Vigilada Mineducación

DEFINICIÓN DE UNA METODOLOGÍA PARA ANÁLISIS DE DISCURSO BASADO EN LINGÜÍSTICA COMPUTACIONAL Y TÉCNICAS DE APRENDIZAJE DE MÁQUINA

Definition of a methodology for discourse analysis based on computational linguistics and machine learning techniques

DAIAN PAOLA FAJARDO BECERRA

Tesis de Maestría

Asesor

Edwin Nelson Montoya Múnera

UNIVERSIDAD EAFIT

ESCUELA DE CIENCIAS

MAESTRÍA EN CIENCIAS DE LOS DATOS Y LA ANALÍTICA

MEDELLÍN

2023

Definición de una metodología para análisis de discurso basado en lingüística computacional y técnicas de aprendizaje de máquina

Tesis enviada para la Maestría (M.S) de Ciencia de Datos y Analítica de la Universidad EAFIT en Medellín, Colombia.

Autor: Daian Paola Fajardo Becerra

Director: Edwin Nelson Montoya Múnera

Codirector: Leandro Fabio Ariza Jiménez

Fecha de envío: febrero 24 de 2023

Palabras Clave:

Metodología, Análisis de sentimientos; Análisis de discurso; Analítica de texto; Tokenización, Stopwords, Lemmatization, Procesamiento de lenguaje natural, Bag of words, LDA, Clasificación, precisión del Modelo, Medición, Clustering, Parcing, Lingüística computacional, Information Retrieval.

Declaración de Autoría

Por la presente declaro que soy el único autor del trabajo de fin de máster presentado, titulado: " Definición de una metodología para análisis de discurso basado en lingüística computacional y técnicas de aprendizaje de máquina". Las citas literales o análogas están claramente marcadas y se pueden ver sus referencias en la bibliografía.

Febrero 24, 2023 Daian Fajardo

Agradecimientos

Durante el desarrollo de esta tesis, he recibido un gran apoyo por parte del director de tesis Edwin Montoya y codirector Leandro Ariza, por lo cual quisiera expresar mi sincera gratitud por su paciencia, motivación entusiasmo y orientación, ya que por medio de sus comentarios y correcciones me ayudaron durante el proceso de investigación y redacción logre tener un conocimiento más profundo en técnicas para el procesamiento del lenguaje natural con técnicas de aprendizaje de máquina.

También doy gracias a mi familia, ya que sin ellos no podría haber finalizado la maestría que durante los tiempos de cuarentena me ayudaron a continuar estudiando, para poder cumplir esta meta.

Resumen

Las diferentes acciones realizadas por un ente regulador del estado, generan múltiples opiniones entre los ciudadanos, las cuales forman debates entre las personas haciendo que se encuentren de acuerdo, desacuerdo o parcialmente de acuerdo con las decisiones o estrategias planteadas. Con el fin de conocer las opiniones de los ciudadanos, en Chile se origina un proyecto llamado "*Tenemos que hablar Chile*" el cual realizaba preguntas estructuradas a un grupo de ciudadanos, donde la respuesta de cada persona era clasificada por el moderador. Dicha etiqueta fue utilizada para diferentes análisis de discurso que se empezaron a desarrollar sin ningún orden específico.

Este proyecto fue replicado en Colombia, bajo la misma dinámica para así conocer las opiniones de los ciudadanos, sin embargo, las técnicas utilizadas fueron diferentes al proyecto chileno. Como resultado, se observa que a pesar de que ambos proyectos tenían la misma dinámica y buscaban un resultado similar, no se pudo reutilizar las técnicas desarrolladas en el proyecto de Chile en Colombia. Debido a esto, la propuesta de este proyecto de maestría busca la implementación de una metodología que permite usar diferentes técnicas de análisis de discurso basado en lingüística computacional y aprendizaje de máquina que dote al equipo de analistas con un esquema de etapas las cuales contarán con herramientas y técnicas de *Natural Language processing* (NLP, por sus siglas en inglés) para mejorar la eficiencia de este tipo de proyectos.

Dentro de este proyecto se puede destacar las fortalezas del director quien tiene una alta experiencia en *Machine Learning* (ML, por sus siglas en ingles) y de NLP, además de las fortalezas del codirector con un amplio entendimiento del proyecto de "*Tenemos que Hablar Colombia" (TQHC),* y finalmente el estudiante de este proyecto con una base en la Maestría de Ciencia de los Datos y Analítica para generar una investigación sobre las técnicas de NLP. Por conveniencia, este proyecto utiliza el idioma inglés para los términos referentes a la ciencia de los datos, con el fin de evitar sesgos en su traducción, sin embargo, su explicación y modo de aplicación serán dados en español.

Contenido

Lista de Figuras

Lista de Tablas

1. Introducción

1.1. Planteamiento del problema

Este proyecto se enmarca en la iniciativa de TQHC (https://tenemosquehablarcolombia.co/), proyecto inspirado en una iniciativa previa en Chile (https://www.tenemosquehablardechile.cl/). Estos proyectos consisten en una serie de entrevistas a miles de ciudadanos para indagar acerca de diferentes temáticas que podrían utilizar las administraciones públicas para tomar acciones. Las entrevistas realizadas son estructuradas y las respuestas de los ciudadanos se transcriben en forma estructurada y con oraciones bien formadas, lo que facilitará procesos posteriores de análisis de texto, discurso o en general extracción de información y conclusiones útiles que son divulgados en diferentes esquemas de reportes publicados.

Dado el alto volumen de datos, contar con herramientas computacionales que apoyen el proceso de extracción de información, conocimiento y análisis de la información entregada por los participantes se hacen muy necesarias. Tanto en el proyecto de Chile como en el de Colombia, se utilizaron varias técnicas de análisis exploratorio en texto y muchas veces la ejecución de pruebas de tipo *ensayo-error*, lo que hace difícil replicar estos proyectos en otros contextos o países con iniciativas similares. Adicionalmente, los líderes principales del proyecto de Chile apoyaron el desarrollo del proyecto en Colombia y se presentó muchas situaciones exploratorias y nuevamente de ensayo-error, lo que condujo a mucho esfuerzo del equipo técnico en Colombia ocasionando varios reprocesos. Asimismo, en la iniciativa chilena, no tuvo en cuenta el uso de técnicas basadas en aprendizaje de máquina, o nuevos modelos de procesamiento de texto basado en redes neuronales.

En síntesis, la falta de una metodología que ayude a desarrollar este tipo de proyectos hace que los analistas e ingenieros utilicen muchas técnicas exploratorias y del tipo *ensayo-error*, que al ser usadas de una forma no estructurada disminuye la eficiencia de reutilización en otros proyectos similares, lo cual implica un costo en tiempo significativo entre proyectos.

Por esta razón se busca plantear una metodología que por medio una serie de etapas se logre identificar cuáles técnicas son más efectivas en la exploración de nuevos modelos de lingüística computacional, ML o aprendizaje profundo que más se adecuen al procesamiento de texto y entendimiento natural del lenguaje con el fin de extraer de forma automática o semiautomática información relevante de las interacciones que existieron en dichas entrevistas. Esto tendrá como beneficio que los proyectos sean estandarizados, enfocados en los éxitos obtenidos y evitando los errores que se pudieron realizar en proyectos anteriores y así poder replicar en futuros proyectos, mejorando los tiempos de desarrollo y análisis.

1.2. Justificación

Este proyecto busca la forma de crear una metodología que busque la generalización y estandarización en proyectos de análisis de discurso por medio de un conjunto de pasos, diferentes técnicas de análisis exploratorio de texto y ML, que eviten la falta de eficiencia y efectividad mostrada en proyectos similares, donde los analistas requieren invertir un tiempo significativo en la búsqueda y pruebas de diferentes técnicas para extraer resultados, información o conocimiento y adicional a esto, al no poder reutilizar técnicas o fases, hace que estos proyectos no puedan ser replicables fácilmente.

Por otro lado, se puede evidenciar que durante los últimos años, han aparecido grandes avances en las técnicas de NLP basado en ML y *Deep Learning*, esto debido a la disponibilidad de alto poder computacional y de datos de entrenamiento que existen (Fernández J, 2021), por lo que el número de modelos y técnicas es significativo, por lo que la creación de una metodología basada en análisis exploratorio de técnicas clásicas y avanzadas de ML para proyectos de análisis de discurso ayudará a los analistas a un desarrollo eficaz de los proyectos, y así poder enfocarse en el análisis de los resultados obtenidos en los modelos y servir de guía para proyectos futuros.

1.3. Objetivos

General:

Definir una metodología de análisis de discurso en texto que aplique diferentes técnicas de análisis de texto, técnicas de aprendizaje de máquina y lingüística computacional para facilitar el análisis de información textual en entrevistas basadas en preguntas y respuestas de opinión.

Específicos:

- Realizar una revisión del estado del arte de diferentes metodologías empleadas para minería de texto y/o análisis de discurso, también realizar la revisión en técnicas de aprendizaje de máquina aplicado al análisis de texto en contextos similares a este proyecto.

- Realizar una apropiación de las diferentes técnicas de minería de texto y análisis de texto aplicado al proyecto TQHC, con el fin de identificar y entender las diferentes etapas y las diferentes técnicas/modelos empleados, los cuales servirán de base al conjunto de modelos y técnicas a emplear en la construcción de la metodología.

- Analizar y seleccionar las técnicas/modelos que apoyan el análisis de discurso desde la lingüística computacional.

- Proponer una metodología mediante un conjunto de etapas y de técnicas/modelos más adecuados para proyectos que requieren un análisis en texto de opiniones y respuestas a preguntas abiertas bien formadas en diferentes contextos de uso, evaluando y seleccionando las técnicas que mejor apoyan el análisis de discurso mejorando la replicabilidad en otros proyectos similares.

- Aplicar la metodología propuesta teniendo en cuenta los datos y contexto del proyecto TQHC.

1.4. Metodología del proyecto

La metodología que se seguirá dentro de este proyecto es una adaptación del modelo del ciclo de vida de los datos de la metodología *Team Data Science Process* (TDSP por sus siglas en ingles), donde por medio de un proceso estándar se logra la estandarización de procesos de minería y analítica de datos, obteniendo un gran dominio en necesidades del negocio, al igual que el de los datos (Microsoft Documents, 2021).

Esta adaptación comenzará por un preprocesamiento de datos, seguido por un análisis exploratorio de datos (*Exploratory Data Analysis* - EDA - por sus siglas en inglés) en el cual existirá un comparativo de técnicas y finalmente la construcción de diferentes clasificadores que se analizarán, para así seleccionar el modelo que tenga un buen desempeño con el fin de lograr seleccionar el más adecuado para este proyecto.

Dentro de la primera fase de la metodología propuesta, se busca una recolección y entendimiento de datos. Dentro de la recolección se tiene como objetivo obtener un conjunto de datos de las diferentes sesiones realizadas en el proyecto TQHC, los cuales contienen respuestas a preguntas estructuradas sobre un tema específico y el cual se espera responda ¿Qué mantendría?, ¿Qué cambiaría? y ¿Qué mejoraría? Dentro del entendimiento de datos, se espera un dominio sobre las diferentes variables y metadatos que tiene el conjunto de datos.

En la segunda fase, se requiere realizar un preprocesamiento de datos, el cual buscará realizar las diferentes tareas de limpieza y preparación del texto a analizar, algunos ejemplos como la separación de las palabras de una oración a una lista de tokens, agrupación de la misma palabra así esta tenga diferentes formas gramaticales dentro del texto y demás técnicas revisadas en el marco teórico y revisión de la literatura de este proyecto.

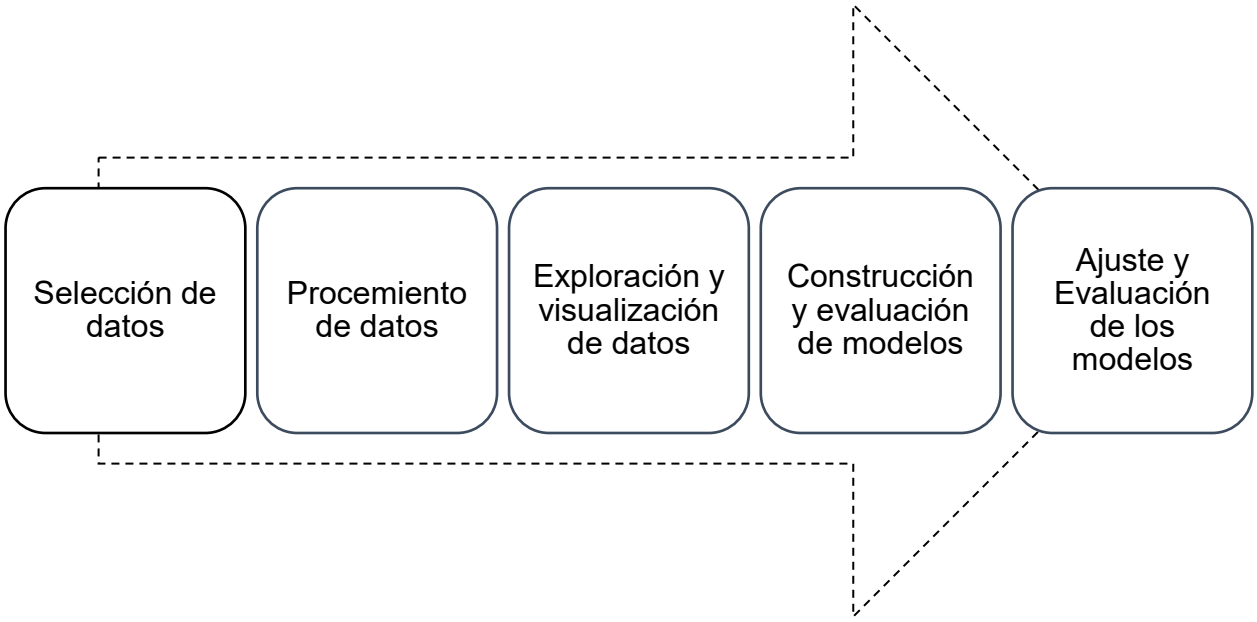
Dentro de la tercera fase se realiza un EDA y la visualización de los diferentes resultados, esta exploración sistemática del conjunto de datos ayudará a conocer las palabras con mayor frecuencia, tópicos importantes, análisis de sentimientos entre varias técnicas más que serán aplicadas en el capítulo 4 de este proyecto.

Dentro de la cuarta fase se busca la construcción de clasificadores que ayuden a la correcta identificación del indicador verbal de las respuestas de los ciudadanos. Dentro de esta fase, se realizará la evaluación de dichos modelos con el fin de obtener el desempeño de cada clasificador.

En la última fase, se realizará ajustes a los diferentes clasificadores con el fin de mejorar su desempeño y así poder ser reevaluados y poder seleccionar el más adecuado para el proyecto. Esta metodología se puede observar en la Figura 1.

Figura 1: Metodología del proyecto

Adaptación del ciclo de vida de los datos TDSP (Azure, 2021)



1.5. Estructura del proyecto

Este proyecto cuenta con 6 capítulos, en el cual el primer capítulo se puede encontrar el problema de investigación, objetivos, y metodología del proyecto. Dentro del segundo capítulo se podrá encontrar una revisión de la literatura y marco teórico sobre las técnicas tradicionales y modernas de aprendizaje de máquina en procesamiento del lenguaje natural para el análisis de discurso. En el capítulo 3 se hace la definición de la metodología que se desarrolla dentro del proyecto, luego, ésta es aplicada en el capítulo 4 y en el último capítulo 5 se extraen las conclusiones y se exponen los trabajos futuros.

2. Marco teórico y estado del arte

2.1. Marco teórico

2.1.1. Metodologías en minería de datos

Las metodologías en minería de datos, son una forma de estructurar proyectos por medio de pasos y etapas que permitan la exploración y análisis de datos de una manera estructurada y ordenada con el fin de alcanzar los objetivos de un proyecto, guiando las etapas de preprocesamiento de datos, determinación de modelo y análisis de resultados.

Debido a que este proyecto se enfoca en la creación de una metodología, se realiza una adaptación de TDSP que se basa en la implementación de una metodología ágil desarrollada por el equipo de Microsoft, el cual se enfoca en la creación de un proceso de ciencia de datos, estructurado en cuatro componentes principales. Dentro del primer componente se encuentra el ciclo de vida de los datos, donde se busca tener un entendimiento de los objetivos del negocio a nivel de datos, proyectos y problema a resolver, seguido por la selección y entendimiento de los datos, para luego realizar un modelamiento de datos, que incluye ingeniería de características, transformación, entrenamientos y evaluación de modelos para finalizar con la evaluación y monitoreo constante del modelo implementado. El segundo componente busca dar una estructura a nivel de documentación del proyecto el cual tiene como expectativa un reporte sobre cada una de las fases del ciclo de vida de los datos. El tercer componente busca la investigación de diferentes arquitecturas a utilizar. Finalmente, en el último componente se proporciona un conjunto inicial de herramientas y códigos para poner en marcha la adopción de TDSP dentro de un equipo (Microsoft Documents, 2021).

Otro tipo de metodología que se basa en el ciclo de vida de los datos, es *Knowledge Discovery in Databases* (KDD, por sus siglas en inglés), la cual fue propuesta por Fayad en 1996, en el que se espera la eliminación de ruido y por medio la etapa de 4 fases. En la primera fase se selecciona los datos, al mismo tiempo se desarrolla un entendimiento del problema y aplicaciones a utilizar, seguido por una etapa de preprocesamiento, que

busca la limpieza de los datos y eliminación del ruido de por medio de la búsqueda de características útiles; luego se implementa una fase de transformación donde se realiza EDA, se crea la hipótesis del modelo y se construye el modelo que se será la solución del problema y finalmente, en la última fase se realiza una evaluación por medio de visualizaciones y así encontrar patrones o el resultado del modelo, dentro de este proceso también se busca la comprobación de la solución y posibles correcciones que se deban realizar. (Hotz N, 2021).

Adicional a estas dos metodologías, existe *Cross Industry Standard Process for Data Mining* (CRISP-DM, por sus siglas en ingles) y *Sample, Explore, Modify, Model, Asses* (SEMMA, por sus siglas en inglés) que están enfocadas a la minería de datos para grandes volúmenes de datos. CRISP-DM es una metodología propuesta por la comunidad europea en 1999, el cual se enfoca en desarrollar una plataforma para minería de datos y así lograr estandarizar los procesos de minería y analítica por medio del entendimiento del negocio/proyecto y de los datos, seguido por una preparación que incluye la selección, limpieza y estructura de los datos, para luego realizar un modelamiento e implementación de los modelos que resolverán el problema que por consiguiente serán evaluados, buscando responder los objetivos del negocio/proyecto y finalizar con una fase de desarrollo para así realizar las automatizaciones de procesos necesarias y poder mantener y monitorear la solución (Huber S, Wiemer H, 2019).

La metodología SEMMA, tiene un total de 5 fases que corresponden al nombre de la metodología, buscando una muestra de datos que pueda ser subdivido en diferentes conjuntos de datos de pruebas, luego entrar a una fase de exploración, y así encontrar las variables más importantes para el modelo, seguido por una fase de transformación de variables y eliminación de ruido o valores atípico que puedan afectar los modelos, una vez estas tres fases están finalizadas, se entra a la creación de modelos con diferentes técnicas para que así se pueda resolver el problema planteado y finaliza el modelo con un evaluación en cada una de las técnicas que se desarrollaron para cada uno de los modelos y así poder realizar un reporte final que contenga dicha evaluación y solución (Shfique U, Qaiser H, 2014).

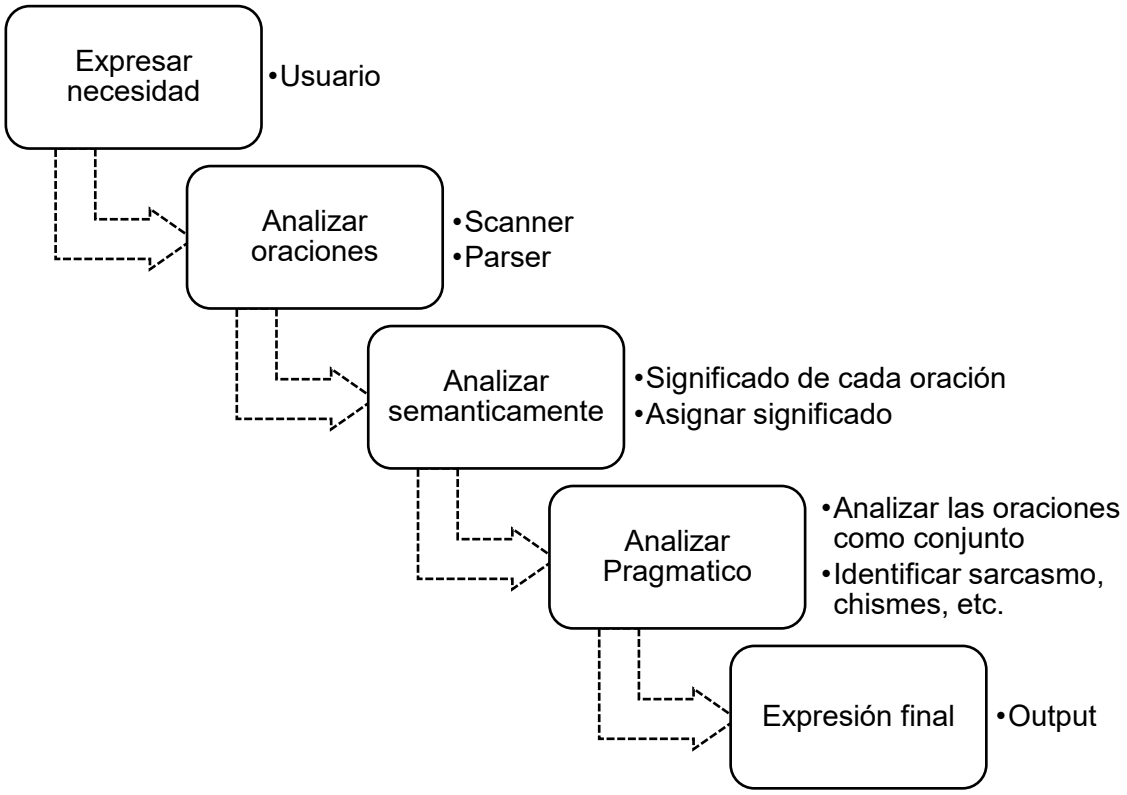
2.1.2. Procesamiento de lenguaje natural

Procesamiento de lenguaje natural (*Natural Language Processing* – NLP- por sus siglas en inglés) es un campo de la ciencia de datos y la Inteligencia Artificial (*Artificial intelligence* – AI por sus siglas en inglés) que se originó durante la segunda guerra mundial, donde se buscaba una forma eficiente traducir diferentes textos de un idioma a otro (Brookshear J, 1993). NLP busca que las computadoras puedan analizar el lenguaje humano por medio un conjunto de componentes léxicos que son organizados por reglas que pueden ser sintácticas o gramaticales y así llegar a darle un sentido semántico, por esto está técnica se basa en la sintaxis y la semántica, donde el la sintaxis se encarga de dar el orden correcto de los componentes léxicos y la semantica tiene como objetivo dar un significado correcto a cada oración. Con el fin de identificar si las palabras y oraciones de los textos se encuentran correctas se debe realizar dos análisis que son fundamentales para NLP, el primero es el *Scanner* que es un analizador lexicográfico, encargado de identificar los componentes léxicos de la oración y el segundo *Parser* que es un analizador sintáctico para identificar si se cumple un orden gramatical entre los elementos identificados por el *Scanner* (Vicente M, Barros C, Lloret E, Peregrino S, 2007).

Dentro de la arquitectura de NLP se pueden identificar 5 niveles (ver figura 2), iniciando con un nivel fonológico, donde las palabras se relacionan con los sonidos, luego estas palabras se empiezan a construir a partir de unidades de significado más pequeñas o también llamadas morfemas, este nivel es llamado morfológico. El tercer nivel es sintáctico y las palabras pueden unirse para formar oraciones, y así cada palabra tiene una estructura y regla dentro de la oración, desde aquí se pasa al cuarto nivel semántico, que busca el significado de las palabras y como significado de cada una le da un sentido a una oración y se finaliza en el nivel pragmático, en el cual las oraciones son analizadas y entender el contexto en el que se usa, y así observar el impacto de su significado (Cortez A, Vega H, Pariona J, 2009).

Figura 2: NLP

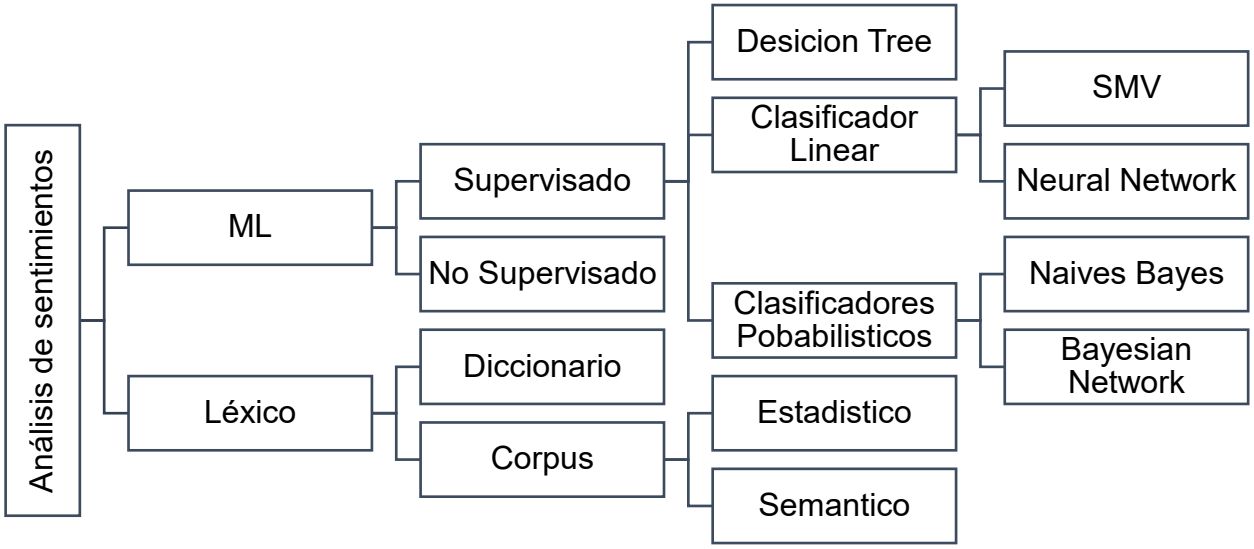
Adaptado de Procesamiento de lenguaje Natural, Cortez A, Vega H, Pariona J (2009).



NLP contiene varias subáreas, como se muestra en la figura 3, que se utilizarán dentro de este proyecto. La primera de ellas es el análisis de sentimientos, el cual es una técnica que busca identificar las emociones que se encuentran en un texto, el objetivo es dar una connotación positiva, negativa o neutral. Los tipos de métodos utilizados para el análisis de sentimientos, están basados en léxico y aprendizaje de máquina con métodos supervisados y no supervisados, en la figura 3 se puede visualizar las diferentes técnicas y algunos ejemplos de cada una (Ahmed, El Tazi, & Hossny A, 2015).

Figura 3: Técnicas de clasificación de sentimientos

Adaptado de Sentiment classification techniques, Medhat et al (2014).



El método basado en léxico, se basa en el análisis de sentimientos por cada palabra o frase, en el cual por medio de una ejecución manual o por medio de una creación de una semilla de palabras que se expande automáticamente como se puede ver en la tabla 1.

Tabla 1: Tipos de lexicón

Tabla adaptada de Yadav & Elchuri, 2013

Tipo de Lexicón	
Palabras de sentimiento	"Las palabras de sentimiento positivo y negativo tienen una puntuación de sentimiento de +1 o -1 para indicar la polaridad respectiva"
Palabras de negación	"Son las que invierten la polaridad del sentimiento, normalmente aparecen antes de la palabra de sentimiento."
Palabras de negación ciega	"Operan a nivel de frase y señalan la ausencia o presencia de algún sentido no deseado en una característica del producto."

Palabras de división	“Las palabras de división son las que se utilizan para dividir las frases en cláusulas. La lista de palabras de división se compone de conjunciones y signos de puntuación.”

Dentro de las técnicas para el análisis de sentimientos que son dadas desde el aprendizaje de máquina de forma supervisada, se puede encontrar la propuesta por Pang y Lee en 2002 en donde introdujeron por primera vez un análisis de sentimientos realizado con aprendizaje de máquina con un modelo de entrenamiento de opiniones de películas etiquetados con los algoritmos de Naive Bayes, Clasificación de máxima entropía y *Support Vector Machine* (SVM, por sus siglas en ingles). Basados en este estudio en 2014 Dong y Wei propusieron una red neuronal recursiva adaptativa la clasificación de sentimientos en redes sociales como Twitter y a este tipo de red neuronal lo llamaron “Coooolll” (Tang, Wei, Qin, Liu y Zhou, 2014).

Los métodos no supervisados se puede encontrar uno de los estudios que empezó con la combinación entre conjunto de datos con una parte de datos etiquetados y otra parte no etiquetados con el fin de reducir costos para el análisis de sentimientos, fue propuesta por Tan y Cheng en 2014, donde propusieron un clasificador Naive Bayes adaptado, sin embargo, Zhou y Chen en ese mismo año lograron tener una precisión del 79,4% con un modelo semi supervisado por medio de *Deep Belief Networks* (DBN, por sus siglas en inglés) (S. Zhou, Chen, & Wang, 2014).

La segunda subárea que será utilizada en este proyecto, será la clasificación de texto, la cual es utilizada en NLP para poder organizar y categorizar textos de documentos en diferentes etiquetas por medio de palabras claves. La clasificación de documentos es necesaria debido a que categorizar documentos consume un tiempo considerable debido al volumen de datos y estructura en la que vienen los documentos, adicionalmente es necesario mantener un criterio consistente a largo tiempo, evitando errores humanos por subjetividad (Kowsari K, 2019).

En la actualidad se puede realizar la clasificación de textos de forma automática o manual. Dentro de la forma manual, se puede encontrar la forma en que se realizó el proyecto *Tenemos que hablar Colombia* que es por medio de un anotador que interpreta

el contenido que del documento, texto o frase y lo categoriza dentro de unas etiquetas específicas.

La forma automática está basada en modelos de aprendizaje de máquina, NLP u otras técnicas de AI que, de forma automática, clasifica el documento, texto o frase en una etiqueta con la ventaja de ser más exacto y con menor de consumo de tiempo. Para construir un clasificador automático, se puede realizar por medio de un enfoque basado en reglas, las cuales son construidas por medio de grupos organizados que tienen un conjunto de reglas lingüistas que son elaboradas de forma manual, luego, se espera que el sistema empiece la organización de documentos por los elementos semánticamente relevantes que detectaran las categorías a la que pertenece su contenido.

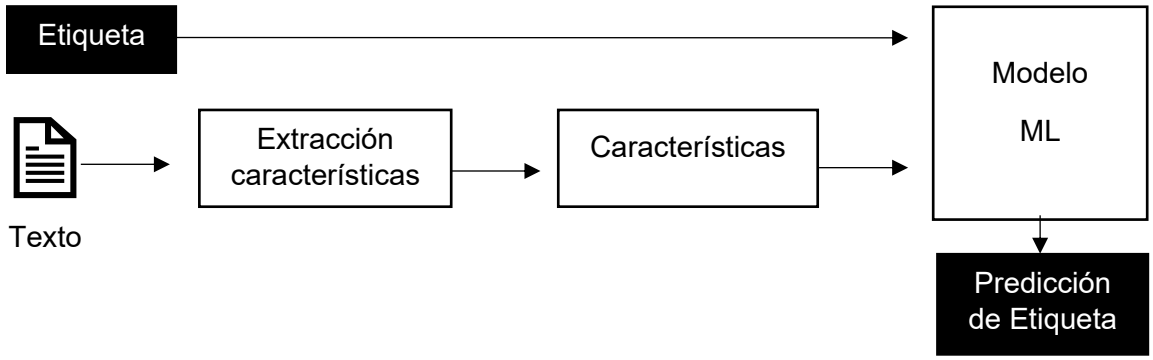
La segunda forma de construir un clasificador automático y la que será utilizada dentro de este proyecto es por medio de ML, el cual aprende la clasificación de documentos por medio de la historia y observaciones que se tienen en el pasado, para este tipo de clasificación es necesario contar con un conjunto de datos que ya tenga asignado una etiqueta para cada texto, documento o frase, y así los modelos de ML podrán aprender por medio de asociaciones dentro de las palabras del documento y su relación con la etiqueta que tiene asignada. Para este tipo de clasificadores es necesario realizar una extracción de características con el fin de realizar una representación numérica en forma de vector[1] y de allí alimentar un modelo de ML para la clasificación de texto[2] (ver figura4) (Kowsari K, 2019).

[1] La representación numérica será explicada con mayor detalle en los títulos 2.1.5 de este proyecto

[2] Los clasificadores de texto se explicarán en mayor detalle en el titulo 2.1.8 de este proyecto

Figura 4: Clasificación de Texto

Adapatado Text calssifcation Algorithms de Kowsary K, 2019.



Dentro de NLP existen varias subáreas que se explicarán sin entrar en detalles ya que no tendrán una aplicación directa en este proyecto, pero que dan un mayor entendimiento de este campo de estudio. Entre las primeras subáreas que se encuentra en NLP es *Natural Language Generation* (NLG, por sus siglas en inglés) que busca cómo se podría construir aplicaciones que puedan producir por sí mismas textos con una alta calidad, este tipo de texto que es generado, también puede ser convertido a un formato vocal mediante servicios como conversión de texto a voz (Vicente M, Barros C, Lloret E, Peregrino S, 2007.

Otra subárea es *Natural Language Understanding,* (NLU, por sus siglas en inglés) donde el objetivo principal es interpretar un mensaje con el fin de dar un significado y así poder conocer su intención, utilizando los análisis de la sintáctica y semántica de un texto. NLU también realiza una relación entre las palabras y las frases y así poder encontrar la intención en los diferentes textos. Otra rama de NLP es *Information Retrieval* (IR por sus siglas en inglés) el cual busca procesar textos y poder recuperar partes específicas del documento por medio de palabras claves, esto es logrado por medio de la organización, almacenamiento, recuperación y evaluación de la información de documentos. *Machine Translation* (ML, por sus siglas en inglés) es una subárea encargada de la traducción automática de textos a diferentes idiomas, manteniendo el significado del texto de entrada y evitando ambigüedades en el texto de salida (Vicente M, Barros C, Lloret E, Peregrino S, 2007).

2.1.3. Análisis de discurso

Existe diferentes tipos análisis al querer realizar un estudio de discurso, los cuales serán utilizados dentro de este proyecto, uno de ellos es el análisis frecuencial que busca tener un listado de palabras que tenga mayor frecuencia o que aparezca más veces y así proporcionar información sobre el tema del texto. La forma más común de visualizar este tipo de análisis es por medio de histogramas y nube de palabras donde se observa las palabras con mayor frecuencia con una fuente más notoria (en tamaño y color) que las demás. Otra forma de revisar la frecuencia de palabras, es con la extracción de características, las frecuencias de la palabra se pueden identificar con la extracción de características por medio de *CountVectorizer* que será utilizado para la construcción de BoW[3]. Este tipo de análisis proporciona una información sobre el tema que se trata en el documento, importancia de palabras, palabras claves, *Stopwords*[4] entre muchos más (Firti A, Rachamadita A, Muhammad A, 2019).

Un segundo estudio que se aplicará en el desarrollo de este proyecto, será el análisis de sentimientos el cual busca evaluar o determinar el tipo de emociones que genera un texto. La información dada por una persona puede ser separada en hechos que es considerado como objetivo y declaraciones de algo que ya ha sucedido y que puede existir evidencia (Firti A, Rachamadita A, Muhammad A, 2019).

Dentro del análisis de discurso también se busca una relación entre aquellas palabras que pueden ser sustituidas (Paradigmática) y aquellas que pueden ser combinadas (Sintagmática). Para poder identificar dichas palabras paradigmáticas, se calcula la similaridad que tienen dentro del documento, y para identificar las palabras sintagmáticas se busca la frecuencia que dos palabras aparecen juntas y se compara con su coocurrencia individual, este tipo de relación ayuda con los análisis de morfemas los cuales logran clasificar por medio de una etiqueta a la categoría gramatical se puede utilizar el analizador morfológico, el cual por medio de *Part of Speech and Tagger* (POS Tagging, por sus siglas en inglés) se busca asignar una etiqueta de categoría a los tokens

[3] BoW será explicado en mayor detalle dentro del título 2.1.5 de este proyecto

[4] Stopwords será explicado en mayor detalle el título 2.1.4 de este proyecto

o a una oración. Uno de los usos más frecuentes es la identificación de sustantivos, verbos, adjetivos etc. Durante este procesamiento se puede llegar a tener una estructura de la frase y el significado de las palabras que pertenecen a dicha oración. POS analiza el nivel más bajo de la estructura sintáctica de la oración y así logra realizar un etiquetado al discurso. En la tabla 2 se puede observar las diferentes etiquetas que se le pueden asignar a las palabras (Patel R, 2017).

Tabla 2: Etiquetas POS Tagging:

Adaptado de Patel R, 2017

Tipo	POS	Ejemplo
Adjetivo	ADJ	Grande, Viejo, Verde, Primero
Adposición	ADP	En, Durante
Adverbio	ADV	Muy, mañana, abajo, donde, alli
Auxiliar	AUX	Soy, deberia
Conjunción	CONJ	y, o, pero
Conjunción Coordinativa	CCONJ	y, o, pero
Determinante	DET	un, una
Interjección	INTJ	psst, ouch, bravo, hello
Sustantivo	NOUN	chica, gato, árbol, aire, belleza
Numero	NUM	1, 2017, uno, siete, IV, MMXIV
Partible	PART	’s, not
Pronombre	PRON	Yo, ella, yo mismo, alguien
Nombre Propio	PROPN	Mary, John, London, NATO, HBO
Puntuación	PUNCT	., (,), ?
Conjunción subordinante	SCONJ	Si, mientras
Símbolos	SYM	$, %, §, ©, +, −, ×, ÷, =, :), 😝
Verbo	VERB	Correr, comer
Otros	X	sfpksdpsxmsa
Espacios	SPACE	

Otra forma de analizar los morfemas, es dado por *Name Entity Recognition (*NER, por sus siglas en inglés) es una técnica que puede lograr escanear un artículo y extraer entidades que fundamentan el texto y así poder clasificarlas en categorías predefinidas como organizaciones, fechas, nombres de personas, localizaciones, entre otras. Este proceso se basa en la identificación de frases sustantivas por medio de un análisis sintáctico y así tener un etiquetado de la parte de la oración, luego en un segundo paso se clasifican las frases y finalmente se desambigua la entidad. Al igual que en POS Tagging, NER asigna una etiqueta a las palabras, las cuales se pueden ver en la tabla 3 (Patel R, 2017).

Tabla 3: Etiquetas NER

Adapatado Patel R, 2017

Tipo	Descripción
PERSON	Nombre de personas
NORP	Nacionalidades, grupos religiosos o políticos
FAC	Edificios, Aeropuertos, puentes
ORG	Compañías, agencias, institutos
GPE	Países, ciudades
LOC	Montañas, ríos
PRODUCT	Objetos, vehículos
WORK_OF_ART	Títulos de libros, canciones
LAW	Nombre de documentos
LANGUAGE	Idioma
DATE	Fechas o periodos de tiempo
TIME	Cuando es menor a un día
PERCENT	Porcentajes
MONEY	Valores monetarios
QUANTITY	Medidas de distancia o peso
ORDINAL	Números ordinales
CARDINAL	Números de todo tipo

Otro tipo de estudios que existen para NLP son los análisis sintácticos los cuales buscan las reglas gramaticales de una oración y así conocer la validez de una oración. Este tipo de análisis se apoya en técnicas como *Parser* y *Syntactic Parser*, donde se identifican sintagmas y su función dentro de la oración.

Existe un análisis sintáctico superficial apoyado por técnicas como *chunks* y un análisis sintáctico profundo que se apoya en técnicas como *Deep* y *Full Parsing*. Con el fin de poder realizar *Parsing*, es necesario utilizar una estructura gramatical que está definida por medio de un modelo matemático lingüístico (Sankart D. 2018).

$$G = (V_T, V_N, P, S)$$

V_T = *Conjunto finito de símbolos terminales ej. Nombre, verbos, adj, adv.*

V_N = *Conjunto finito de símbolos no terminales ej. el, la, los, niño, estudia, corre.*

P = Conjunto finito de reglas.

S = Axioma inicial desde donde se recorrerán las secuencias.

Para poder revisar si se cumple un orden gramatical correcto, se utilizan los árboles de análisis sintáctico, ya que este es una representación de diferentes categorías sintéticas de la oración. En la figura 5 se puede apreciar un ejemplo de un árbol sintético con una estructura gramatical definida, donde:

$$V_T = \{NP, VP, ADJP, ADVP, PP\}$$

V_T = *Conjunto finito de símbolos terminales ej. Nombre, verbos, adj, adv.*

NP = Noun Phrase, Sustantivo que actúa como palabra principal.

VP = Verb Phrase. Verbo que actúa como palabra principal.

ADJP = Adjective Phrase, Adjetivo como palabra principal.

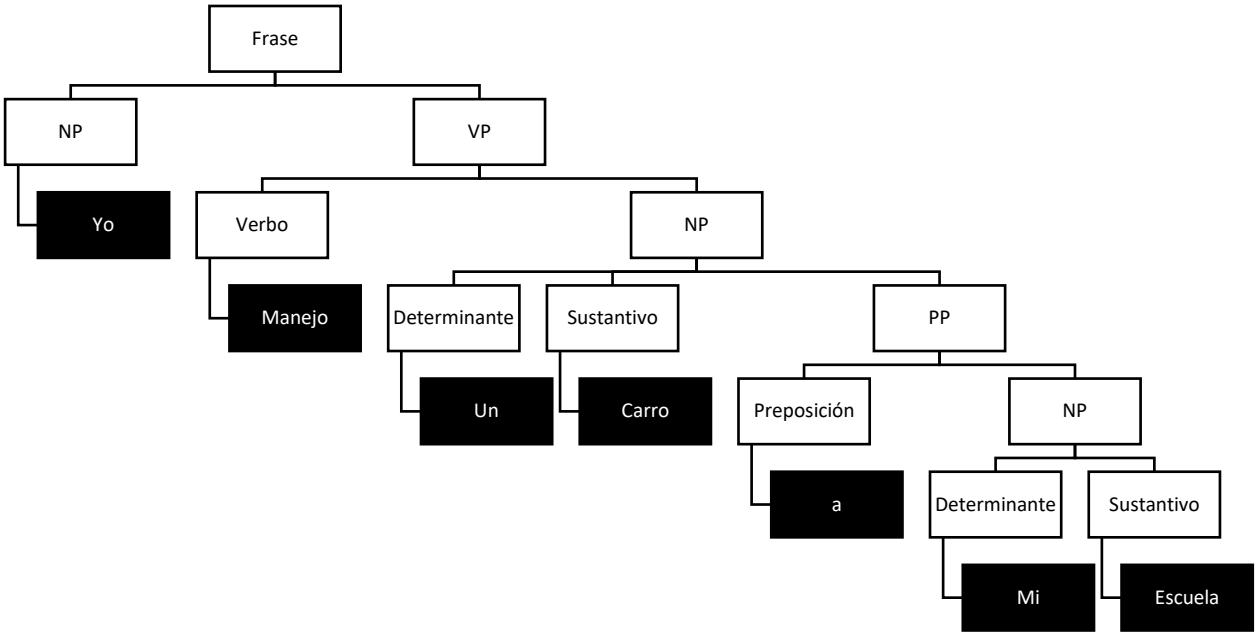
ADVP = Adverb Phrase, Adverbio como palabra principal.

PP = Prepositional Phrase, Preposición como palabra principal.

$$V_N = \{Yo, manejo, un, carro, a, mi, escuela\}$$

Figura 5: Ejemplo de Syntax Tree, NLP

Adaptado de sintax Tree. (Sankart D. 2018)



Finalmente, el ultimo tipo de análisis con el que cuenta NLP es el semántico, el cual busca dar un significado a cada palabra, evitando ambigüedad. Dentro de este análisis se pueden usar técnicas como *Word Sense Disambiguation* (WSD, por sus siglas en ingles) el cual está enfocado a activar las palabras por su uso y contexto en las frases, para la construcción de WSD es necesario el contar con un diccionario el cual contendrá las frases que se les removerá la ambigüedad y un corpus que tiene un alto nivel de anotación que contiene los sentidos correctos o de destino (Firti A, Rachamadita A, Muhammad A, 2019).

2.1.4. Preparación de documentos

La preparación de documentos o preprocesamiento de datos de texto, es un paso preliminar que se realiza antes de empezar con la exploración, análisis, modelado y visualización de datos, y realizar la limpieza necesaria y evitar errores en los modelos.

Se debe comenzar con una limpieza del texto con el objetivo de eliminar caracteres, palabras, símbolos especiales como URLs, emojis y otros caracteres especiales que no son requeridos para el análisis de texto de acuerdo a un contexto concreto de aplicación.

Después de esta limpieza, es necesario un proceso de Tokenización el cual consta de dividir cadenas de texto en tokens, para esto hay que aclarar que una palabra es “un elemento formal (de al menos un morfema) cuyo patrón de ocurrencia puede describirse en términos de una serie ordenada de forma única de otros elementos léxicos que ocurren en su entorno” (Webster J, 1992). Este paso también es conocido como segmentación de texto o análisis léxico, y para lograr realizar esta Tokenización, se pueden emplear diferentes técnicas como *Consonante-Vocal* (CV, por sus siglas en ingles) que es empleado para lenguajes como el coreano, chino que tienen una forma de escritura diferente a la occidental. Para los idiomas occidentales existen técnicas como sílaba, morfemas, sub-palabra, palabra, *byte pair encoding* (el cual busca representar un texto utilizando el menor número de tokens por medio de compresión de datos) (Kyubyong P, Jooong L. 2020).

Dentro de la normalización, se espera colocar todo el texto o documento en igualdad de condiciones, esto con el fin de poder realizar un procesamiento de forma uniforme, este procedimiento se puede ejecutar siguiendo una serie de pasos los cuales empiezan con la preparación y EDA (Mayo M, 2017).

Uno de los pasos importantes a destacar en esta primera fase es la eliminación de *Stopwords,* las cuales se definen como palabras con una frecuencia alta, pero con poca importancia. Para realizar la eliminación de *Stopwords* se puede realizar por medio de un modelo clásico, el cual elimina las palabras desde una lista de palabras creadas. Este tipo de listas se puede obtener por medio de librerías que ayudan no solo a adquirir las

palabras a eliminar, sino que también se pueden agregar nuevas palabras o excluir palabras que sean necesarias para el estudio, algunas librerías como *Natural Language Toolkit* (NLTK, Por sus siglas en ingles) que contiene alrededor de 179 palabras de parada, para el idioma inglés, también existe la librería *SpaCy* ya que contiene 369 palabras en inglés, adicional *Gensim* que también es una librería que usa una moderna estadística para machine learning por lo que la lista de palabras es diferente y contiene un total de 337 palabras y finalmente *scikit-learn* que es una librería utilizada para machine learning y considerada como una de las más poderosas contiene un total de 319, en los últimos análisis, se puede ver que Scikit-learn y SpaCy han tenido los mejores resultados, y entre estas dos librerías, la diferencia es casi nula (Khanna C, 2021).

Otros métodos que son efectivos para la eliminación de *Stopwords* son los métodos basados en la ley de *Zip* (Zip'f Law) (Z-Methods) en el que se adiciona a la lista de *Stopwords* las palabras con mayor frecuencia (TF-High) y aquellas palabras que solo aparecen una vez (TF1). También existe un modelo que tiene en cuenta la información mutua (MI, por sus siglas en ingles) que es un método supervisado el cual revisa la información entre la palabra y el documento y así determinar qué tanta información da una palabra en específico. Una palabra con un bajo índice indica que no es relevante para el documento por lo que puede ser eliminada. Y un tercer método propuesto en 2005, es un muestreo aleatorio (TBRS, por sus siglas en inglés) donde se detecta las palabras vacías iterando sobre pequeñas muestras de textos separados y que se seleccionan aleatoriamente (Khanna C, 2021).

La segunda fase para la preparación de documentos es la aplicación de *lemmatization* y *stemming*, que durante varios estudios de clasificación de documentos o identificación de tópicos se busca realizar *lemmatization* con el fin de no perder información que puede ser esencial para las diferentes tareas, por esto, en 2001 Alkula R propuso una conversión de cadenas de caracteres con texto plano y con palabras con significado, mostrando que podría mejorar el sistema de recuperación de información.

Stemming (Raghavan, P. y Schütze, H. 2008) es definido como un proceso donde se elimina cualquier sufijo, prefijo, infijo o circunfijos de la palabra. (ej. "encontrarás" = "encontr"). En este proceso se realiza es una búsqueda de las diferentes variantes de las

palabras con su raíz, sin embargo, en varias técnicas que se conocen de *stemming* se basan en cortar las palabras haciendo que se pierda el significado o el contexto de esta, por lo que este proceso se debe realizar con mayor cuidado y en algunos análisis de texto se puede llegar a omitir. Uno de los algoritmos más usados en *stemming* es el de Porter (*snowball*) que fue creado en 1980 y que ha sufrido varias actualizaciones, siendo la más reciente en 2014 que se enfoca en 5 fases para reducción de palabras, dentro de cada paso se aplican reglas hasta que una de ellas supera las condiciones, este algoritmo ha sido el más popular por lo que se han sugerido varias modificaciones que han mejorado el algoritmo básico.

Otros algoritmos utilizados para stemming son Lovin y Lancaster los cuales son propuesto en 1968 donde se elimina el sufijo más largo de cada palabra, luego está recodificada y así buscar su palabra raíz, este algoritmo es bastante rápido y manejaba plurales irregulares, sin embargo, la deficiencia de este algoritmo recae en su falta de sufijos por lo que era poco fiable, este algoritmo fue mejorado por Paice y Husk el cual agrego al algoritmo un modo iterativo que contiene 120 reglas indexadas, el cual busca encontrar la regla aplicable por el último carácter de la palabra, cada regla indica si es la eliminación o sustitución de un término, sin embargo, este algoritmo es considerado uno de los más pesados y también muestra que se hace exceso de *stemming*. Dawson Stemmer es una extensión de Lovis con una lista mucho más completa con un total de 1200 sufijos, los sufijos se almacenan en orden inverso indexados por su longitud y última letra, una desventaja que se ve en este algoritmo es su complejidad en la implementación.

Lemmatization (Raghavan, P. y Schütze, H. 2008) está definido como el proceso de capturar formas canónicas basadas en el lema de una palabra. (ej. “encontrarás” = “encontrar”). A diferencia de *Stemming*, el proceso de *lemmatization* requiere tener un conocimiento completo del vocabulario y un análisis morfológico con el fin de lematizar correctamente cada una de las palabras. El análisis morfológico de las palabras tiene un mayor beneficio ya que la palabra llega a su forma raíz evitando que se pierda información cuando se requiere realizar algún proceso de recuperación.

Los métodos que se encuentran en la literatura para la creación de *lemmatization* tienen una gran variedad, ya que algunos se basan en la similitud de cadenas para poderlas convertir en su raíz, como lo es el *N-Grammer Stemmer*, normalmente es una cadena de n caracteres que son adyacentes, donde su objetivo principal es el enfoque de palabras similares. Este tipo de algoritmo tiene muchas aplicaciones, sin embargo, ocupa demasiada memoria y almacenamiento para dar un lugar a cada n-grama y sus indices.

Otro tipo de tecnicas, es *Hidden Markov Model Stemmer* (HMM, por sus siglas en inglés) el cual es un método basado en aprendizaje no supervisado, el cual no requiere un conocimiento previo de la lingüística del conjunto de datos y se rige por funciones de probabilidad, para esto, calcula la probabilidad de cada camino y encuentra el camino más probable en el grado de autómatas. También, existe un modelo que se basa en el corpus, busca un mejor rendimiento del Stemmer generando agrupaciones de un léxico sin ninguna entrada lingüista, este tipo de modelo tiene por nombre *Yet Another Suffix Striper Stemmer* (YASS, por sus siglas en inglés) propuesto por Prasenjit Majumder.

Finalmente, existen unos métodos mixtos, donde se utiliza la estadística de *lemmatization* y los métodos tradicionales del stemming. El primero de ellos es *Inflectional and Derivational Methods*, el cual hace un análisis de la morfología como el de la derivativa, este tipo de algoritmos también están basados en corpus. Los algoritmos son de inflexión ya que sus variantes de las palabras están relacionadas con variaciones sintácticas específicas de la lengua y la derivación está relacionada con POS de una frase en la que aparece la palabra.

El segundo método mixto es el *Corpus Based Stemmer*, este algoritmo fue propuesto por Xu y Crof y busca mejorar el *Stemmer* propuesto por Porter por medio de métodos estadísticos buscando una modificación automática de las clases de confluencia que han dado lugar a la raíz común y así adaptarlas al corpus del texto. Y como ultimo método mixto se encuentra *Context Sensitive Stemmer* que fue propuesto por Funchun Peg en donde las palabras de entrada predicen las variantes morfológicas que ayudan a reducir el coste computacional y mejora la precisión (Mohan V, 2020).

Como último paso para la normalización del texto, se requiere transformar el texto todo en mayúscula o minúscula, eliminar puntuación, convertir números a sus equivalentes

en palabras, y adicionalmente la substitución o filtrado que también es conocido como eliminación de ruido, donde se busca eliminar cualquier encabezado, pie de página, markups HTML, XML, metadata, extraer datos que sean valiosos en formatos JSON o bases de datos. Normalmente este tipo fase se maneja con expresiones regulares. (Mayo M, 2017).

2.1.5. Representación de documentos

Para realizar estudios con NLP, es necesario trabajar con vectores de alta dimensión que representan las palabras que se encuentran dentro de un texto, por lo que puede existir un gran número de elementos que no generen información y si se requiere realizar ajustes a los modelos que se construyan, es necesario realizar una representación de características la cual podrá ser procesada por un algoritmo.

Para la creación de esta representación de características, existe la técnica de *Term frequency-inverse document frequency* (TF-IDF, por sus siglas en inglés) la cual realiza una cuantificación de las palabras en un conjunto de documentos, donde TF busca el número de veces que aparece una palabra dentro de un documento, y para realizar el cálculo, se realiza una normalización del valor de la frecuencia, dividiendo la frecuencia entre el número total de palabras en el documento, y para esto es necesario realizar una vectorización de los documentos (Wu C, Luk R, Wong K, Kwok K. 2008).

$$tf(t,d) = \frac{Conteo\ de\ t\ en\ d}{Número\ total\ de\ terminos\ en\ d}$$

t = término

d = documentos

La segunda parte de esta representación de características es dada por DF, donde la frecuencia del documento es el recuento de apariciones del término en el conjunto de

documentos del corpus (el conjunto total de documentos). Se busca saber el número de documentos donde la palabra está presente.

$$df\,(t) = Ocurrencia\ de\ los\ términos\ N\ documentos$$

N = Conteo de corpus

Corpus = total de documentos

La última parte IDF, busca la información que está brindando una palabra, y con esto, se espera que las palabras con mayor frecuencia (y las *Stopwords*) tengan una puntuación más baja, ya que no generan información relevante.

$$idf(t) = \frac{N}{df}$$

En caso que las palabras no existan, el df = 0, por lo que se debe realizar una suavización en la fórmula de la siguiente manera:

$$idf(t) = log\ \left(\frac{N}{df + 1}\right)$$

Finalmente, TF-IDF se puede expresar de la siguiente manera:

$$tf - idf = tf(t, d) * log\ \left(\frac{N}{df + 1}\right)$$

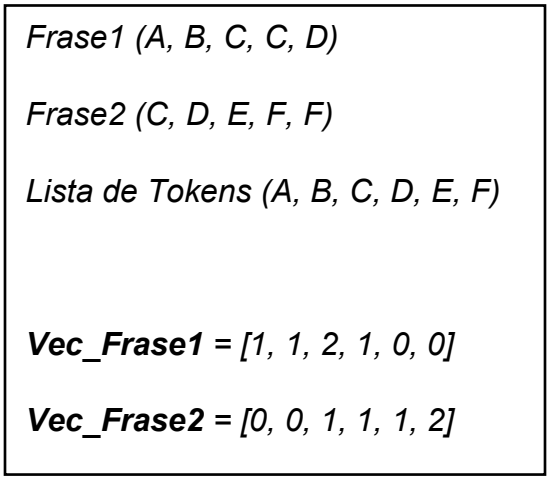
Otras técnicas utilizadas para la representación de documentos, es por medio de *Bag of words* (BoW, por sus siglas en inglés) es la forma más simple para representar por medio vectores el texto, realizando una simplificación el contenido del texto. Dentro de esta

representación se omite la gramática y el orden lógico de las palabras, pero sí es importante conocer la frecuencia con la que aparece cada palabra.

El algoritmo que se construye para la creación del BoW, empieza separando cada termino en tokens, para así crear un vector de palabras con todos los tokens que existen en el texto, y luego evaluar la frecuencia de cada token dentro de cada frase. En la figura 6 se puede observar un ejemplo de esta representación (M. Kanakaraj, R. Guddeti, 2015).

Figura 6: BoW

Adaptación: Introduction to Bag of Words. Great learnings (2022)



Otras técnicas que son usadas dentro de la representación de documentos son Word2Vec el cual es una técnica basada en vectores a partir de palabras que utiliza una red neuronal con el fin de aprender asociaciones de palabras a partir del corpus y así llegar a identificar palabras que sean sinónimas o entender las relaciones entre palabras y así lograr completar oraciones. Este se logra por medio de la similaridad del coseno, donde el ángulo entre los vectores sea cercano a 1. Para aprender las palabras en *embedding* se puede utilizar los modelos de *Continues Bag of words* (CBOW, por sus siglas en inglés) que aprende prediciendo las palabras actuales basándose en su contexto o modelos de *Continues Skip-Gram* que aprende prediciendo las palabras

circundantes a las que se les ha dado una palabra actual. Existe una modificación en Word2Vec, llamado Doc2Vec, donde la gran diferencia radica en agregar un vector de identificación al párrafo y así lograr obtener una representación numérica adicional del documento (Karani D. 2018).

Una extensión adicional de Word2Vec, es *Global Vectors* (Glove), el cual se enfoca en combinar las estadísticas globales de técnicas de factorización matricial como *Latent Semantic analysis* (LSA, por sus siglas en ingles), y con la forma de aprendizaje por medio del contexto como Word2Vec. Por lo que GloVe construye una matriz explícita de palabras-contexto o coocurrencia de palabras del corpus.

Finalmente, una técnica que no tiene en cuenta la construcción de Word2Vec, es *One hot encoding,* el cual asigna una dimensión a cada palabra del vocabulario así representando cada palabra con un vector binario exceptuando el índice de la palabra. Con este tipo de representación, todas las palabras tienen una misma distancia y similitud, por lo cual este método proporciona información si una palabra es diferente de la otra (Karani D. 2018).

2.1.6. Reducción de dimensionalidad

Debido al volumen de datos que se obtiene al realizar el procedimiento de representación de documentos, es necesario la reducción de la dimensionalidad y así lograr un modelo con mayor eficiencia. Para esto se puede utilizar las siguientes técnicas.

- *Principal Component Analysis* (PCA, por sus siglas en inglés): es una forma de realizar una reducción de la dimensión de los grandes conjuntos de datos intentando mantener la mayor parte de la información original. Para NLP se puede utilizar SPCA para análisis de componentes principales dispersos, ya que PCA tiene una limitación donde sus componentes principales son esencialmente combinaciones lineales de las variables originales. El objetivo principal de SPCA es forzar un número de cargas menos importantes que lleguen a cero, por lo que da lugar a vectores propios dispersos, para poder llegar a este objetivo el SPCA se basa en la matriz de covarianza (Drikvandi R, Lawal O. 2020).

- *CountVectorizer*: Es un método el cual convierte un texto en vectores que dan información sobre el número de frecuencia de la palabra. Dentro del proceso de *CountVectorizer*, se genera un vocabulario en caso que este no exista, por medio de un estimador que extra el vocabulario, luego de esto se seleccionan las palabras de mayor tamaño en frecuencia del tamaño del vocabulario, al final se genera un vector que cuenta con tres partes, la longitud, los índices y los valores del vector (Dhamija V, 2019).

- *HashingTF*: es un transformador que toma un conjunto de y los convierte en vectores de longitud fija, que pueden ser considerados como un BoW. Luego los términos son mapeados a índices por medio de la función Hash y se calcula la frecuencia de los términos respecto a los índices asignados. Este tipo de técnica tiene un desempeño optimo y un bajo costo, ya que evita el alto procesamiento al mapear termino con índice, como lo es el caso del *CountVectorizer*. Sin embargo, un problema que tiene este tipo de técnica es que se puede sufrir de *Hash collisions* donde varios términos pueden tener se convierten el mismo después del *Hashing* (Documentación Spark 2.1.0. 2021).

- *T-distribuite Stochastic Neighbor Embedding* (T-SNE, por sus siglas en inglés): Es un algoritmo creado en 2008 el cual se busca la distancia euclídea y la transforma en probabilidad condicional, encontrando así la probabilidad que un término X seleccionado para la observación vecina si las observaciones siguieran una distribución en densidad gaussiana con centro x_j. Al finalizar el algoritmo encuentra el conjunto con baja dimensionalidad asociado, pero debido a este tipo de procedimiento, este algoritmo consume demasiados recursos, siendo un algoritmo de bajo desempeño, ya que debe buscar la probabilidad en cada par de observaciones (Canavate, 2020).

- *Uniform Manifold Approximation and Projection for Dimension Reduction* (UMAP, por sus siglas en inglés): es un algoritmo complejo que puede ser usado para

reducción de dimensionalidad, este algoritmo es similar al T-SNE, pero a diferencia de este, no es de reducción linear, si no que se basa en asunciones que se toman de los datos. Este algoritmo se posiciona en una dimensión n en función de otra dimensión $n\text{-}1$ y así sucesivamente construyendo bloques que son llamados *k-simplex*. Cuando los puntos no se encuentran uniformemente distribuidos, se formarán los *k-simplex* por medio de *k-simplex* difusos, que indican que la distancia entre los puntos no será un valor sino una función (Leland M, Healy J, Melville J. 2018).

2.1.7. Técnicas análisis exploratorio de datos en documentos

EDA ayuda a los analistas a tener una perspectiva completa de los datos que van a ser utilizados para el estudio y así lograr identificar las técnicas de preprocesamiento y modelamiento que se deberían utilizar dentro del estudio, teniendo como objetivo el identificar cuáles son los posibles problemas que podría tener un conjunto de datos y adicionalmente, conocer el contenido que se tiene en el conjunto de datos.

EDA es uno de los primeros pasos después del preprocesamiento de datos que se realiza, con el fin de conocer las generalidades y calidad del conjunto de datos. Algunas de las técnicas más comunes y utilizadas en minería de datos para NLP son:

- Análisis Estadístico: dentro de este análisis se busca examinar tanto la importancia de las palabras, por medio de un análisis de frecuencia, donde se mide el texto por las veces que una palabra aparece dentro del documento. Y la longitud que tiene cada frase, dentro de un texto o número de palabras utilizadas para realizar gramaticalmente una oración; este tipo de análisis se utiliza con el fin de conocer qué tipo de técnicas de aprendizaje de máquinas puede ser utilizado, ya que una frase corta puede ser más difícil de leer o de extraer información (Lavin, Matthew J, 2020).

- Exploración de *N-GRAM*: son secuencias continuas de palabras o tokens en un documento, son llamados *bi-grams* la secuencia de dos palabras y *tri-grams* la

secuencia de tres palabras. por lo que en combinación con *CountVectorizer* se puede realizar un análisis frecuencial de los *N- GRAM* puede ayudar a tener un mejor contexto del uso de las palabras dentro del documento. Sin embargo, los *N-GRAM*, también son utilizados como un tipo de predicción por medio del modelo de Marcov (Jurfsky D, Martin h, 2023).

- Modelación de tópico: es un proceso no supervisado de aprendizaje de máquina, en el cual se busca identificar los temas que sobresalen en el documento. Dentro del análisis exploratorio, se puede realizar una modelación de tópico por medio de *Latent Dirichlet Allocation* (LDA, por sus siglas en inglés) una técnica muy usada para el modelamiento de tópicos para análisis exploratorios de texto, en el cual cada documento está representado por una distribución de temas y cada tema está representado por la distribución de palabras (Kapadia S, 2019). Este concepto se explicará en mayor detalle dentro de las técnicas no supervisadas de ML que se encuentran en el capítulo 2.1.8 de este proyecto.

- *Wordcloud*: es un tipo de gráfico utilizado con el fin de visualizar las palabras más frecuentes que se encuentran en un documento, este es generado de forma estadística como un resumen del texto completo. Las nubes de palabras son utilizadas para tener un punto de partida en el análisis profundo, ya que ayuda a identificar si un texto es relevante o tiene información específica. Sin embargo, uno de los grandes problemas que puede tener la nube de palabras es que proporciona un resumen estadístico de palabras aisladas basado en su frecuencia sin tener en cuenta la lingüística de las palabras y las relaciones entre ellas (F Heimerl, S Lohmann, S Lange, 2014).

- Exploración por medio de *POS Tagging* y *NER*: como se explicó con anterioridad, estos son métodos que asigna etiquetas a las palabras utilizadas en una frase. Si bien son métodos complejos, ya que una palabra puede ser utilizada de diferente manera en una frase y dependerá de su contexto, son una buena herramienta para utilizar en un exploratorio de datos para texto, ya que ayudará al analista a tener mayor entendimiento del conjunto de datos (Shahul E, 2022).

- Complejidad de texto: (T Tanprasert, D Kauchak, 2021) es un cálculo para formular un índice de legibilidad a un documento o texto. En la actualidad existen varias fórmulas para encontrar este puntaje. Dentro de este proyecto se utilizará "*FleschKincaid Grade Level"* el cual indica la complejidad que tiene la oración por medio del puntaje dado. Esta asignación de dificultad se puede ver en la tabla 4:

Tabla 4: Puntaje de complejidad de palabras

Adaptado de Flesch Reading Ease and the Flesch Kincaid Grade Level

Puntaje	Dificultad
90 -	Muy fácil de leer
80 – 89	Fácil de leer
70 – 79	Ligeramente Fácil
60 – 69	Estándar
50 – 59	Ligeramente Difícil
30 – 49	Difícil
0 -29	Confusa.

La fórmula utilizada para este puntaje de *FleshKincaid Grade Level* es:

$$206.835 - 1.015\left(\frac{total\ palabras}{total\ frase}\right) - 84.6\left(\frac{total\ silabas}{total\ palabras}\right)$$

2.1.8. Técnicas de ML en Texto

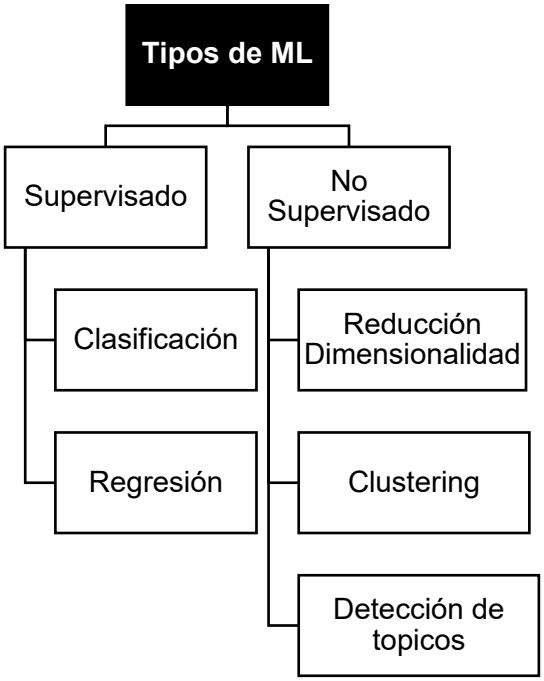
Existen diferentes técnicas tradicionales que son utilizadas para la clasificación, detección de tópicos, predicción de texto y muchas otras aplicaciones para NLP. Dentro de los métodos propuestos para clasificación que serán los tipos de modelos que se crearan en este proyecto, se propone en la investigación de Mikolov y Le (2014)

tecnologías como Doc2Vec el cual convierte palabras en vectores característicos utilizando dos modelo *Paragraph vector distributed memory* (PV-DM, por sus siglas en inglés) el cual consiste en una red neuronal que aprende la repetición de las palabras y *Paragraph Vector – Distribute Bag of Words* (PV-DBOW, por sus siglas en inglés) que ignora esas palabras de entrada pero busca la forma de obligar al modelo a predecir las palabras de salida y finalmente dentro de su se utiliza Máquinas de soporte vectorial (SVM, por sus siglas en inglés), el cual ha demostrado una efectividad del 89%.

Como segunda metodología se puede ver *Latent Semantic Analysis* (LSA, por sus siglas en inglés) que fue patentado en 1998 por varios científicos de datos, entre ellos Deerwester S, Dumais S, Furnas G, Harshman R el cual busca la relación entre documentos y los términos que contiene para así poder crear conceptos.

Dentro de las técnicas que se pueden utilizar en análisis de discurso basadas en aprendizaje de máquina, se dividen en técnicas supervisadas, que requieren de una etiqueta que ayude a su clasificación y técnicas no supervisadas las cuales no requieren de una etiqueta para aprender. Dichas técnicas se pueden visualizar en mejor detalle en la figura 7.

Figura 7: Tipos de ML



2.1.8.1. Técnicas supervisadas

Las técnicas supervisadas de ML se basan en una función que mapea la entrada a una salida, donde la función matemática ayuda a predicción de valores a partir de una etiqueta que debe existir en el data set por lo que para las técnicas supervisadas se requiere que los datos estén previamente categorizados. Algunas técnicas utilizadas para clasificación son:

- Logistic Regression: es una técnica de ML que puede ayudar a la clasificación de texto de forma binaria. Este tipo de técnicas utiliza una combinación ponderada de las características de entrada y pasando por una función sigmoidea hace que se transforme su clasificación en un número real de entrada entre 0 y 1. Existen varios tipos de Logistic Regression, como lo es la binomial para clasificar solamente en dos valores; Multinomial, para la clasificación entre 3 o más clases que no tengan un orden especifico; y finalmente, la Ordinal, para clasificación entre 3 o más clases que dependan de un orden.

Dentro de esta clasificación, es necesario la revisión de la función de costo, que es usada para calcular el error del modelo, dentro de este proyecto se calculará por medio de *Cross-Entropy* la cual aumenta a medida que la probabilidad predicha diverge de la etiqueta real (Arya N, 2022).

- *Naive Bayes*: este un clasificador de texto simple y no iterativo que supone, que cada característica del documento es independiente de cualquier otra característica que exista, por lo que, no existen conexiones entre características. Existen 4 tipos de clasificadores en *Naive Bayes* que son *Gaussian Naive Bayes, Multinomial Naive Bayes, Bernoulli Naive Bayes* y estimación semi supervisada de parámetros (Aggarwal A, Singh J, Gupta K. 2018).

- *K-Nearest Neighbour* (KNN, por sus siglas en inglés): es una técnica de aprendizaje de máquina donde el algoritmo busca clasificar los documentos en un espacio euclidiano con puntos de distancia entre dos documentos. K es el número de vecinos más cercano que se quiere encontrar, cuando K=1 se conoce como el vecino más próximo.

  Dentro de este algoritmo se debe tener en cuenta la selección correcta del número de K, con el fin de evitar un gran coste computacional o un sesgo al tomar un número muy pequeño de K, esto se puede realizar por medio del método del codo, el cual busca identificar el K optimo (Naviani A, 2018).

- *Decision Tree*: es una técnica de clasificación binaria que basa en reestructura un conjunto de datos de entrenamiento, donde el documento requerirá ser categorizado desde su nodo raíz hasta los nodos hijos (con la estructura de un árbol) cada nodo raíz puede tener únicamente dos hijos que a su vez puede tener otro nivel de clasificación.

  Un árbol de decisión tiende a tener sobre ajuste, debido a que construye una hipótesis de forma rápida que reduce el error del conjunto de entrenamiento e ignora el crecimiento del error en el conjunto prueba. Para evitar este sobreajuste se utilizan técnicas como *Pre-pruning* y *Pos-pruning* que detiene el crecimiento

del árbol antes de que este haga una clasificación perfecta del conjunto de entrenamiento o retroceder en la clasificación una vez el árbol ya fue totalmente creado (Aggarwal A, Singh J, Gupta K. 2018).

- Métodos basados en vectores: existen dos tipos de técnicas basadas en vectores, la primera es basada en el centroide que busca determinar el vector componente normal para la clasificación, luego este es establecido como vector centroide. El segundo método son las SMV que están basadas en la división de datos permitiendo el cruce de características en el modelo de clasificación y así obtener la etiqueta más adecuada para el documento. Una vez realizado este paso, se espera crear un gráfico multidimensional donde cada dimensión es específica a una única característica, esta división se realiza introduciendo un número de hiperplanos entre los vectores de soporte. (Wang, Xian-Jia, Guan-Tian, Ke-Xin, Hai-Bo, Zhen-Song. 2020)

2.1.8.2. Técnicas no supervisadas

Las técnicas de aprendizaje automático no supervisadas se refieren a aquellas que no requieren que un conjunto de datos esté previamente categorizado o clasificado. Dentro de estas técnicas se pueden destacar:

- LDA: es una técnica de aprendizaje automático, que busca reducir la dimensionalidad de un documento sin perder las características estadísticas. LDA es un modelo generativo, ya que puede crear un documento a partir de conjuntos de temas, y a su vez puede ser utilizado como una herramienta para extraer tópicos de un corpus, esto hace que LDA tenga dos suposiciones importantes, la primera que cada corpus tiene una mezcla de temas latentes y que estos a su vez son distribuciones sobre las palabras que encuentran dentro del documento; la segunda suposición se basa en que los documentos son generados por medio de una lista de palabras cuyo orden no es importante (Doumit S, Minai A. 2012).

- Hidden Markov Model (HMM, por sus siglas en inglés): es un modelo probabilístico que se utiliza para explicar las características probabilísticas de un proceso aleatorio, este tipo de explicación se deriva de un conjunto de distribuciones probabilísticas. Este tipo de técnicas en NLP se utiliza para *POS-Tagging* ya que busca la probabilidad de transición de una secuencia, por ejemplo, se busca la probabilidad que después de un sustantivo la siguiente palabra sea un verbo o un verbo modal. Otras aplicaciones en NLP son la clasificación, traducción, reconocimiento de palabras escritas a mano entre otras. (Verma Y. 2021)

- Información Mutua Puntual (PMI, por sus siglas en inglés): es una técnica de aprendizaje de máquina no supervisado que busca la probabilidad de coocurrencia de dos palabras, teniendo en cuenta el hecho de que podría ser causada por la frecuencia de las palabras individuales. Por lo tanto, el algoritmo calcula la probabilidad (logarítmica) de coocurrencia escalada por el producto de la probabilidad de ocurrencia individual. Se debe tener claro que la probabilidad de la palabra “a” y la probabilidad de la palabra “b” son independientes, pero si su probabilidad marginal es igual a 1 significa que las dos palabras no forman un concepto, y si al contrario su resultado se acerca a 0 es probable que las palabras expresan un concepto único.

- *K-means:* es un tipo de algoritmo que ayuda a la agrupación en un número K de clústeres, donde cada clúster estará definido por K Centroide que representará el punto central del clúster. Este tipo de algoritmo es iterativo ya que a medida que se mueva el centroide, sus puntos cercanos se definirán por la distancia que cada uno de ellos tenga con el punto central, por lo que en cada iteración si el centroide se mueve, se recalculará la distancia entre los puntos y el centroide. Para llegar a definir el número de clústeres (K) se puede utilizar diferentes tipos de técnicas, como el “método del codo”, o el *Silhouette Puntaje* (Sá L, 2018).

2.1.8.3. Métricas para la evaluación de clasificadores

Existen varias métricas que ayudan a conocer si un clasificador tiene un buen o mal desempeño, por esto, existen diferentes métricas que ayudarán a la evaluación del modelo, que el conjunto total de ellas, ayudará a la decisión de ajustar, aprobar o cambiar de modelo.

Dentro de este proyecto se tendrá en cuenta cuatro (4) tipos de métricas, que son *Acurracy, Precision, F1-puntaje* y *Recall*. Pero antes de conocer estas definiciones, es necesario conocer *Confusion Matrix* (Narkhede S, 2018) la cual es una matriz de $n * n$ (dependiendo del número de clases que se tengan en el modelo) que utiliza para describir el rendimiento del modelo, donde cada fila de la matriz representa la clase real, mientras que las columnas representan la clase predicha (ver figura 8).

Figura 8: Confusion Matrix

Adaptado de Narkhede S, 2018

		Clase Predicha	
		Positivos	**Negativos**
Clase Real	**Positivos**	*Verdaderos Positivos*	*Falsos Positivos*
	Negativos	*Falsos Negativos*	*Verdaderos Negativos*

- *Verdadero Positivo* (TP por sus siglas en inglés): mostrará el valor real y el predicho están bien clasificados.
- *Verdadero Negativo* (TN por sus siglas en inglés): mostrará el valor real y el predicho están bien clasificados.
- *Falso Positivo* (FP por sus siglas en inglés): el valor real es negativo pero su valor predicho es positivo, esto también es llamado error tipo II.
- *Falso Negativo* (FN por sus siglas en inglés): el valor real es positivo y su valor predicho es negativo, esto también es llamado error tipo I.

Una vez esta matriz es calculada, se podrán realizar los cálculos de las métricas para la evaluación que podrán ser apreciadas en la tabla 5 (Kulkarni H, 2021).

Tabla 5: Métricas de Evaluación

Adaptado Kulkarni H, 2021

Métrica	Descripción	Formula
Precision	El valor de la precisión se encuentra entre el 0 y 1, dentro de todos los positivos predichos, cual es el porcentaje realmente positivo.	$\frac{TP}{TP + FP}$
Recall	Del total de positivos, qué porcentaje son positivos predichos. Se debe tener en cuenta que al incrementar el *Recall,* hará que disminuya el *precision* y viceversa, a este efecto se le llama *Precision/Recall trade off.*	$\frac{TP}{TP + FN}$
F1-puntaje	La puntuación F1 es la media armónica de la precisión y la recuperación y es una medida mejor que la exactitud. La puntuación F1 se convierte en 1 sólo cuando tanto *precision* como la *recall* son 1. La puntuación F1 es alta sólo cuando *precision* y *recall* son altas.	$2 * \frac{Precision * Recall}{Precision + Recall}$
Acurracy	El *Acurracy* representa el número de instancias de datos clasificadas correctamente sobre el número total de instancias de datos.	$\frac{TN + TP}{TN + FP + TP + FN}$

2.1.8.4. Métodos de resampling

Es importante tener en cuenta que los conjuntos de datos pueden encontrarse desbalanceados, lo que indica que la distribución de las clases del conjunto de datos tiene una asimetría importante. Este tipo de desbalance puede causar grandes problemas en los modelos de clasificación, ya que debido al desbalance que existe hace que los algoritmos ignoren por completo la clase minoritaria. Para mejorar este tipo de desbalanceo se puede aplicar dos técnicas, en las cuales se muestrea aleatoriamente el conjunto de entrenamiento de datos, ya sea eliminado de la clase mayoritaria o duplicar ejemplos de la clase minoritaria (Brownlee J, 2020).

Random Oversampling es la estrategia la cual puede duplicar ejemplos de la clase minoritaria y se añaden al conjunto de datos de entrenamiento, este proceso se realiza seleccionando unas muestras del conjunto de entrenamiento original, para luego añadir el conjunto de datos original y así poder seleccionarlos nuevamente.

Esta técnica es eficaz si el algoritmo se ve afectado por una distribución sesgada y en los que múltiples ejemplos duplicados para una clase determinada pueden influir en el ajuste del modelo. Este tipo de estrategias ayudan a mejorar modelos de SMV, Árboles de decisión y modelos basados en redes neuronales, sin embargo, se debe tener en cuenta que el reajuste de estas clases puede aumentar el error generalizado, el cual mejora el desempeño del conjunto de datos de entrenamiento, pero empeora el desempeño del conjunto de datos de pruebas (Maldonado S, Lopéz J, Vairetti C. 2019)

La técnica de *Random Undersampling* se basa en eliminar ejemplos de datos de la clase mayoritaria y así eliminarlos del conjunto de entrenamiento. Esto ayude a reducir el número de ejemplos de la clase mayoritaria, y, como ventaja adicional, este tipo de técnicas se puede repetir dentro del conjunto de entrenamiento, hasta conseguir la distribución en la clase deseada. Sin embargo, al realizar esta eliminación, se pueden perder ejemplos que, en sus características, le agregan valor al clasificador. Dado que los ejemplos que toma el *Random Undersampling* son totalmente aleatorios, no hay

forma de excluir de esta estrategia ejemplos buenos o más ricos en información (Prusa J, Khoshgoftaar T, Dittman D, Napolitano A. 2015)

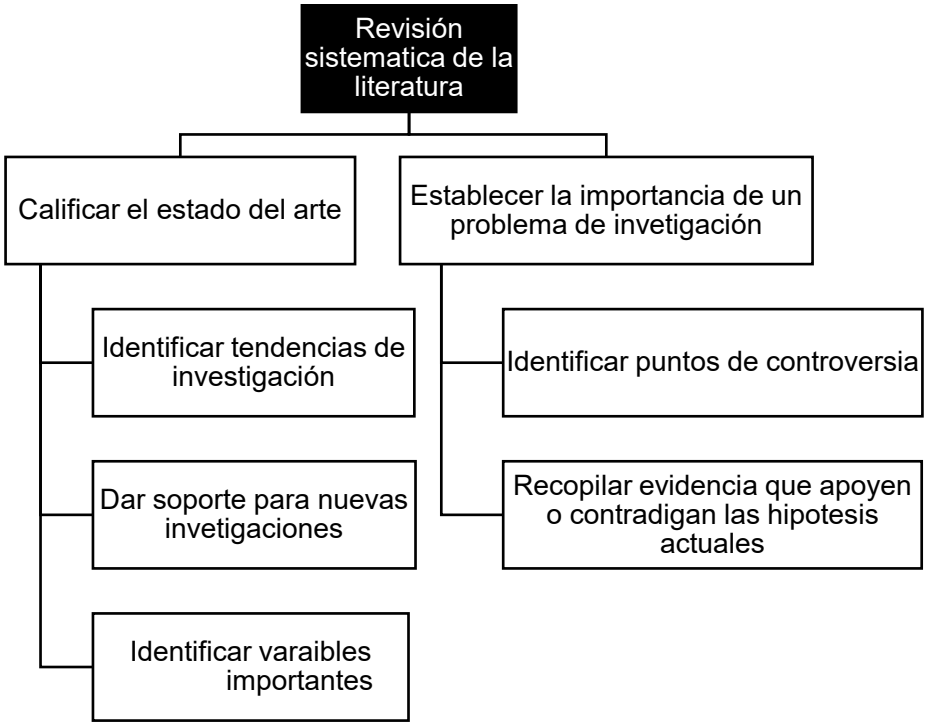
2.2. Estado del arte

Dentro de este trabajo de grado se busca implementar una metodología, que por medio de diferentes técnicas, se logre una mejor exploración y preparación de texto, por medio de técnicas tradicionales y de análisis de discurso, con el fin de poder crear una estructura en los proyectos que permitan su replicabilidad.

Para esto se realizará una revisión sistemática de la literatura que se evaluará de la siguiente manera (ver figura 9):

Figura 9: Contribuciones de SRL

Adaptado de Revisión sistemática de la literatura: Estado actual de la inteligencia artificial en las instituciones de educación superior (2020)



2.2.1. Metodologías para proyectos de análisis de discurso.

Dentro de estudios similares realizados en proyectos de maestría en universidades como Universidad técnica de Múnich, Universidad técnica de Viena, Universidad de Twente, entre otras, se muestra que los estudios de análisis del discurso, análisis de sentimientos, clasificación de tópicos y otros temas relevantes para NLP, utilizan metodologías de minería de datos basados en el ciclo de vida de datos como lo son KDD, CRISP-DM, SEMMA y TDSP siendo CRISP-DM una de las más populares. En general todas estas metodologías cuentan con 5 etapas que permiten el desarrollo de los proyectos en forma ordenada y ayudando a que se puedan replicar los procesos en diferentes proyectos que involucren el desarrollo de un modelo para el análisis de datos.

- Selección de datos: seleccionar y utilizar un conjunto de datos idóneo para el estudio, análisis y modelamiento.
- Preprocesamiento de datos: utilizar técnicas específicas para la limpieza y ajustar el conjunto de datos a utilizar.
- Exploración y visualización de datos: realizar una revisión del conjunto de datos con el fin de entender, conocer e identificar patrones.
- Construcción de modelos: crear los modelos que serán utilizados en el análisis de discurso.
- Evaluación de modelos: evaluación de los modelos construidos y revisión de puntos de mejora

2.2.2. Redes neuronales para NLP y análisis de discurso

Una de las primeras prácticas que se utilizaban para en ML para NLP eran obtener las variables del BoW y estas eran utilizadas para que el algoritmo aprendiera por medio de la frecuencia de las palabras, luego se utilizan diferentes técnicas de representaciones matriciales estáticas del texto, donde cada una de las palabras del documento está codificada por un vector (*embeddigns*). A finales del 2017, se empezó a incursionar en los modelos basados en redes neuronales recurrentes combinados con *embeddigns*, logrando definir el significado general de cada palabra y la posición que debería tener en cada oración.

Dentro del mismo periodo de tiempo que se empezó a investigar las redes neuronales, Google realizó un nuevo artículo "*Attention is all you need*" donde presentan la arquitectura *Transformer*, el cual cambió las capas recurrentes de las anteriores redes neuronales por capas de atención donde se buscaba "codificar cada palabra de una frase en función del resto de la secuencia, permitiendo así introducir el contexto en la representación matemática del texto" (Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez N, Kaiser L, Polosukhin I. 2017). Dentro la arquitectura de *Transformer* también presenta *embedding* posicionales que tiene diferentes aplicaciones en traducciones, clasificación de documentos, identificación de entidades, respuesta de preguntas, generación de texto, resumen y entre otros.

Actualmente se están desarrollando modelos *Reformer* y *Compressive Transformers* el cual se puede apreciar en el artículo *"Reformer the efficient transfer*" escrito por Kltaev N, Kalse L y Levskaya A (2020) que propone una nueva arquitectura donde se realizaría una sustitución de los productos puntos que realiza la arquitectura de *transformer* por un *hash* sostenible y así lograr reducir el uso de memoria. Finalmente, dentro del artículo "*Compressive transformer for long range sequence model*" escrito por Rae J, Potapenko A, Jayakumar S, Hiller C y Lillercrap T (2019), se busca un construir un transformador comprensivo por medio de un modelo de secuencia que tenga como objetivo la comprensión de memoria basado en el aprendizaje de secuencias de largo alcance.

2.2.3. Deep learning

Deep learning viene originalmente de *Artificial Neural Network* (ANN, por sus siglas en ingles) y es una de las tecnologías más recientes para la creación de análisis de datos, generación de conocimiento, predicción y muchos más objetivos que se pueden utilizar en diferentes aspectos de la ciencia de datos, la ventaja de utilizar aprendizaje profundo sobre los métodos tradicionales se basa en su rendimiento cuando se tiene un volumen de datos considerable. Dentro de estas técnicas se emplean algoritmos como *Multi-Layer Perceptron* (MLP, por sus siglas en ingles) el cual es un algoritmo de aprendizaje supervisado que tiene una arquitectura de aprendizaje profundo, constituida por una red totalmente conectada que consiste en una capa de entrada y una capa de salida para tomar decisiones o predicción sobre la señal de entrada y finalmente una o más capas ocultas que se encuentran dentro de la capa de entrada y salida. Cada capa está entrelazada y se conectan por medio de pesos determinados que se activan con diferentes funciones de transferencia (algunos ejemplos son *ReLU, Tanh, Sigmoid, Softmax*) que determinarán la salida de la red (Thorn J, 2019).

También se encuentran *Convolutional Neural Network* (CNN, por sus siglas en inglés) la cual consta de una arquitectura por medio de una capa de entrada, varias capas convolucionales, agrupación de capas que están totalmente conectadas y una capa de salida. Este tipo de algoritmo aprende directamente de los datos sin necesidad de

extracción manual de características. Cada una de las capas del CNN considera parámetros optimizados para obtener resultados significativos y reducir su complejidad, adicional a esto, CNN tiene una optimización para reducir el sobreajuste por medio de un abandono que se puede ver en redes típicas. Las redes neuronales convolucionales están diseñadas específicamente para tratar la variabilidad de las formas 2D (S. Albawi, T. A. Mohammed y S. Al-Zawi. 2017).

Otros algoritmos como *Long Short-Term Memory Recurrent* Neural Network (LSTM-RNN) que permite procesar una secuencia de entradas y retener su estado mientras procesa la siguiente secuencia de entradas, la arquitectura de esta red se basa en 3 puertas, una puerta de olvido que decide qué información se memoriza y qué información ya no es útil, una puerta de entrada que determina que información que entrara a la red y una puerta de salida que decide y controla las salidas. Estas tres puertas trabajan de forma cooperativa para controlar el flujo de información. Para poder mantener la información en memoria, esta red neuronal artificial cuenta con bucles de retroalimentación en la capa recurrente. Las redes LSTM son utilizadas para el análisis de datos secuenciales, clasificaciones, predicciones basadas en datos de series temporales, detección de anomalías y especialmente en NLP.

En aprendizaje profundo, se puede encontrar algoritmos como *Self-organization MAP* (SOM, Por sus siglas en inglés) que es un tipo de red neuronal artificial que tiene un enfoque de aprendizaje no supervisado, el cual tiene una arquitectura donde los nodos compiten por el derecho a responder a un subconjunto de datos de entrada, este algoritmo tiene un aprendizaje competitivo por medio de una función de vecindad para reservar las propiedades en el espacio de entrada. SOM es utilizado para agrupación de datos, y mapeo de conjuntos de datos de alta dimensionalidad ya que en la unidad de salida el nodo ganador es aquel cuyos pesos de enlace entrante se acerca al patrón de entrada (a través de distancia euclidiana), por tal razón, también es utilizado y conocido como algoritmos para reducción de dimensionalidad.

Dentro del aprendizaje profundo, existen algoritmos para la reducción de dimensionalidad tal como *Auto Encoder* (AE, por sus siglas en inglés). Este es un tipo de red neuronal artificial que tiene un enfoque no supervisado el cual busca aprender la

representación de un conjunto de datos e ignorar el ruido para ayudar a la reducción de dimensionalidad. Dentro de la arquitectura de esta red, se encuentra un codificador que comprime la entrada que es aquella que genera los datos y un decodificador reconstruye la entrada. Este tipo de red es utilizado para reducción de dimensionalidad, extracción de características, codificación eficiente y el modelado generativo.

Otro algoritmo con el mismo objetivo es *Restricted Boltzmann Machine* (RBM, por sus siglas en inglés), un tipo de red neuronal estocástica que es utilizado para reducción de dimensionalidad, clasificación, regresión, aprendizaje de características, modelamiento de temas y muchos más, el cual cuenta con dos tipos de nodos. El primer nodo es llamados visibles que se pueden medir y los segundos nodos son los ocultos que no pueden ser medidos. Este tipo de algoritmo tiene una base de aprendizaje no supervisado en el cual cada nodo está conectado con todos los demás lo que ayuda a identificar y comprender anomalías aprendiendo sobre el funcionamiento del sistema (Fischer, A., Igel, C. 2012).

Finalmente, *Deep Transfer Learning* (DTL, por sus siglas en inglés o Deep TL) es un tipo de red neuronal se basa en el aprendizaje por transferencia y es un método utilizado cuando hay insuficiencia de datos de entrenamiento o cuando no todos los datos cuentan con la etiqueta necesaria para realizar modelos de alta complejidad. Este algoritmo utiliza modelos pre entrenados aprendidos en un dominio de origen para tareas en el dominio de destino. Los tipos de aprendizaje que se pueden ver dentro de este algoritmo son aprendizaje de transferencia inductivo, donde la tarea de destino varía de la tarea de origen; aprendizaje por transferencia transductiva en la cual la tarea de origen y de destino son la misma pero los dominios son diferentes y aprendizaje de transferencia no supervisado donde la tarea de destino es diferente de la tarea de origen, pero están relacionadas entre sí. Este tipo de algoritmo tiene amplias aplicaciones, como NLP, clasificación de sentimientos, clasificación de imágenes, reconocimiento de habla, entre otras (Sarker H, 2021).

3. Definición de la Metodología

3.1. Introducción a una metodología para análisis de discurso

Para realizar el análisis de discurso en proyectos que requieren la respuesta de un grupo de participantes a preguntas estructuradas, se necesita la creación de una metodología que conste de una serie de etapas que permitan la exploración de diferentes configuraciones, librerías, modelos que al finalizar su ejecución generen conocimiento a los analistas.

La adaptación de las metodologías CRISP-DM y TSDP se utilizan para analizar las diferentes respuestas de los colombianos y chilenos, ya que, ambas consideran el ciclo de vida de los datos, diferentes técnicas de preparación de texto, aprendizaje de máquina supervisado y no supervisado, además de la utilización de técnicas tradicionales y modernas y así lograr la automatización y replicación en futuros proyectos.

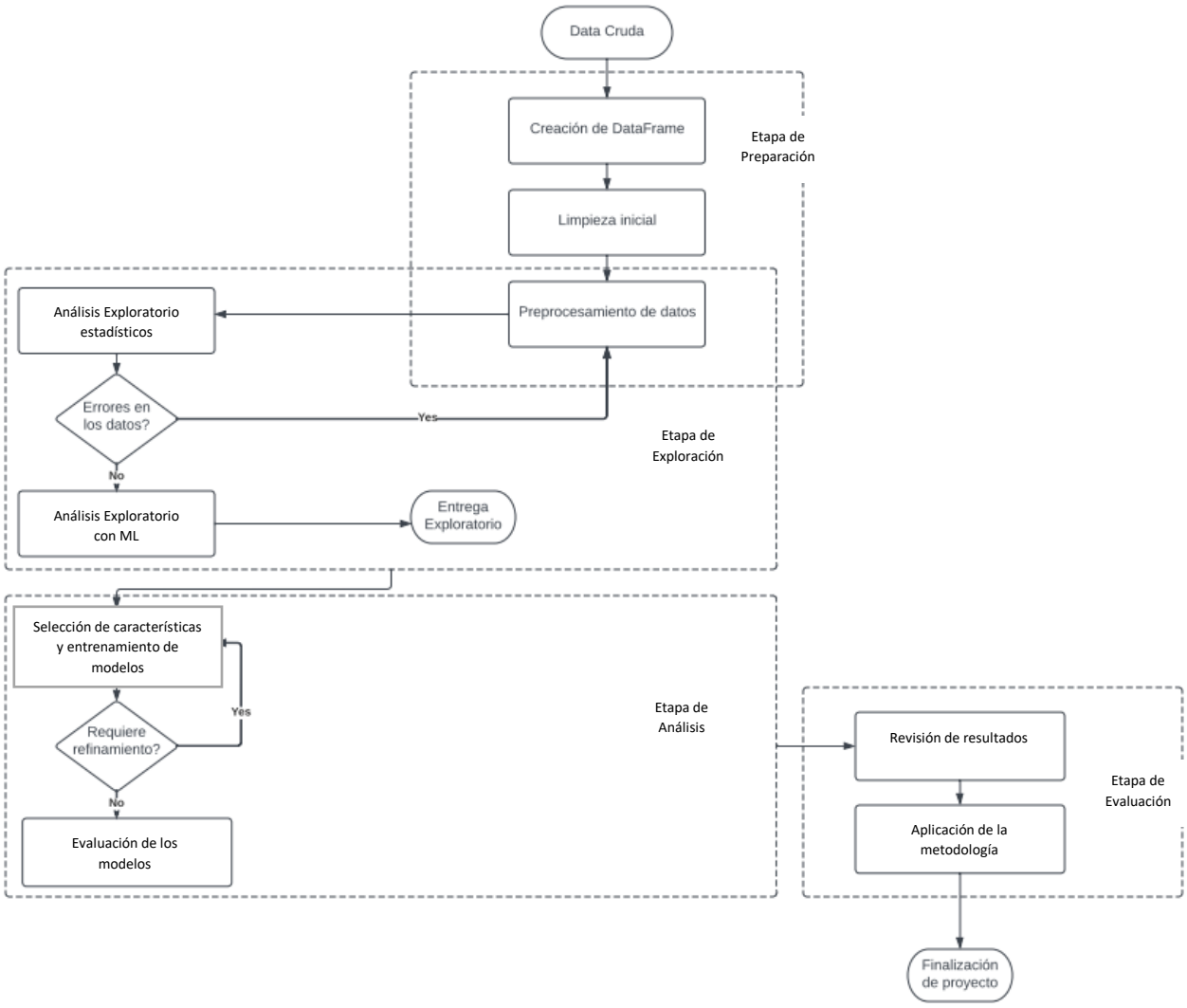
La metodología de este proyecto se basará en 5 etapas fundamentales, las cuales servirán de guía tanto para la preparación de la metodología, como para su aplicación en el proyecto *TQHC* (ver tabla 6 y figura 10).

Tabla 6: Metodología para análisis de discurso

Etapas	Descripción
Entendimiento del negocio	Dentro de esta primera etapa se busca conocer la razón y el alcance del proyecto con el fin de establecer cuáles serán las técnicas y herramientas que se utilizarán en las siguientes etapas.
Preparación de datos	Dentro de esta etapa, se busca estructurar los datos en un DataFrame estándar, para poder utilizar las técnicas de EDA, también, se espera realizar la limpieza del texto y el preprocesamiento de este, adicional a la representación de

	documentos que será utilizado para algunos modelos de exploración y de análisis.
Exploración de datos	En la exploración de datos se busca utilizar técnicas clásicas y de aprendizaje de máquina que empezará a mostrar y generar conocimiento sobre el texto que se está analizando. Dentro de esta etapa se realiza cualquier refinamiento sobre la limpieza y preprocesamiento ya que se puede empezar a evidenciar fácilmente si existen palabras que afecten los modelos, o si algunas palabras no tuvieron un preprocesamiento adecuado.
Creación de modelos	Dentro de esta etapa se busca realizar modelos tradicionales, y de aprendizaje de máquina supervisado y no supervisado con el cual se puedan extraer resultados para la toma de decisiones. Dentro de esta etapa, se busca la construcción de diferentes modelos, con el fin de comparar y poder seleccionar el modelo óptimo para el conjunto de datos seleccionado para este proyecto.
Ajuste y Evaluación	Durante la realización de esta etapa, se debe realizar refinamiento de los modelos si es necesario, ya sea con técnicas de reducción de dimensionalidad, balanceo de clases, modificación de hiperparámetros, entre otras formas de mejora de desempeño. Luego de cada ajuste que se le haga al modelo, se realizará una evaluación con las métricas de desempeño.

Figura 10: Metodología



3.2. Entendimiento del negocio y datos

Para realizar un entendimiento de negocio, se busca conocer el tipo de proyecto y el problema que se espera resolver. Por esto se puede revisar los títulos 1.1 y 1.2 de este proyecto, en los cuales plantea el problema y la justificación de la realización de este proyecto. Dentro del entendimiento de los datos, se debe tener en cuenta que existen tres tipos de datos, que se presentan en los proyectos TQHC y Chile, los cuales son:

- Datos conversacionales: son datos que se encuentran relacionados con los textos que fueron registrados como respuesta de los participantes durante los encuentros programados durante el proyecto. Dentro de este grupo, se busca crear la primera estructura en la cual los datos pasan por una serie de transformaciones para que pueda ser utilizados en los modelos.
    - Datos originales: son los datos que no contienen ninguna transformación.
    - Datos limpios: son datos que requieren una limpieza debido a que pueden contener errores de digitación, con anomalías por lo cual deba ser descartado, con símbolos especiales que no son necesarios, entre otras correcciones.
    - Datos analíticos: son los datos que ya están limpios, preprocesados con técnicas como *Stopwords, stemming, lemmatization* y entre otros, que podrán ser utilizados para la exploración y modelamiento de datos.

- Indicadores verbales: dentro del conjunto de datos, se puede observar que las respuestas dadas por los participantes, contiene un indicador verbal, el cual indica una clasificación que se la da a esta respuesta dependiendo de la emoción, valor, fines o consecuencias negativas que tengan la respuesta. Este indicador es asignado de forma manual por el moderador de la entrevista. Los indicadores verbales que se manejan dentro del proyecto *TQHC,* se encuentran en la tabla 7.

Tabla 7: Indicador Verbal

Etiqueta	Descripción Corta	Descripción
EM	Apela a emociones	Existe o expresa una emoción cuando responde las preguntas.
RE	Apena a Reglas	Justifica las respuestas con reglas de la sociedad, o constitucionales
BI	Apela a Fines	Básico Instrumental, el participante hace alusión a un fin o propósito.
CL	Clasificaciones	Justifica las respuestas donde clasifica o colando una etiqueta a la persona responsable.

CN	Consecuencias Negativas	Si no se realiza el cambio o mejora, entonces puede existir una emoción.
NUL	Ninguna de las anteriores	El argumento no coincide con ninguna categoría
VA	Valor	El participante hace alusión a un valor

- Anonimización de los datos: dentro del conjunto de datos se debe tener en cuenta la aceptación de términos y condiciones, adicional a la protección de datos, por tal motivo se realizó una anonimización de datos en la cual no se puede realizar una identificación directa de las personas que fueron entrevistadas.

Una vez se tiene un entendimiento de los datos y de su estructura, se requiere crear un realizar una estructuración de los datos, por medios de una tabla en memoria, la cual buscará estandarizar el esquema de los datos para que estos puedan ser procesados y así poder aplicar los métodos de preparación, análisis, modelamiento y evaluación de datos.

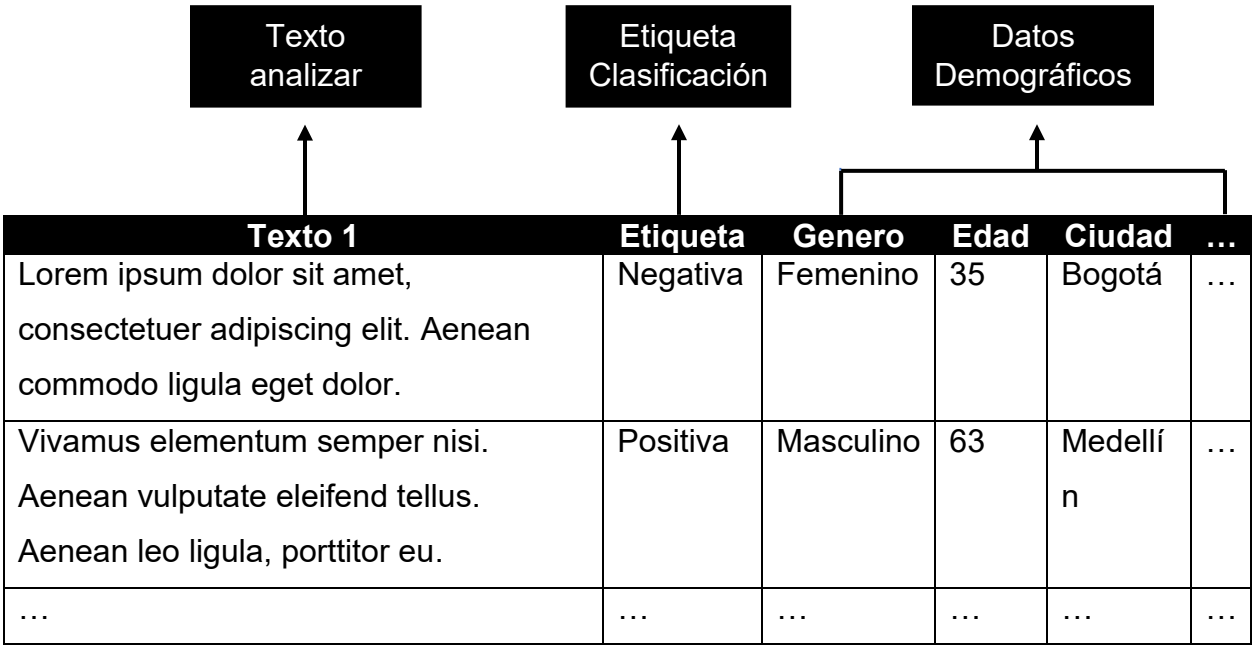
3.3. Preparación de datos

3.3.1. Estructuración de datos

Con el fin de poder replicar la metodología que se propone en este proyecto, y así centralizar los datos en una tabla de consulta, es necesario crear una estructura de dos dimensiones (líneas y columnas) en el cual, las columnas representaran los atributos de los datos, y en las líneas los valores en cada uno de estos atributos, teniendo en cuenta que cada línea representará un conjunto de datos de un participante especifico.

Por lo cual se busca crear una tabla en memoria (DataFrame) por medio de la librería de pandas de Python, en la cual se espera tener el texto que se requiere analizar, la etiqueta de clasificación, y datos demográficos que ayudara al exploratorio de datos. En la figura 11, se observa la estructura de datos que será utilizada para este proyecto.

Figura 11: DataFrame de entrada



Texto 1	Etiqueta	Genero	Edad	Ciudad	…
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor.	Negativa	Femenino	35	Bogotá	…
Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu.	Positiva	Masculino	63	Medellín	…
…	…	…	…	…	…

3.3.2. Limpieza de datos

Con el fin de eliminar ruido y errores que tenga los diferentes textos, se debe realizar una limpieza inicial que consta de los siguientes pasos:

- Eliminación de caracteres especiales, puntuación y errores de tipografía: es importante eliminar los errores que puedan afectar el estudio, con el fin de tener un buen conjunto de datos, esta eliminación de caracteres especiales, puntuación y errores de tipografía se realizará por medio de expresiones regulares, que son cadenas de texto que se pueden usar y que tienen una sintaxis especial, que permite hacer coincidir patrones, (que son series de letras y símbolos que corresponden a un texto) y así lograr encontrar cadenas de texto específicas, eliminar o reemplazar caracteres que no se requieran. Es importante aclarar, que para la lengua castellana es necesario mantener la acentuación de las palabras, ya que esto puede cambiar el significado de las palabras.

- Convertir texto en minúsculas: debido a que algunas librerías y metodologías son sensibles a las mayúsculas y minúsculas, se realizará un procedimiento para que todas las palabras se encuentren en minúscula y así poder implementar un preprocesamiento de datos idóneo, este paso puede ser logrado también con expresiones regulares.

- Eliminar palabras que contengan combinación de letras y números: con Python se puede identificar fácilmente si una palabra contiene caracteres con o sin números, por medio del método *isalpha(),* que por medio de una respuesta booleana indicará si una cadena de texto tiene un valor atípico a revisar para que posteriormente sea eliminado. Se debe tener en cuenta que este tipo de revisión debe ser más exhaustiva si las respuestas son científicas o académicas, ya que se marcara como FALSE o palabras a eliminar palabras como “word2vec”, “U.K” y diferentes ecuaciones.

- Eliminar espacios innecesarios: por error humano, se puede encontrar que existen espacios que no son necesarios, y estos podrían afectar el estudio, ya que aumentaría el número de tokens.

- Eliminar líneas que tengan valores nulos y que no agreguen valor al estudio: pueden existen participaciones que no tienen alguna respuesta, o por error humano se ha duplicado su entrada, también, pueden existir respuestas que no estén completas y estas se pueden detectar por su longitud.

3.3.3. Pre-procesamiento de datos

Una vez se tiene el texto de una forma limpia y gramaticalmente correcta, se requiere realizar un proceso en el cual se pueda estandarizar el texto a analizar, en un formato amigable para los algoritmos, el cual se espera que los modelos traten las palabras similares o la misma palabra, pero con diferente conjugación como si esta fuera la misma.

Debido a que las técnicas de preprocesamiento dependen de la aplicación en la que se trabajará, se realizará una aplicación de las técnicas más comunes, y se espera guardar cada conjunto de datos preprocesado en un archivo diferente, que será llamado para cada uno de los modelos propuestos.

- Tokenización: para realizar este proceso en el cual se divide una cadena de texto en tokens, se utilizará la librería *spaCy*, la cual se deberá cargar un modelo de español previamente entrenado, luego, se llamará este modelo con cada una de las entradas de los participantes y así lograr tener los tokens de cada una de las respuestas.

- Eliminar *Stopwords*: dentro de este proceso se busca eliminar esas palabras que en el español aparecen frecuentemente y que no generan información, como los son artículos, pronombres entre otros que han sido explicados en el marco teórico. Dentro de la librería de *spaCy*, se puede utilizar la librería *lang.es.stop_words* el cual contiene una lista predefinida de las Stopwords que se podrán eliminar, adicional, se puede utilizar una lista extendida en caso de que se requiera eliminar otra palabra que no genere información y no se encuentre dentro de la lista.

- *Lemmatization*: para realizar este proceso de convertir una palabra en su forma base (*lemma*), también se utilizará la librería de Python *spaCy*, el cual tiene un modelo predeterminado para el idioma español que como resultado mostrará las formas bases de las palabras convertidas en tokens.

- *Stemming:* con el fin de poder agrupar más palabras y así eliminar dimensionalidad al conjunto de datos para los modelos de clasificación se realizará un preproceso de stemming sobre el conjunto de datos ya lematizado y sin Stopwords. Se realizará una prueba con el fin de identificar el mejor stemming para el idioma español, por medio de los algoritmos de *Porter, Snowball, word tokenizer.*

3.3.4. Representación de documentos y características

Contando con un conjunto de datos, limpio y con un preprocesamiento, se realizará una representación de documentos, con el fin de poder alimentar los modelos que se construirán dentro de este proyecto, adicionalmente este tipo de representación ayudará a conocer la ocurrencia de las palabras dentro de un corpus. Dentro de este capítulo se utilizarán las tres (3) métodos explicados dentro del marco teórico:

- BoW: donde no se tiene en cuenta la secuencia de las palabras, por lo cual trata cada una de las palabras independiente. Dentro de este proyecto se utilizará dos formas la creación de BoW, por medio de *CountVectorizer*, el resultado esperado de BoW es una matriz con vectores de gran dimensionalidad.

- TF-IDF: dentro de esta fase del proyecto, también se busca conocer cuáles son esas palabras que son representativas, por lo construirá por medio de sklearn *TFidfVectorizer* una matriz que se basa en la idea que el peso de una palabra en un documento o texto debería ser proporcional a su frecuencia y una función inversa del número de documentos en los que aparece. El método de sklearn es muy parecido al utilizado en BoW (*CountVectorizer*), pero su mayor diferencia es que para realizar este cálculo, se utiliza la formula explicada en el marco teórico. Utilizando la matriz generada con TF-IDF también se puede calcular la similaridad entre las respuestas de los ciudadanos, por medio de *similaridad del coseno* el cual dará un valor entre 0 y 1, donde 0 indicara que no hay similitud y 1 que las respuestas son idénticas. Para implementar esta similitud, se utilizará el paquete de sklearn.

- *Work2Vec:* usando una red neuronal con pocas capas, *Work2Vec* intenta crear relación entre palabras y así disminuir la dimensionalidad de los vectores, por lo que w2v intenta resolver algunos problemas que existen en BoW como lo es la alta dimensionalidad y asumir la independencia de las palabras. Para la creación de este modelo, se utilizará la librería gensim.

Cabe destacar que Work2Vec es muy usado para la predicción de palabras, sin embargo, dentro de este proyecto se utilizará también con el fin de tener mayor conocimiento del conjunto de datos y utilizarse en posibles modelos futuros.

3.4. Análisis exploratorio de datos (EDA)

EDA es la etapa más importante para el desarrollo de cualquier proyecto de datos, incluyendo aquellos que constan de la creación de modelos de aprendizaje de máquina. Por esto, para el procesamiento de lenguaje natural se buscará en esta sección la implementación de las técnicas más utilizadas para entender los diferentes textos a los que se podrían enfrentar los analistas de datos.

- Exploración de variables demográficas: el primer paso del EDA es conocer la distribución de las variables del conjunto de datos. Este tipo de análisis ayudaran a que el analista pueda identificar otro tipo de estudios que se puedan realizar, algunos ejemplos de esto son, ¿cuáles son las palabras más frecuentes utilizadas por un grupo de personas que viven en el campo a aquellas que viven en la ciudad? ¿Puede existir un cambio en el análisis de sentimientos por genero? Para poder responder este tipo de preguntas, es necesario conocer la parte más básica del Conjunto de datos.

- Análisis estadístico de texto: dentro de este primer acercamiento a los datos, se buscará revisar la frecuencia en la que aparece las palabras y la longitud de las frases, esto con el fin de conocer las características del texto que se analizará, también el conocer la longitud de los tokens. Para este tipo de análisis se utilizará *Python* y se realizará una visualización por medio de histogramas (data continua) y gráficos de barras (datos categóricos).

- Exploración de *N-GRAM*: conociendo los *N-GRAM* se puede tener mayor entendimiento del contexto en que las palabras con mayor frecuencia se están utilizando. Para poder realizar esta exploración se utilizara la librería *nltk.util* de

Python, también se utilizará *CountVectorizer* para construir una representación del vocabulario y se realizará una visualización por gráfico de barras de *bi-grams* y *tri-grams*.

- Modelamiento exploratorio para detección de tópicos LDA: como primera técnica de aprendizaje de máquina no supervisada se utilizará el modelo LDA para la extracción de tópicos principales en todas las respuestas dadas por los ciudadanos. LDA es un modelo probabilístico generativo que asume que cada tema es una mezcla sobre un conjunto subyacente de palabras, y que cada documento es una mezcla sobre un conjunto de probabilidades de tema. Dentro de la creación de este modelo se realizará por medio de la librería *gensim* para así poder realizar un modelado de temas, el cual está representado por la distribución de palabras, para luego profundizar en la exploración de los datos de cada tema o grupo de temas y *pyLDAvis* para la visualización en un gráfico interactivo.
  Dentro de este grafico interactivo, se puede realizar la modificación de λ, el cual se debe interpretar como la puntuación de relevancia de la palabra clave ya que su fórmula es:

$$\frac{palabra}{tema\ k}$$

  K = El tema seleccionado

  Adicional se realizará una prueba de coherencia con el fin de conocer el número de tópicos que sería el óptimo para este modelo y así tener un mejor resultado.

- Frecuencia de palabras con *Wordcloud*: para la creación de una nube de palabras en *Python* se requiere que dentro de su preprocesamiento se construya un corpus de palabras, y así poder generar un gráfico el cual mostrará la frecuencia e importancia de una palabra por un color y tamaño de fuente más notorio que otras palabras con menor importancia. Se construirá *Wordcloud* con el fin de identificar

aquellos términos que tienen una gran frecuencia, después de las diferentes modificaciones que se han realizado.

- Análisis de sentimientos: dentro del análisis exploratorio de texto más comunes se puede destacar el análisis de sentimientos, con el fin de determinar si la respuesta, opiniones o comentarios son negativos, neutrales o positivos, para esto se utilizará dos paquetes de Python y el servicio de *Comprehend* de *AWS*. El primer paquete que se utilizará es *Textblob* que es una técnica que se puede construir por medio de la librería *NLTK,* el cual funciona por medio de dos propiedades, la primera un número flotante que se encontrará en un rango de -1 y 1 donde 1 es positivo y -1 negativo, a esta propiedad se le llama polaridad; la segunda propiedad es la subjetividad en la que los ciudadanos están formando sus opiniones o respuestas, y esta está representada por un valor de punto flotante que se encuentra en el rango de 0 a 1.

  El segundo paquete es *Balance aware dictionary and sentimental reasoner* (*Vader,* por sus siglas en inglés*)* es una librería preconstruida de analizar sentimientos basados en reglas de léxico que está especializada en la detección de sentimientos negativos. Con Vader se espera analizar los sentimientos desde un diccionario que contiene las probabilidades de que un texto sea positivo, negativo o neutral.

  Y finalmente se construirá un modelo por medio del servicio *detect_sentiment* de *AWS Comprehend*. Este servicio realiza análisis de sentimientos en tiempo real en los primero 500 caracteres de cada frase, luego de esto ignora el resto de los caracteres dados en los datos de entrada. Para el uso de este tipo de servicios es necesario establecer en los parámetros el lenguaje a utilizar, que para los fines de este proyecto se utilizará *'es' (*abreviación para indicar que es español). Este tipo de servicios da como resultado una inferencia de sentimiento predominante que puede ser Positivo, Negativo, Neutral o Mixto, donde cada sentimiento tiene un puntaje, y seleccionando el sentimiento con mayor puntaje.

- POS Tagging: con el fin de conocer los verbos, adjetivos, sustantivos más utilizados por los ciudadanos al dar sus respuestas se empleará un análisis de POS Tagging, sin embargo, como se explicó en el marco teórico, se debe prestar atención al análisis realizado al resultado, ya que POS Tagging no tiene en cuenta el contexto en el cual se utiliza la palabra. Para este análisis se realizará un análisis con *spaCy* y el servicio de *AWS Comprehend batch_detect_syntax.*
  El servicio de AWS es llamado por medio de una API, el cual busca la sintaxis de las palabras que conforman la oración, se debe tener en cuenta que este tipo de servicio está disponible para frases con un número menor o igual a 25 palabras, el resultado dado es un JSON que por cada palabra se asigna la etiqueta.

- NER: con el fin de conocer las personas, organizaciones o entidades de las cuales se habla dentro de las respuestas de los ciudadanos, se empleará NER, el cual clasifica el texto dentro de tipos de entidad. Debido a que el proyecto se encuentra en español, se realizará NER con las librerías de *spaCy, nltk,* y adicional se realizará un análisis con *AWS* por medio de la API *batch_detect_entities,* el cual detecta entidades como persona, localización, organización, eventos, cantidades y otros, para su utilización se requiere ingresar el lenguaje y la oración a analizar la cual no puede tener más de 25 palabras.

- Exploración de la complejidad del texto: para el análisis exploratorio es muy importante conocer que tan difícil de leer es la respuesta dada por el ciudadano, para esto se buscará dar un número de legibilidad del texto. Como se explicó en el marco teórico, se tomará el índice de "*FleschKincaid Grade Level" y* así identificar el nivel de dificultad que tienen las respuestas al ser leídas. Con el fin de conocer cómo se afecta este indicador, se realizará un análisis con el archivo inicial y contra el archivo preprocesado.

- Árbol de palabras: como ultimo método para la exploración de conjunto de datos, se realizará un árbol de palabras, que es una visualización que se deriva de *Wordcloud*, con la gran diferencia que este método busca representar por medio

de un gráfico las palabras con mayor frecuencia después de una palabra seleccionada como nodo (sufijos y prefijos). Dentro de este proyecto se busca encontrar aquellas palabras por medio del paquete *WordTree* de *Python*.

3.5. Modelos de análisis de discurso

Dentro de los modelos para el análisis de discurso, se busca desarrollar un clasificador para el indicador verbal, por medio de aprendizaje de máquina supervisado, que, como requisito fundamental para este tipo de entrenamiento, se necesita tener una etiqueta dentro del conjunto de datos, la cual será la que el modelo aprenderá. Para efectos de este proyecto, se utilizará el indicador verbal como etiqueta de clasificación.

Los modelos de clasificación se construirán basándose en BoW y TF-IDF, los cuales tendrán varias comparaciones entre conjunto de datos con *lemmatization,* y *stemming,* al igual que la identificación de palabras de poca frecuencia y de poca relevancia.

Con el fin de conocer cuál es el mejor clasificador, se crearán varios modelos de clasificadores, como lo son *Logistic Regression, KNN, Decision Tree, Random Forest, Naives Bayes y Multi-Layer Perceptrom Classifier* (MPL por sus siglas en ingles), que servirá de línea base, con el fin de conocer el desempeño de los modelos y así en pasos posteriores, se buscará realizar una mejora en cada uno y llegar a seleccionar el más adecuado para el proyecto.

Para la creación de la competencia o línea base, se construirá el algoritmo con los siguientes pasos:

- Preprocesamiento de datos
- Determinar la etiqueta
- Separar el conjunto de datos en datos de entrenamiento y de prueba
- Extraer las características del texto
- Ajuste del modelo al conjunto de datos de entrenamiento
- Creación de la predicción *(y_pred)*

- Evaluación del modelo, definiendo *y_test* y *y_pred*, por medio de la matriz de confusión, Acurracy, Presicion, recall y F1-Puntaje
- Ajustes del modelo

Debido a que existe un desbalanceo de clases en el conjunto de datos de entrenamiento, en el cual algunas etiquetas tienen muy pocos datos, causando que el modelo ignore por completo la clase minoritaria, por esto, se realizará un ajuste de las diferentes etiquetas donde se aplicarán técnicas de *resampling* realiza una nueva versión del conjunto de datos de entrenamiento en el cual se selecciona unos ejemplos de datos creando así una nueva distribución de las clases.

Se debe tener en cuenta que este tipo de técnicas no suponen nada sobre los datos y no se utiliza ninguna heurística, por lo que son técnicas rápidas de aplicar y ejecutar, adicionalmente pueden ser utilizada para clasificación binaria o de multiclase. También cabe destacar que este tipo de técnicas solo se aplicaran en el conjunto de entrenamiento, ya que la intención que se tiene es influir en el ajuste del modelo.

3.6. Ajuste y evaluación de modelos

Una vez se tenga un conjunto de datos limpio y preprocesado, un conjunto de datos de entrenamiento balanceado y las primeras ejecuciones de los diferentes clasificadores, se realizará una selección de los mejores tres (3) para así realizar los diferentes ajustes que se podrán tener en cuenta para cada modelo.

En general, existen tres pasos para mejorar los desempeños de cualquier clasificador:

1. Realizar un balance de clases: en este caso, se revisará si los conjuntos de entrenamiento requieren volver a pasar por un proceso de *Under/Over Sampling.* También se podrán eliminar las clases que no sean significativas para el modelo.

2. Reducción de la dimensionalidad: se realizará una reducción en la dimensionalidad de los modelos creado con la BoW y TF-IDF con el fin de detectar

el desempeño de los clasificadores al eliminar tokens que no sean esenciales. Se buscará el percentil más adecuado a mantener con el fin de no perder información importante para los clasificadores.

3. Sintonización de hiper parámetros: dentro de este paso, se realizará modificaciones a los hiperparámetros de cada uno de los clasificadores, buscando así la mejora de su desempeño

Para la evaluación del desempeño de los modelos se construirá una matriz de confusión, para luego evaluar con las métricas de *precision, recall, f1-puntaje* y *acurracy.* Esto ayudará a seleccionar el mejor clasificador para la automatización de la clasificación del indicador verbal.

4. Aplicación de la metodología para el análisis de discurso

4.1. Preparación de datos

4.1.1. Estructuración de datos para analítica

Para estructuración de datos, se deben construir tablas en memoria, que al utilizar el lenguaje Python en este proyecto, se utilizará DataFrames, en donde se debe comenzar concatenando diferentes columnas que generen un único texto, con el fin de realizar análisis de discurso. Durante el proyecto TQHC se puede apreciar que existen varias columnas con texto para analizar, ya que, este tipo de entrevistas tuvieron 3 ciclos de participación, por lo que se concatenarán los textos de la siguiente manera:

- La primera columna registra la respuesta dada por el participante en el Ciclo 1 sobre lo que quiere cambiar, mejorar o mantener en Colombia, y la segunda columna donde se da la razón de su respuesta.
- Se mantendrá solamente la razón por la cual el participante considera que existe un tema más importante de entre los mencionados por él mismo y los demás participantes en el Ciclo 1.
- Solo se mantendrá las acciones que se proponen para el tema que se priorizó en el Ciclo 3 del encuentro al cual asistió el participante.

Luego de la concatenación de columnas, se revisa cuales columnas no son necesarias para el estudio y así proceder a eliminarlas y disminuir el volumen de datos. Para realizar esta eliminación se hace una revisión de:

- Columnas que tienen más del 80% de valores nulos
- Columnas que no agregan valor al estudio, algunos ejemplos de estos son nombre de carpetas, id de entrevistador, etc.
- Columnas que tenga un valor fijo.
- Columnas que tenga un mismo valor en más del 80% del conjunto de datos. Sin embargo, para el análisis de outliers se revisa la información de estas columnas,

ya que existen valores como “el participante ha sido víctima de conflicto armado”, “zona a la que pertenece”, “grupo etario” que puede ser importante para los analistas revisar por separado.

Finalmente, se selecciona la etiqueta con la cual se clasificará y analizará las respuestas, en este caso se tomará el indicador verbal del ciclo 1.

Después de los pasos anteriores, se generó un DataFrame, el cual contiene la siguiente estructura (ver tabla 8):

Tabla 8: DataFrame Inicial

Columna	Descripción
Texto_1	Respuestas y razón de esta respuesta de los participantes en el primer ciclo de preguntas.
Texto_2	Tema que se considera más importante y razón por la cual es considerado importante para el participante.
Texto_3	Tema a priorizar y razón por la cual es considerado importante para el participante
Etiqueta_Texto_1	Indicador verbal, el cual si el participante apela a emociones (EM), reglas (RE), fines (BI), clasificaciones (CL), consecuencias negativas (CN) o ninguna de los anteriores (NUL). Este indicador verbal es asignado para el texto_1 y puede tomar uno o dos valores.
Etiqueta_Texto_2	Indicador verbal, el cual si el participante apela a emociones (EM), reglas (RE), fines (BI), clasificaciones (CL), consecuencias negativas (CN) o ninguna de los anteriores (NUL). Este indicador verbal es asignado para el texto_2 y puede tomar uno o dos valores.

Cambiar_mejorar_mantener	Pregunta que el participante decidió contestar en el texto_1. Toma los siguientes valores: 1=Cambiar, 2=Mejorar, 3=Mantener, 0=Sin respuesta.
Responsable	Actor o agente (persona, grupo u organización) que el participante considera tiene el poder o la capacidad para hacer lo propuesto en el texto_3
Genero	Género del participante. Toma los siguientes valores: Masculino; Femenino; No binario; No sabe/No responde.
Grupo _etario	Grupo etario del participante. Toma los siguientes valores: De 8 a 13 años; De 14 a 25 años; De 26 a 36 años; De 37 a 47 años; De 48 a 58 años; Más de 58 años.
Departamento	Indica el departamento al que pertenece el participante.
Escolaridad	Nivel de escolaridad del participante. Toma los siguientes valores: Ninguno; No sabe/No responde; Primaria incompleta; Primaria completa; Bachillerato incompleto; Bachillerato completo; Tecnológico; Profesional; Posgrado.
Municipio	Indica el municipio al que pertenece el participante
Región	Región de Colombia donde reside el participante. Toma los siguientes valores: Centro; Antioquia-Eje cafetero; Caribe; Pacífico; Llanos orientales; Amazonía.

Como se indicó con anterioridad, se busca generar un segundo DataFrame con las columnas adicionales para estudios de datos atípicos (*outliers*), cual contiene las columnas de la tabla anterior y adicional las columnas de la tabla 9.

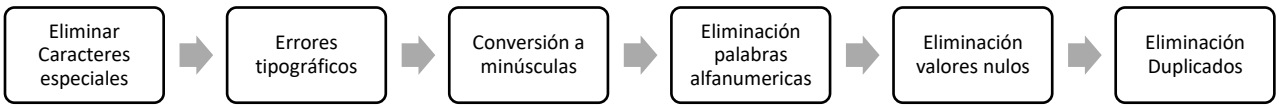
Tabla 9: DataFrame inicial con columnas outliers

Columna	Descripción
Zona	Donde reside el participante. Toma los siguientes valores: Urbana; Rural.
Victima_conflicto	Si el participante ha sido víctima del conflicto armado. Toma los siguientes valores: No; Sí; No sabe/No responde
Grupo_étnico	Grupo étnico al que dice pertenecer el participante. Toma los siguientes valores: Ninguno; Afros-raizales-palenqueros; Pueblos y comunidades indígenas; Pueblo Rom o Gitano

4.1.2. Limpieza de datos

El siguiente diagrama de flujo presentado en la figura 12, muestra la aplicación para la limpieza de datos:

Figura 12: Diagrama de flujo para limpieza de datos



En particular, se comenzará con la limpieza de caracteres especiales. Esto se puede hacer por medio de expresiones regulares, que por medio de funciones anónimas se especificando el conjunto de caracteres que se requiere eliminar. En el caso de esta tesis se eliminan los caracteres [,\.!?&]

Como segundo paso, se busca identificar errores tipográficos que puedan afectar el estudio, este tipo de errores se pueden encontrar al familiarizarse con el conjunto de datos. Nuevamente se utiliza las expresiones regulares con el fin de modificar los errores

tipográficos más comunes como lo son la acentuación, abreviación no convencional, palabras con errores de ortografía y digitación (ver tabla 10).

Tabla 10: Ejemplos errores de tipografía

```
df['texto_1'].map(lambda x: re.sub('publica', 'pública', x))
df['texto_1'].map(lambda x: re.sub('ddhh', 'derechos humanos', x))
df['texto_1'].map(lambda x: re.sub('campesisno', 'campesino', x))
```

Por conveniencia, este tipo de limpieza, se repetirá durante todo el proyecto, debido a que nuevas palabras pueden aparecer al ejecutar EDA o la creación de los modelos. Acto seguido, se realiza una conversión del texto a minúscula, esto es posible por medio del método *lower* del lenguaje *Python.* Este paso ayudará a que las palabras se logren agrupar para el paso de *tokenización* y representación de características.

Para este proyecto se eliminarán las palabras que tengan letras y números, pues se presume que corresponden a textos incorrectamente digitados, ya que las respuestas de los ciudadanos no son de connotación científica o académica. Por medio de la función *isalpha()* de *Python* se puede realizar esta revisión, por lo que se crea una función, donde se transformará el texto en tokens, para después identificar si cada token tiene o no algún carácter que no sea una letra o vocal. Finalmente, solo se preservan en el texto aquellas palabras que cumplan esta condición.

Luego, se realiza una revisión de valores nulos, con el fin de eliminar aquellas líneas que no tengan datos en la columna texto, ni en la etiqueta, ya que ambos campos son fundamentales para los modelos que se realizarán dentro de este proyecto.

Como último paso, se realiza una revisión de duplicados teniendo en cuenta el subconjunto de datos de texto_1 y la etiqueta para esta. El propósito de dicha identificación es eliminar redundancia de información para así evitar errores en los modelos que se desarrollaran dentro del análisis exploratorio y los clasificadores.

Al cabo de este proceso, se obtiene un conjunto de datos con un total de 4595 líneas con 13 atributos.

4.1.3. Preprocesamiento de datos

Con el fin de realizar diferentes análisis de discurso, y ver la afectación que tiene los diferentes métodos de preprocesamiento de datos en NLP, se aconseja generar tres (3) DataFrames que serán guardados como archivos pickle, los cuales se utilizarán a lo largo del proyecto con el fin de generar comparaciones (ver tabla 11).

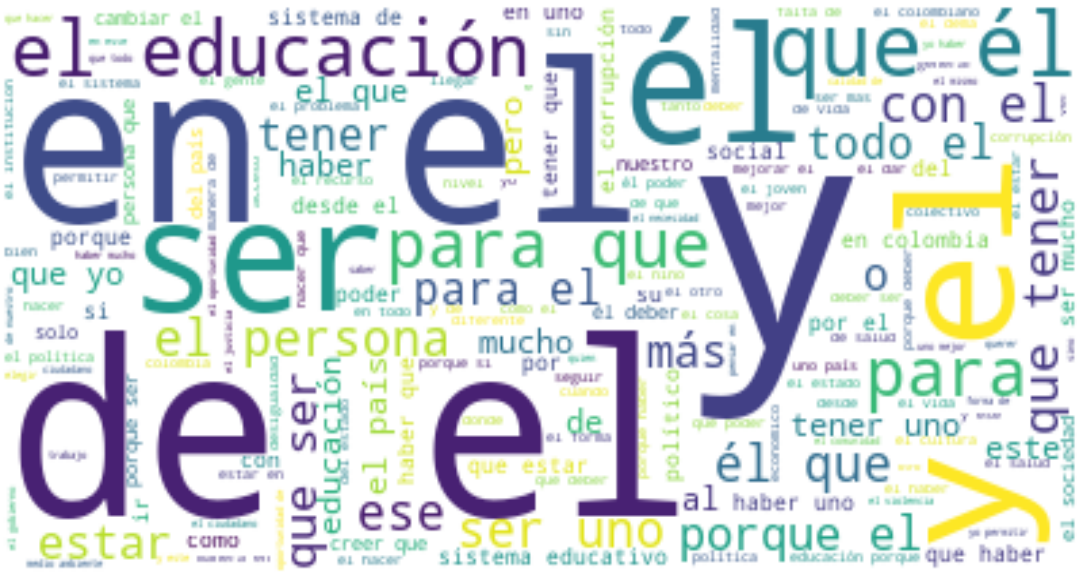
Tabla 11: DataFrames preprocesados

Preprocesamiento	Razón de creación	Nombre de archivo
Lemmatization* y Eliminación de *Stopwords	reducir la dimensionalidad y eliminar las palabras muy frecuentes que no generan información	*Lemma_no_Stopwords*
Lemmatization	Observar el impacto de *Stopwords* en los modelos	*Lemma*
Lemmatization, Stemming* y eliminación de *Stopwords	Analizar el impacto del acortamiento de las palabras en los diferentes modelos.	*Stemming*

4.1.3.1. Lemmatization y Stopwords

Con el fin de demostrar la importancia de la eliminación de las Stopwords, lemmatization y stemming, se genera una primera *Wordcloud* (ver Figura 13) con la que se puede evidenciar que las palabras con mayor frecuencia son “en” “el” “y” “de” Nótese que son palabras que a pesar de tener una gran frecuencia no generan información.

Figura 13: Wordcloud sin preprocesamiento de datos



De esta forma, el primer preprocesamiento que se realiza es la *lemmatization y la* eliminación *de Stopwords* sobre el texto_1.

Este preprocesamiento se realiza por medio de la librería *spaCy* y el modelo *es_core_news_md. En particular,* se construye una función que convierte las respuestas en tokens, y estos tokens, tienen un procesamiento de *lemmatization* y clasifica cuales son las palabras que son *Stopwords* para así eliminarlas. El proceso que se debe tener en cuenta para esta función es:

- Primero, se declara una variable que contendrá una lista de Stopwords que para fines de este proyecto también es tomada desde la lista de spaCy debido a que es la lista más amplia que existe para el español.
- Luego, se construye la función que crea los tokens para así identificar el lema de cada uno.
- Finalmente, se elimina las palabras que se encuentran dentro de la variable de las Stopwords.

Una vez realizamos este preprocesamiento, podemos evidenciar el cambio que se genera en la nube de palabras, como se muestra en la figura 14.

Figura 14: Wordcloud después de Preprocesamiento basado en la lematización y eliminación de Stopwords

Una vez eliminadas las *Stopwords* y habiendo obtenido las palabras con sus respectivos *lemmas*, se procede a realizar una limpieza de duplicados, para evitar que la misma palabra se repita, y que afecte el análisis frecuencia. Como ejemplo de la necesidad de realizar este paso se puede apreciar en la tabla 12.

Tabla 12: Eliminación de lemmas duplicados

Frase con palabras duplicadas	Frase con eliminación de palabras
visión social educación educación deber capacidad persona talento innato desigualdad base	visión social educación deber capacidad persona talento innato desigualdad base

4.1.3.2. Lemmatization

Como se explicó con anterioridad, se busca guardar un archivo con *Stopwords* y así poder ver el impacto sobre el idioma español que podrían tener los diferentes modelos en los análisis exploratorios y de creación de clasificación con aprendizaje de máquina. Para esto, se realiza una modificación en la función explicada en el numeral 4.1.3.1 donde solo se mantiene la creación de tokens para así identificar los lemmas de cada

uno. Como resultado se tiene un texto con lemmas y con *Stopwords.* En la tabla 13 se puede observar la diferencia entre el preprocesamiento realizado en el numeral 4.1.3.1 y el 4.1.3.2.

Tabla 13: Diferencia entre Lemmatization y Lemmatization & Stopwords

Lemma_no_Stopwords	Respuestas solo con Lemmatization
visión social educación deber capacidad persona talento innato desigualdad base	visión social de el educación nuestro educación deber en el capacidad de el persona en su talento innato sin tener en cuenta el desigualdad de base

4.1.3.3. Stemming

Los modelos empleados de lematización pueden tener limitantes para el idioma español, lo que se puede evidenciar en que no todas las palabras son reducidas a su forma base. Debido a esto se busca un modelo de stemming para así poder reducir el número de tokens, lo que conducirá a tener mejores modelos que utilicen BoW o TF-IDF.

Con el fin de encontrar el mejor método de stemming para el idioma español, se comparó el resultado obtenido tras aplicar tres diferentes métodos a una frase con diferentes formas de la misma palabra. La Tabla 14 muestra los resultados obtenidos.

Tabla 14: Tipos de Stemming

Stemming	Resultado
Porter	`['sistema', 'educativo', 'educación', 'abandonado', 'abandonar', 'abandono']`
Snowball	`['sistem', 'educ', 'educ', 'abandon', 'abandon', 'abandon']`

Lancaster	['Sistema', 'educativo', 'educación', 'abandonado', 'abandonar', 'abandono']

La tabla 14 evidencia que el *stemmer* que más agregación de tokens podría generar es el método *Snowball*, que, si bien hace un recorte de las palabras, también puede hacer perder el significado de las mismas.

En la tabla 15 se puede observar la diferencia entre una frase con el proceso de *lemmatization* y un texto pasado por el stemmer Snowball.

Tabla 15: Resultados Stemming

Lemmatization	**Stemming**
visión social educación deber capacidad persona talento innato desigualdad base	vision social educ deb capac person talent innat desigualdad bas

4.1.4. Representación de documentos

Como ya se ha mencionado, se aplicarán diferentes modelos de aprendizaje de máquina para el procesamiento de lenguaje natural con el fin de realizar tanto un análisis de discurso como una clasificación de las respuestas. Por consiguiente, se debe transformar el texto en vectores y en el caso de esta tesis se opta por obtenerlos por medio de las técnicas BoW y TF-IDF.

4.1.4.1. BoW

Para la construcción de la bolsa de palabras, se realizará un procedimiento de extracción de tokens desde las respuestas de los participantes- Luego, se revisará la frecuencia de estas palabras y finalmente se construirá un vector de palabras que se basará en la frecuencia del vocabulario que tiene el cuerpo del documento.

Para la construcción de BoW, se utilizan los conjuntos de datos *Lemma_no_Stopwords*, y *Stemming,* y con ayuda de la librería de *Sklearn* y *CountVectorizer*, se generará la matriz de vectores de palabras aplicando el método *fit_transform* de *CountVectorizer.* La matriz obtenida se transforma en un DataFrame con el fin de poder tener una visualización amigable, que permita la identificación de tokens y sus frecuencias.

Como ejemplo, se puede observar que para línea de texto "*Voto voluntario voto obligatorio revisar la abstención elección legislativo ejecutivo, encontrar elección colombiano habilitar votar votar"* en BoW se puede encontrar los tokens voluntario, obligatorio, revisar, abstención, legislativo, ejecutivo, encontrar, colombiano y habilitado 1 vez y las palabras votar, voto, elección 2 veces (ver figura 15).

Figura 15: BoW

```
bow_df.head()
```

ocación	volar	voltear	voluntad	voluntariado	voluntario	volver	volvimos	votación	votado	votalidad	votante	votar	votir	voto	voucher	voz
0	0	0	0	0	1	0	0	0	0	0	0	2	0	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Como primer resultado, se obtiene una matriz con un tamaño de 4595x5443 para el conjunto de datos *Lemma_no_Stopwords*, y para el conjunto de datos *Stemming*, una matriz de 4595x3900, por lo que se está reduciendo el conjunto de datos en 28.34%. Sin embargo, se debe tener en cuenta que la agrupación de las palabras de stemming no son la base raíz, por lo que puede que pueden ser diferentes palabras que sean similares en sus caracteres iniciales.

Gracias a la BoW, se obtiene información sobre la frecuencia que tiene cada token, por lo que se puede generar un nuevo conjunto de datos con la eliminación de tokens cuya frecuencia sea mínima, esto con el fin de ayudar a la mejora de los modelos de clasificación que se expondrán en el numeral 4.3 de este proyecto.

Para revisar el impacto de eliminar tokens con poca frecuencia, se revisa cual es porcentaje que representa los tokens dentro de la BoW, en el cual se observa que al eliminar los tokens que tienen una frecuencia de 1 sola vez, se eliminaría el 50% de la información del conjunto de datos (ver tabla 16).

Tabla 16: Tokens y % Representación de acuerdo a su frecuencia

Frecuencia	% Representación
1	50.37%
2	14.27%
3	7.27%
4	4.46%
5	2.97%

4.1.4.2. TF-IDF

Para la creación de la matriz TF-IDF, se utilizará el mismo procedimiento de la BoW, con la diferencia que se utiliza la función *TfidfVectorizer* que ayuda a calcular la importancia del token dentro del conjunto de datos.

Para la construcción de esta matriz se inicializa el *TfidfVectorizer* de la librería *Sklearn* y luego se pasa la columna con el texto que se quiere analizar, y para poder tener una visualización humanamente posible, se transforma la matriz a un DataFrame de pandas, donde el nombre de las columnas serán los diferentes tokens que se obtienen por medio del método *get_features_names().*

Este tipo de matriz se utiliza para encontrar la importancia de los tokens en cada una de las frases, donde 0.0 es el valor con menor importancia. Como resultado se obtiene una matriz con un tamaño de 4595x5443 para el conjunto de datos *Lemma_no_Stopwords.*

En la figura 16, se puede observar un ejemplo de los tokens y su importancia para cada una de los textos.

Figura 16: Ejemplo de TF-IDF

abstención	absurdo	abuchear	abuelito	abuelo	abundar	aburrir	abusar	abusivo	abuso	acabado	acabalidad
0.340108	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Al igual que la bolsa de palabras, se puede realizar una eliminación de tokens que tengan una importancia no significativa para el conjunto de datos. Para esto se revisa el porcentaje de representación en cada token y así excluirlos del estudio en caso de requerir un ajuste en los modelos de clasificación (ver tabla 17).

Tabla 17: % Representación TF-IDF

Importancia	% Representación
0 -10%	0%
10% - 20%	0%
20% - 30%	2.7%
30% - 40%	24.2%

De acuerdo a la tabla anterior, se puede concluir que se podría realizar una eliminación de los tokens que tengan un valor menor al 0.3 y esto afectará solo en un 2.7% del conjunto de datos, por lo que se tendrá un matriz con un tamaño de 4595x5295.

Adicional a este tipo de análisis, gracias a TF-IDF se puede revisar la similaridad que se tiene entre los tokens, y así reducir aún más esta matriz, por lo que por medio de la función *cosine_similarity* de la librería *sklearn* se realiza una comparación de esta matriz contra ella misma.

Así pues, se puede evidenciar que los tokens que tienen una similaridad mayor al 90% son 133, por lo cual se proceden a eliminarse, y como resultado se genera una matriz final de tamaño 4595 * 5162.

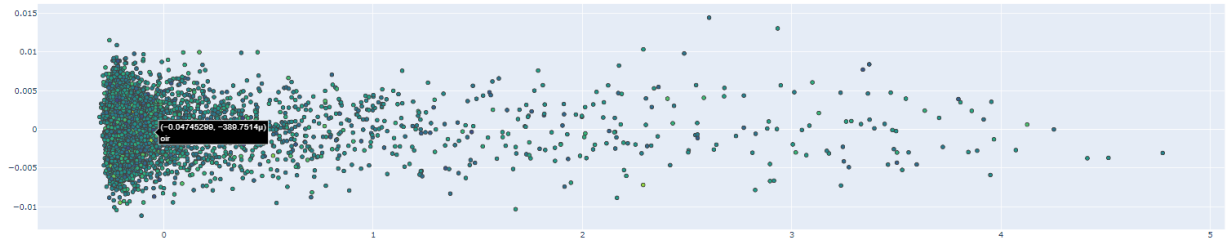
4.1.4.3. Work2Vec

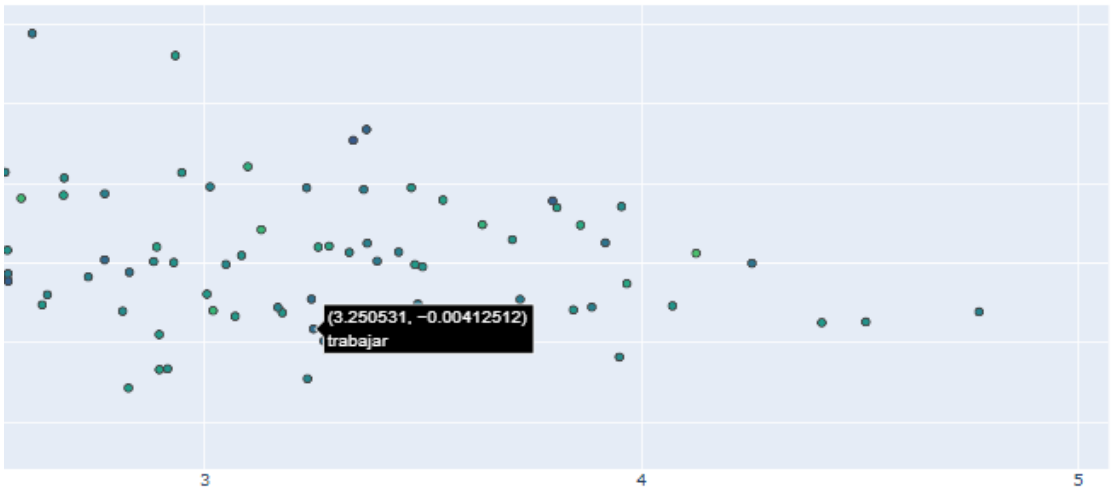
Para la creación de work2vec se requiere realizar la instalación de modelo de español de *spaCy es_core_news_md* y se realiza *tokenización* del conjunto de datos *Lemma_no_Stopwords*, para así crear el corpus y los vectores. Se utiliza la técnica de PCA con el fin de poder colocar los resultados en un DataFrame y así poder realizar un gráfico con *Plotly.*

El grafico de *plotly* es interactivo, que puede ser visualizado en la figura 17, en donde se logra observar a profundidad la distribución de todos los tokens del conjunto de datos. Este tipo de modelos ayuda a poder identificar tokens que sean similares por medio de la similitud del coseno, también es posible evaluar las analogías y encontrar la palabra menos parecida o que no coincide con las demás.

Figura 17: Ejemplo de visualización Work2Vec

	x	y	word
0	1.187320	-0.002257	voto
1	-0.196334	0.001694	voluntario
2	0.028995	-0.002529	obligatorio
3	0.452755	0.006945	revisar
4	-0.062202	0.000850	abstención





Este tipo de modelado puede ser utilizado en futuros proyectos como la predicción de texto y así ayudar a la calidad del conjunto de datos evitando errores de mala digitación, que ayudará a la mejora de todos los modelos en general.

4.2. Análisis exploratorio

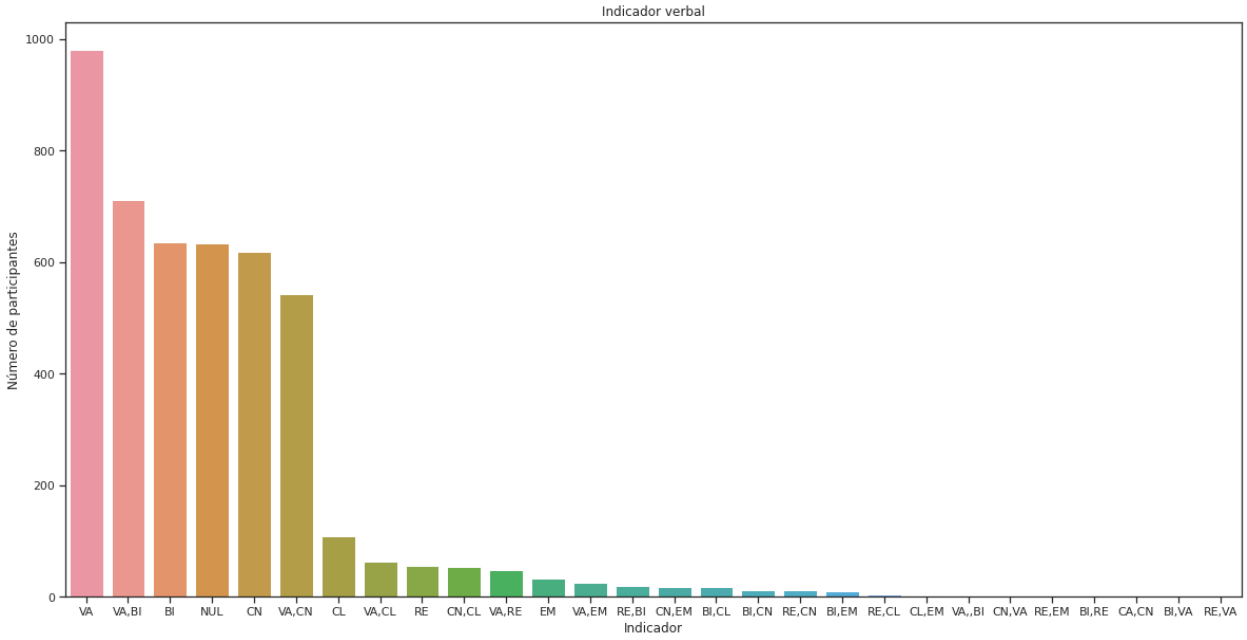
4.2.1. Entendimiento de variables

Dentro de este primer acercamiento al conjunto de datos, se espera tener un conocimiento sobre las variables que se están manejando, por esto, es necesario realizar un análisis descriptivo de su distribución, media, mediana y moda en cada variable.

Dentro de este proyecto, se quiere conocer la distribución de la etiqueta del indicador verbal, ya que esta será utilizada en los modelos de clasificación. Para esto se revisa por medio de un histograma los valores únicos que contiene el conjunto de datos, y el número de respuestas que tienen esta clasificación.

En la siguiente ilustración se puede encontrar que el indicador VA tiene una representación del 22%, seguido por una combinación entre el indicador VA y BI de un 15%, en un tercer lugar esta solamente el indicador BI con una representación del 13%, seguido por NUL y CN con el mismo porcentaje del 13% (ver imagen 18).

Figura 18: Clasificador de indicador verbal



El indicador verbal, es la etiqueta utilizada para la clasificación de respuestas, por tal motivo, se realiza un análisis más profundo, conociendo su impacto dentro las respuestas dadas por género, región y zona y así poder identificar si existe algún factor externo que afecte este tipo de etiqueta que en la actualidad se asigna de forma manual por el modulador de la entrevista.

En la figura 19 (*pivot_table*) se puede observar que el indicador verbal no tiene ninguna afectación por el género del participante, ya que se mantiene la misma frecuencia que la vista en la imagen anterior.

Figura 19:Tabla pívot, número de respuestas por indicador verbal y genero

etiqueta_texto_1	BI	BI,CL	BI,CN	BI,EM	BI,RE	BI,VA	CA,CN	CL	CL,EM	CN	CN,CL	CN,EM	CN,VA	EM	NUL	RE	RE,BI	RE,CL	RE,CN	RE,EM	RE,VA	VA	VA,,BI	VA,BI	VA,CL	VA,CN	VA,EM	VA,RE	All
genero																													
Femenino	278	6	2	7	0	0	1	51	2	278	13	9	1	15	255	22	11	2	7	1	0	429	0	316	24	229	11	18	1988
Masculino	317	11	10	2	1	1	0	51	0	295	35	9	0	16	318	30	6	1	5	0	1	476	1	347	35	280	10	25	2283
No binario	2	0	0	0	0	0	0	2	0	3	1	0	0	0	2	0	0	0	0	0	0	4	0	2	1	2	1	0	20
No sabe/No responde	2	0	0	0	0	0	0	0	0	3	0	0	0	0	1	0	0	0	0	0	0	2	0	1	0	1	0	0	10
All	599	17	12	9	1	1	1	104	2	579	49	18	1	31	576	52	17	3	12	1	1	911	1	666	60	512	22	43	4301

Sin embargo, al realiza un análisis descriptivo sobre la zona, se si puede ver que existe un cambio en el orden de la etiqueta, donde empieza a tener mayor relevancia el estar en una zona rural, donde VA continua en primer lugar con un 20% pero las consecuencias negativas representan el 16% del conjunto de datos (ver figura 20).

Figura 20: Tabla pívot, número respuestas por indicador verbal y zona

etiqueta_texto_1	BI	BI,CL	BI,CN	BI,EM	BI,RE	BI,VA	CA,CN	CL	CL,EM	CN	CN,CL	CN,EM	CN,VA	EM	NUL	RE	RE,BI	RE,CL	RE,CN	RE,EM	RE,VA	VA	VA,,BI	VA,BI	VA,CL	VA,CN	VA,EM	VA,RE	All
zona																													
Rural	53	4	5	1	0	0	0	10	0	78	5	2	0	4	70	4	1	0	1	0	0	94	0	66	10	53	3	4	468
Urbana	531	13	7	8	1	1	1	92	2	488	42	16	1	27	490	45	15	3	10	1	1	796	1	578	47	446	19	37	3719
All	584	17	12	9	1	1	1	102	2	566	47	18	1	31	560	49	16	3	11	1	1	890	1	644	57	499	22	41	4187

Finalmente, se puede identificar, que el indicador verbal también es afectado de forma mínima por región, ya que, dependiendo de la región del país, las consecuencias negativas tienen mayor peso que como en el caso de Antioquia – Eje Cafetero como puede ser visualizado en la figura 21.

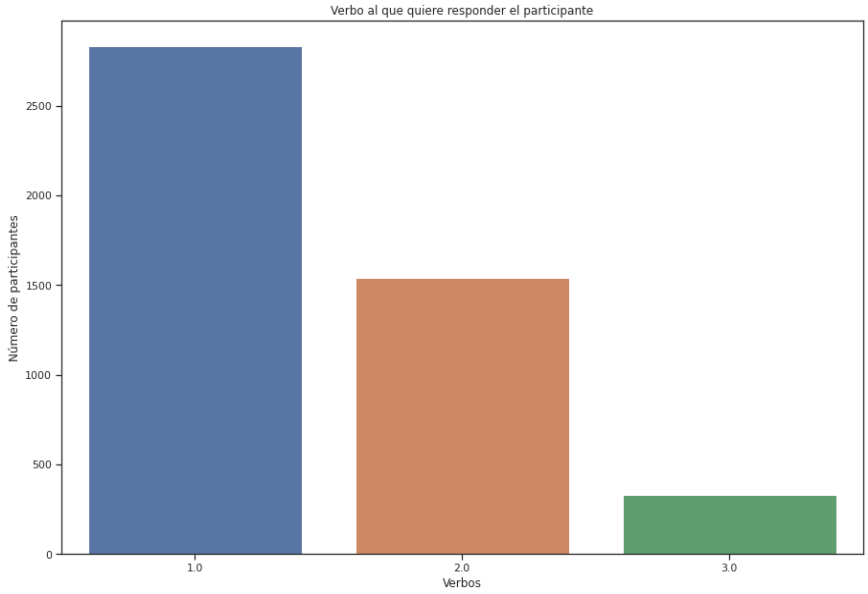
Figura 21: Tabla pívot, número de respuestas por indicador verbal y región

etiqueta_texto_1	BI	BI,CL	BI,CN	BI,EM	BI,RE	BI,VA	CA,CN	CL	CL,EM	CN	CN,CL	CN,EM	CN,VA	EM	NUL	RE	RE,BI	RE,CL	RE,CN	RE,EM	RE,VA	VA	VA,,BI	VA,BI	VA,CL	VA,CN	VA,EM	VA,RE	All
region																													
Amazonia	4	0	1	0	0	0	0	3	0	8	1	0	0	0	4	0	0	0	0	0	0	9	0	10	1	6	0	2	49
Antioquia-Eje cafetero	177	6	6	4	0	0	0	35	1	196	16	8	0	15	188	19	6	2	5	0	0	297	0	185	21	129	6	12	1334
Caribe	97	3	0	0	0	1	0	14	0	104	5	1	0	4	101	6	2	0	2	0	0	133	0	116	3	77	5	8	682
Centro	230	6	5	4	1	0	1	39	1	194	19	7	1	11	212	13	7	1	4	1	0	350	1	273	24	222	9	16	1652
Llanos orientales	12	0	0	0	0	0	0	1	0	7	2	1	0	0	6	0	0	0	0	0	0	14	0	7	1	7	1	1	60
Pacifico	66	2	0	1	0	0	0	10	0	60	4	1	0	1	55	11	2	0	1	0	1	94	0	60	8	63	1	4	445
All	586	17	12	9	1	1	1	102	2	569	47	18	1	31	566	49	17	3	12	1	1	897	1	651	58	504	22	43	4222

Dentro de esta consulta ciudadana, se les pide a los participantes indicar que verbo entre “cambiar”, “mejorar” o “mantener” que quisieran contestar, por lo que es necesario conocer cuál es el comportamiento de esta variable dentro del conjunto de datos.

En el siguiente histograma presentado en la figura 22, se puede ver que el 60% de los participantes prefieren seleccionar preguntas con el verbo cambiar, seguido por un 32% el verbo mejorar y un 7% el verbo mantener.

Figura 22: Verbo que se quiere responder



Al realizar el mismo análisis anterior por región, zona y género, se puede ver en la figura 23 que esta distribución no cambia.

Figura 23: Tablas pívot número de participantes por verbo y datos demográficos

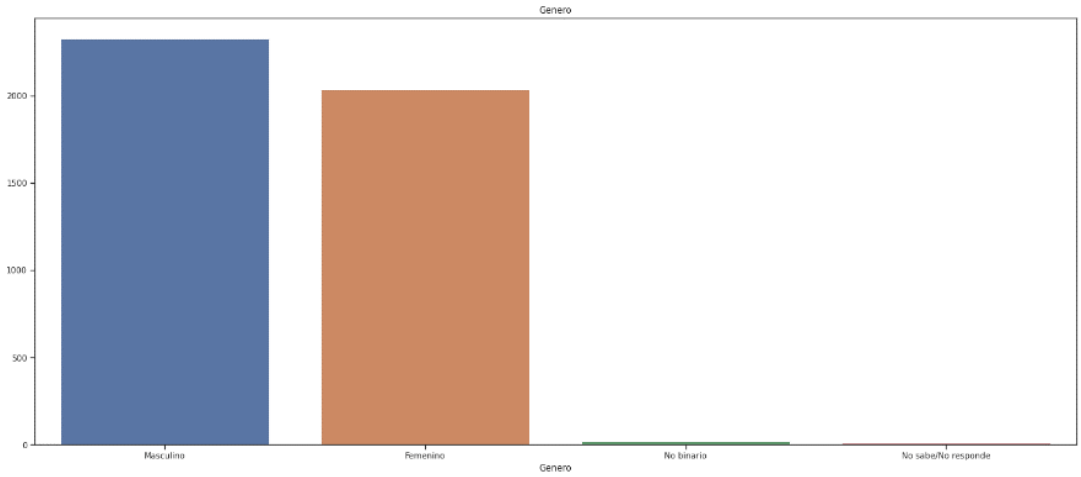
cambiar_mejorar_mantener	1.0	2.0	3.0	All
genero				
Femenino	1157	726	149	2032
Masculino	1468	696	164	2328
No binario	13	7	0	20
No sabe/No responde	6	3	1	10
All	2644	1432	314	4390

cambiar_mejorar_mantener	1.0	2.0	3.0	All
region				
Amazonía	37	15	2	54
Antioquia-Eje cafetero	764	467	123	1354
Caribe	392	274	36	702
Centro	1063	501	112	1676
Llanos orientales	45	9	7	61
Pacífico	297	139	27	463
All	2598	1405	307	4310

cambiar_mejorar_mantener	1.0	2.0	3.0	All
zona				
Rural	281	173	24	478
Urbana	2292	1224	280	3796
All	2573	1397	304	4274

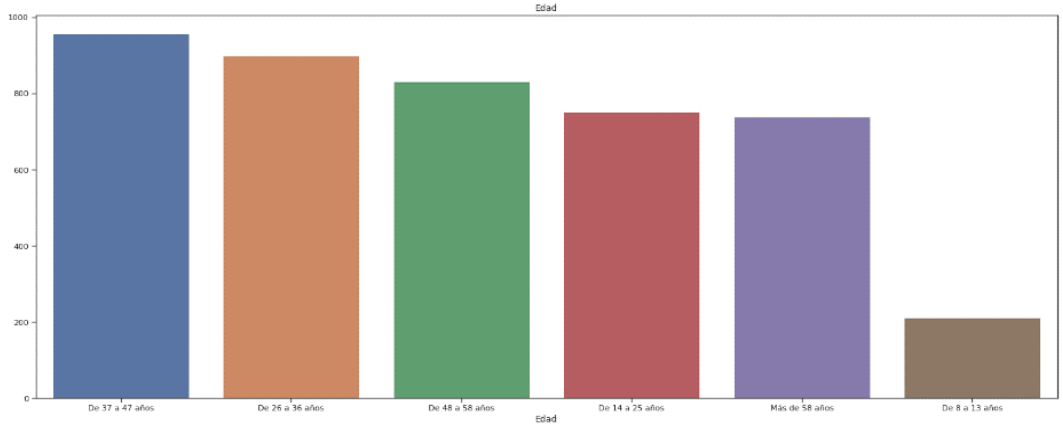
Finalmente, se revisa cual es la distribución que tiene el género dentro del conjunto de datos, donde se identifica que el género masculino representa el 53% del conjunto de datos, seguido por las mujeres con un 46% y un 1% entre genero no binario y aquellos participantes que no contestaron esta pregunta (ver figura 24).

Figura 24: Genero que participan en el conjunto de datos



Como se evidencia en la figura 25, se encuentra el rango de edad de los participantes está distribuida en 22% personas que se encuentran entre los 37 a 47 años, seguidos por participantes de 26 a 36 años, y con un 19% participantes con edades entre 14 a 25 años.

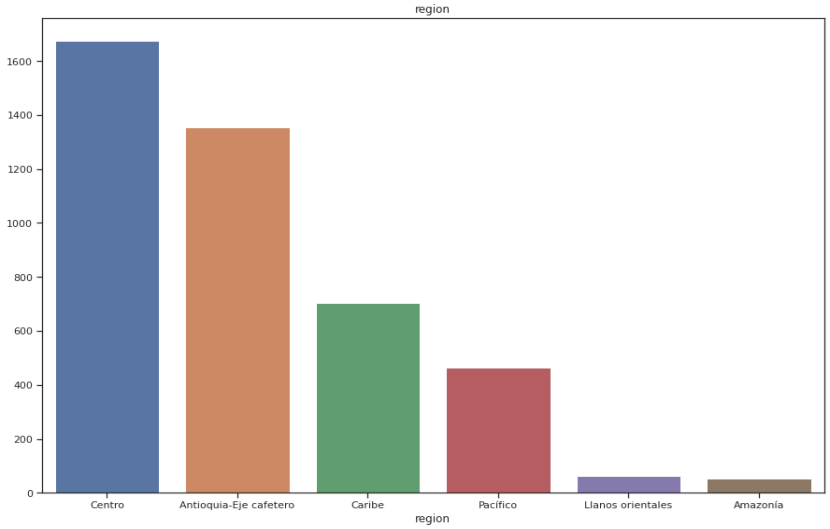
Figura 25: Edades de participantes



Dentro de las regiones que participan dentro de este tipo de encuestas se encuentran la región central que tiene una participación del 38%, seguida por la región Antioquia y eje

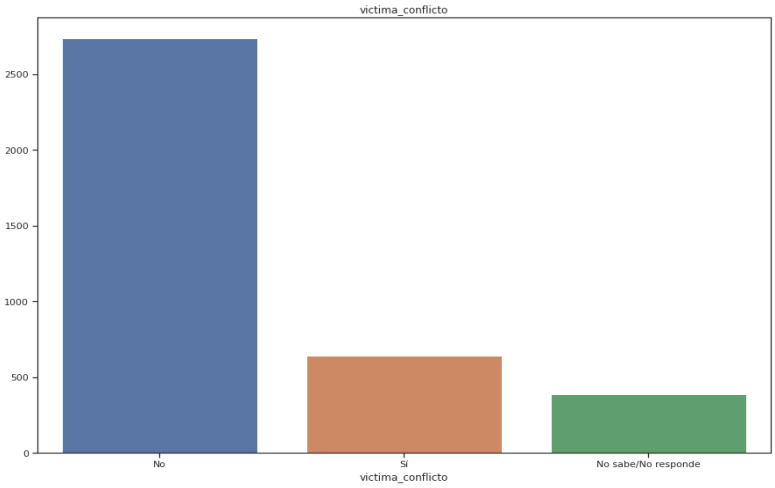
cafetero con 31% seguidas por las regiones Caribe, Pacífico, Llanos orientales y Amazonía que solo tiene una participación del 1% (ver figura 26).

Figura 26: Regiones en el conjunto de datos



Finalmente, la última variable que se busca analizar es conocer si los ciudadanos fueron víctimas del conflicto armado, donde solo existe una minoría de 10%. Este tipo de análisis puede ayudar a identificar outliers, y realizar un estudio diferente dentro de este tipo de participantes (ver figura 27).

Figura 27: Victimas de conflicto en el conjunto de datos

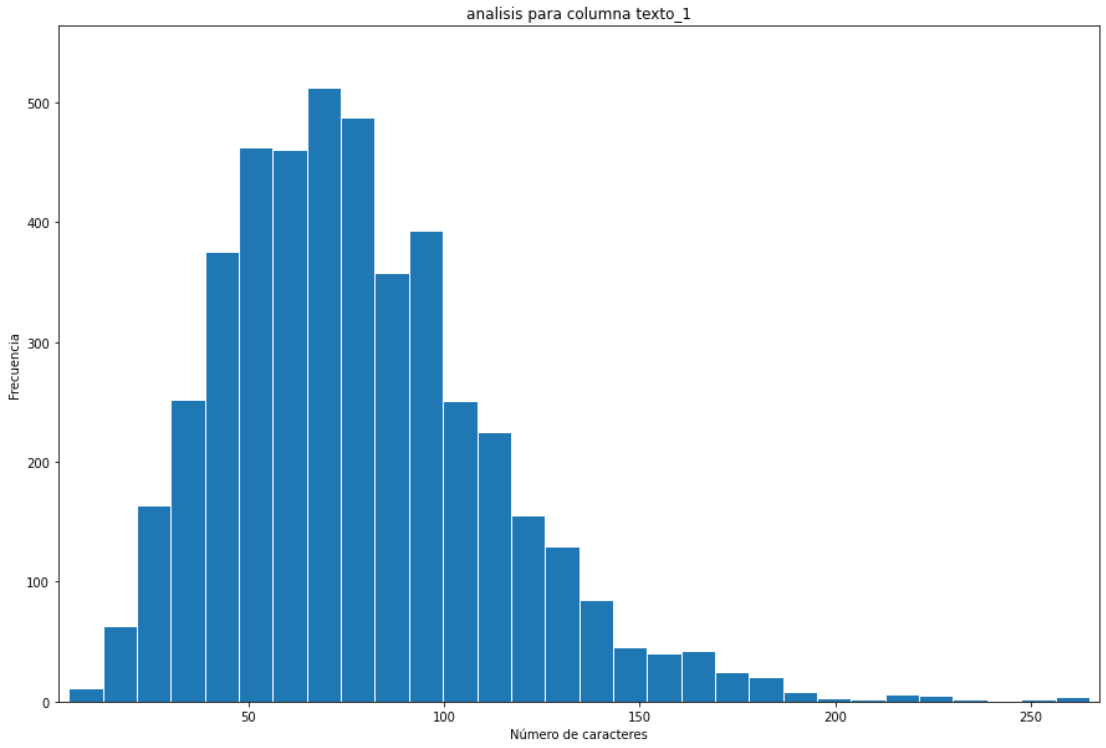


4.2.2. Análisis estadístico

El análisis estadístico realizado en esta sección busca la exploración fundamental de las características del texto, para esto, se visualizarán en gráficos de barra e histogramas. Lo primero a revisar es el número de caracteres que están presentes en cada frase, y así poder revisar la longitud aproximada de cada respuesta dada.

Por medio de la librería *matplotlib* podemos visualizar en la figura 28 que el rango de caracteres utilizados para las respuestas del ciclo 1 está entre 10 hasta 250, con algunos valores extremos, por encima de 260.

Figura 28: Análisis de conteo de caracteres



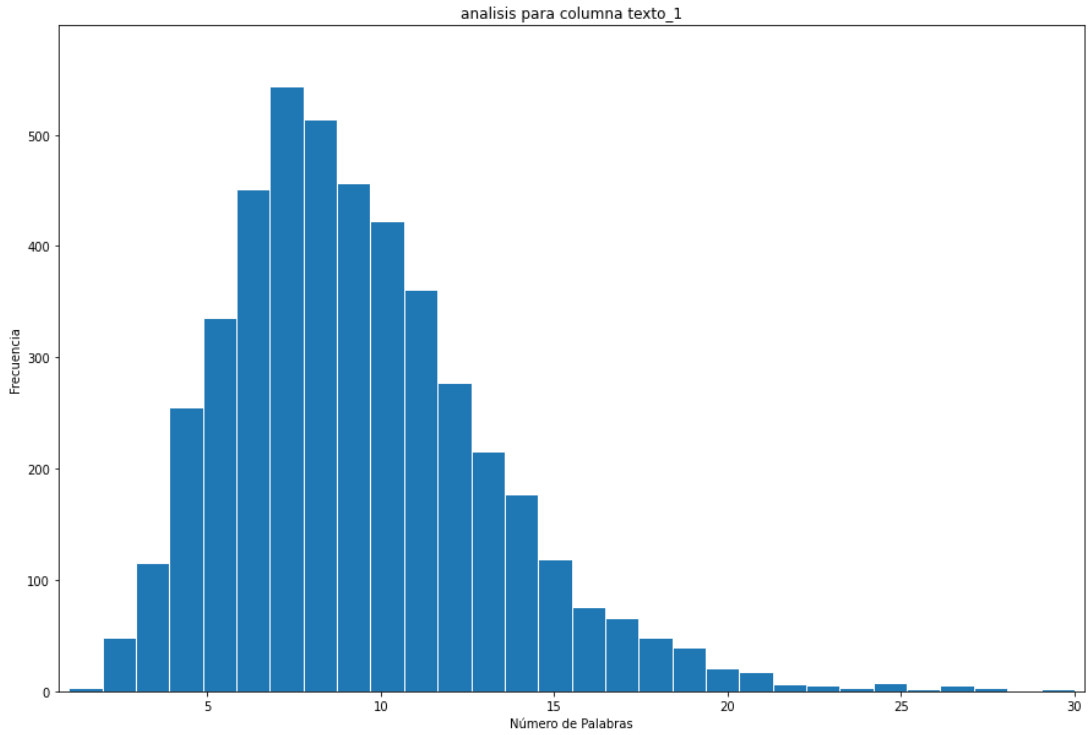
Con este tipo de análisis se puede encontrar aquellas palabras que no hayan sido eliminadas en el proceso de *Stopwords* y que, para el idioma español, no generan

información, como aquellas que tienen una longitud menor o igual a 3 caracteres y así excluir estas palabras del estudio.

Como resultado de este primer análisis, dentro del texto_1, no existen tokens con una longitud menor estrictamente a 5, por lo que no se excluye ninguna palabra.

Luego de esto, se busca conocer el número de palabras utilizadas en las respuestas de los participantes. En la figura 29 se presenta el histograma, en el que se puede visualizar que el número de palabras utilizadas en respuestas del ciclo 1 se encuentra entre 2 a 12, donde se ve un gran pico en utilizar 8 palabras, también se observa con algunos datos atípicos que llegan hasta 30 palabras por respuesta.

Figura 29: Análisis número de palabras por respuesta



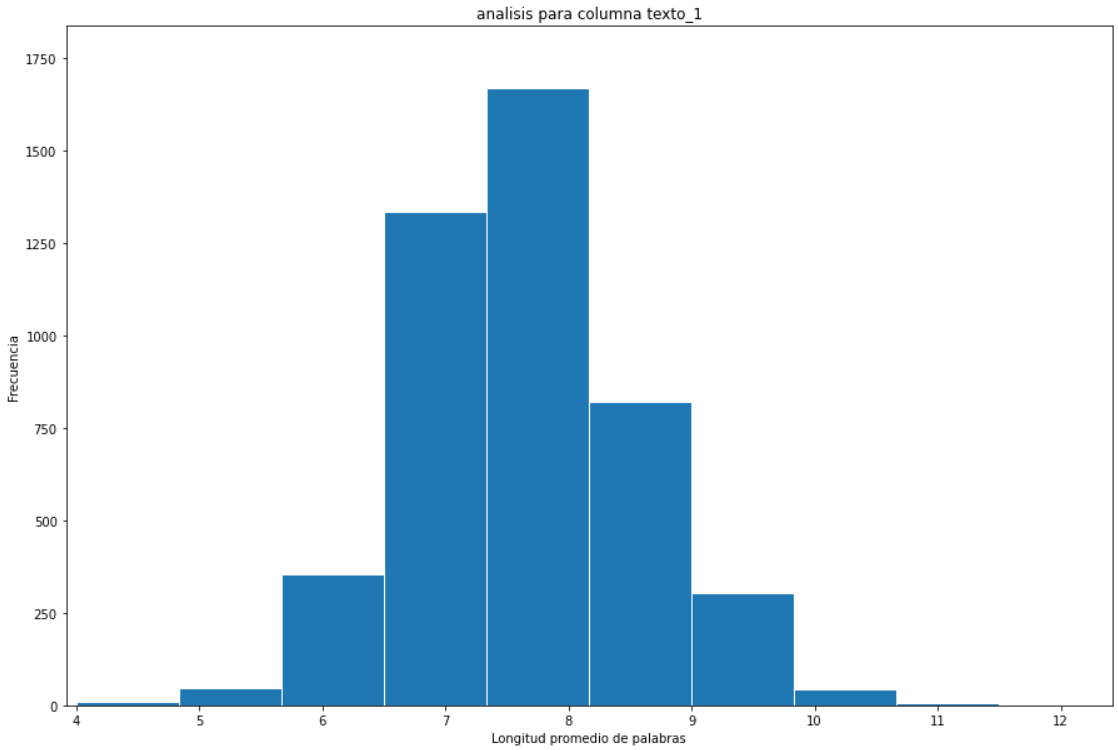
Se puede realizar una revisión a aquellas respuestas que tienen entre una y 4 palabras como respuesta y estas pueden eliminarse, con el fin de reducir la dimensionalidad, y evitar errores en los modelos, ya que, para un modelo supervisado de aprendizaje de

máquina, una frase con 3 tokens es difícil de categorizar por la poca cantidad de información que genera este tipo de respuesta.

Como resultado, se puede observar que existen 51 respuestas que tienen un máximo de tres palabras por respuesta, entre las cuales se encuentran "*educación precario", "corrupción", "salud malo".* Se procede a excluir dichas respuestas.

Finalmente se busca la longitud promedio de las palabras usadas en las respuestas, donde vemos que la longitud más común se encuentra entre 7 y 9 palabras (ver figura 30.

Figura 30: Análisis de longitud de palabras

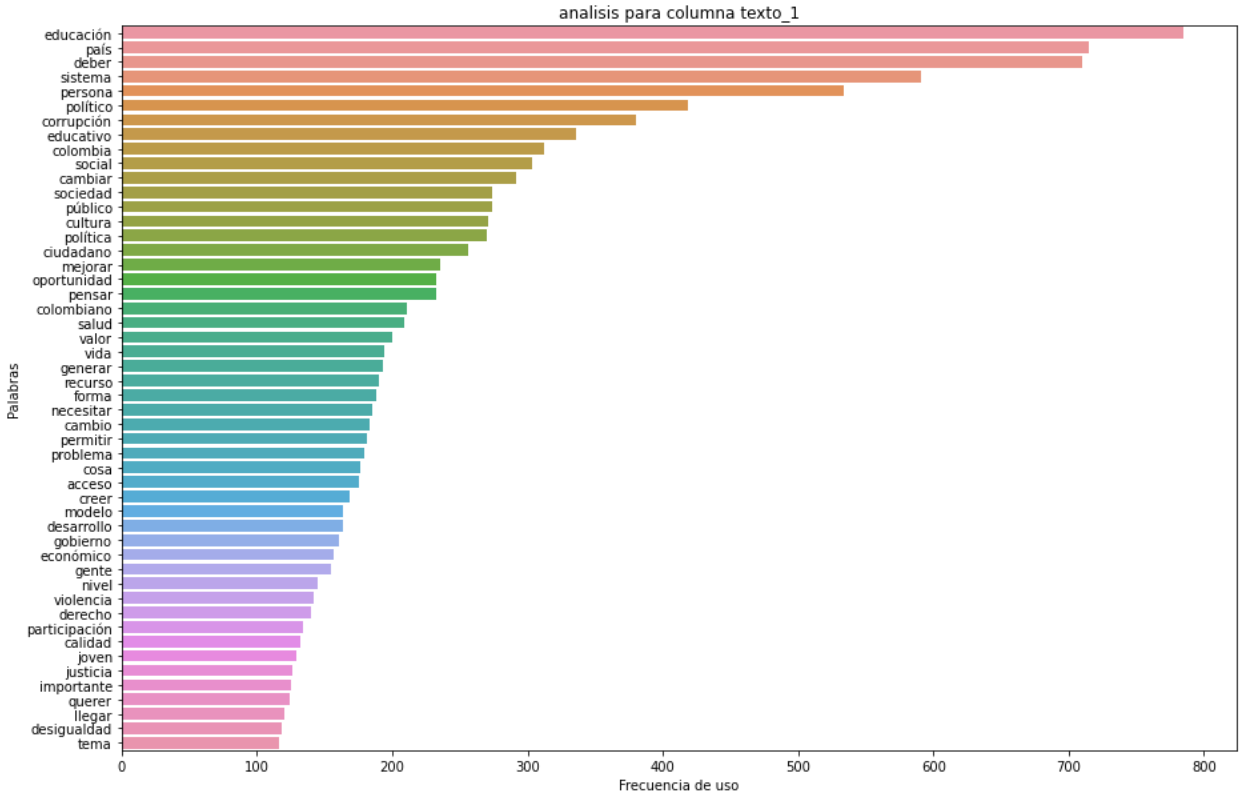


Nuevamente, se busca aquellas respuestas que entre sus palabras existen tokens con una longitud menor a 4, con el fin de excluirlas del estudio. Dentro de este estudio no se encuentra tokens para excluir.

Este tipo de análisis es requerido para análisis como análisis de sentimientos, *POS Tagging* y *NER* donde se utilizará los modelos pre entrenados de AWS, los cuales tienen limitaciones para utilizar el *Free Tier.* Para conocer más detalle de este tipo de restricciones se puede visualizar los numerales 4.2.6, 4.2.7 y 4.2.8.

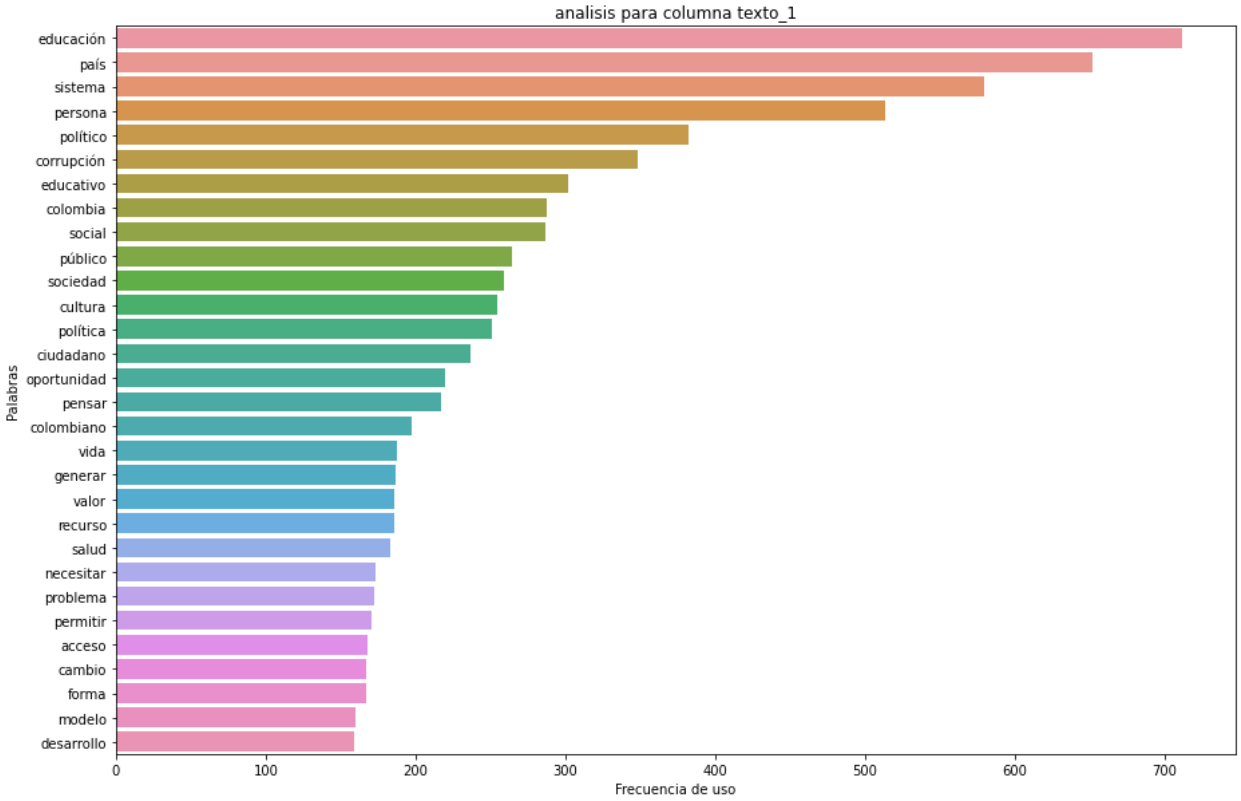
Finalmente, al igual que la nube de palabras presentado con anterioridad, se puede visualizar en la figura 31, cuáles son las palabras que tienen mayor frecuencia entre las respuestas de los ciudadanos. Las palabras con mayor frecuencia son *"educación", "país", "deber", "sistema", "político", "corrupción".*

Figura 31: Palabras con mayor frecuencia



Se puede evidenciar que existen varios verbos modulares, que tienen una gran frecuencia y que pueden generar ruido en los próximos modelos, por lo cual se excluyen palabras como *"deber", "cambiar", "mantener", "mejorar", "cosa" y "tema"* el cual nos da una nueva frecuencia (ver figura 32).

Figura 32: Frecuencia de palabras con modificación



Gracias a la eliminación de palabras modales, se pueden ver que dentro del top 20 de palabras más frecuentes se encuentra ahora “desarrollo”, “modelo”, “forma”, “acceso” entre otras.

Como resultado final se guarda un archivo plano, en formato pickel, que será utilizado en los siguientes modelos con un total de 4566 respuestas.

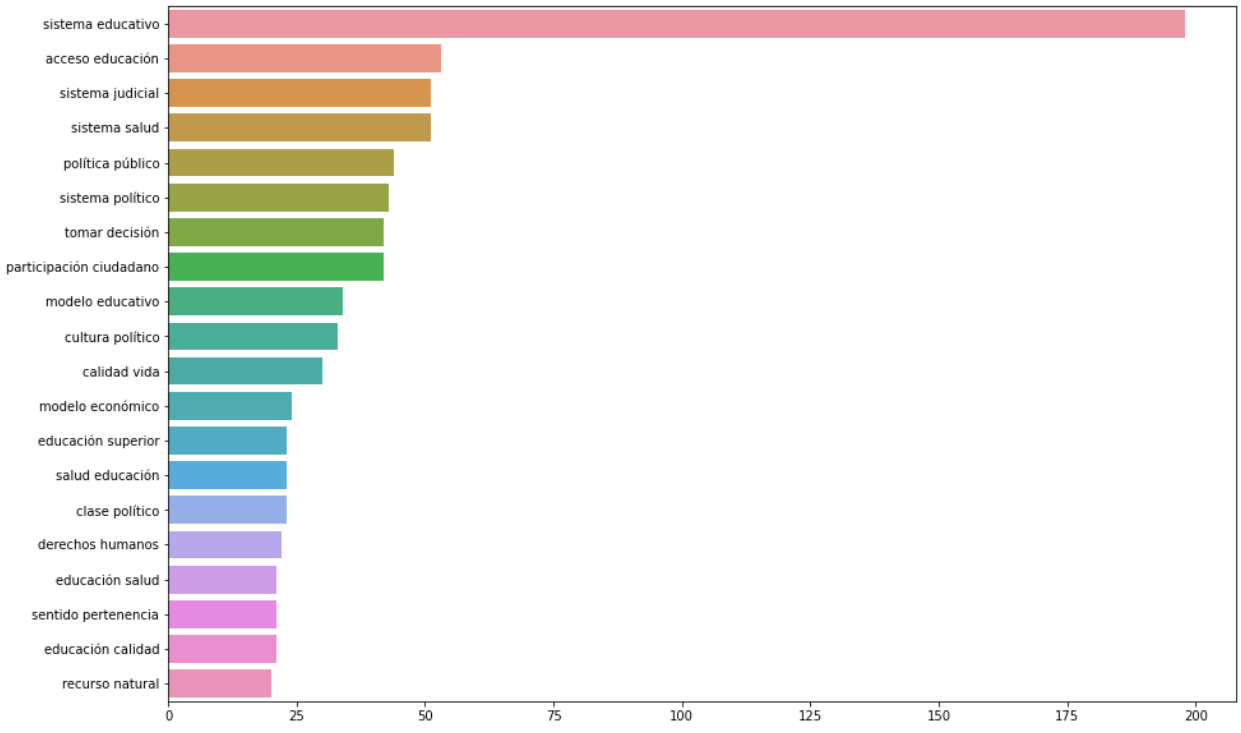
4.2.3. Exploración de N-Grams

Conociendo los *N-grams* más frecuentes, se puede tener un mejor entendimiento del contexto en el cual se utiliza las palabras con mayor frecuencia. Para la creación de los diagramas de barras que indicarán cuales son los *N-gram*, es necesario generar una BoW y una matriz TF-IDF utilizando el conjunto de datos del análisis estadístico, y así lograr encontrar los N-gram más frecuentes.

Para este método, se debe adicionar en *CountVectorizer* el argumento *ngram range* el cual es una tupla que contiene el límite inferior y superior del rango de n valores, por lo que al pasar (2,2) generará solamente *bi-grams*, pero agregar (1,3) generará *Ngrams* cuando n es igual a 1,2 y 3.

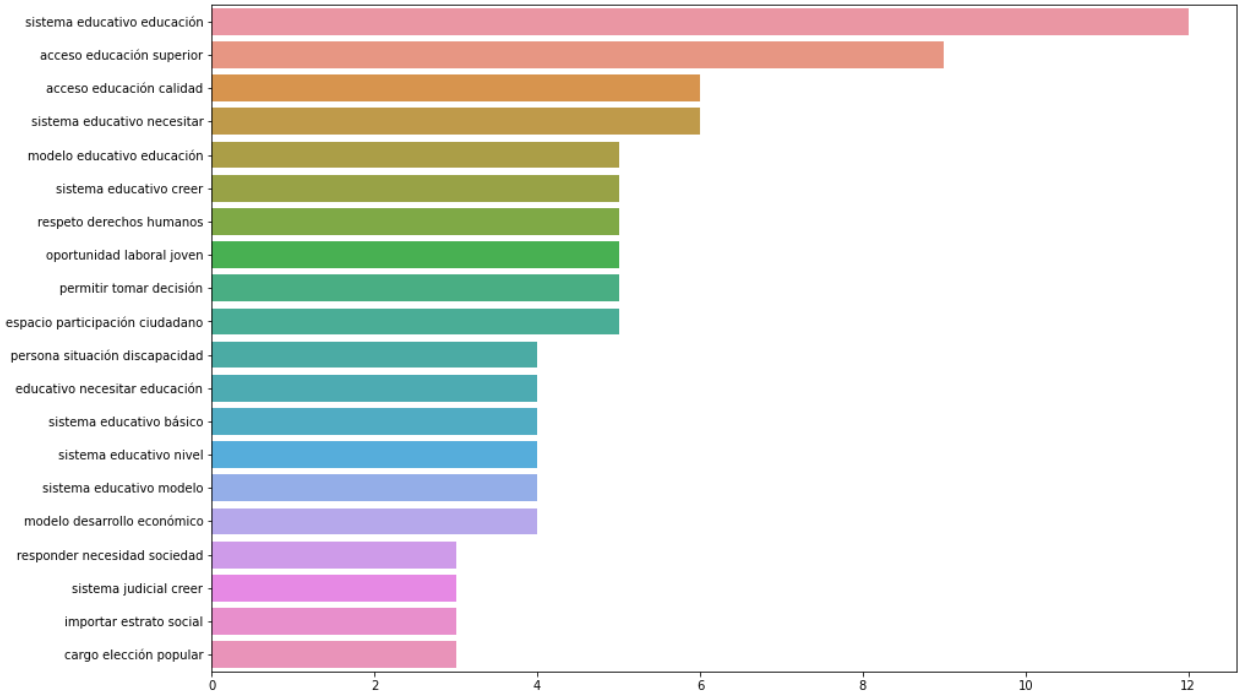
En las figuras 33 y 34, se puede observar que los *bi-grams* y *tri-grams* más frecuentes., En el caso de los *bi-grams,* los más frecuentes son "*sistema educativo"* seguido por "*acceso educación", "sistema judicial",* y "*sistema salud".*

Figura 33: Bi-grams con verbos modales



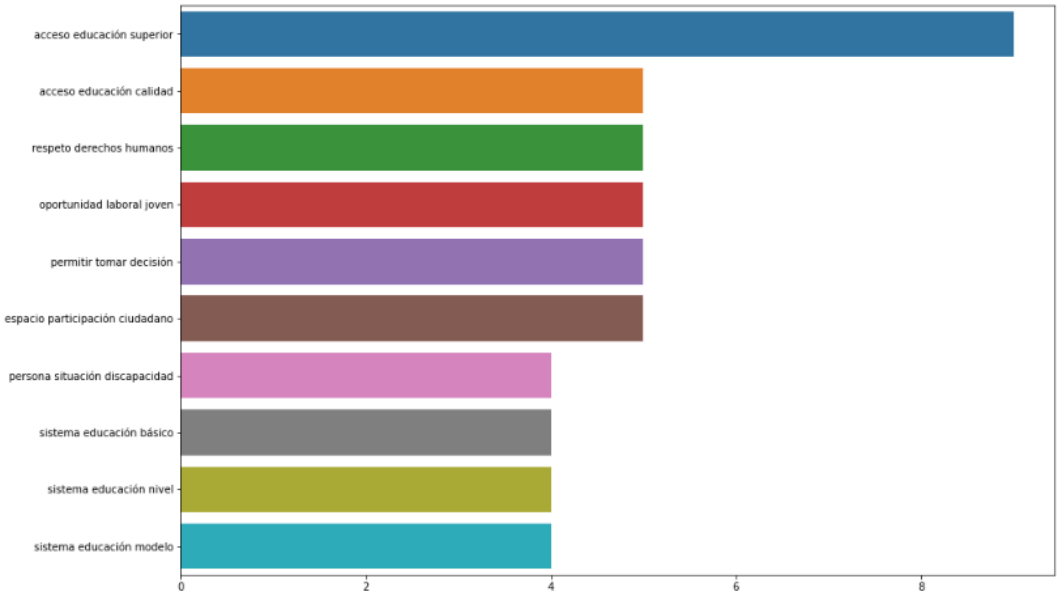
En cuanto a los *tri-grams,* entre los más comunes se encuentran "*sistema educativo educación", "acceso educación superior", "acceso educación calidad", "sistema educativo necesitar".* Como puede observarse, los temas que tienen mayor relevancia están relacionados con la educación.

Figura 34: Tri-grams con verbos modales



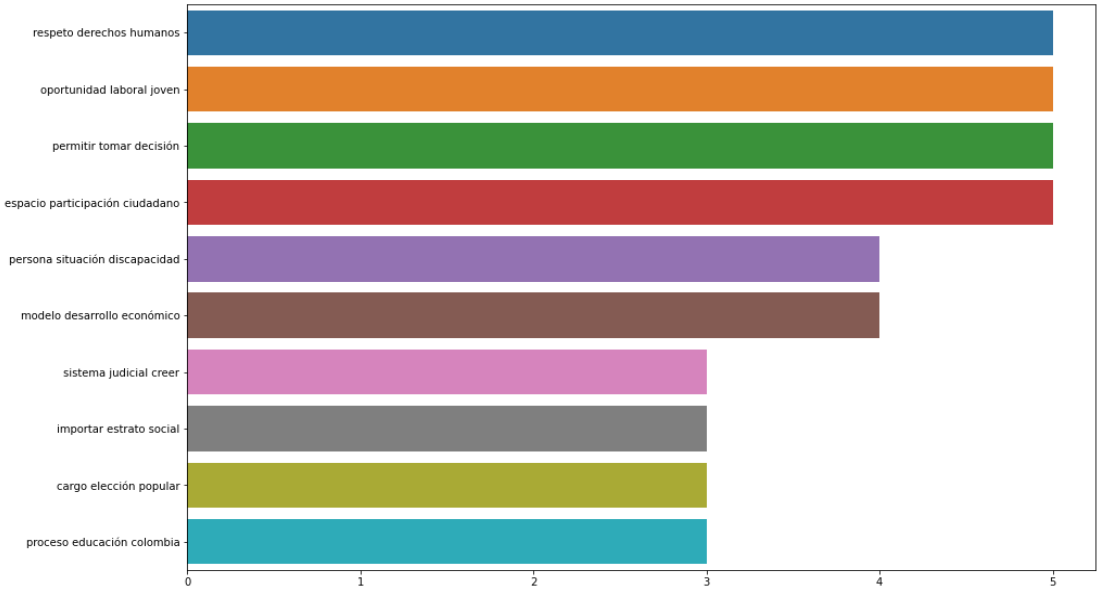
Gracias al análisis del *bi-grams* y *tri-grams* se detectaron otros tokens que se podrían eliminar del estudio como *"necesitar", "responder"* y unificar la palabra "*educativo"* con "*educación"*. Con esta modificación, se observa, que para *tri-grams,* se encuentra nuevos resultados, en los cuales los resultados más populares son: *"acceso educación superior", "acceso educación calidad", "respeto derechos humanos"* y *"oportunidad laboral joven"* (ver figura 35 y 36)

Figura 35: Tri-grams modificado



Con el fin de conocer *tri-grams* que no estén relacionados a *"sistema educación" y "acceso educación"* se realiza una exclusión de estos *bi-grams* y así poder revisar que otro tipo de *tri-grams* se encuentran dentro de las respuestas de los ciudadanos. Como resultado, se pueden ver "*respeto derechos humanos", "oportunidad laboral jóvenes", "espacios participación ciudadanos", y "personas situación discapacidad".*

Figura 36: Tri-grams sin sistema de educación



4.2.4. Modelación LDA

Para la creación del modelo de LDA se debe tener en cuenta que se trabajará con dos parámetros. En primer lugar, el parámetro *alfa*, que representa la densidad de los temas de los documentos, por lo que un alfa más alto, supone que los documentos están formados por más temas y dan lugar a una distribución de temas más específica por documento. En segundo lugar, el parámetro beta, que representa la densidad por tema y palabra, por lo que una beta más alto, indica que los temas están formados por la mayoría de palabras y por ende una distribución de palabras más específicas por tema.

Para poder conocer los temas más relevantes, se emplea el archivo que se encuentra preprocesado, y al igual que los *N-grams*, se eliminan los verbos modales con el fin de eliminar el ruido que estos puedan generar. Finalmente, se toma la columna a analizar y es convertida a lista.

Con el fin de realizar este modelo, se requiere una preparación del texto para el análisis con LDA, por lo que se debe transformar el texto a una lista de texto en tokens y luego convirtiéndolo el objeto tokenizado en un corpus y un diccionario.

Por medio de la librería *gensim* se crea el modelo LDA, donde se mantiene los parámetros por defecto a excepción del número de temas, que para el análisis exploratorio se define como 10, en el cual cada tema será la combinación de palabras claves, por lo que cada una de estas palabras contribuye a un peso ponderado en un tema determinado.

Luego, para visualizar en la figura 37, estas palabras claves y sus pesos en los tópicos se utiliza la librería pprint. En la siguiente imagen se puede ver el resultado esperado, donde el tema 0 está representado por las palabras como “*corrupción”, “país”, “sentir”, “malo”, “cuidar”, “persona”, “casa”, “político”, “ley” y “via”*

Figura 37: Resultado LDA

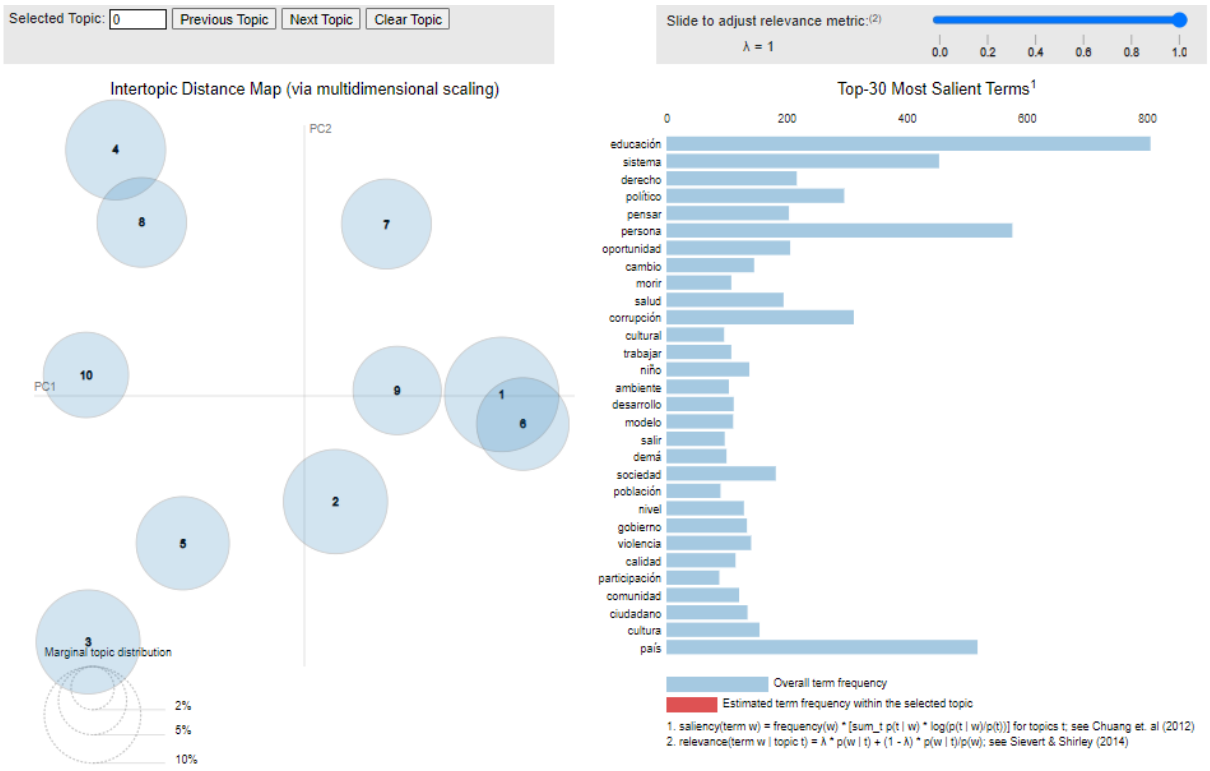
```
[(0,
  '0.030*"corrupción" + 0.027*"país" + 0.022*"sentir" + 0.019*"malo" + '
  '0.018*"cuidar" + 0.014*"persona" + 0.014*"casa" + 0.014*"político" + '
  '0.014*"ley" + 0.013*"vía"'),
 (1,
  '0.035*"derecho" + 0.028*"sistema" + 0.023*"recurso" + 0.021*"educación" + '
  '0.020*"calle" + 0.018*"servicio" + 0.017*"salud" + 0.017*"campo" + '
  '0.016*"justicia" + 0.015*"público"'),
 (2,
  '0.033*"cambio" + 0.032*"cultural" + 0.023*"violencia" + 0.023*"querer" + '
  '0.021*"cultura" + 0.020*"país" + 0.018*"conflicto" + 0.017*"escuchar" + '
  '0.015*"aprender" + 0.012*"colombia"'),
 (3,
  '0.035*"sistema" + 0.033*"morir" + 0.027*"salud" + 0.025*"persona" + '
  '0.019*"cumplir" + 0.018*"quedar" + 0.017*"educación" + 0.015*"colombia" + '
  '0.015*"acceder" + 0.011*"encontrar"'),
 (4,
  '0.035*"persona" + 0.023*"niño" + 0.018*"gente" + 0.017*"educación" + '
  '0.016*"vivir" + 0.015*"basura" + 0.015*"vida" + 0.015*"pueblo" + '
  '0.014*"común" + 0.014*"atención"'),
 (5,
  '0.037*"persona" + 0.036*"pensar" + 0.025*"político" + 0.024*"trabajar" + '
  '0.024*"política" + 0.023*"ciudadano" + 0.022*"demá" + 0.020*"participación"
  '+ 0.019*"comunidad" + 0.016*"ayudar"'),
 (6,
  '0.032*"oportunidad" + 0.030*"político" + 0.029*"educación" + '
  '0.023*"ambiente" + 0.023*"sistema" + 0.022*"social" + 0.020*"país" + '
  '0.018*"persona" + 0.017*"desigualdad" + 0.016*"corrupción"'),
 (7,
  '0.091*"educación" + 0.029*"país" + 0.026*"sistema" + 0.021*"sociedad" + '
  '0.021*"nivel" + 0.021*"desarrollo" + 0.021*"modelo" + 0.020*"calidad" + '
  '0.017*"derecho" + 0.016*"valor"'),
 (8,
  '0.020*"llegar" + 0.018*"animal" + 0.017*"respeto" + 0.017*"falta" + '
  '0.015*"corrupción" + 0.015*"persona" + 0.015*"dejar" + 0.014*"frente" + '
  '0.014*"caso" + 0.013*"importar"'),
 (9,
  '0.023*"salir" + 0.023*"gobierno" + 0.022*"población" + 0.018*"futuro" + '
  '0.014*"respetar" + 0.014*"creer" + 0.014*"difícil" + 0.012*"paz" + '
  '0.012*"experiencia" + 0.011*"atender"')]
```

Finalmente, para la visualización del modelo y así poder realizar un análisis e interpretación adecuada, se requiere la instalación de *pyLDAvis* de la librería *gensim_models* y se debe realizar la creación de una carpeta en la cual se almacenará archivos pickles con un archivo pre-preparado y un archivo html que es grafico interactivo que permite la interpretación de cada uno de los temas y la relación de las palabras con cada tópico.

En las siguientes ilustraciones, se puede seleccionar de forma manual un tema específico para conocer los términos más frecuentes y su relevancia cuando λ = 1 o cuando λ = 0.4. Este tipo de configuración se utiliza para asignar un nombre interpretable al tema. Dentro del grafico generado con pyLDAvis, se puede observar las 30 palabras más relevantes por tópico, que, al seleccionar un tema en específico, muestra cuales son las más frecuentes dentro de ese tema.

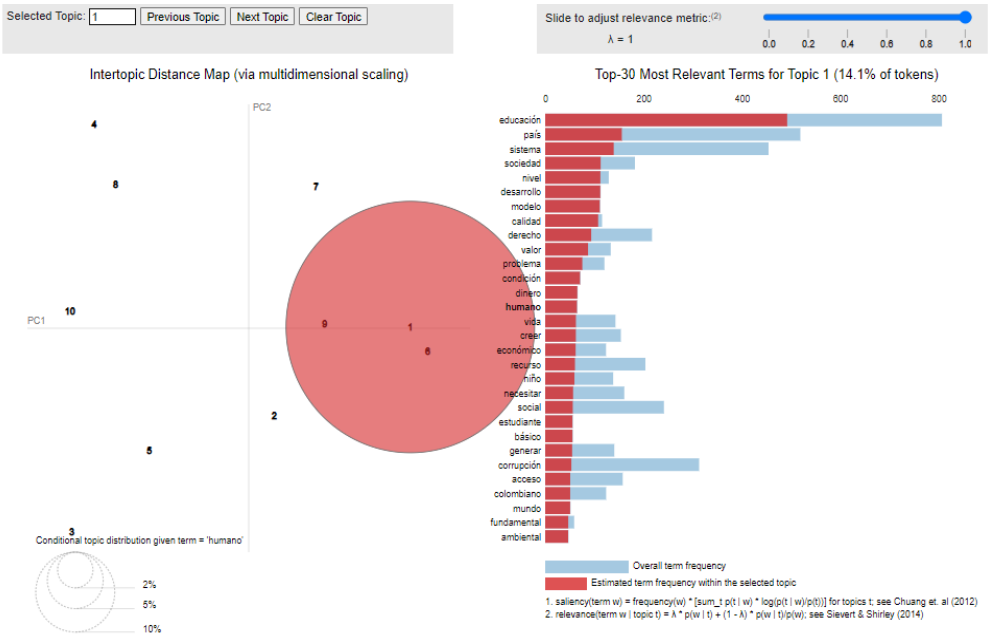
En el lado izquierdo, el área de cada círculo representa la importancia del tema en relación con el corpus y la distancia entre el centro de los círculos indica la similitud entre los temas (ver figura 38).

Figura 38: Modelo LDA



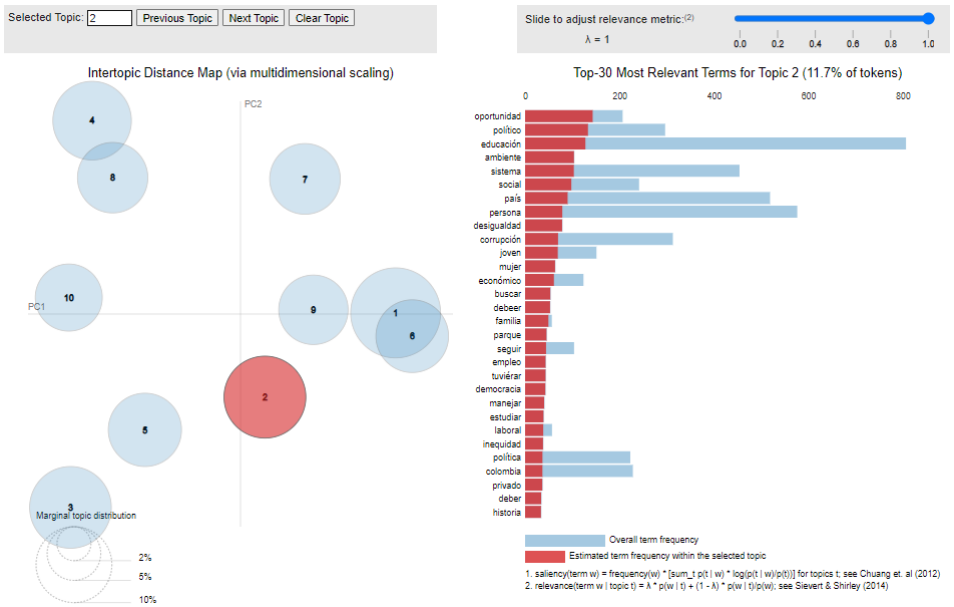
Como se puede observar en la figura 39, para el tópico 1, la palabra "*educación"* tiene la mayor relevancia dentro de los tokens, seguido por los tokens "*país"* y "*sistema".*

Figura 39: Modelo LDA tópico 1



Ya para el tópico número 2, se pueden empezar a ver el top de palabras cambia al igual que su relevancia para el tópico, ya que, dentro de este tópico, se puede observar que tokens como "*oportunidad", "político", "educación", "ambiental", y "sistema"* se encuentra dentro del top 5 de este tópico (ver figura 40).

Figura 40: Modelo LDA tópico 2

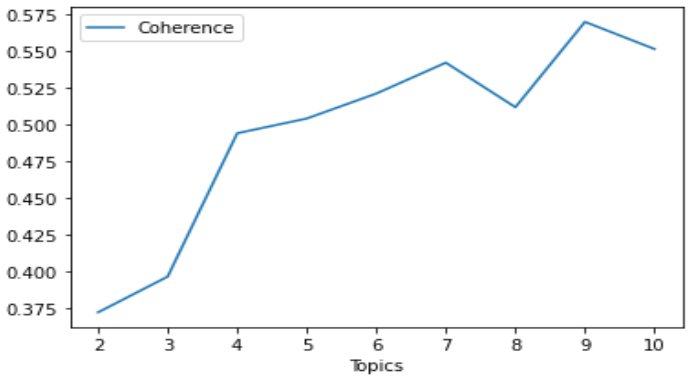


Como resultado se puede ver que al mantener los parámetros por defecto puede no generar los tópicos más importantes, por lo que para evidenciar si el número de tópicos seleccionados es el adecuado, se realiza una prueba de coherencia, donde se muestra que la coherencia de este modelo es de 0.3514.

La prueba de coherencia ayuda a medir el grado de similaridad semántica entre las palabras con mayor puntaje en los temas, y así distinguir si los temas son interpretables semánticamente. Como se explicó en el marco teórico, la medida de coherencia que se utiliza en esta tesis es *C_v* que revisa la similitud del coseno de las palabras más importantes del tema.

Para mejorar este modelo y tener mejores tópicos, se realiza un ajuste en los hiperparámetros, como lo es el número de temas (K), la densidad del documento-tema (alpha), y la densidad de la palabra-tema (beta). Para esto se desarrolla una función en la cual se realizarán pruebas en secuencia, cambiando un parámetro a la vez mientas que los otros dos se quedan constantes y así encontrar la combinación que más favorezca al modelo (ver figura 41).

Figura 41: Coherence LDA



A partir de la puntuación de coherencia, se puede apreciar que el *C_v* tiene un pico en número de temas 9. Luego se debe evaluar cuáles serán los valores de alfa y beta a utilizar. Y con estos nuevos parámetros se realiza un nuevo modelo LDA el cual tiene una coherencia de 0.5607 si se utiliza Alpa= Asymmetric y Beta= 0.31 (ver figura 42).

Figura 42: cv_puntaje

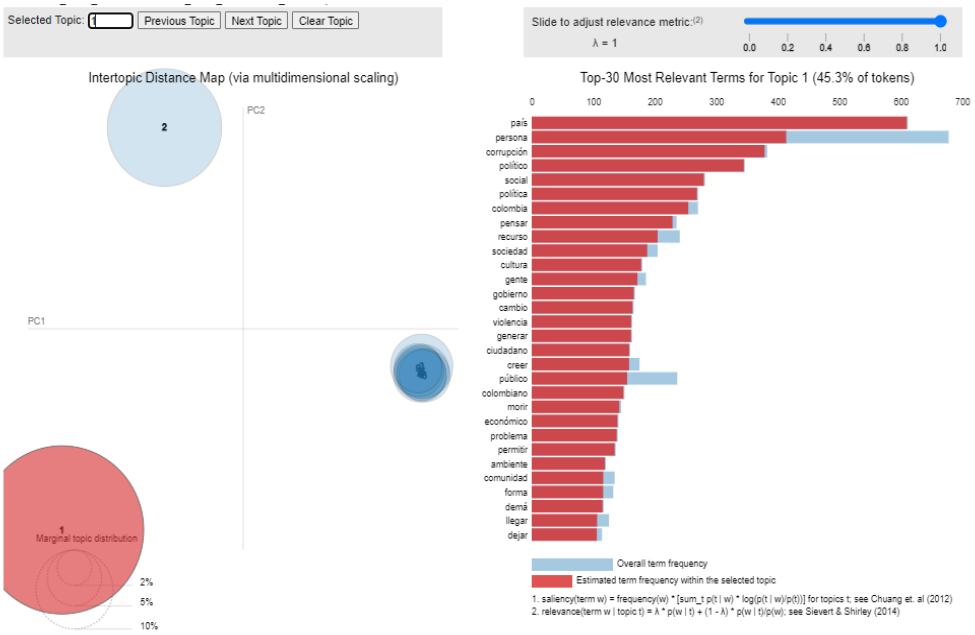
	Validation_Set	Topics	Alpha	Beta	Coherence
482	100% Corpus	9	0.01	0.61	0.470218
217	75% Corpus	9	0.31	0.61	0.460422
223	75% Corpus	9	0.61	0.9099999999999999	0.423929
228	75% Corpus	9	0.9099999999999999	0.9099999999999999	0.416944
236	75% Corpus	9	asymmetric	0.31	0.569741
502	100% Corpus	9	symmetric	0.61	0.453494

	Validation_Set	Topics	Alpha	Beta	Coherence
485	100% Corpus	9	0.31	0.01	0.388428
236	75% Corpus	9	asymmetric	0.31	0.569741
507	100% Corpus	9	asymmetric	0.61	0.534966
488	100% Corpus	9	0.31	0.9099999999999999	0.439529
509	100% Corpus	9	asymmetric	symmetric	0.450981

Como resultado, se puede observar que el modelado de tópicos cambió significativamente. En particular, la frecuencia de los ítems dentro de cada tópico tiene una mayor importancia, haciendo que los temas puedan ser más concretos y más intuitivos para los analistas.

En la figura 43 se puede visualizar que los tokens más significativos son *"país", "persona", "corrupción", "político" y "social".*

Figura 43: LDA k= 9 tópico 1



En las figuras 44 y 45 de LDA, se puede observar como a pesar de que se los temas son parecidos al estar estos solapados, los tokens y sus frecuencias cambian, ya que para el tópico 3, se tiene como top 5 los tokens como *"pensamiento", "atención", "parque",* "comunicación" y "escuchar", mientras que para el tópico 5 los tokens más significativos son *"futuro", "seguridad", "policía", "reconocer" y "leticia".*

Figura 44: LDA k=9 tópico 3

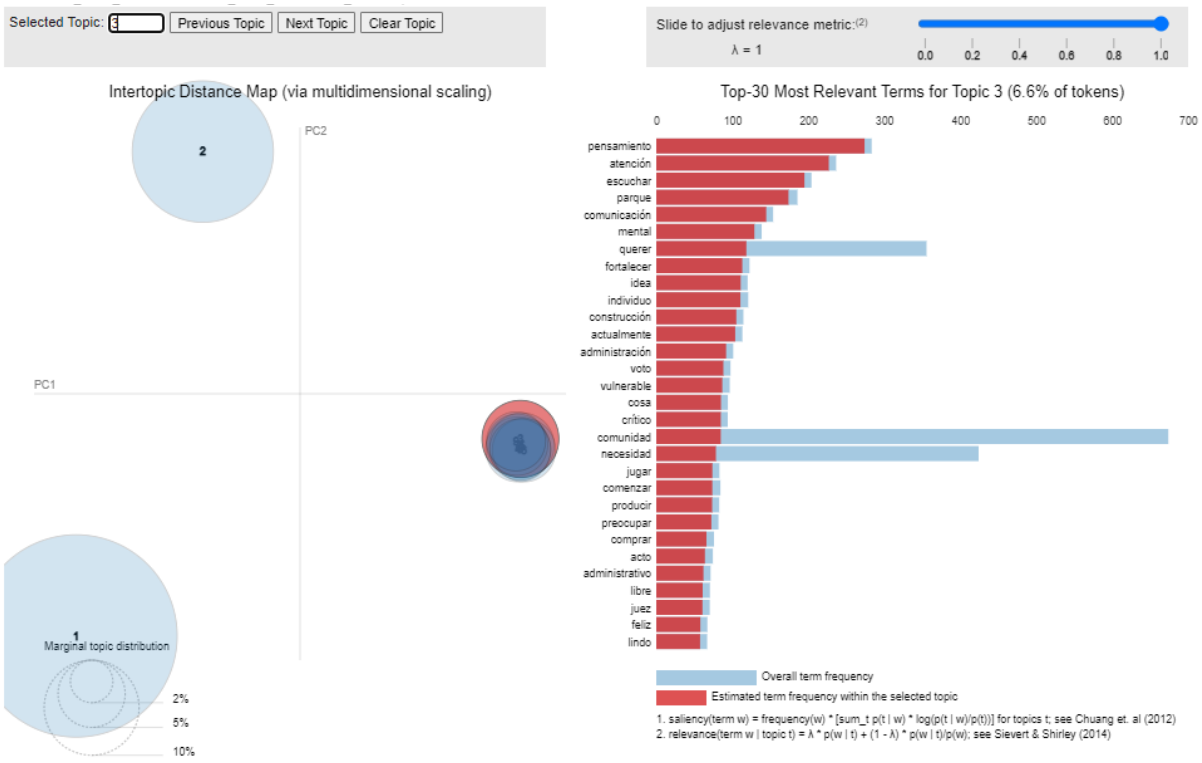
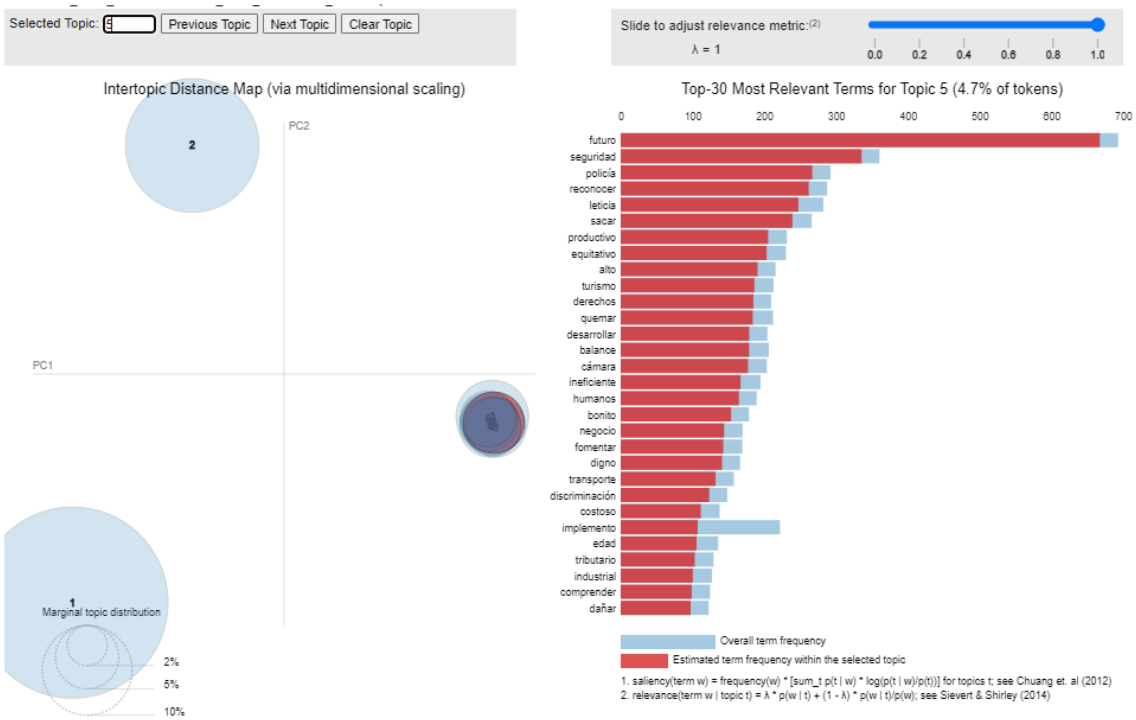


Figura 45: LDA k=9 tópico 5



El top 10 de palabras más relevantes para los tópicos se presentan en la tabla 18.

Tabla 18: Tópicos LDA

Tópico - % Tokens	Palabras
1 – 45.3%	País, persona, corrupción, político, social, política, Colombia, pensar, recurso, sociedad
2 – 22.1%	Educación, sistema, persona, salud, derecho, oportunidad, acceso, niño, vida, joven
3 – 6.6%	Pensamiento, atención, escuchar, parque, comunicación, mental, querer, fortalecer, idea, individuo
4 – 4.7%	Pagar, acabar, valer, plata, enfocado, herramienta, avanzar, eps, ciencia, inseguridad
5 – 4.7%	Futuro, seguridad, policía, reconocer, leticia, sacar, productivo, equitativo, alto, turismo
6 – 4.5%	Mujer, violar, razón, familia, momento, recibir, mecanismo, ejercer, promesa, genero.
7 – 4.2%	Vivir, real, imagen, árbol, escuela, vender, rio, respirar, bomba, líder
8 – 4.1%	Ayudar, estudiante, agua, colegio, ayuda, juventud, expresión, personal, humanidad, maltrato
9 – 3.9%	Querer, corrupto mentalidad, vía, suficiente, solución, unión, problemática, abierto, objetivo

4.2.5. Wordcloud

Para la construcción de la nube de palabras, se utiliza la librería *Wordcloud*, la cual solo requiere indicar el texto y el número de palabras que se quieren graficar. Por tanto, se toma el texto_1 y como se observa en la figura 46, se genera un gráfico con las primeras 100 palabras con mayor frecuencia, entre más uso se tiene de una palabra, el grafico la mostrará con una fuente mayor y un color más intenso, por lo que palabras como *"país",*

“educación”, “persona”, “corrupción”, “político”, “Colombia”, “sistema educativo” son palabras que tienen la mayor frecuencia dentro de las respuestas de los ciudadanos.

Figura 46: Wordcloud

4.2.6. Análisis de sentimientos

Para el análisis de sentimientos, se realizará un estudio sobre el conjunto de datos completo y el conjunto de datos solamente con la etiqueta CN (consecuencia negativa) con el fin de identificar si existe alguna relación entre esta etiqueta y el sentimiento negativo. Como se pudo revisar en el numeral 4.2.1 este indicador verbal representa el 26.73% del conjunto de datos, por lo que aparece en 1274 respuestas.

Adicional a esto, se realizará una comparación entre los archivos *Lemma_no_Stopwords*, y el conjunto de datos *Lemma*, con el fin de identificar si los Stopwords afectan este tipo de modelo. Como se explicó en el capítulo 3, se realizará análisis de sentimientos por medio de *TextBlob, Vader y AWS comprehend*.

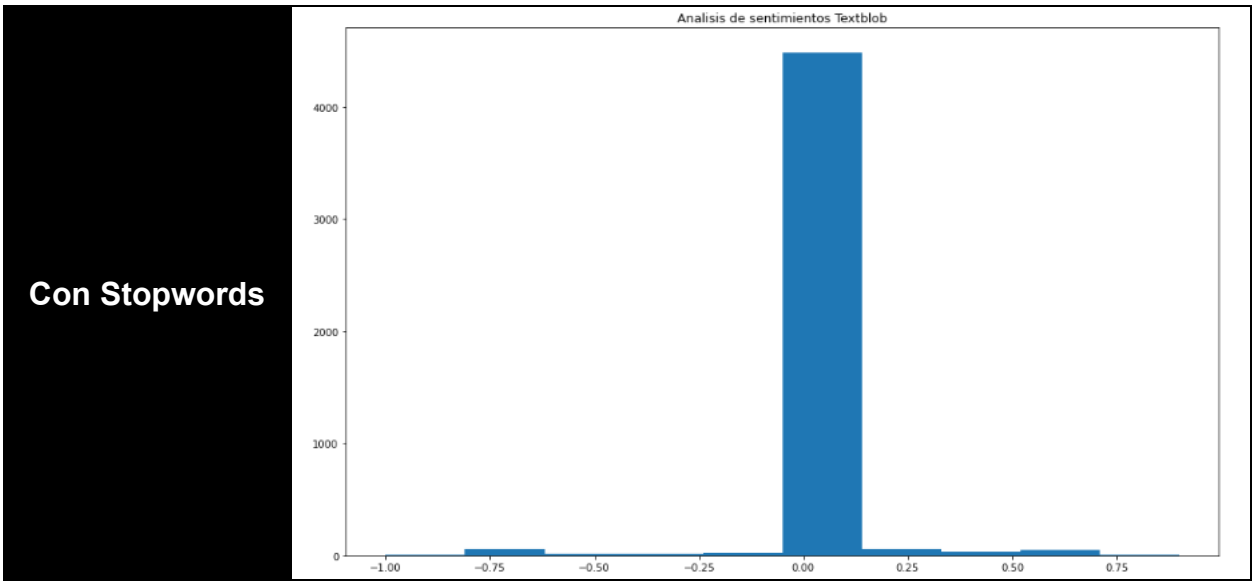
4.2.6.1. TextBlob

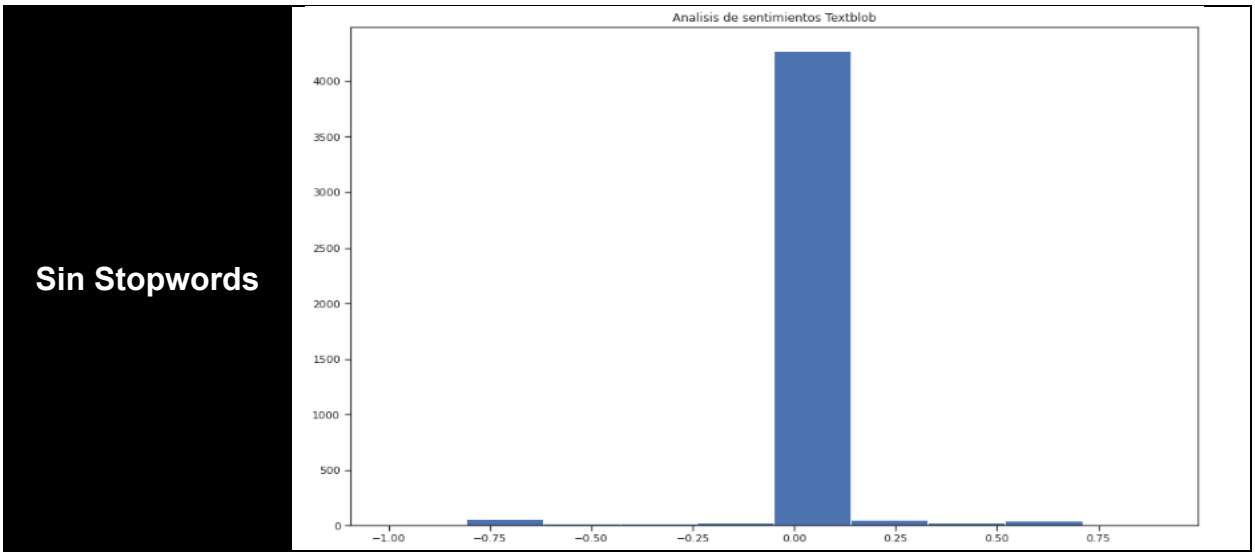
Para la creación del modelo de análisis de sentimientos y así conocer si las respuestas dadas por los participantes son positivas, negativos o neutrales se utilizará la librería *TextBlob* de *NLTK* de Python. En este ejercicio se estudiarán dos propiedades:

- Polaridad: se refiere a un número decimal que se encuentra entre -1 y 1, donde 1 es positivo y -1 significa negativo.
- Subjetividad: se refiere si el tema está basado en las opiniones personales o en los sentimientos, este también se encuentra representado por un numero decimal que se encuentra ente 0 y 1.

Por medio de un histograma, que se puede visualizar en a figura 47, se aprecia que la gran mayoría de las respuestas se encuentran en un nivel neutro en ambos conjuntos de datos.

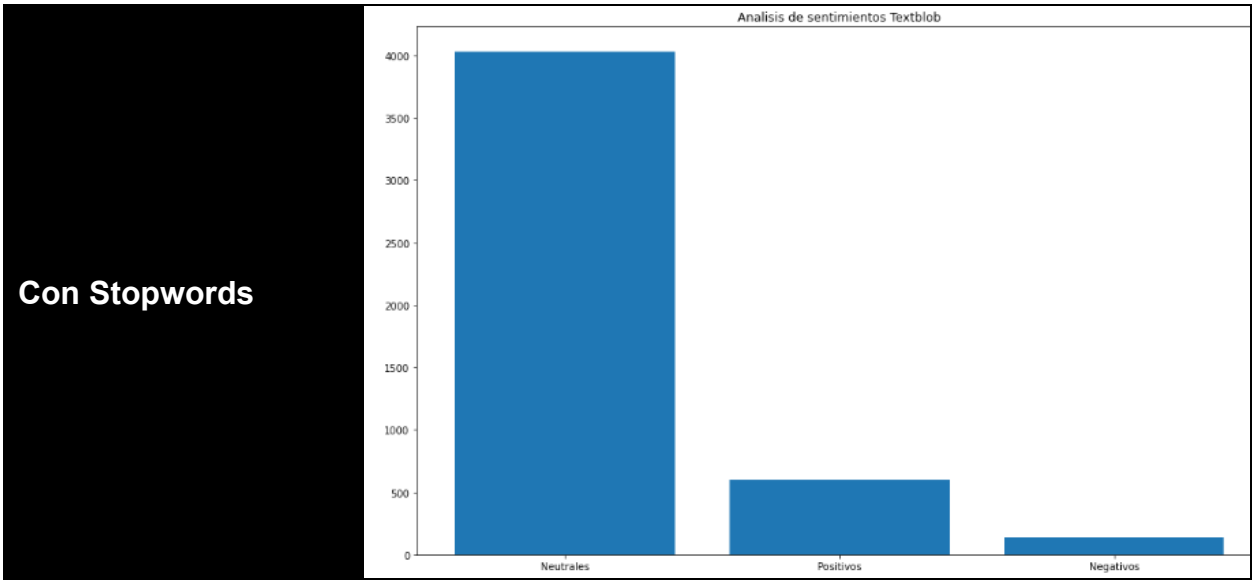
Figura 47: Análisis de sentimientos Textblob

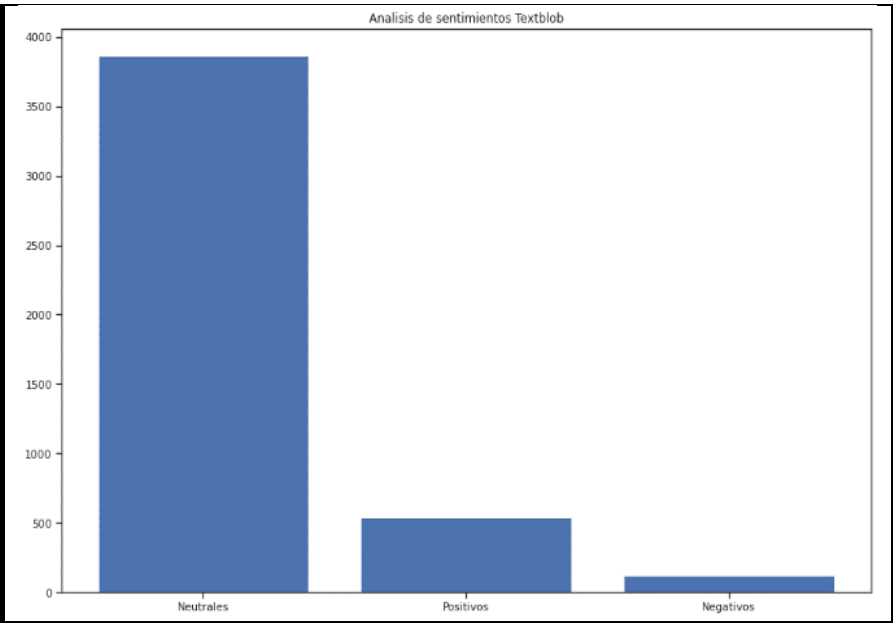




Para tener un mayor entendimiento sobre los sentimientos, se construye una nueva columna en la cual si el puntaje de polaridad es menor a 0 se agregue la etiqueta Negativo, cuando este sea igual a 0, que este sea Neutral o Positivo si es mayor estrictamente a 0. En el siguiente gráfico podemos observar cómo el 85% de las respuestas el modelo las está tomando como neutrales, y solamente 12% positivas y 2.8% negativas en ambos conjuntos de datos (ver figura 48).

Figura 48: Análisis de sentimientos



Sin Stopwords	

Al realizar la revisión de las respuestas en cada sentimiento en los dos conjuntos de datos, se puede evidenciar que el cambio es mínimo, por lo que se demuestra que el análisis de sentimientos con la librería *Textblob* no tiene un gran impacto el utilizar un conjunto de datos sin *Stopwords*. A continuación, se puede observar algunos ejemplos de las respuestas de cada participante en cada sentimiento en la tabla 19.

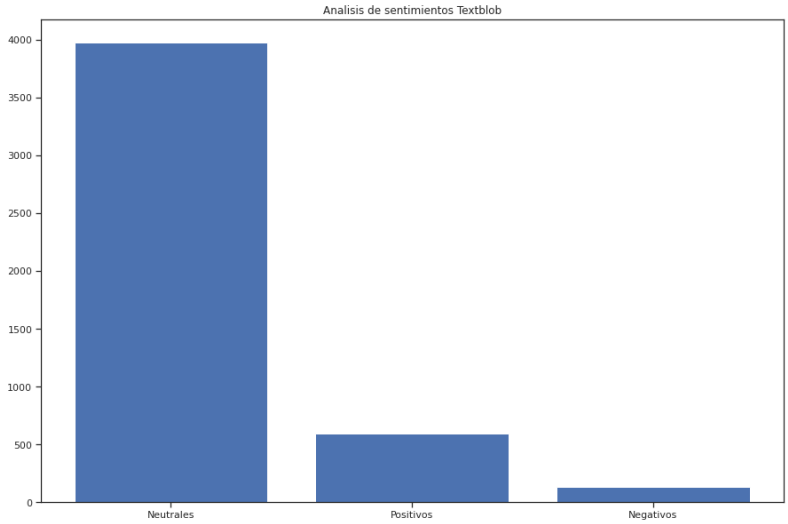
Tabla 19: Ejemplo Resultado análisis de sentimientos TextBlob

Sentimiento	Con Stopwords	Sin Stopwords
Positivo	▪ deber haber uno pacto social real para que estar en uno país más equitativo e incluyente porque no ser uno país equitativo e incluyente ▪ diversidad y riqueza cultural él deber resaltar el cultura de cada uno de el región el problema ser que a veces no aceptar el diferencia que tanto yo caracterizar ▪ sistema de elección para el cargo de elección popular para establecer regla claro en el período de el que él elegir y el puesto necesario en cada cargo	▪ deber pacto social real país equitativo incluyente ▪ sistema elección cargo popular establecer regla período elegir puesto necesario ▪ matriz cultural referente cultura patriarcal violento

Neutral	▪ voto voluntario por el voto obligatorio porque si uno revisar el abstención en el elección para el legislativo y el ejecutivo encontrar que en el último elección de cada colombiano habilitado para votar votar y no él hacer ▪ manera en el que el colombiano agenc nuestro interés individual y colectivo porque no existir mucho espacio en el que el persona él poder encontrar con el demá en interés común ▪ cobertura a nivel educativo como ver en país desarrollado como singapur a medida que él adoptar uno proceso educativo para todo el persona tener mayor capacidad de oferta laboral y condición económico que generar mejor calidad de vida	▪ voto voluntario obligatorio revisar abstención elección legislativo ejecutivo encontrar colombiano habilitado votar ▪ colombiano agenc interés individual colectivo existir espacio persona encontrar demá común ▪ cobertura nivel educación país desarrollado singapur medida adoptar proceso persona capacidad oferta laboral condición económico generar calidad vida
Negativo	▪ visión social de el educación nuestro educación deber en el capacidad de el persona en su talento innato sin tener en cuenta el desigualdad de base ▪ corazón uno gran parte de el población ser mucho vulnerable y ser importante que uno ejercicio de solidaridad salir del corazón ▪ base emocional porque tener uno historia que yo poner en uno espacio emocional de venganza de guerra de arrogancia de dolor y desde ahí él conversar y él actúar en todo el ámbito	▪ visión social educación capacidad persona talento innato desigualdad base ▪ corazón población vulnerable importante ejercicio solidaridad salir ▪ base emocional historia espacio venganza guerra arrogancia dolor conversar actuar ámbito

Finalmente, se busca revisar si la etiqueta CN afecta el sentimiento de los participantes. Así pues, se crea un conjunto de datos donde su indicador verbal sea CN únicamente, y con esto podemos evidenciar que utilizando la librería *TextBlob,* Como se puede observar en la figura 49, esta etiqueta no tiene un mayor impacto para el sentimiento negativo, ya que solo el 2.8% tienen este tipo de sentimiento.

Figura 49: Análisis sentimiento Textblob CN



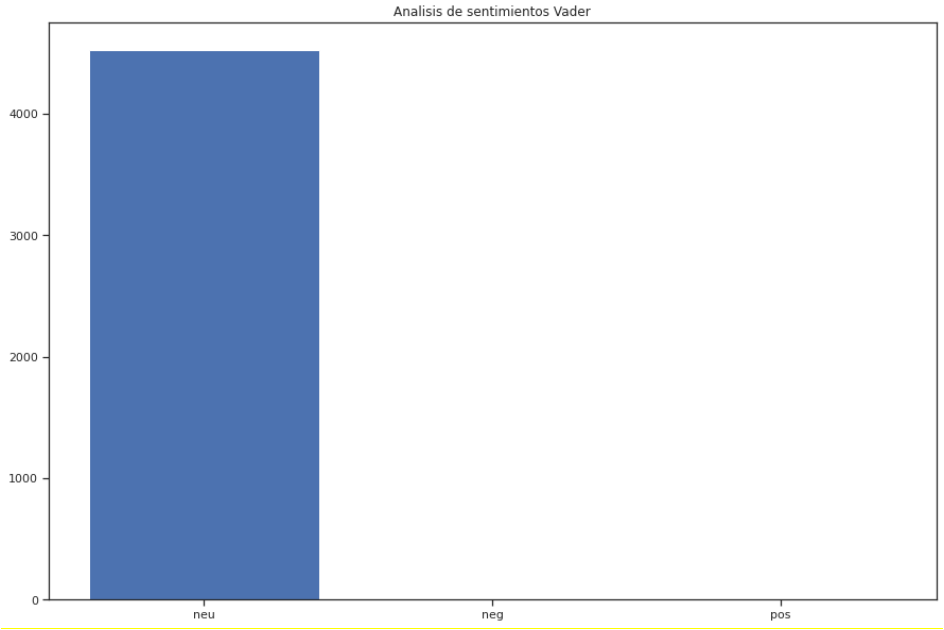
4.2.6.2. Vader

Vader es una librería que es utilizada para el análisis de sentimientos, especialmente en la detección de sentimientos negativos. En particular, este modelo devuelve un diccionario que contiene probabilidades del texto de ser positivo, negativo o neutro.

Para la creación del modelo se debe descargar *NLTK.download('vader_lexicon')* *sid* = *SentimentIntensityAnalyzer()* Ya que estos son necesarios para la configuración. Luego es necesario la definición de una función en la cual se buscará la creación del puntaje de polaridad.

Para poder visualizar los resultados, se crea un histograma que se puede observar en la figura 50, el cual nos muestra que solo existe cinco respuestas que es considerada con un sentimiento negativo, mientras que el 94% de las respuestas son tomadas como neutras:

Figura 50: Análisis de sentimientos Vader



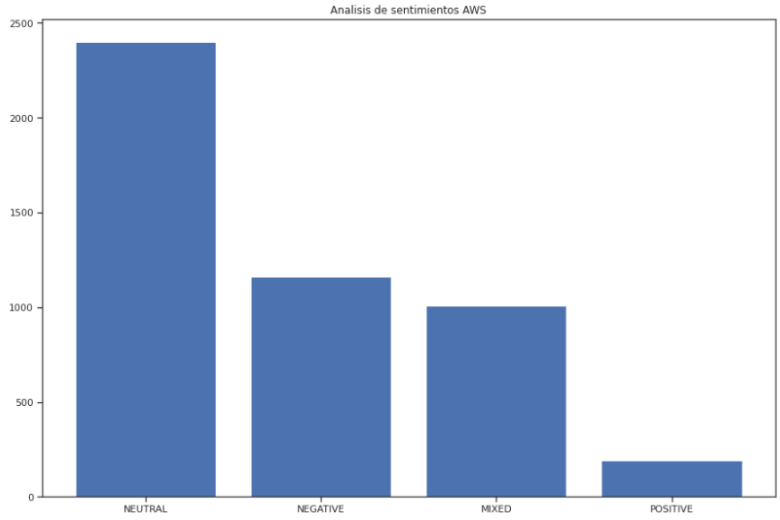
Las respuestas que son detectadas como negativas son: *"institucionalidad crisis enfrentar erradicar corrupción", "nación país atraso terrible", "paz momento crisis", "salud pandemio preparado crisis sanitario", "delincuencia error país"* las cuales tienen palabras como crisis, corrupción terrible y error que son palabras que ayudan a detectar este tipo de sentimientos en Vader.

4.2.6.3. AWS comprehend – Detect sentiment

Con el fin de tener un modelo que tenga mejor precisión para el idioma español, se utiliza API de AWS para utilizar los servicios de *Comprehend*, específicamente para *detect_sentiment*, para esto se realiza la instalación de varias herramientas como *boto3, awscli, jedi, urlib3 V 1.25.4.* Una vez instalados todos los paquetes se procede a crear la función de análisis de sentimientos con el código de lenguaje *'es'* (Abreviación de español en AWS).

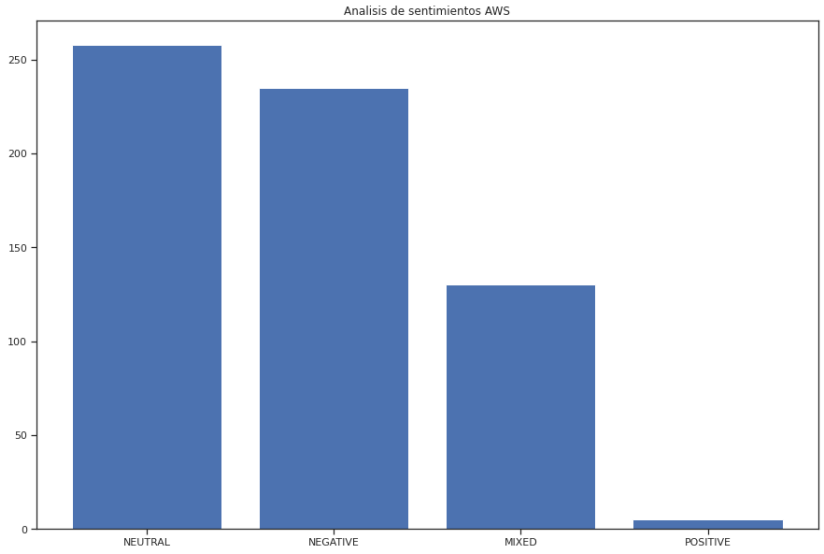
Como resultado se puede observar en la figura 51 que el 50% de las respuestas tienen un sentimiento neutro, pero a diferencia de las librerías Textblob y Vader, existe un 24% de respuesta que son negativas y 21% como mixtas, dejando solo un 4% como positivas.

Figura 51: Análisis de sentimientos AWS



Buscando su relación con la etiqueta CN (Consecuencia Negativa) que evaluamos al principio, se puede observar en la figura 52 que aumenta el porcentaje de respuestas negativas a un 37% mientras que las neutrales bajan a un 41%, por lo que se puede observar que esta etiqueta si afecta el sentimiento de las respuestas dadas por los ciudadanos.

Figura 52: Análisis de sentimientos AWS con CN



Algunos ejemplos de las respuestas etiquetadas como CN dentro del análisis de sentimientos son los mostrados en la tabla 20.

Tabla 20: Análisis de sentimiento AWS CN

Sentimiento	Respuestas
Neutra	▪ financiación del sistema educativo actualmente haber uno reforma educativo donde él recargar a el municipio en materia presupuestal y el municipio no contar con el recurso suficiente para brindar uno servicio de calidad frente a el educación básico y medio ▪ sistema educativo ser necesario cambiar el sistema educativo pues en el funcionamiento del sistema actual mucho no poder ingresar a el educación universitario el cupo en el universidad no ser sificient y tampoco él él ofrecer otro oprtunidad de preparación
Negativo	▪ enfoque desarrollisto porque ser uno modelo que privilegiir capital de transnacional y que dejar a colombia con uno economía de explotación de materia primo ▪ manera de mostrar nuestro país el culpa de que el extranjero tener miedo a conocer cosa en colombia él tener el colombano por el malo forma en el que haber mostrar el país
Mixto	▪ sistema de funcionamiento de el educación y salud porque él basar en el recaudo de consumo de licor cigarrillo y otro impuesto y este hacer que el departamento dependar de el vicio ▪ participación de el juventudes en territorio en tema de paz el joven querer dar su opinión y participar pero haber ser mucho difícil

Positivo	▪ planeación y el ejecución de el proyecto en el institución planear mucho y hacer mucho burocracia y ya todo el entusiasmo él gastar en planear y al pasar a el acción él acabar el energía ▪ cultura facilista nuestro cultura facilista hacer que querar ser millonarios de uno momento a otro

Con estos ejemplos se puede ver que existe una mala interpretación del sentimiento positivo, debido a que, por palabras como planeación, ejecución, facilista, cultura, se toman como positivas cuando estas se podrían clasificar como negativas (intuitivamente). En consecuencia, al finalizar este modelo se debe realizar una revisión más profunda sobre las respuestas dadas, ya que dentro de la documentación de AWS nos muestra que el modelo de análisis de sentimientos tiene un *acurracy* aproximado del 56%.

4.2.7. POS Tagging

El análisis de discurso por medio de POS Tagging, se utiliza con el fin de conocer o etiquetar las partes de sintaxis de las palabras que conforman una oración. Esto llega a ser muy útil en el momento de conocer cuáles son los verbos, adjetivos, pronombres, sustantivos que se utilizan por indicador verbal, por sentimientos, o por variable demográfica.

Dentro de este análisis se revisará los conjuntos de datos preprocesado con las nuevas columnas de análisis de sentimientos, con el fin de identificar aquellos verbos, adjetivos, pronombres y sustantivos del sentimiento negativo que fue detectado por *AWS Comprehend.* Dentro de este análisis, se comparará el modelo de spaCy contra AWS *batch_detect_syntax.*

4.2.7.1. spaCy aplicado POS Tagging

Para la construcción de este modelo, es necesario la instalación de *spaCy* al igual que su modelo en español *es_core_news_md*. Después se construyen dos funciones, las cuales servirán para crear la etiqueta en las palabras, y el conteo de cada etiqueta por palabra.

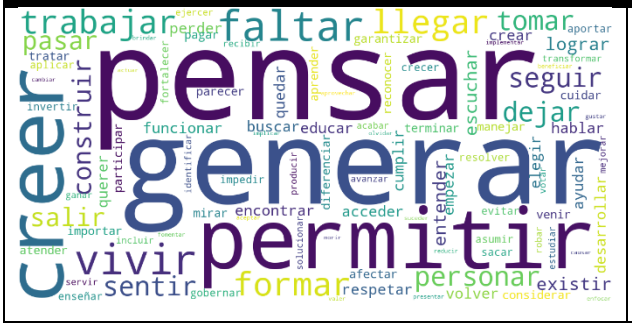
El resultado esperado es la creación de dos columnas nuevas las cuales tendrá el número de palabras con la etiqueta seleccionada, y una columna que muestra las diferentes palabras. Esta última columna será utilizada para los diferentes análisis que se requieran hacer (ver figura 53).

Figura 53: POS Conjunto de datos con Verbos

	texto_1	num_verb	verbos
0	voto voluntario obligatorio revisar abstención...	3	revisar, encontrar, votar
1	colombiano agenc interés individual colectivo ...	3	existir, personar, encontrar
2	visión social educacióncapacidad persona talen...	1	personar
3	cobertura nivel educación país desarrollado si...	2	adoptar, generar
4	sistema educación aprendizaje significativo pe...	1	elegir

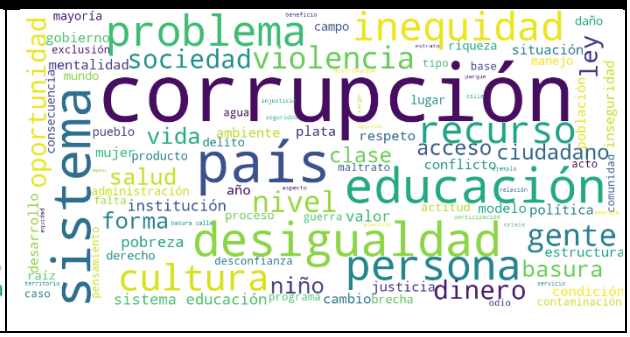
En la tabla 21 se puede observar los verbos más utilizados en conjunto de datos con *"pensar", "generar", "permitir", "creer",* y en la parte izquierda los verbos más utilizados cuando se limita el conjunto de datos a aquellas respuestas con sentimiento negativo, donde los verbos como *"faltar", "dejar", "perder" y "morir"* toman mayor relevancia.

Tabla 21: POS spaCy Verbos

Conjunto de datos completo	Sentimiento Negativo
	

Se realiza el mismo análisis, pero con los sustantivos, con el fin de conocer cuáles son los términos más utilizados por los ciudadanos. Se puede observar un gran impacto en el subconjunto de datos con sentimiento negativo, donde los sustantivos con mayor frecuencia son "*corrupción", "desigualdad", "problema", "inequidad", "sistema", "educación" y "país",* mientras que en el conjunto de datos completo se observa *"país", "sistema", "educación", "persona", "cultura", "oportunidad"* (ver tabla 22).

Tabla 22: POS spaCy Sustantivos

Conjunto de datos completo	Sentimiento negativo
	

Como tercer análisis de POS se revisa los pronombres utilizados (ver tabla 23), con el fin de identificar localizaciones, responsables, actores principales en las respuestas de los ciudadanos, en el cual también existen palabras comunes en los dos conjuntos de datos, tales como *"Colombia", "gobierno", "congreso", "constitución", "ciudad" "país" y*

“comunidad”. Esto nos indica que sin importar el sentimiento que tenga el ciudadano, los actores principales son los mismos.

Tabla 23: POS spaCy Pronombres

Conjunto de datos completo	Sentimiento Negativo
	

Finalmente, se realiza un análisis descriptivo del uso de verbos, sustantivos, adjetivos y pronombres en ambos conjuntos de datos, en el cual se puede ver en promedio por respuesta los ciudadanos pueden utilizar 2 verbos y 3 sustantivos, a pesar de que existen datos atípicos donde existen respuestas que utilizan un total de 10 verbos y 12 adjetivos.

También se puede observar en la tabla 24, un cambio con el número de verbos utilizados en las respuestas de sentimiento negativo, donde la mediana nos indica 1 verbo por respuesta y 3 sustantivos, lo que hace que estas respuestas sean más cortas o concisas.

Tabla 24: Análisis descriptivo POS

Conjunto de datos completo

	num_verb	num_adj	num_sus	num_np
median	2.000000	2.000000	3.000000	0.000000
mean	1.877592	2.386414	3.640935	0.480591
amin	0.000000	0.000000	0.000000	0.000000
amax	10.000000	12.000000	15.000000	8.000000

Sentimiento Negativo

	num_verb	num_adj	num_sus	num_np
median	1.000000	2.000000	3.000000	0.000000
mean	1.666667	2.266932	3.237716	0.414343
amin	0.000000	0.000000	0.000000	0.000000
amax	7.000000	12.000000	12.000000	5.000000

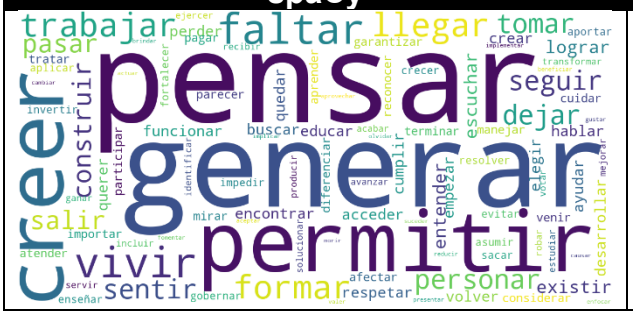
4.2.7.2. AWS comprehend – Batch detect syntax

Para la revisión de sintaxis de las respuestas de los ciudadanos, se puede utilizar los servicios de *AWS Comprehend*, con la *API batch_detect_syntax.* Sin embargo, este servicio tiene una limitante donde solo se puede utilizar frases que tengan un máximo de 25 palabras, por lo que se hace un subconjunto de datos, donde se elimina el 1% del conjunto de datos.

Al igual que el análisis de sentimientos, se requiere realizar una instalación de *boto3, awscli, jedi, urllib3-1.25.* Luego de instalar todos estos paquetes, se debe construir una función en la cual se llame la API, y se le pueda asignar la etiqueta a cada una de las palabras. Se debe tener en cuenta que debido al tipo de output que este archivo genera, se debe traer la información donde se encuentre el TAG == a la etiqueta que se desee y el tamaño del diccionario varía de acuerdo al número de tokens que tenga la frase.

Por medio de una nube de palabras se busca ver las diferencias entre el modelo creado con spaCy y con AWS donde se puede observar en la tabla 25, que los verbos con mayor relevancia se mantienen, como lo son "*pensar", "generar", "permitir" "creer"* y que en AWS muestra algunos verbos que a pesar de estar considerados en spaCy no tienen la misma frecuencia como lo es *"forma", "vivir", "permitir".*

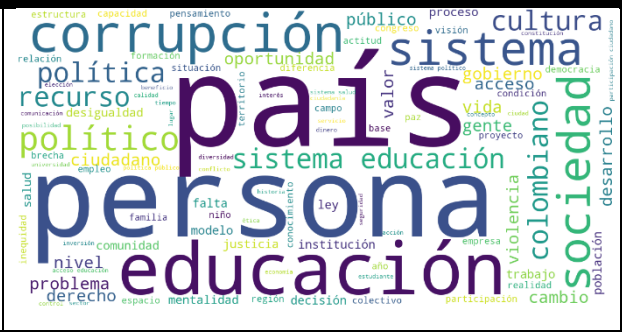
Tabla 25: AWS POS verbos

spaCy	AWS
	

Dentro de la comparación contra los sustantivos utilizados (ver tabla 26), se ve un leve cambio entre los tokens que se identificaron en spaCy y en AWS debido a que sigue

apareciendo con gran frecuencia palabras como *“educación”, “corrupción”, “sistema”, “país”, “persona” “cultura” y “político”*, pero en AWS se empieza “*persona*” tiene mayor importancia que “*educación*” al igual que “*país*” tiene mayor relevancia que “*sistema*”, adicional se pueden detectar nuevos como *“sociedad”, “cultura” y “recurso”.*

Tabla 26: AWS POS sustantivos

spaCy	AWS
	

Finalmente, en los pronombres se puede evidenciar un cambio significativo, donde aparecen nuevas palabras como “*afrodecendientes”, “microtrafico”, “bylling”, “lgtbiq*”, sin embargo, se observa en la tabla 27 que dentro de este modelo se encuentran también días de la semana, meses del año y las iniciales de este proyecto TQHC, haciendo que se pierdan varios actores, por lo que, para el análisis de pronombres, se obtienen mejores resultados el modelo creado con spaCy.

Tabla 27: AWS POS Pronombres

spaCy	AWS
	

4.2.8. Reconocimiento de entidades nombradas (NER)

Dentro del reconocimiento de entidades se busca encontrar entidades con las etiquetas personas, localidades y organizaciones. Para esto, se utilizará el conjunto de datos creado en el numeral 4.2.7 de este proyecto, con el fin de realizar nuevamente una comparación, de la eficiencia del modelo con un texto completo, y con la columna que contenga lo sustantivos detectados por *AWS comprehend* (ver tabla 28).

Tabla 28: Conjunto de datos NER

Tipo de archivo	Razón de creación	Nombre de archivo
Texto Completo	Se busca encontrar las entidades de las respuestas de los ciudadanos	*Respuestas_completas*
Sustantivos detectados por AWS	Se busca identificar las entidades en los sustantivos obtenidos por el proceso POS Tagging con AWS comprehend	*Sustantivos_aws*

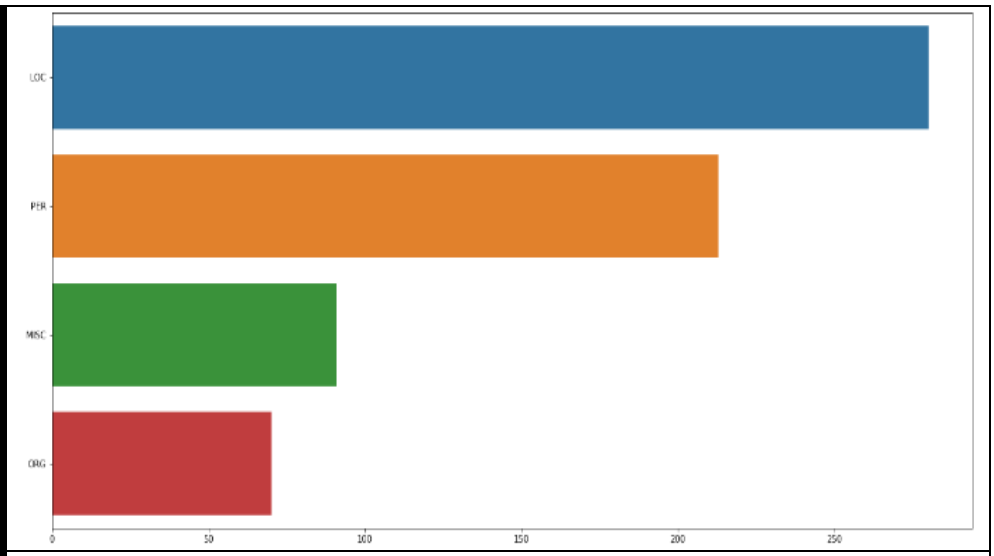
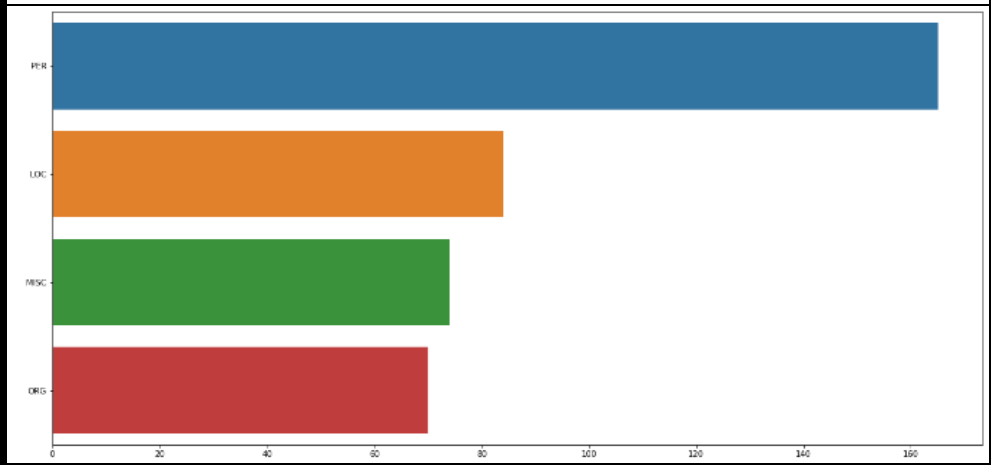
4.2.8.1. spaCy aplicado a NER

Para el modelo creado con *spaCy* se necesita instalar el paquete de español de spaCy *es_core_news_md* y se inicia el modelo *es_core_news_md.load().* Luego, es necesario crear una función en la cual se busca encontrar las entidades que se detectan en spaCy y el número de tokens para así poder visualizar su distribución en el conjunto de datos.

Por medio de *matplotlib*, se crea un gráfico de barras para así poder tener una mejor visualización de las diferentes entidades encontradas (ver tabla 29), donde tanto para el conjunto de datos *Respuestas_completas* y *Sustantivos_aws* se encuentran las

entidades de persona, localización, organización y *misc* que es una entidad que se da a aquellos tokens que no pueden ser clasificadas en alguna entidad.

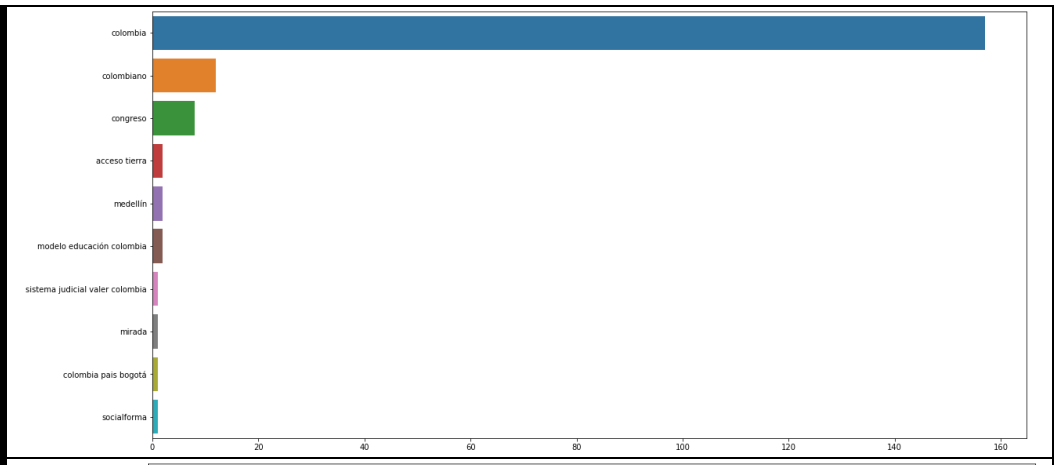
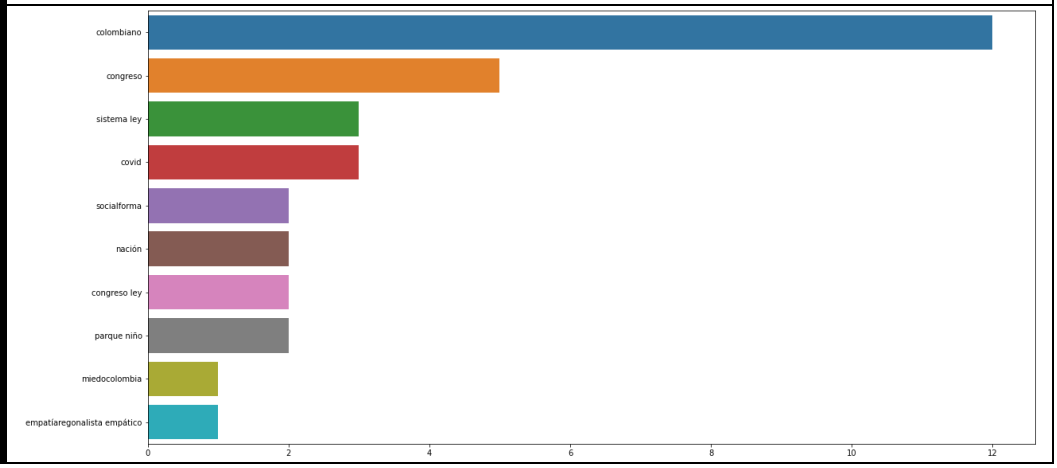
Tabla 29: Entidades spaCy

Respuestas completas	
Sustantivos aws	

Para la entidad 'LOC' se puede observar como para el texto completo más del 80% de las entidades se encuentra con el token "*Colombia*", luego es seguida por un porcentaje menor al 5% otros tokens como "*colombiano*", "*congreso*", "*Medellín*". Sin embargo, al revisar solamente el conjunto de datos *Sustantivos_aws*, se obtiene una mejor información ya que las localizaciones son "*Colombia", "congreso", "nación", "parque"* donde a pesar que "Colombia" tiende a mantenerse en el top 1, se puede obtener mayor información de las otras localizaciones. Cabe resaltar que, dentro de los resultados

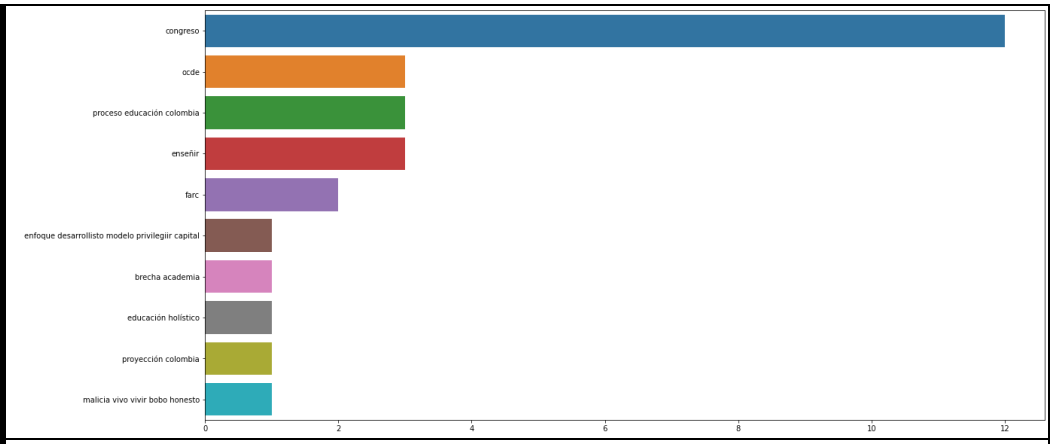
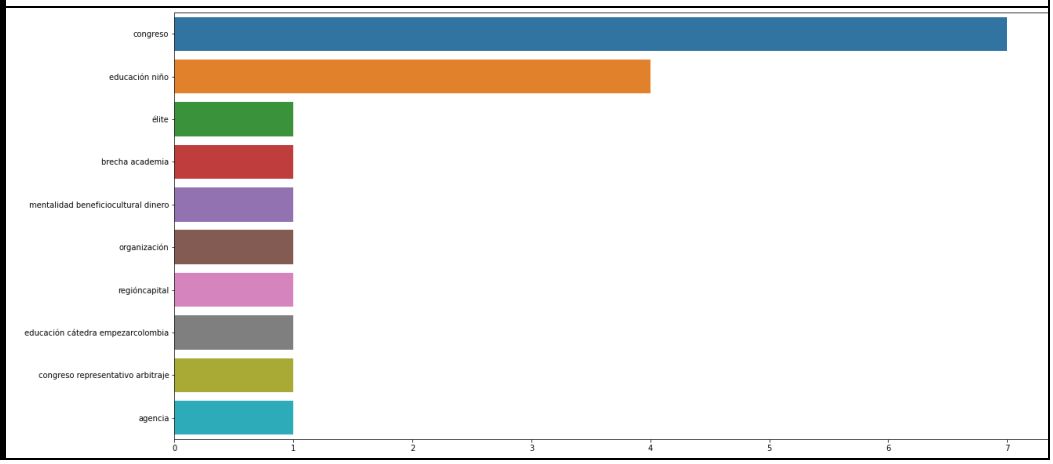
dados el conjunto de datos con los sustantivos, existen etiquetas que no son necesariamente LOC, como lo son "*COVID" y "empatía*", lo que nos indica un error dentro de este modelo (ver tabla 30).

Tabla 30: Diagrama NER LOC

Respuestas _completas	
Sustantivos _aws	

Dentro de las entidades identificadas como 'ORG' (ver tabla 31) se pueden encontrar observar que se encuentra con mayor frecuencia "*congreso"* como la más popular para el conjunto de datos *Respuestas_completas* y para *Sustantivos_aws*, indicando que esta es la organización más importante que es actor o responsable de lo que los cuidados quieren cambiar, mejorar o mantener.

Tabla 31: Diagrama NER ORG

Respuestas _completas	
Sustantivos _aws	

Sin embargo, tanto en LOC como ORG se puede ver que existen entidades que intuitivamente no se clasificarían como localización tales como "*acceso tierra", "modelos de educación Colombia", "sistema de ley" y "sistema ley"* o en el caso de organización "*proceso educación Colombia", "enseñar", "brecha académica", "educación niño" o "elite"* por esto es necesario revisar otro tipo de modelos con el fin de reducir los errores en la identificación de identidades.

4.2.8.2. NLTK aplicado a NER

Por medio de la librería NLTK se crea un segundo modelo para el reconocimiento de entidades. Para la construcción del modelo, se requiere la instalación de *coreNLP* y la descarga de *punkt, averaged_perceptron_tagger, maxent_ne_chunker, words.*

Luego es necesario construir la función para la detección de entidades, que en este caso solo se realizarán para el texto completo y así lograr comparar la efectividad de este modelo contra el construido de spaCy.

Dentro de las organizaciones detectadas en el reconocimiento de entidades con NLTK se observa en la figura 54 el token *"congreso"* con la mayor frecuencia, seguida por entidades que al igual que spaCy no necesariamente son organizaciones, tales como "*educación", "ocde", "Colombia"* y "*proceso".* Por lo que se puede ver que a nivel organizaciones, NLTK tiene las mismas limitaciones que spaCy.

Figura 54: NLTK NER ORG

Dentro de las entidades reconocidas como localización, nuevamente se puede ver que los tokens con mayor frecuencia son "*Colombia", "congreso"* y algunas ciudades como *"Bogotá", "Medellín"* y algunos lugares tales como *"parque", "capital" e "institución"* sin embargo, al igual que spaCy, existen algunas entidades que claramente no son localizaciones como *"vivir", "colombiano", "educación"* (ver figura 55).

Figura 55: NLTK NER LOC

Con el fin de evaluar los modelos creados por spaCy y por NLTK, se realiza un tercer modelo por medio de API de AWS y así lograr identificar las entidades de organización y localización dentro del texto.

4.2.8.3. AWS comprehend – Batch detect entities

Para poder utilizar esta API, al igual que el análisis de sentimientos y POS, se requiere la instalación de los paquetes de *boto3, awscli, jedi, urllib3-1.25.4* y luego por medio del servicio comprehend se hace un llamado a *batch_detect_entities.*

Debido a que este servicio tiene una limitante de 25 palabras por oración, se crea una columna adicional en el conjunto de datos, para eliminar aquellas oraciones que no cumplen con este requerimiento, eliminando así un 1% del conjunto de datos. Debido a que esto era una limitación para POS Tagging, se utiliza el DataFrame creado para ese modelo.

Para crear el modelo se debe realizar una función que por medio de un ciclo agregue los tokens que pertenecen a cada una de las entidades en una lista que luego será transformada a una celda del DataFrame.

Finalmente, por medio una nube de palabras se puede observar en la figura 56 los tokens con mayor frecuencia en cada una de las etiquetas, donde para Localización se encuentran palabras como "*Colombia", "Bogotá", "caribe", "Latinoamérica"* que a simple vista son palabras de varias secciones geográficas, donde se encuentran ciudades importantes de Colombia como la "*capital", "Medellín", "leticia", "buenaventura", "Carreño", "Palermo", "Aruaba",* donde actualmente se encuentra una crisis social, cultural y económica. También se puede encontrar lugares tales como "*calle"* y "*vereda"* que, a pesar de ser lugares genéricos, son lugares en los que los ciudadanos pasan la mayor parte de su tiempo (ver figura 56).

Figura 56: NER AWS LOC

Finalmente, para la etiqueta de organizaciones, se encuentran nuevamente el país "*Colombia*" como uno de los más frecuentes, seguido de otras entidades como "*congreso", "universidad", "nacional", "gobierno", "senado*", donde se puede dar una interpretación que son aquellas entidades normalmente públicas las cuales tienen una acción dentro de las respuestas de los participantes (ver figura 57).

Figura 57: NER AWS Organización

A diferencia del modelo creado con NLTK y spaCy, se reduce el número de palabras que no pertenece a organizaciones y localizaciones. No obstante, estas no desaparecen en su totalidad, por lo que al revisar el puntaje del modelo se observa que el nivel de confianza (puntaje) que se les da a estas palabras es menor al 40%, por lo que se podría realizar un filtro en el cual solo muestre aquellas palabras que tengan un puntaje mayor al 60% y así tener una mejor precisión en este modelo.

4.2.9. Complejidad del texto

Dentro de la complejidad del texto, se utiliza el conjunto de datos inicial, sin ningún preprocesamiento, ya que el preprocesamiento no tiene afectación en este modelo, pero si se quiere conocer la complejidad de la respuesta original del ciudadano.

También, es necesario instalar el paquete *textstat* y de allí el paquete *flesch_reading_ease* el cual asignará el puntaje a cada una de las frases del conjunto de datos.

Dentro de la tabla 58, se puede observar el porcentaje de las respuestas dadas por cada nivel de complejidad, adicional a un ejemplo para lograr un mejor análisis.

Figura 58: Complejidad del texto

Nivel	%	Ejemplo
Muy fácil	0.6%	policía ya ser insostenible
Fácil	1.3%	sistema salud porque deber tratar mejor el salud mental
Ligeramente Fácil	3.2%	atención a el población de calle el habitante de calle él poder formar para aportar en el campo
Estándar	7.1%	justicia para que él cambiar el paradigma de que el justicia no funcionar
Ligeramente Difícil	13.9%	proceso de elección No es lógico que unas personas inviertan mucho dinero en una campaña, para luego llegar con compromisos que generan el proceso de corrupción.
Difícil	39.3%	voto voluntario por el voto obligatorio Porque si uno revisa la abstención en las elecciones para el legislativo y el ejecutivo, encuentra que en las últimas elecciones de cada 100 colombianos habilitados para votar, votan 40 y 60 no lo hacen.
Confusa	26.1%	mirada centralista que tener en colombia sobre el propio pais el mirada que él tener ser mucho centralista ver el mirada de bogotá de para allá todo ser provincia en el sentido de que el país girar en torno a bogotá mirada desde el centralismo tener que cambiar en colombia no poder seguir tanto dependiente desde uno punto central del país igualmente desde todo el región que tener uno mirada mucho de él mismo cada región estar mucho ensimismado y solo mirar hacia ser cuando haber uno necesidad de mirar hacia el centro no ser así en otro país él percibir completamente diferente en el sentido de ese mirada hacia el mundo

Se puede evidenciar que, la mayor parte del Dataset tiene un índice que muestra las respuestas como confusas o difíciles de leer, esto se debe al tipo de ecuación que se utiliza, ya que se tiene en cuenta el número de palabras que se utilizan en sus frases y al número de silabas. También, este tipo de índice se utiliza para los textos mayormente escritos en inglés, donde las palabras tienen menor número de silabas que el español, al igual que en español no existen contracciones.

4.2.10. Árbol de palabras

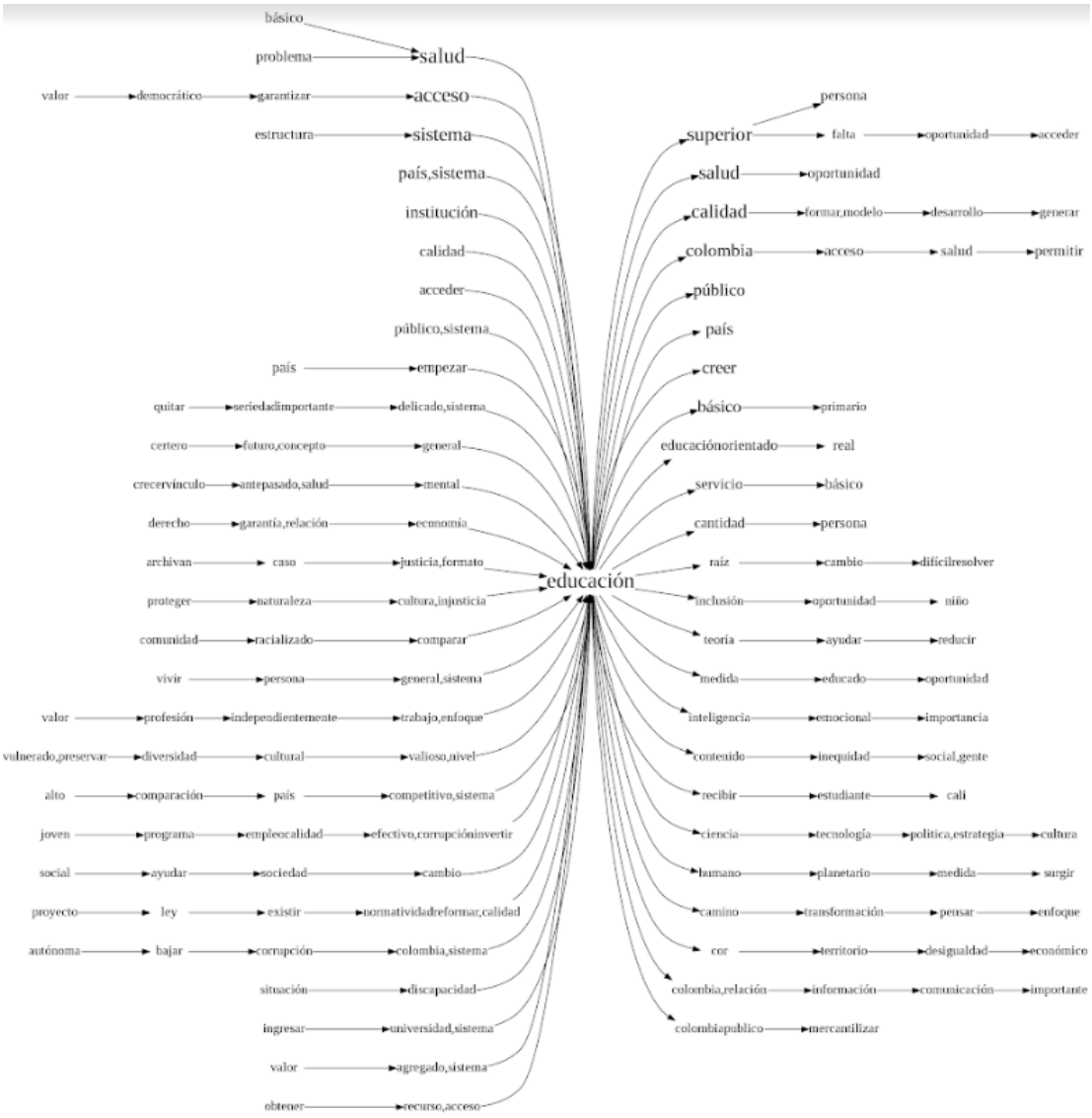
Para la creación del árbol de palabras, se seleccionará las palabras que se encontraron con mayor frecuencia en el análisis estadístico, los cuales son: *"educación", "pais", "sistema", "corrupción" y "Colombia".* Luego se creará una cadena completa de texto con todas las respuestas, para así contar con un único documento que servirá de entrada para el árbol de palabras.

Finalmente, el árbol será creado con la función *search_and_draw*, el cual por medio de la función *render* se creará un archivo .png que mostrará la visualización.

Se debe tener en cuenta que a pesar de mostrar las palabras sufijos y prefijos que tiene la palabra nodo, esta se encuentra fuera de contexto, por lo que se puede tener una idea general de lo que los ciudadanos hablan. También cabe resaltar que dentro de esta librería se puede dar mayor prioridad y por ende mayor frecuencia a las palabras que tienen una fuente más grande y su orden dentro del gráfico.

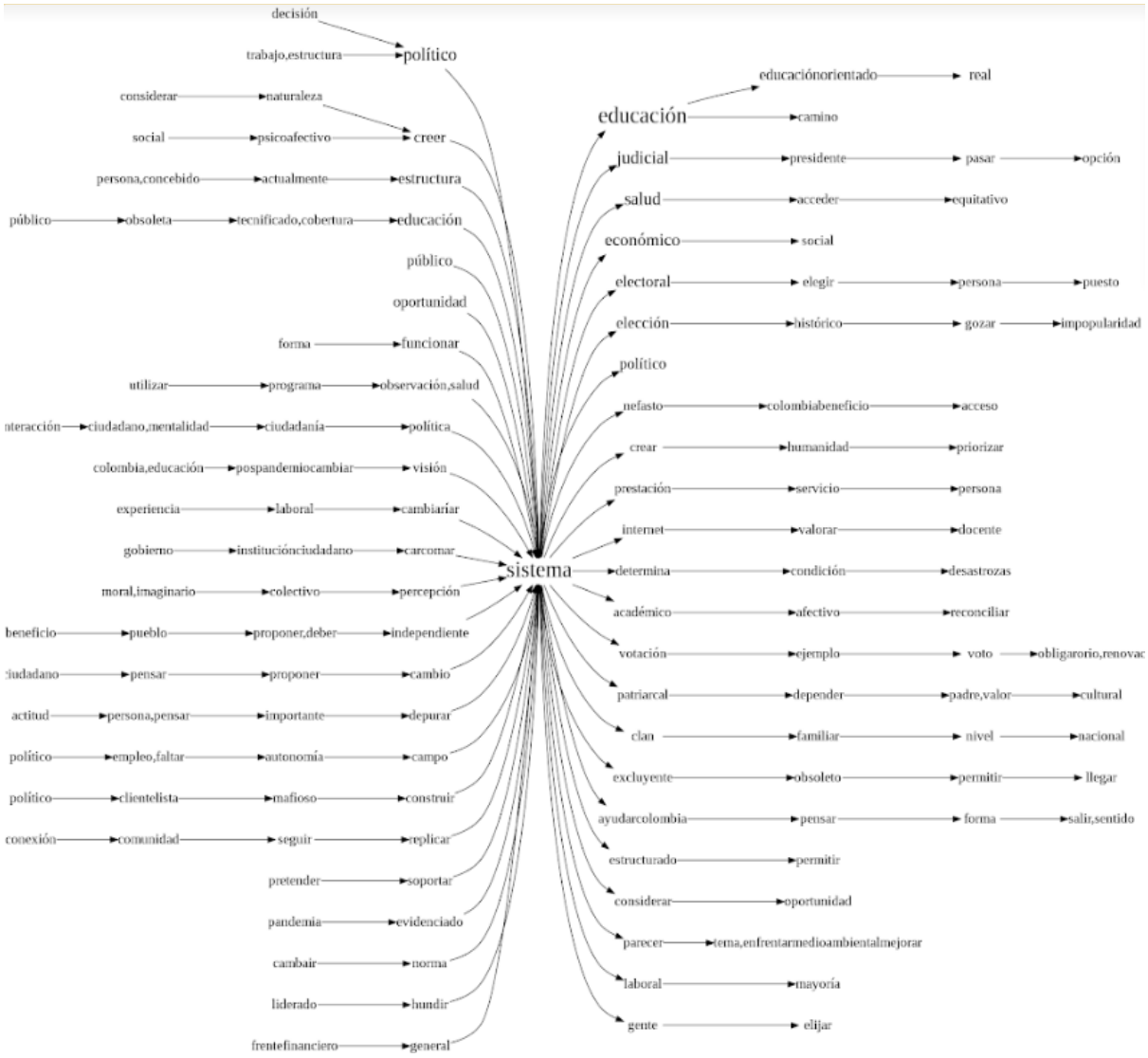
Dentro de la figura 59 se puede visualizar como las palabras más frecuentes utilizadas como sufijo del token "*educación" son "acceso", "institución", "empezar*" y sus prefijos son "*superior", "calidad", básico*" con lo cual nos indica que en las respuestas existe una gran preocupación sobre la educación en Colombia.

Figura 59: Árbol de palabras educación



Otro ejemplo que se puede visualizar en la figura 60, con el token nodo "*sistema*" donde existen prefijos *"político", "creer", "publico" y "educación*" y algunos sufijos que a su vez también tienen palabras populares como "*educación – real*", "*internet – valorar – docente*", "*crear – humanizar – priorizar*".

Figura 60: Árbol de palabras sistema



El árbol de palabras ayuda a tener una visión general de unas palabras especificas dentro del estudio. Sin embargo, se debe tener en cuenta que estas palabras no tienen un contexto y que adicional, para poder crear este árbol, es necesario la creación de un texto único, por lo que afecta los sufijos únicos que pueda tener en unas respuestas.

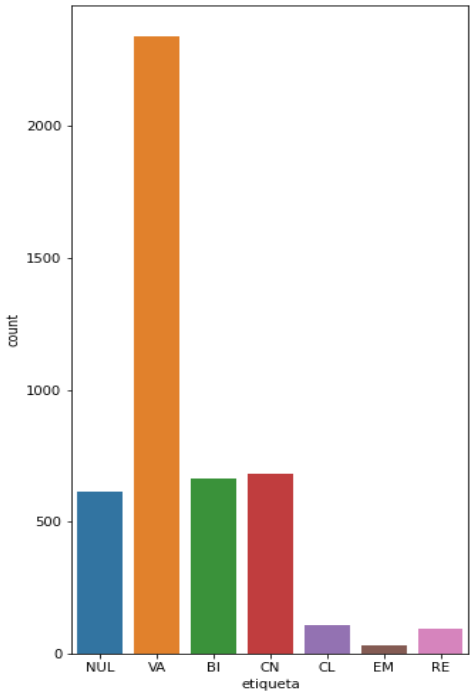
4.3. Modelos de análisis de discurso

Como línea base, se realiza la creación de una competencia de clasificadores, utilizando los valores que sus hiperpárametros tienen por defecto, con base en la BoW y TF-IDF creado con el conjunto *Lemma_no_Stopwords*.

Para la implementación de esta competencia, se debe realizar un ajuste al conjunto de datos, especialmente en la etiqueta, ya que actualmente esta puede tener varios valores, como se pudo observar en el análisis exploratorio, y debido a que el tamaño del conjunto de datos no es lo suficientemente grande, no se buscará crear un clasificador multiclase. Por lo tanto, solo se mantiene el primer indicador verbal asignado a la respuesta dada por el ciudadano.

Por lo que la nueva distribución de la etiqueta se puede observar en la figura 61 cómo el 51.6% de las respuestas fueron clasificadas como VA, seguidas por CN con 15% y luego por BI con un 14,6%, y donde el conjunto de datos tiene una etiqueta con una representación mínima del 0,7%.

Figura 61: Distribución de la etiqueta



Notoriamente se observa un desbalanceo en la etiqueta, que afectara significativamente los modelos de clasificación, por lo que se debe realizar una evaluación de los diferentes métodos para balance de clases.

Adicional al ajuste de la etiqueta, se busca eliminar o reemplazar los demás valores demográficos que sean nulos, ya que estos podrán ayudar a mejorar los clasificadores. Para esto, gracias al análisis exploratorio, se llenan los valores nulos con el valor con la mediana más alta.

4.3.1. Línea base

Para la creación de la línea base, se define la variable X como aquella que tiene el texto y demás datos demográficos que ayudaran a la clasificación, y la variable Y, la cual será solamente la etiqueta que se intenta buscar. Una vez definidas las variables, se utiliza el método de *count_vectorizer* y así crear la bolsa de palabras, la cual solo aplicará para la variable X. Luego, se debe realizar la separación del conjunto de datos para entrenamiento (67%) y de prueba (33%).

Finalmente, para crear la competencia de clasificadores, se crea una lista en la cual se encuentra los clasificadores sin modificación en los hiperparámetros los cuales se pueden apreciar en la tabla 32.

Tabla 32: Hiperparámetros iniciales

Clasificador	Hiperparámetros
Logistic Regression	Multi_class = 'multinomial', Solver = 'lbfgs', C = 5
Decision Tree	Critererion = 'entropy'
Random Forest	N_estimators = 10
Naive Bayes	Alpha = 1.0, Fit_prior = True, Class_prioir = None
KNN	N_neighbors = 5, Weights = 'unirform'
MLP	Alpha = 1.5, Max_iter = 500

Luego, por medio de una iteración, se inicializa los clasificadores, y después el clasificador se ajusta a los datos de entrenamientos para así crear la predicción del modelo con el conjunto de datos de pruebas. Finalmente, se revisa su desempeño por medio de un reporte de métricas, a saber: *precisión*, *recall*, *f1puntaje* y *accuracy*.

Al crear BoW se obtiene una matriz con 6528 variables, por lo que al crear una bolsa de palabras con *count vectorizer* y estableciendo un máximo de características a 1000, se realiza una comparación en cada uno de los clasificadores, y se ve que no existe una mejora en el modelo que sea significativa en ninguna de las métricas.

También se puede ver que los clasificadores tienen un desempeño muy bajo, ya que todos los *accuracy* se encuentran por debajo del 50%. Adicionalmente, el *F1-Puntaje* de cada una de las clases está por debajo del 20%, y solo para el valor VA (que es que tiene mayor participación en el conjunto de datos) tiene un *F1-puntaje* mayor al 70%. Este tipo de desempeño es esperado debido al gran desbalanceo de las clases que se tienen en el conjunto de datos (ver tabla 33).

Tabla 33: Reporte Métricas BoW en diferentes clasificadores

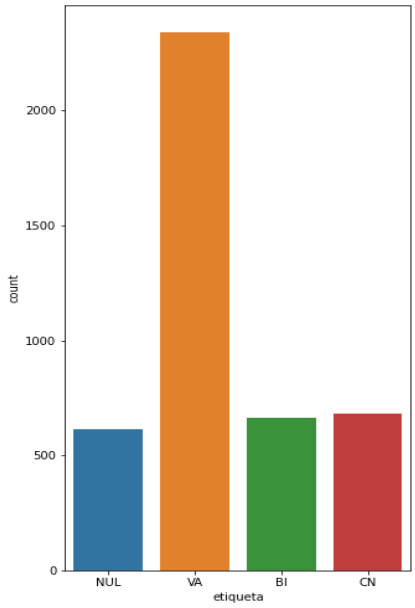
Clasificador	BoW	BoW max feature = 1000
Logistic Regression	Reporte de metricas precision recall f1-score support BI 0.30 0.24 0.27 217 CL 0.33 0.10 0.16 29 CN 0.33 0.27 0.29 228 EM 0.00 0.00 0.00 13 NUL 0.19 0.18 0.18 190 RE 0.75 0.08 0.14 38 VA 0.62 0.74 0.67 782 accuracy 0.49 1497 macro avg 0.36 0.23 0.25 1497 weighted avg 0.46 0.49 0.47 1497	Reporte de metricas precision recall f1-score support BI 0.30 0.25 0.27 217 CL 0.17 0.10 0.13 29 CN 0.30 0.28 0.29 228 EM 0.00 0.00 0.00 13 NUL 0.19 0.19 0.19 190 RE 0.50 0.08 0.14 38 VA 0.62 0.71 0.66 782 accuracy 0.48 1497 macro avg 0.30 0.23 0.24 1497 weighted avg 0.46 0.48 0.46 1497
Decision Tree	Reporte de metricas precision recall f1-score support BI 0.33 0.21 0.26 217 CL 0.06 0.03 0.04 29 CN 0.25 0.14 0.18 228 EM 0.00 0.00 0.00 13 NUL 0.16 0.15 0.15 190 RE 0.11 0.05 0.07 38 VA 0.57 0.73 0.64 782 accuracy 0.46 1497 macro avg 0.21 0.19 0.19 1497 weighted avg 0.41 0.46 0.42 1497	Reporte de metricas precision recall f1-score support BI 0.25 0.24 0.24 217 CL 0.04 0.03 0.04 29 CN 0.21 0.20 0.21 228 EM 0.00 0.00 0.00 13 NUL 0.16 0.23 0.19 190 RE 0.06 0.03 0.04 38 VA 0.59 0.58 0.58 782 accuracy 0.40 1497 macro avg 0.19 0.19 0.19 1497 weighted avg 0.40 0.40 0.40 1497

Random Forest	Reporte de metricas precision recall f1-score support BI 0.23 0.10 0.14 217 CL 0.14 0.03 0.06 29 CN 0.35 0.18 0.23 228 EM 0.50 0.08 0.13 13 NUL 0.15 0.10 0.12 190 RE 0.25 0.03 0.05 38 VA 0.56 0.82 0.67 782 accuracy 0.49 1497 macro avg 0.31 0.19 0.20 1497 weighted avg 0.41 0.49 0.42 1497	Reporte de metricas precision recall f1-score support BI 0.24 0.18 0.21 217 CL 0.06 0.03 0.04 29 CN 0.29 0.18 0.22 228 EM 0.00 0.00 0.00 13 NUL 0.17 0.21 0.19 190 RE 0.14 0.05 0.08 38 VA 0.60 0.70 0.65 782 accuracy 0.45 1497 macro avg 0.21 0.19 0.20 1497 weighted avg 0.42 0.45 0.43 1497
Naive Bayes	Reporte de metricas precision recall f1-score support BI 0.41 0.09 0.15 217 CL 0.50 0.03 0.06 29 CN 0.39 0.12 0.19 228 EM 0.00 0.00 0.00 13 NUL 0.28 0.06 0.10 190 RE 0.00 0.00 0.00 38 VA 0.55 0.94 0.70 782 accuracy 0.53 1497 macro avg 0.30 0.18 0.17 1497 weighted avg 0.45 0.53 0.43 1497	Reporte de metricas precision recall f1-score support BI 0.29 0.18 0.23 217 CL 0.18 0.07 0.10 29 CN 0.38 0.26 0.31 228 EM 0.00 0.00 0.00 13 NUL 0.21 0.14 0.17 190 RE 0.17 0.03 0.05 38 VA 0.59 0.80 0.68 782 accuracy 0.50 1497 macro avg 0.26 0.21 0.22 1497 weighted avg 0.44 0.50 0.46 1497
KNN	Reporte de metricas precision recall f1-score support BI 0.15 0.13 0.14 217 CL 0.00 0.00 0.00 29 CN 0.24 0.07 0.10 228 EM 0.00 0.00 0.00 13 NUL 0.13 0.54 0.21 190 RE 0.00 0.00 0.00 38 VA 0.58 0.32 0.42 782 accuracy 0.27 1497 macro avg 0.16 0.15 0.12 1497 weighted avg 0.38 0.27 0.28 1497	Reporte de metricas precision recall f1-score support BI 0.10 0.12 0.11 217 CL 0.12 0.03 0.05 29 CN 0.22 0.10 0.13 228 EM 0.00 0.00 0.00 13 NUL 0.16 0.56 0.25 190 RE 0.00 0.00 0.00 38 VA 0.61 0.35 0.44 782 accuracy 0.29 1497 macro avg 0.17 0.17 0.14 1497 weighted avg 0.39 0.29 0.30 1497
MLP	Reporte de metricas precision recall f1-score support BI 0.39 0.20 0.26 217 CL 0.00 0.00 0.00 29 CN 0.46 0.22 0.30 228 EM 0.00 0.00 0.00 13 NUL 0.23 0.11 0.15 190 RE 0.00 0.00 0.00 38 VA 0.59 0.89 0.71 782 accuracy 0.54 1497 macro avg 0.24 0.20 0.20 1497 weighted avg 0.46 0.54 0.47 1497	Reporte de metricas precision recall f1-score support BI 0.41 0.15 0.22 217 CL 0.00 0.00 0.00 29 CN 0.40 0.15 0.22 228 EM 0.00 0.00 0.00 13 NUL 0.27 0.11 0.15 190 RE 0.00 0.00 0.00 38 VA 0.57 0.91 0.70 782 accuracy 0.53 1497 macro avg 0.24 0.19 0.18 1497 weighted avg 0.45 0.53 0.45 1497

Para mejorar los diferentes clasificadores, se busca realizar un balance de clases por medio de *Oversampling* y *Undersampling.* Estas técnicas ayudan a aquellos conjuntos de datos que tienen un sesgo en la distribución de clases al muestrear aleatoriamente el conjunto de entrenamiento. En la técnica de *Oversampling* se elimina algunos récords de la clase mayoritaria y en *Undersampling* se duplican los récords de la clase minoritaria.

Sin embargo, como se observó en el análisis de la etiqueta, existen algunas clases cuya representación es menor al 3% en todo el conjunto de datos, por tal razón, se deben eliminar y luego aplicar las diferentes técnicas de balanceo. Las clases a eliminar son CL que representa 2.8%, EM que fue la etiqueta usada solamente en un 0.7% y finalmente RE que tiene un 2% (ver figura 62).

Figura 62: Nueva Distribución de la etiqueta



El nuevo conjunto de datos tiene un tamaño de 4302, donde se tiene un total de 4 clases (BI, CN, NUL y VA). Donde el conjunto de entrenamiento tiene un total de 2882 y conjunto de pruebas 1420.

Con el fin de conocer el desempeño de los diferentes clasificadores, se realiza una construcción con TF-IDF en el cual se observa que para *Logistic regression* y KNN tienen un mejor desempeño aumentando su *acurracy* en un 6%, sin embargo, para *Decision Tree, Random Forest y Naive Bayes* disminuye en un 3%, estos resultados se pueden visualizar en la tabla 34.

Tabla 34: Reporte Métricas TF-IDF

Clasificador	TF-IDF	TF-IDF max feature = 1000
Logistic Regression	Reporte de metricas precision recall f1-score support BI 0.31 0.24 0.27 191 CN 0.42 0.22 0.29 233 NUL 0.23 0.09 0.13 221 VA 0.62 0.85 0.72 775 accuracy 0.55 1420 macro avg 0.40 0.35 0.35 1420 weighted avg 0.48 0.55 0.49 1420	Reporte de metricas precision recall f1-score support BI 0.27 0.27 0.27 191 CN 0.36 0.24 0.29 233 NUL 0.25 0.14 0.17 221 VA 0.63 0.77 0.70 775 accuracy 0.52 1420 macro avg 0.38 0.36 0.36 1420 weighted avg 0.48 0.52 0.49 1420
Decision Tree	Reporte de metricas precision recall f1-score support BI 0.16 0.21 0.18 191 CN 0.25 0.20 0.22 233 NUL 0.24 0.17 0.20 221 VA 0.60 0.65 0.62 775 accuracy 0.44 1420 macro avg 0.31 0.31 0.31 1420 weighted avg 0.43 0.44 0.43 1420	Reporte de metricas precision recall f1-score support BI 0.18 0.21 0.19 191 CN 0.23 0.19 0.21 233 NUL 0.20 0.21 0.21 221 VA 0.61 0.60 0.61 775 accuracy 0.42 1420 macro avg 0.31 0.31 0.31 1420 weighted avg 0.43 0.42 0.42 1420
Random Forest	Reporte de metricas precision recall f1-score support BI 0.25 0.13 0.17 191 CN 0.30 0.10 0.15 233 NUL 0.20 0.06 0.09 221 VA 0.58 0.87 0.70 775 accuracy 0.52 1420 macro avg 0.33 0.29 0.28 1420 weighted avg 0.43 0.52 0.44 1420	Reporte de metricas precision recall f1-score support BI 0.21 0.19 0.20 191 CN 0.29 0.16 0.21 233 NUL 0.23 0.11 0.15 221 VA 0.59 0.77 0.67 775 accuracy 0.49 1420 macro avg 0.33 0.31 0.31 1420 weighted avg 0.43 0.49 0.45 1420
Naive Bayes	Reporte de metricas precision recall f1-score support BI 1.00 0.01 0.02 191 CN 1.00 0.01 0.02 233 NUL 0.00 0.00 0.00 221 VA 0.55 1.00 0.71 775 accuracy 0.55 1420 macro avg 0.64 0.25 0.19 1420 weighted avg 0.60 0.55 0.39 1420	Reporte de metricas precision recall f1-score support BI 0.38 0.04 0.08 191 CN 0.46 0.05 0.09 233 NUL 0.14 0.00 0.01 221 VA 0.55 0.98 0.71 775 accuracy 0.55 1420 macro avg 0.38 0.27 0.22 1420 weighted avg 0.45 0.55 0.41 1420
KNN	Reporte de metricas precision recall f1-score support BI 0.23 0.25 0.24 191 CN 0.35 0.20 0.26 233 NUL 0.20 0.14 0.17 221 VA 0.62 0.75 0.68 775 accuracy 0.50 1420 macro avg 0.35 0.33 0.33 1420 weighted avg 0.46 0.50 0.47 1420	Reporte de metricas precision recall f1-score support BI 0.16 0.27 0.20 191 CN 0.40 0.01 0.02 233 NUL 0.14 0.64 0.24 221 VA 0.53 0.08 0.14 775 accuracy 0.18 1420 macro avg 0.31 0.25 0.15 1420 weighted avg 0.40 0.18 0.14 1420
MLP	Reporte de metricas precision recall f1-score support BI 0.43 0.02 0.03 191 CN 0.17 0.00 0.01 233 NUL 0.00 0.00 0.00 221 VA 0.55 0.99 0.70 775 accuracy 0.54 1420 macro avg 0.29 0.25 0.19 1420 weighted avg 0.38 0.54 0.39 1420	Reporte de metricas precision recall f1-score support BI 0.37 0.04 0.07 191 CN 0.62 0.02 0.04 233 NUL 0.00 0.00 0.00 221 VA 0.55 0.99 0.71 775 accuracy 0.55 1420 macro avg 0.39 0.26 0.20 1420 weighted avg 0.45 0.55 0.40 1420

También se puede observar que al igual que la construcción de BoW, el desempeño de los clasificadores para las métricas *acurracy, f1-puntaje, recall y precisión* tienen un mejor desempeño en TF-IDF si no se agrega un máximo de características.

4.3.1.1. Oversampling

Para la construcción de *Oversampling*, se realiza una duplicidad aleatoria con ejemplos de la clase minoritaria y así se añaden al conjunto de datos de entrenamiento. Esto se logra importando la librería *imblearn.over_sampling* de *RandomOverSampler*, de allí se declara una variable que iniciara el modelo para ser aplicado sobre el conjunto de entrenamiento. Este tipo de técnica genera un conjunto de entrenamiento con 1566 respuesta en cada una de las etiquetas (BI, CN, NUL y VA).

Dentro de la técnica de *Oversampling* se puede establecer la estrategia que es quiere seguir, en el caso de este proyecto, se quiere revisar la estrategia *minority* la cual solo balancea la etiqueta minoritaria, manteniendo el resto igual. Como resultado, se obtiene 1566 respuestas para VA y para NUL, dejando BI con 475 muestras y CN 449.

En la tabla 63 se puede dar una comparación entre el reporte de métricas entre *Oversampling* sin estrategia y con estrategia minoritaria, donde se puede ver una mejora de 2 a 3% si se utiliza un Oversampling con estrategia.

Figura 63: Resultados Oversampling

Clasificador	Oversampling	Oversampling Minorista
Logistic Regression	Reporte de metricas precision recall f1-score support BI 0.23 0.30 0.26 191 CN 0.35 0.32 0.33 233 NUL 0.26 0.25 0.26 221 VA 0.67 0.64 0.65 775 accuracy 0.48 1420 macro avg 0.38 0.38 0.38 1420 weighted avg 0.49 0.48 0.49 1420	Reporte de metricas precision recall f1-score support BI 0.26 0.26 0.26 191 CN 0.38 0.25 0.30 233 NUL 0.25 0.29 0.27 221 VA 0.65 0.69 0.67 775 accuracy 0.50 1420 macro avg 0.38 0.37 0.37 1420 weighted avg 0.49 0.50 0.49 1420

Decision Tree	Reporte de metricas precision recall f1-score support BI 0.15 0.14 0.15 191 CN 0.25 0.16 0.19 233 NUL 0.17 0.15 0.16 221 VA 0.58 0.66 0.62 775 accuracy 0.43 1420 macro avg 0.29 0.28 0.28 1420 weighted avg 0.40 0.43 0.41 1420	Reporte de metricas precision recall f1-score support BI 0.14 0.13 0.13 191 CN 0.21 0.10 0.13 233 NUL 0.23 0.29 0.26 221 VA 0.60 0.66 0.63 775 accuracy 0.44 1420 macro avg 0.29 0.29 0.29 1420 weighted avg 0.42 0.44 0.42 1420
Random Forest	Reporte de metricas precision recall f1-score support BI 0.23 0.28 0.25 191 CN 0.31 0.17 0.22 233 NUL 0.22 0.24 0.23 221 VA 0.64 0.68 0.66 775 accuracy 0.47 1420 macro avg 0.35 0.34 0.34 1420 weighted avg 0.47 0.47 0.47 1420	Reporte de metricas precision recall f1-score support BI 0.19 0.13 0.15 191 CN 0.36 0.10 0.15 233 NUL 0.26 0.39 0.32 221 VA 0.62 0.72 0.67 775 accuracy 0.49 1420 macro avg 0.36 0.33 0.32 1420 weighted avg 0.46 0.49 0.46 1420
Naive Bayes	Reporte de metricas precision recall f1-score support BI 0.25 0.38 0.30 191 CN 0.33 0.35 0.34 233 NUL 0.25 0.23 0.24 221 VA 0.70 0.61 0.65 775 accuracy 0.48 1420 macro avg 0.38 0.39 0.38 1420 weighted avg 0.51 0.48 0.49 1420	Reporte de metricas precision recall f1-score support BI 0.34 0.11 0.17 191 CN 0.42 0.12 0.19 233 NUL 0.24 0.34 0.28 221 VA 0.61 0.78 0.68 775 accuracy 0.51 1420 macro avg 0.40 0.34 0.33 1420 weighted avg 0.49 0.51 0.47 1420
KNN	Reporte de metricas precision recall f1-score support BI 0.13 0.17 0.14 191 CN 0.22 0.12 0.16 233 NUL 0.16 0.65 0.26 221 VA 0.72 0.14 0.23 775 accuracy 0.22 1420 macro avg 0.31 0.27 0.20 1420 weighted avg 0.47 0.22 0.21 1420	Reporte de metricas precision recall f1-score support BI 0.17 0.04 0.06 191 CN 0.22 0.02 0.03 233 NUL 0.16 0.85 0.27 221 VA 0.61 0.15 0.24 775 accuracy 0.22 1420 macro avg 0.29 0.26 0.15 1420 weighted avg 0.42 0.22 0.18 1420
MLP	Reporte de metricas precision recall f1-score support BI 0.26 0.32 0.29 191 CN 0.33 0.29 0.31 233 NUL 0.29 0.33 0.31 221 VA 0.68 0.64 0.66 775 accuracy 0.49 1420 macro avg 0.39 0.40 0.39 1420 weighted avg 0.51 0.49 0.50 1420	Reporte de metricas precision recall f1-score support BI 0.41 0.16 0.23 191 CN 0.45 0.12 0.20 233 NUL 0.27 0.40 0.32 221 VA 0.63 0.78 0.70 775 accuracy 0.53 1420 macro avg 0.44 0.37 0.36 1420 weighted avg 0.52 0.53 0.49 1420

Realizando una comparación entre BoW sin límite de características y Oversampling con estrategia minorista, se puede observar que existe una mejora en todas las métricas de 2% para la *logistic regression*, *random forest y Naives Bayes* y una desmejora para los clasificadores *decisión tree y KNN*. También se evidencia que ninguno de los clasificadores tiene un *accuracy* mayor al 50% por lo que los modelos no son óptimos para producción, por esta razón, se busca una nueva alternativa de *resampling.*

4.3.1.2. Undersampling

Para realizar el submuestreo aleatorio, selecciona aleatoriamente puntos de datos de la clase mayoritaria y así poder tener el mismo número de muestras de la clase minoritaria.

Para la preparación del *Undersampling* se debe importar la librería *imblearn.under_sampling* de *RandomUnderSampler,* y al igual que *Oversampling* se debe realizar una declaración de una variable para inicializar el modelo y luego aplicarla al conjunto de datos de entrenamiento. Este tipo de técnica genera un conjunto de datos con 392 muestras por cada clase (BI, CN, NUL, VA).

Una de las técnicas *Undersampling* es establecer la estrategia *mayority* en la cual, solamente se minimiza el número de muestras para la clase mayoritaria a la minoritaria, dejando el resto de las clases igual, esto genera que BI tenga un numero de 476, CN con 449, y NUL y VA con 392 muestras.

En la tabla 34 se puede observar que el generar una estrategia aumenta el *acurracy y precision* en un 11% en los modelos de *logistic regression*, *Decision Tree, Random Forest, Naives Bayes*, KNN y MLP.

Tabla 35: Reporte de Métricas Undersampling

Clasificador	Undersampling	Undersampling Mayorista
Logistic Regression	Reporte de metricas precision recall f1-score support BI 0.23 0.39 0.29 191 CN 0.27 0.36 0.31 233 NUL 0.25 0.42 0.31 221 VA 0.72 0.40 0.52 775 accuracy 0.40 1420 macro avg 0.37 0.39 0.36 1420 weighted avg 0.51 0.40 0.42 1420	Reporte de metricas precision recall f1-score support BI 0.27 0.25 0.26 191 CN 0.36 0.21 0.27 233 NUL 0.27 0.21 0.24 221 VA 0.63 0.75 0.68 775 accuracy 0.51 1420 macro avg 0.38 0.36 0.36 1420 weighted avg 0.48 0.51 0.49 1420
Decision Tree	Reporte de metricas precision recall f1-score support BI 0.20 0.30 0.24 191 CN 0.20 0.30 0.24 233 NUL 0.19 0.36 0.25 221 VA 0.62 0.28 0.39 775 accuracy 0.30 1420 macro avg 0.30 0.31 0.28 1420 weighted avg 0.43 0.30 0.32 1420	Reporte de metricas precision recall f1-score support BI 0.18 0.18 0.18 191 CN 0.32 0.16 0.21 233 NUL 0.18 0.12 0.15 221 VA 0.58 0.72 0.64 775 accuracy 0.46 1420 macro avg 0.32 0.30 0.30 1420 weighted avg 0.42 0.46 0.43 1420

Random Forest	Reporte de metricas precision recall f1-score support BI 0.19 0.39 0.26 191 CN 0.22 0.18 0.20 233 NUL 0.20 0.46 0.28 221 VA 0.69 0.30 0.42 775 accuracy 0.32 1420 macro avg 0.33 0.33 0.29 1420 weighted avg 0.47 0.32 0.34 1420	Reporte de metricas precision recall f1-score support BI 0.21 0.12 0.15 191 CN 0.40 0.12 0.18 233 NUL 0.17 0.05 0.08 221 VA 0.57 0.87 0.69 775 accuracy 0.52 1420 macro avg 0.34 0.29 0.28 1420 weighted avg 0.43 0.52 0.44 1420
Naive Bayes	Reporte de metricas precision recall f1-score support BI 0.22 0.44 0.29 191 CN 0.28 0.39 0.33 233 NUL 0.26 0.23 0.24 221 VA 0.72 0.49 0.58 775 accuracy 0.42 1420 macro avg 0.37 0.39 0.36 1420 weighted avg 0.51 0.42 0.45 1420	Reporte de metricas precision recall f1-score support BI 0.36 0.12 0.17 191 CN 0.34 0.10 0.15 233 NUL 0.21 0.04 0.06 221 VA 0.57 0.93 0.71 775 accuracy 0.54 1420 macro avg 0.37 0.29 0.27 1420 weighted avg 0.45 0.54 0.44 1420
KNN	Reporte de metricas precision recall f1-score support BI 0.09 0.13 0.11 191 CN 0.20 0.03 0.05 233 NUL 0.16 0.75 0.26 221 VA 0.71 0.05 0.10 775 accuracy 0.17 1420 macro avg 0.29 0.24 0.13 1420 weighted avg 0.46 0.17 0.12 1420	Reporte de metricas precision recall f1-score support BI 0.20 0.13 0.16 191 CN 0.41 0.03 0.06 233 NUL 0.20 0.66 0.30 221 VA 0.62 0.42 0.50 775 accuracy 0.36 1420 macro avg 0.36 0.31 0.25 1420 weighted avg 0.46 0.36 0.35 1420
MLP	Reporte de metricas precision recall f1-score support BI 0.24 0.40 0.30 191 CN 0.28 0.39 0.33 233 NUL 0.25 0.42 0.32 221 VA 0.74 0.40 0.52 775 accuracy 0.40 1420 macro avg 0.38 0.40 0.37 1420 weighted avg 0.52 0.40 0.43 1420	Reporte de metricas precision recall f1-score support BI 0.29 0.23 0.25 191 CN 0.41 0.18 0.25 233 NUL 0.32 0.22 0.26 221 VA 0.63 0.82 0.71 775 accuracy 0.54 1420 macro avg 0.41 0.36 0.37 1420 weighted avg 0.50 0.54 0.50 1420

4.4. Ajuste y evaluación de modelos

Se puede observar que en los reportes de métricas de los modelos en el numeral 4.3.1 de este proyecto, tiene diferente comportamiento dependiendo del clasificador, método de construcción y *resampling* utilizado.

Con el fin de poder enfocarse en el top 3 de clasificadores y así poder realizar los ajustes necesarios para reevaluar los modelos, se hace una interpretación del reporte de métricas, dando como resumen la tabla 35.

Tabla 36: Comparación de evaluación de cada clasificador

Clasificador	BoW	TF-IDF	Undersampling	Oversampling
Logistic Regression	*Acurracy* de 49% y *precision, recall, f1puntaje* por debajo del 30% para todas las clases, exceptuando VA. No hay balanceo de clases	*Acurracy* del 55% y con *Precision* muy variable entre clases, donde la menor es del 31%, y con *recall* menor al 25% para todas las clases exceptuando VA. No hay balanceo de clase	Se construye con el modelo TF-IDF, con un *acurracy* del 51%, y su *precisión* se encuentra por debajo del 40% para todas las clases exceptuando VA. Se utiliza la estrategia *Mayority.*	Se construye con el modelo TF-IDF, hay un *acurracy 50%*, pero la *precision* aumenta un 2% en las clases. Se usa estrategia *Minority.*
Decision Tree	*Acurracy* de 46% y *precision, recall, f1puntaje* por debajo del 25% para todas las clases, exceptuando VA. No hay balanceo de clases	*Acurracy* del 44% y con *Precision* por debajo del 30% para todas las clases, y con *recall* menor al 15% para todas las clases exceptuando VA. No hay balanceo de clase	Se construye con el modelo TF-IDF, con un *acurracy* del 46%, y su *precisión* se encuentra por debajo del 30% para todas las clases exceptuando VA con un 58%. Su *Recall* se mantiene en por debajo del 20% para todas las clases exceptuando VA. Se utiliza la estrategia *Mayority.*	Se construye con el modelo TF-IDF, hay un *acurracy 44%*, pero la *precision* aumenta un 2% en las clases. Se usa estrategia *Minority*
Random Forest	*Acurracy* de 49% y *precision, recall, f1puntaje* por debajo del 40% para todas las clases, exceptuando VA.	*Acurracy* del 52% y con *Precision* es por debajo del 60% para todas las clases, y con *recall* menor al 15% para todas	Se construye con el modelo TF-IDF, con un *acurracy* del 52%, y su *precisión* se encuentra por debajo del 40% para todas las	Se construye con el modelo TF-IDF, hay un *acurracy 49%*, pero la *precision* aumenta un 1% en las clases.

	No hay balanceo de clases	las clases exceptuando VA. No hay balanceo de clases.	clases exceptuando VA y un *recall* debajo del 10% para todas las clases exceptuando VA con un 87% Se utiliza la estrategia *Mayority.*	Se usa estrategia *Minority*
Naive Bayes	*Acurracy* de 53% y *precisión debajo del 50%* para todas las clases*, recall,* menor al 15% para todas las clases exceptuando VA que está en 94%. No hay balanceo de clases	*Acurracy* del 55% y con *Precision* del 100% para todas las clases, exceptuando VA. todas las clases, y con *recall* menor al 1% para todas las clases exceptuando VA. No hay balanceo de clases.	Se construye con el modelo TF-IDF, con un *acurracy* del 54%, y su *precisión* se encuentra por debajo del 40% para todas las clases exceptuando VA. Se utiliza la estrategia *Mayority*	Se construye con el modelo TF-IDF, hay un *acurracy 51%*, pero la *precision* aumenta un 10% en las clases, el *Recall* aumenta para las demás clases en un 10%. Se usa estrategia *Minority*
KNN	*Acurracy* de 27% y *precision, recall, f1puntaje* por debajo del 30% para todas las clases, exceptuando VA. No hay balanceo de clases	Se tiene un *Acurracy* del 50%, con *precision* por debajo del 60%, y recall, por debajo del 25% para todas las clases, exceptuando VA. No hay balanceo de clases.	Se construye con el modelo TF-IDF, con un *acurracy* del 36%, y su *precisión* aumenta en un 7% para todas las clases Se utiliza la estrategia *Mayority*	Se construye con el modelo TF-IDF, hay un *acurracy 22%*, pero la *precision* aumenta un 6% en las clases. Se usa estrategia *Minority*
MLP	*Acurracy* de 54% y *precisión* por debajo del 45% para todas las clases, recall por debajo del 25%	*Acurracy* del 54%, y con *precision* menor al 55% para todas las clases, y un *recall* por menor al 1% para	Se construye con el modelo TF-IDF, con un *acurracy* del 54%, y su *precisión* aumenta en un	Se construye con el modelo TF-IDF, hay un *acurracy 53%*, pero la *precision* aumenta

	para todas las clases No hay balanceo de clases	todas las clases, exceptuando VA. No hay balanceo de clases.	10% para todas las clases. Se utiliza la estrategia *Mayority*	un 15% en las clases. Se usa estrategia *Minority*

Para lograr una mejor evaluación de los modelos, se tomarán los modelos de *Logistic regression, Random Forest* y Naives Bayes, para realizar varios ajustes a los modelos y así poder tener una mejor evaluación.

4.4.1. Ajustes en los Modelos

Para realizar ajustes y así realizar nuevas evaluaciones de los modelos, se utilizará la técnica y estrategias con la que se obtuvo mejor desempeño, y de allí se realizará nuevos ajustes con el fin de observar cómo estos ayudaran a los modelos a aprender a clasificar de una manera correcta las respuestas en sus respectivos indicadores verbales.

4.4.1.1. Logistic Regression

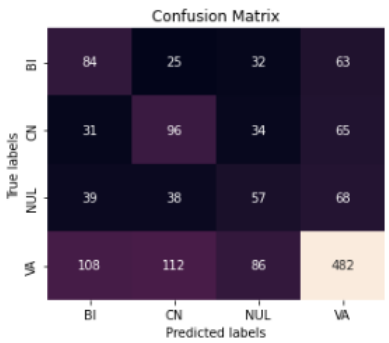
Para *Logistic Regression* se utilizará el modelo construido por TF-IDF, eliminado los tokens que tengan una importancia menor al 40% manteniendo el 73.1% del conjunto de entrenamiento. Al realizar la separación del conjunto de datos, se obtiene un conjunto de entrenamientos tiene 2881 y de prueba 1420.

Como ajustes al modelo, se agrega el hiperparámetros *Class_imbalance* el cual revisa el balance de las clases sin usar las técnicas de *Undersampling* ni *Oversampling*. Este modelo da como resultado con acurracy del 51%, y precisión, recall y f1-puntaje tienen un puntaje por debajo del 40% para todas las clases exceptuando VA. Dentro de la matriz de confusión se observa que se han clasificado correctamente las 84 respuestas para la etiqueta BI, que representa el 32% y se observa que el 42% de las respuestas son clasificadas como VA en lugar de BI. Para el clasificador CN y NUL también se están clasificando 41% en VA, estos resultados pueden ser visualizados en la tabla de 36.

Tabla 37: Métricas para Logistic Regression

Reporte de Métricas	Matriz de confusion

	precision	recall	f1-score	support
BI	0.32	0.41	0.36	204
CN	0.35	0.42	0.39	226
NUL	0.27	0.28	0.28	202
VA	0.71	0.61	0.66	788
accuracy			0.51	1420
macro avg	0.41	0.43	0.42	1420
weighted avg	0.54	0.51	0.52	1420



4.4.1.2. Random Forest

Dentro del clasificador construido por medio de *Random Forest* se observó que el mejor desempeño se da por medio de la construcción por TF-IDF y la utilización de la técnica *Undersampling*, por lo que al igual que en la *Logistic Regression*, se realiza un ajuste al modelo eliminado aquellas variables que tienen una importancia menor al 40% pasando de 6262 a 4603 características, quedando así el mismo conjunto de datos utilizado en el clasificador *Logistic Regression.*

El conjunto tiene un total de 4301 que es dividido en conjunto de entrenamiento con 2881 y de pruebas 1420 el cual está dividido en las clases BI con 462, CN con 455, NUL con 411 y VA con 1553. Como se indicó con anterioridad, se busca un balanceo de clases por medio de la técnica *Undersampling* con estrategia de *Majority* el cual genera un balance de clases donde BI, CN y NUL permanecen con los mismos valores y VA es recortado a 411 respuestas.

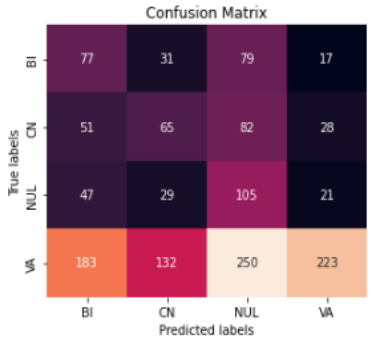
Como resultado, se obtiene que existe un *acurracy* del 33% que a pesar de percibirse como que, disminuyó el desempeño del clasificador, se deben tener en cuenta las demás

métricas, donde la clasificación se encontraba correcta para la etiqueta VA, dejando las demás etiquetas con puntajes de 0 para *presión, recall y f1-puntaje* (ver tabla 37).

Tabla 38: Reporte Métricas Random Forest

Reporte de Métricas	Matriz de confusion

	precision	recall	f1-score	support
BI	0.22	0.38	0.27	204
CN	0.25	0.29	0.27	226
NUL	0.20	0.52	0.29	202
VA	0.77	0.28	0.41	788
accuracy			0.33	1420
macro avg	0.36	0.37	0.31	1420
weighted avg	0.53	0.33	0.35	1420

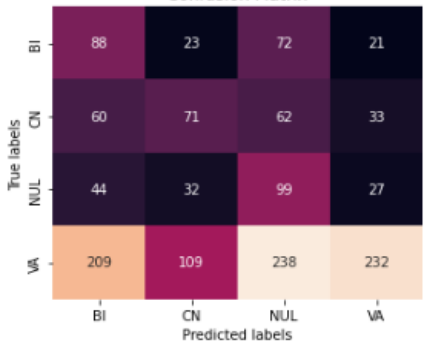


Una de las técnicas que se utilizan para mejorar el *accuracy* del *Random Forest* es agregar árboles extra, por lo que se utiliza la librería *ExtraTreeClassifier* el cual demuestra una mejora en el modelo de 2% en su *acurracy*. También se puede observar, que entre 2% a 6% las métricas de *f1-puntaje* y *recall* para todas las etiquetas del conjunto de datos (ver tabla 38).

Tabla 39: Métricas ExtraTreeClassifier

Reporte de Métricas	Matriz de confusion

	precision	recall	f1-score	support
BI	0.22	0.43	0.29	204
CN	0.30	0.31	0.31	226
NUL	0.21	0.49	0.29	202
VA	0.74	0.29	0.42	788
accuracy			0.35	1420
macro avg	0.37	0.38	0.33	1420
weighted avg	0.52	0.35	0.37	1420



Otras técnicas utilizadas para mejorar el clasificador es el ajuste de los hiperparámetros del modelo, para el caso de *Random Forest*, se tiene en cuenta que existen varios parámetros que se encuentran por defecto, por lo que se busca modificarlos y así buscar la optimización del algoritmo. Para esto, se utiliza *GridSearchCV* el cual ayuda a realizar una iteración de los diferentes opciones y combinaciones para los hiperparámetros por medio de *cross validation* (CV) ayudando así a seleccionar las mejores opciones.

Para esto, se busca mejorar los siguientes hiperparámetros:

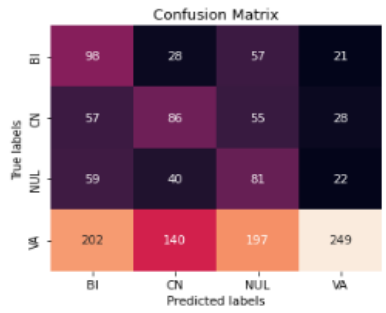
- max_features: número de características a tomar en cuenta a la hora de dividir el nodo
- max_depth: número de niveles de los árboles
- criterion
- min_samples_split: mínimo número de datos colocados en el nodo antes que el nodo se separe.
- n_estimators: número de árboles que se deben tener en cuenta
- min_samples_leaf: número de datos permitidos en cada hoja del nodo
- Bootstrap: método de muestreo de puntos de datos (con o sin sustitución)

Una vez realizada la iteración para buscar la mejor combinación de hiperparametros del modelo, se encuentra que se deben usar para *Random Forest* de este proyecto lo siguientes hiperparametros:

```
rfc1=RandomForestClassifier(random_state=53, max_features='log2',  max_d
epth=None, criterion='entropy', min_samples_split =5, min_samples_leaf =
 1, bootstrap = True)
```

Como resultado se observa que *accuracy* aumenta un 1% y el f1-puntaje también tiene un aumento de 3% para todas las clases, pero afectando precisión y recall, que disminuyen en un 2% respecto al *Random Forest* con extra árboles, estos resultados se pueden visualizar en la tabla 39.

Tabla 40: Reporte de métricas Random Forest

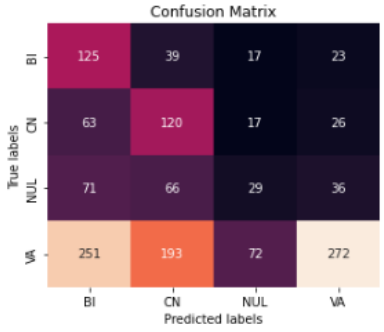
Reporte de Métricas	Matriz de confusion
precision recall f1-score support BI 0.24 0.48 0.32 204 CN 0.29 0.38 0.33 226 NUL 0.21 0.40 0.27 202 VA 0.78 0.32 0.45 788 accuracy 0.36 1420 macro avg 0.38 0.39 0.34 1420 weighted avg 0.54 0.36 0.39 1420	

4.4.1.3. Naives Bayes

Dentro del clasificador de Naives Bayes, se utilizará el modelo construido con TF-IDF y al igual que los clasificadores de *Logistic Regression* y *Random Forest*, se realizará una disminución del 40% del conjunto de datos, donde al separarlos en conjunto de pruebas de 1420 y un conjunto de entrenamiento de 2881 al cual se le aplica la técnica de *Undersampling* con el fin de mejorar el balanceo de clases.

Al construir este clasificador combinando TF-IDF y un balanceo de clases se obtiene un *acurracy* del 38%, donde la precisión solo tiene un porcentaje importante (76%) para el clasificador VA, y donde el f1-puntaje es menor al 50% para todas las etiquetas (ver tabla 40).

Tabla 41: Métricas para Naives Bayes

Reporte de Métricas	Matriz de confusion
precision recall f1-score support BI 0.25 0.61 0.35 204 CN 0.29 0.53 0.37 226 NUL 0.21 0.14 0.17 202 VA 0.76 0.35 0.48 788 accuracy 0.38 1420 macro avg 0.38 0.41 0.34 1420 weighted avg 0.53 0.38 0.40 1420	

Al igual que *Random Forest,* para realizar ajustes que ayuden al desempeño del modelo, se pueden realizar modificaciones en sus hiperparámetros. En el caso de NB, se busca modificar su *Alpha,* el cual, por medio de la figura 41 se puede observar que el mejor acurracy del 39.79% se da con Alpha 0.5, por lo que, al realizar dicha modificación al modelo se observa que el *acurracy* aumenta solamente en 1% y un cambio casi nulo en las demás métricas, por lo que no mejora significativamente el clasificador (ver tabla 41).

Figura 64: Alpha para NB

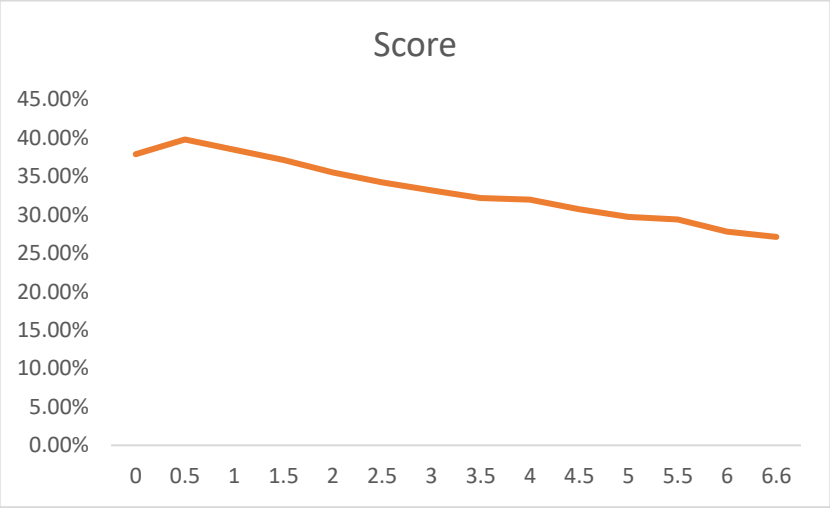
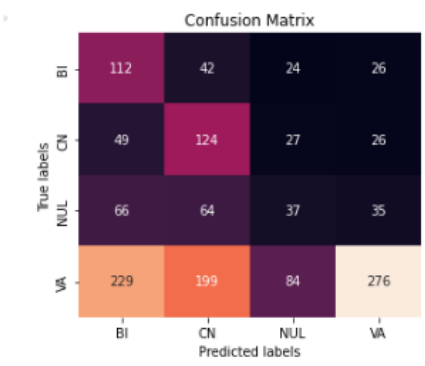


Tabla 42: Métrica para NB

Reporte de Métricas				Matriz de confusion
	precision	recall	f1-score	support
BI	0.25	0.55	0.34	204
CN	0.29	0.55	0.38	226
NUL	0.22	0.18	0.20	202
VA	0.76	0.35	0.48	788
accuracy			0.39	1420
macro avg	0.38	0.41	0.35	1420
weighted avg	0.53	0.39	0.40	1420



5. Conclusiones y Trabajos Futuros

5.1. Conclusiones

Dentro de este trabajo se ha propuesto una metodología para el análisis de discurso basado en lingüística computacional y técnicas de aprendizaje de máquina, aplicados al proyecto *TQHC,* el cual constaba de preguntas abiertas que eran contestadas por diferentes ciudadanos. Dentro de la metodología, se puede concluir que:

1. Se realizó una revisión de las metodologías, modelos y tecnologías basados en lingüística computacional y aprendizaje de maquina en NLP, en proyectos de opiniones populares, como lo son, chatbots, Tweeter, preguntas abiertas, y con el fin de conocer tópicos, análisis de sentimientos y clasificación de texto en diferentes etiquetas.

2. Existen varias limitantes en el preprocesamiento de datos en las técnicas de *lemmatization* y *stemming* en la librería de spaCy para idioma español, ya que se puede evidenciar que existen palabras que, a pesar de estar correctamente escritas, no las devuelve a su forma raíz. Este tipo de limitante afecta los modelos ya que aumenta la dimensionalidad del conjunto de datos, debido a que los tokens tienen múltiples variantes de la misma palabra, por lo que cada palabra en su forma original se considera una característica distinta, aunque tenga el mismo significado.

3. Para análisis de sentimientos no existe un cambio significativo al usar o no las *Stopwords*, también, se puede observar que los modelos no son afectados si se alimenta el modelo con stemming. Adicionalmente, los modelos creados con AWS dan mejores resultados que los obtenidos por *Vader* y *Textblob*, ya que AWS Comprehend ha sido entrenado con diferentes datos para el idioma español, utilizado técnicas avanzadas de aprendizaje automático, incluyendo análisis de emociones y detección de ironía y así ofrecer una compresión más profunda del texto y su sentimiento.

4. Dentro de las técnicas utilizadas para *POS Tagging* tanto el modelo utilizado en AWS como en SpaCy, muestran unos resultados óptimos al identificar verbos, sustantivos y pronombres. La eficiencia de los diferentes modelos muestra que el algoritmo es capaz de etiquetar con precisión y consistencia las palabras del texto e identificar correctamente los patrones de la gramática y escritura del idioma español.

5. En general, tanto los modelos NLTK, spaCy y AWS Comprehend, tienen varias limitantes para el idioma español en el reconocimiento de entidades, dando una confiabilidad del 40% únicamente.

6. Dentro de la creación de modelos para la clasificación de texto utilizando como etiqueta el indicador verbal, no se presentaron los mejores resultados, los cuales pueden ser dados por las limitantes del conjunto de datos de entrenamiento, ya que no cuenta con datos suficientes para que los clasificadores logren aprender correctamente las características para poder clasificarlo en el correcto indicador verbal. Sin embargo, la metodología, modelos y técnicas de evaluación de modelos son adecuados y darían resultados buenos en la medida que mejore la calidad de los datos de entrenamiento, por lo que en trabajos futuros puede ser reutilizado este proyecto para nuevos conjuntos de datos.

7. Las técnicas de *Oversampling* y *Undersampling* ayuda a mejorar el modelo un 10% frente a la métrica del *accuracy*, pero para obtener esta mejoría, es necesario el uso de estrategias que para el caso de NLP son *minority* y *mayority.*

8. Se construye una metodología basada en análisis de discurso, que se encuentra conformada por un conjunto de etapas, las cuales tienen técnicas y modelos para proyectos que requieran el análisis de texto de opiniones, o respuestas a preguntas abiertas.

9. Se aplicó la metodología al proyecto *TQHC*, logrando tener un entendimiento completo del conjunto de datos, y encontrando información gracias al EDA y creación de modelos, que puede ser utilizada por los analistas de este proyecto.

5.2. Trabajo futuro

A continuación, se listan varias líneas de trabajo futuro que no fueron desarrollados dentro de este proyecto debido a que se encontraban fuera del alcance de este trabajo.

1. Realizar modelos de Clustering con el fin de poder tener un mejor agrupamiento de las respuestas dadas por los ciudadanos y así verificar los indicadores verbales, para luego poder correr nuevamente los modelos de clasificación.

2. La metodología de este proyecto está dada en secuencia de pasos la cual tiene conjunto de técnicas que pueden ser utilizadas para cualquier proyecto de análisis de discurso, por lo que se propone utilizar dicha metodología en un conjunto nuevo.

3. Utilizar work2vec para predicción de texto y evitar errores de digitación o gramática.

6. Referencias

1. Microsoft Documents. (2021). *What is the Team Data Science Process?* Recuperado de https://docs.microsoft.com/en-us/azure/architecture/data-science-process/overview
2. Hotz N. (2021). *Methodologies for data mining and data analytics.* Data Science Process Alliance.
3. Huber S, Wiemer H. (2019). *DMME: Data mining methodology for engineering applications – a holistic extension to the CRISP-DM model.* Recuperado de: https://www.sciencedirect.com/science/article/pii/S2212827119302239
4. Shfique U, Qaiser H. (2014). *A Comparative study of Data Mining process model (KDD, CRISP-DM, SEMMA)*. International Journal of innovation and scientific Research pp 217-222.
5. Brookshear J. (1993). *Teoría de la computación*. Wesley iberoamericana Wilmington Delaware.
6. Vicente M, Barros C, Lloret E, Peregrino S. (2007). *La generación de lenguaje natural*. Universidad de Alicante, Departamento de lenguajes y sistemas informáticos, (1-13).
7. Cortez A, Vega H, Pariona J. (2009). *Procesamiento Lenguaje Natural*. Revista de investigación Universidad nacional Mayor de san Carlos (45-54).
8. Firti A, Rachamadita A, Muhammad A. (2019). *Sentimental analysis of social media twitter with case of anti LGTB Campaign in Indonesia using Naïve Bayes, Decision Tree and Random Forest Algorithm*. La quinta conferencia internacional sobre sistemas de información. (765-772)
9. Patel R. (2017). *Sentimental Analysis on Twitter Data Using Machine Learning*. Laurentian University, Ontario Canada.
10. M. Kanakaraj,. R. Guddeti, (2015). "NLP based sentiment analysis on Twitter data using ensemble classifiers," 2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN),
11. Sankar D. (2018). *Understanding Language Syntax and Structure*: *A Practitioner's Guide to NLP*. Guia practica para el procesamiento de lenguaje natural.
12. Webster J. (1992). *Tokenización as the initial phase in NLP.* Universidad de Hong Kong, 14 conferencia de lingüística computacional. Vol 4.
13. Kyubyong P, Jooong L. (2020). *An Emperical Study f tokenización Strategy for Various Korean NLP Tasks*. Universidad de ciencias y tecnologías Pohang.
14. Mayo M. (2017). *A general approach to preprocessing text data, Data preparation*. Recuperado de https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html
15. Mayo M. (2017). *A Framework for Approaching textual data science task, Modeling NLP*. Recuperado de https://www.kdnuggets.com/2017/11/framework-approaching-textual-data-tasks.html

16. Spark Documentation 2.1.0 (2021). *HashingTF and CountVectorizer.* Recuperado de: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.HashingTF.html
17. S. Albawi, T. A. Mohammed y S. Al-Zawi. (2017). *Understanding of a convolutional neural network.* International Conference on Engineering and Technology (ICET),
18. T Tanprasert, D Kauchak (2021). *Flesch-kincaid is not a text simplification evaluation metric.* Association for Computational Linguistics. Recuperado de: https://aclanthology.org/2021.gem-1.1
19. Maldonado S, Lopéz J, Vairetti C (2019). *An alternative SMOTE oversampling strategy for high-dimensional datasets.* Universidad de los Andes. Facultad de Ingenieria y ciencias aplicadas. Chile.
20. Jurfsky D, Martin H (2023). *Speech and Language Processing.* Standnford. *N-gram Lenguage Models*
21. Leland M, Healy J, Melville J (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.* Cornell University, Recuperado de: https://arxiv.org/abs/1802.03426
22. Raghavan, P. y Schütze, H. (2008). *Stemming and lemmatization. Introduction to information retrieval*. Cambridge: Cambridge University Press.
23. Khanna C. (2021). *A quick guide on choosing the right text normalization approach.* Recuperado de https://towardsdatascience.com/stemming-lemmatization-which-one-is-worth-going-for-77e6ec01ad9c
24. Alkula R. (2021). *Stemming and lemmatization in the clustering of Finnish text documents.* Conferencia internacional de informacion y conocimiento.
25. Mohan V. (2020). *Preprocessing Techniques for Text Mining - An Overview*. Bharathiar University.
26. Sarica S, Luo J (2021). *Stopwords in technical language processing*. Publicación en revista PLOS ONE. Página 16.
27. Wu C, Luk R, Wong K, Kwok K. (2008). *Interpreting TF-IDF term weights as making relevance decisions*. ACM Transactions on Information Systems, (3 - 26).
28. Drikvandi R, Lawal O. (2020). *Sparse Principal Componenet Analysis for Natural Language Procesing*. Annals of Data Science. Artículo en Springer.
29. Dhamija V. (2019). *CountVectorizar and HashingTF*. DataScience Toward Data Sceince. Recuperado de https://towardsdatascience.com/countvectorizer-hashingtf-e66f169e2d4e
30. Canavate (2020). *Concept embedding*. Artículo de Universidad politécnica de Madrid.
31. Karani D. (2018) *Introducción a Word Embedding y Word2Vec*. Toward Data Science, Recuperado de: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa
32. Lavin, Matthew J. (2020) *Analyzing Documents with TF-IDF*. Artículo en Programming Historian.
33. Wang, Xian-Jia, Guan-Tian, Ke-Xin, Hai-Bo, Zhen-Song. (2020). *Urban Real Estate Market Early Warning Based on Support Vector Machine*: A Case Study of Beijing. International Journal of Computational Intelligence Systems.

34. Kapadia S. (2019). *NLP with LDA latent Direchlet Allocation and text clustering to improve classification*. Toward Data Science. Recuperado de https://towardsdatascience.com/nlp-with-lda-latent-dirichlet-allocation-and-text-clustering-to-improve-classification-97688c23d98
35. F. Heimerl, S. Lohmann, S. Lange and T. Ertl. (2014). *Word Cloud Explorer: Text Analytics Based on Word Clouds.* Hawai 47 conferencia internacional en ciencias y sistemas. pp. 1833-1842.
36. Thorn J, (2019). *Deep Learning for NLP: ANNs, RNNs and LSTMs.* Recuperado de: https://towardsdatascience.com/deep-learning-for-nlp-anns-rnns-and-lstms-explained-95866c1db2e4
37. Narkhede S, (2018). *Understanding Confusion Matrix.* Blog Amazon Web Services. Recuperado de: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
38. Kulkarni H (2021). *Evaluation a classification Model for data science*. Article for Data science Blogathon.
39. Blaney J, Romanello M, Philpott M. (2019). *An introduction to Natural Lenguage Processing*. School of Advanced Study University of London.
40. Scott W. Towar Ds Datascience (2019). *TF-IDF from scratch in python on a real-world dataset*. Towar Data Science. Recuperado de: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089
41. Aggarwal A, Singh J, Gupta K. (2018). *A Review of different text categorization techniques*. National institute of technology Kurushetra. International Jour of engineering and technology (11-15).
42. Doumit S, Minai A. (2012) *Online New Media Bias Analysis Using LDA-NLP Approach*. Universidad de Cincinnati.
43. Verma Y. (2021). *A Guide to Hidden Markov Model and its Applications in NLP*. Developers Corner Magazine.
44. Ahmed, K., El Tazi, N., & Hossny, A. (2015). *Sentiment analysis over social Networks: An overview*. Articulo presentado en IEEE para la conferencia internacional de sistemas y cibernética.
45. Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., & Xu, K. (2014). *Adaptive recursive neural network for target-dependent twitter sentiment classification*. Articulo presentado en la 52 reunión anual de procesos de la asociación lingüística computacional. Vol 2.
46. Sarker, I. (2021). *Comprehensive Overview from Neural Network and Deep Learning Perspective*. Artículo publicado en SN COMPUT. SCI. 2, 154
47. Rodriguez H, Sidorov G, Escamilla P (2020). *Estado del arte en técnicas de procesamiento de lenguaje natural para análisis de Malware*. Instituto politécnico nacional. Mexico.
48. Vaca A (2021). *Transformer en procesamiento del lenguaje natural*. Artículo de conocimiento. Instituto de ingeniería del conocimiento.

49. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez N, Kaiser L, Polosukhin I. (2017). *Attention isa ll you need*. Google Research. Recuperado de: https://arxiv.org/abs/1706.03762
50. Kltaev N, Kalse L y Levskaya A (2020). *Reformer: The efficient transformer*. Google research. Recuperado de: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fopenreview.net%2Fpdf%3Fid%3DrkgNKkHtvB&clen=597053&pdffilename=reformer_the_efficient_transfo.pdf
51. Fischer, A., Igel, C. (2012). *An Introduction to Restricted Boltzmann Machines*. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP 2012. Lecture Notes in Computer Science, vol 7441. Springer, Berlin.
52. Rae J, Potapenko A, Jayakumar S, Hiller C y Lillercrap T (2019). *Compressive transformer for long range sequence modelling. Deep Mind London UK*. Computer science, University College London UK. Recuperado de: https://arxiv.org/abs/1911.05507
53. Webster J, Kit C (1992). *Tokenización as initial phae in NLP*. Universidad politécnica de Hong Kong.
54. Hobson L, Cole H, Hannes M. H. (2019). *Natural Lenguage Processing in Action*. Undestanding analyzing and generating text with Python. Chapter 4.
55. Kapadia S. (2019). *Topic Modeling in Python: Latent Dirichlet Allocation (LDA)* recuperado de: https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0
56. Documentación AWS (2022). *API Detect sentiments.* Recuperado de: https://docs.aws.amazon.com/comprehend/latest/APIReference/API_DetectSentiment.html
57. Documentación AWS (2022). *API Batch Detect Entities.* Recuperado de: https://docs.aws.amazon.com/comprehend/latest/APIReference/API_BatchDetectSyntax.html
58. Shahul E (2022). *Exploratory Data Analysis for Natural Language Processing: A Complete Guide to Python Tools*. Recuperado de: https://neptune.ai/blog/exploratory-data-analysis-natural-language-processing-tools
59. Shoku P (2021). *Textstat, How to evaluate readability?* Common lit readability prize. Recuperado de: https://www.kaggle.com/code/yhirakawa/textstat-how-to-evaluate-readability
60. Brownlee J (2020). *Random Oversampling and Undersampling for imbalanced classification.*Recuperado de: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/
61. Prusa J, Khoshgoftaar T, Dittman D, Napolitano A. (2015). *Using Random Undersampling to Alleviate Class Imbalance on Tweet Sentiment Data.* International Conference on Information Reuse and Integration, San Francisco.
62. Scikit-learn 1.2.1. *Tuning the hyper-parameters of an estimator.* Recuperado de: https://scikit-learn.org/stable/modules/grid_search.html

63. Fernández J (2021). *Desarrollo de técnicas basadas en Frameworks Deep Learnin para la caracterización de galaxias distantes.* Universidad de Valladolid. Recuperado de: https://uvadoc.uva.es/handle/10324/50236