publications Co, 2021.

[13] B. Tarnoff, «Weizenbaum´s nightmares: how the inventor of the firts chatbot turned againts AI,» *The Gardian,* 25 07 2023.

[14] M. Battocchia, «Preprocesamiento de texto para NLP,» 23 07 2020. [En línea]. Available: https://matiasbattocchia.github.io/datitos/Preprocesamiento-de-texto-para-NLP-parte-1.html#Tokenizaci%C3%B3n. [Último acceso: 10 12 2022].

[15] A. B. y. P. G. Peter Bruce, Estadística práctica para ciencia de datos con R y Python, Madrid, España: Marcombo, 2022.

[16] TASS, «workshop on Semantic Analysis at SEPLN 2020,» TASS, 2020. [En línea]. Available: http://tass.sepln.org/2020/. [Último acceso: 10 12 2022].

[17] J. G. G. J. P. M. R. H. Paula Rendón Cardona, «Self-Supervised Sentiment Analysis in Spanish to Understand the University Narrative of the Colombian Conflict,» *Applied Sciences,* 2022.

[18] D. R. &. D. Ravichandran, «Semi-Supervised Polarity Lexicon Induction,» *12th Conference of*

] *the European Chapter of the ACL,* p. 675–682, 2009.

[19] E. &. A. A. M.Alshari, «Improvement of Sentiment Analysis based on,» *2017 28th International Workshop on Database and Expert Systems Applications,* 2017.

[20] D. K. J. P. J. T. D. Cutting, «Cluster-based Approach to Browsing Large Document Collections.,» *ACM SIGIR Conference,,* 1992.

[21] E. G. C.J. Hutto, «VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text,» *Proceedings of the International AAAI Conference on Web and Social Media,* vol. 8, nº 1, pp. 216-225, 2014.

[22] S. J. Salim Sazzed, «SSentiA: A Self-supervised Sentiment Analyzer for classification from,» *ELSEVIER,* vol. 4, nº 100026, 2021.

[23] X, «X Developer Plattform,» [En línea]. Available: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/overview. [Último acceso: 16 10 2023].

[24] Scikit-Learn, «scikit-learn.org,» 12 11 2023. [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[25] KeepCoding, «Ejercicio sobre similitud entre vectores,» 05 12 2023. [En línea]. Available: https://keepcoding.io/blog/ejercicio-sobre-similitud-entre-vectores/#:~:text=Con%20el%20uso%20del%20producto,dichos%20vectores%20son%20muy%20similares.

[26] A. a. R. Hernández, «Detección de polaridad por tópicos para textos cortos en,» *Trabajo de Diploma - Universidad Central "Marta Abreu" de las Villas.*

[27] Scikit-Learn., «Scikit-Learn.org,» 12 11 2023. [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[28] E. M. &. A. Azman, «SENTI2VEC: AN EFFECTIVE FEATURE EXTRACTION TECHNIQUE FOR SENTIMENT ANALYSIS,» *Ibb University, Yemen,* pp. 240-251, 2020.