UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA ELECTRÓNICA INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

TESIS

"ANÁLISIS MASIVO DE DATOS EN TWITTER PARA IDENTIFICACIÓN DE OPINIÓN "

PARA OPTAR AL TÍTULO PROFESIONAL DE:

Ingeniero Informático y de Sistemas

PRESENTADO POR:

Br. Albhert Edison Olarte Chullo

Br. Alvaro Ciro Casaverde López

ASESOR:

Mg. Julio Cesar Carbajal Luna

CO-ASESOR:

Mcs student. Errol Wilderd Mamani Condori

Cusco - Perú

2020

Dedicatorias

El presente trabajo esta dedicado a mi familia por haber sido mi apoyo a lo largo de toda mi carrera universitaria y a lo largo de mi vida. A todas las personas especiales que me acompañaron en esta etapa, aportando a mi formaciónn tanto profesional y como ser humano.

Alvaro Ciro Casaverde López

A Dios por darme el talento y pasión por estudiar, a mis padres quienes me dieron vida, educación, apoyo y consejos. A mis compañeros de estudio, a mis maestros y amigos, quienes sin su ayuda nunca hubiera podido hacer esta tesis. A todos ellos se los agradezco desde el fondo de mi alma. Para todos ellos hago esta dedicatoria.

Albhert Edison Olarte Chullo

Abreviaturas

SA Sentiment Analysis

ML Machine Learning

NLP Natural Language Processing

SVM Support Vector Machine

ANN Aritifitial Neuronal Networks

NB Naive Bayes

Resumen

Las redes sociales están jugando un papel muy importante en nuestra sociedad, y los microblogging es una parte importante de la comunicación hoy en día. Esto permite que los usuarios puedan publicar una opinión acerca de un determinado tema haciendo uso de internet y los sitios web en un repositorio grande de información. Las redes sociales como twitter, google+, facebook y whatsapp, contienen cantidad de publicaciones en sus sitios web. Esto hace de estas plataformas una fuente para exploración de información haciendo uso de métodos de Inteligencia Artificial, debido a la masiva cantidad de información que en casos como twitter llegan a ser millones por día alrededor del mundo y hoy necesitamos aprovechar esta vasta información para interpretar los datos con la finalidad de saber cuantas opiniones son realizadas positivamente y cuantas son negativas.

Sin embargo se carece de alguna herramienta capaz de hacer un análisis de todos estos comentarios y determinar una polaridad(si una opinión es positiva o negativa). Para esto se propone desarrollar una herramienta de identificación de texto en twitter usando técnicas de aprendizaje automático y procesamiento del lenguaje natural(Natural Language Processing (NLP)), aplicando procesamiento basado en texto con análisis de sentimientos Sentiment Analysis (SA) y técnicas de Machine Learning(Support Vector Machine (SVM)).

Finalmente se presentará una arquitectura desarrollado en python usando algoritmos y un modelo de Aprendizaje Automático, capaz de determinar una polaridad(opinión positiva o negativa). Además presentamos el análisis de estos datos y el procedimiento para realizar con fines de investigación para el idioma español y en particular se hicieron pruebas con el caso de estudio en Perú obteniendo una precisión con la métrica F score de 81 % en efectividad de reconocimiento de opiniones.

Palabras Clave: *NLP*, *Análisis de sentimiento*, Aprendizaje automático, *twitter dataset*.

Abstrasct

Social networks are playing a very important role in our society, and microblogging is an important part of web communication. This allows users to post an opinion about a certain topic. So social networks such as twitter, google +, facebook and whatsapp, provide a lot of information.

However, there is no tool capable of analyzing all these comments and determining a polarity (if an opinion is positive or negative). For this, it is proposed to develop a text identification tool on twitter using automatic learning techniques and natural language processing (NLP), applying text-based processing with SA and Neural Networks.

Finally, a tool capable of determining a polarity on a conjunctural issue and delimiting it to problems of the region will be presented.

Key words: *NLP*, *Sentiment Analysis*, Machine learning , *Twitter DataSet*.

Índice general

Lista de Figuras

Lista de Tablas

Introducción

Los seres humanos desde que logramos tener nuevas formas de comunicación hemos hecho el uso masivo de estas tecnologías. Por otro lado la creciente influencia de las redes sociales en nuestra moderna sociedad, cantidades enormes de información como contenido de texto han sido generados por usuarios en la emergente tecnología de internet. Además, del intercambio de ideas, nosotros hemos visto un crecimiento de contenido en los medios sociales que ha propiciado que la comunidad científica dediquen grandes esfuerzos a analizar, estructurar y procesar esta información. Estos medios se utilizan para expresar opiniones y sentimientos diversos sobre diferentes aspectos de la sociedad como productos, servicios, aficiones; por eso, las empresas, organizaciones, gobiernos y diferentes colectivos han mostrado su interés en la recolección de opiniones y sentimientos que los usuarios tienen acerca de una actividad determinada.

En los últimos años muchas empresas han intentado explorar la oportunidad de que ofrecen las redes sociales como twitter, además de la naturaleza de contenido (tweets) que esta plataforma ofrece que es la introducción de textos pequeños, plantea nuevas formas de análisis y las técnicas usadas por el procesamiento de lenguaje natural tienen que adaptarse a este entorno muy ruidoso. Por otro lado el análisis de opiniones(sentiment Analysis SA), debemos indicar que el estudio de esta tarea no es nueva mas bien, viene siendo estudiado desde hace una década atrás para el inglés, algunos trabajos pioneros en esta área fueron realizados a través de Aprendizaje Automático (Pang et al., 2002). Otros sin embargo, desde aproximaciones basadas en conocimiento y métodos de clasificación no supervisada (Turney, 2002), principalmente se enfocan en determinar la polaridad, es decir si una opinión es positiva o negativa, muchos de estos se aplicaron sobre opiniones de películas y en posteriores investigaciones se realizaron haciendo combinaciones de estos métodos.

Los trabajos para el español y en redes sociales son mucho mas recientes, para conseguir identificar la opinión expresada en contenido de texto con base en la plataforma twitter es una tarea que representa un reto actual y mucho más aún si consideramos esto mismo para nuestro

entorno local. En los siguientes capítulos vamos primero a definir los aspectos generales, en el Capítulo 1, luego introducimos conceptos relacionados a trabajos en el Capítulo 2. Finalmente desarrollamos la arquitectura en el Capítulo 3 y mostramos nuestros resultados en el Capítulo 4.

Parte I

ASPECTOS GENERALES

Capítulo 1

Aspectos Generales

1.1. Antecedentes

En el procesamiento del lenguaje natural en base a texto está muy extendido especialmente en el análisis de datos masivos en twitter, actualmente existen estudios en inglés de análisis de sentimientos iniciando con la tarea de clasificación de documentos; algunos de estos estudios son basado en el análisis de opinión pública y tenemos los siguientes:

- ***"Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment"*** Tumasjan, Andranik and Sprenger, Timm O and Sandner, Philipp G and Welpe, Isabell M - **Technische Universität München Munich 2010** (Tumasjan et al., 2010).

  Este estudio utiliza el contexto de la elección federal alemana para investigar si Twitter se usa como un foro para la deliberación política y si los mensajes en línea en Twitter reflejan de manera válida el sentimiento político. Usando el software de análisis de texto, realizan un análisis de contenido de más de 100,000 mensajes que contenían una referencia a un partido político. Sus resultados muestran que Twitter se usa ampliamente para la deliberación política. Encontrando que la cantidad de mensajes que mencionan un partido refleja el resultado de la elección. Además, las menciones conjuntas de dos partidos están enlazados y las alianzas políticas del mundo real. Lo que indica que el contenido de los mensajes de Twitter refleja plausiblemente el panorama político local.

- ***"Sentiment analysis algorithms and applications: A survey"*** Medhat, Walaa and Hassan, Ahmed and Korashy, Hoda- **Cairo Campus of CBU, Egypt**: Medhat et al.

(2014)

Este estudio analiza un enfoque en el que un flujo publicitado de tweets desde el sitio de microblogging de Twitter se procesa y clasifica según su contenido emocional como positivo, negativo e irrelevante; y analiza el rendimiento de varios algoritmos de clasificación en función de su precisión y recuperación en tales casos. Además, el documento ejemplifica las aplicaciones de esta investigación y sus limitaciones. En síntesis es un estudio del medio social Twitter para identificar si el contenido es positivo, negativo o irrelevante.

- ***"Minería de opinión y Análisis de Sentimientos"*: Tesis** FELIPE Ignacio Oliva VAldebenito, PUCP, lima Perú VALDEBENITO (2014)

  Este estudio esta enfocado en técnicas para poder clasificar las opiniones y poder determinar si es positiva o negativa, es decir, para ello se plantean distintos escenarios para observar el comportamiento según la representación y clasificación que se utilice.

- ***"Análisis de sentimiento en Twitter: El bueno, el malo y el :("*: Tesis**, Bacerra Martin ,**Universidad de Cordoba,Argentina,2017** Becerra (2017)

  Este trabajo de tesis muestra los pasos necesarios para la recolección de datos mediante twitter y poder armar su propio corpus, además describe los métodos necesarios para efectuar el reconocimiento de opiniones como un caso de Análisis de Sentimientos para el idioma Español - Argentino, usando un método de agrupación *clustering* con un método de aprendizaje no supervisado, pero que sienta las bases para un mejor análisis de sentimientos para el español.

- ***"Análisis de Sentimiento en Twitter de las principales Compañías del Sector Asegurador Español"*** , Francisco José Martínez Martínez, **Universidad de Valencia, España, 2017** Martínez (2017).

  En este trabajo se realiza un análisis de sentimiento en Twitter de las principales compañías del sector asegurador español (PCSAE) mediante la utilización del software estadístico R y de léxicos de sentimiento. Obtener y comprender la gran cantidad de información gratuita que circula por Internet puede ser de gran interés para las empresas

También tenemos que indicar que se encontró páginas web de desarrollo de este tipo de software pero que no ofrecen información adicional sobre el esquema que realizan para el desarrollo del mismo para el idioma en español.

1.2. Problema de Investigación

Recolectar información pública, reconocer la opinión de un grupo de personas, identificar si es positiva o negativa con respecto a un asunto: Estas operaciones que antes nos parecían difíciles y complejas tareas de sociedades y grupos con arduos procesos que llevan días o meses. Este tipo de identificación juega un papel relevante en el ejercicio del poder y en la organización y regulación de las sociedades, esto asu vez ha llevado al desarrollo y evolución de pueblos. Entonces en la actualidad, se tiene una gran cantidad de formas de identificar la opinión pública de las personas con respecto a un asunto de interés social como: sensos, referendus, consultas; de tal modo que nació un nuevo campo para explorar y que estos procesos puedan ser cada vez mas automáticas y sobre todo rápidas y accesibles. El procesamiento de lenguaje natural basado en texto es una área encargada de procesar, analizar y recolectar patrones de información.

El aprendizaje automático por otro lado es un sub área de la ciencias de la computación cuyo objetivo es desarrollar técnicas (programas) que permiten a las computadoras aprender, de forma mas concreta, el procesamiento de lenguaje natural junto con el aprendizaje automático se centran en desarrollar herramientas que analizan información del lenguaje, para así aprender el comportamiento de estos y en un futuro poder predecir ciertos rasgos o tendencias.

En la actualidad uno de los medios mas usados para la interacción de muchas personas y usuarios en internet son las redes sociales y quizá uno de los más concurridos para publicar opinión como una forma de expresión es twitter; así en este proyecto se plantea el uso de este medio para recolectar publicaciones para luego analizarlos e identificar si una opinión es positiva o negativa.

1.2.1. Planteamiento del problema de investigación

¿Cómo implementar una arquitectura para el análisis masivo de datos en twitter para la identificación de opinión positiva o negativa?

1.3. Justificación

Con el incremento de la influencia de medios sociales en nuestra moderna sociedad, una larga cantidad de usuarios comentan, publican y exponen su opinión en internet. Además

de tener intercambio de ideas, vemos un exponencial crecimiento de opinión respecto a cuestiones políticas, salud, educación y eventos relevantes que conciernen a nuestro país. Por lo tanto analizar la opinión pública es sumamente importante para la toma de decisiones y sobre todo tener una forma mas rápida de índice de aceptación con respecto a un tema, pero nuestros métodos convencionales como encuestas y consulta general consumen mucho tiempo y un esfuerzo humano que lleva a costos monetarios considerables. Entonces surge la necesidad de tener este tipo de información de opinión de forma fácil, accesible, automática y en tiempo real.

Con la ayuda del Aprendizaje Automático ahora podemos hacer una análisis de este tipo de información, procesarla y predecir sentimientos ya sea positivo o negativo; ayudando a resolver este problema de impacto social en la toma de decisiones en las organizaciones públicas y privadas. En aplicaciones más prácticas se puede tener una analizador para medios publicitarios y ver la aceptación de un producto, también en una gestión pública y el impacto que esta presentando en una sociedad o grupo a nivel nacional o regional.

A nivel computacional, en el Perú se ha desarrollado pocas investigaciones de este tipo para el idioma español y la arquitectura utilizada puede servir para futuros desarrollos prácticos de software de análisis o la posibilidad de tener implementado un sistema de censos digitales captando la opinión en internet o consulta popular.

1.4. Objetivos

1.4.1. Objetivo General

Implementar una arquitectura para el análisis masivo de datos en twitter para la identificación de opinión.

1.4.2. Objetivos Específicos

- Elegir el DataSet adecuado de recolección de información de medios sociales.
- Implementar modelo de pre procesamiento de texto y su manipulación de datos.
- Aplicar e implementar el algoritmo de Máquinas de Soporte Vectorial.
- Analizar y proponer una configuración de los parámetros necesarios de la arquitectura propuesta que nos permita identificar la opinión positiva o negativa.

1.5. Alcances y Limitaciones

1.5.1. Alcances

- El identificador hará uso de un DataSet extraído de las publicaciones de la red social Twitter.
- En este proyecto no se considera el análisis semántico, es decir se eliminan artículos y conectores.
- El identificador será implementado con una técnica de aprendizaje automático (Maquinas de Soporte Vectorial).

1.5.2. Limitaciones

- Según el estado de arte actual, existe pocos trabajos relacionados en la aplicación a la región y en el idioma español.
- La identificación es más directa por palabras raíz y la relación entre ellas, no se hace un análisis mas detallado de la sintaxis debido a que los mensajes de Twitter presentan mucho ruido(falta ortográfica y gramatical).
- El identificador será implementado como prototipo y fines de investigación, no como producto de software final.

1.6. Metodología

1.6.1. Método de Investigación

Este proyecto de investigación es de alcance exploratorio, por que este tipo de problemas ha sido poco estudiado y abordado antes, menos aún para el idioma español y solo tenemos bases de estudios realizados para el idioma inglés Hernández Sampieri et al. (2014). Por otro lado este tipo de metodología es aplicable a medios sociales y partiendo desde este punto implementar un método específico a un desarrollo de machine learning y deep learning no es posible, así que no se puede aplicar métodos tradicionales, por que hacemos uso de datos de entrenamiento y exploración de nuevas arquitecturas para encontrar patrones y parámetros que no se parecen en nada a los métodos tradicionales. Por otro lado, para tener un guía de

referencia mas plausible para este tipo de desarrollo propuesto tomaremos en cuenta el siguiente método propuesto por Carlos Enrique Perez (2016) para la implementación del identificador de opinión en twitter.

De la figura 1.1 podemos describir los pasos a seguir para el desarrollo de esta investigación como :

1. Define el problema y sus características para luego seleccionar el modelo y la arquitectura adecuada para la identificación de opinión a través de texto basado en publicaciones de Twitter.
2. Realizar la configuración en el modelo de aprendizaje automático utilizando IA, también se realizará el pre-procesamiento de los datos.
3. Entrenar datos y realizar el feedback(retro alimentación) de acuerdo a la obtención de error.
4. Verificar esta prueba de error, si esta excede, entonces se procede a crear una arquitectura mas grande o se hace un reajuste de los parámetros y ver el error otra vez.
5. Verificar si nuestro error es mínimo y entonces continuamos con el entrenamiento y pasamos a la validación de los mismos.
6. Si en el paso quinto el error es alto entonces volvemos al paso tres, que indica que necesitamos mas data, o reajustamos el conjunto de datos del training.
7. Si el error en el paso quinto no es alto, hacemos la validación con los otros datos.
8. Si la cantidad de desaciertos es alta en el paso anterior, entonces buscaremos un dataset de prueba para volver al paso quinto o se puede filtrar los datos para volver al tercer paso.
9. Si los desaciertos es bajo en el paso séptimo, entonces probamos con los datos de prueba.
10. Si existe desaciertos altos en el paso anterior, se busca mas datos de validación y volvemos al paso séptimo.
11. Si los desaciertos es bajo en el noveno paso, podemos tomar como un modelo de arquitectura con sus respectivos hiper-parámetros.

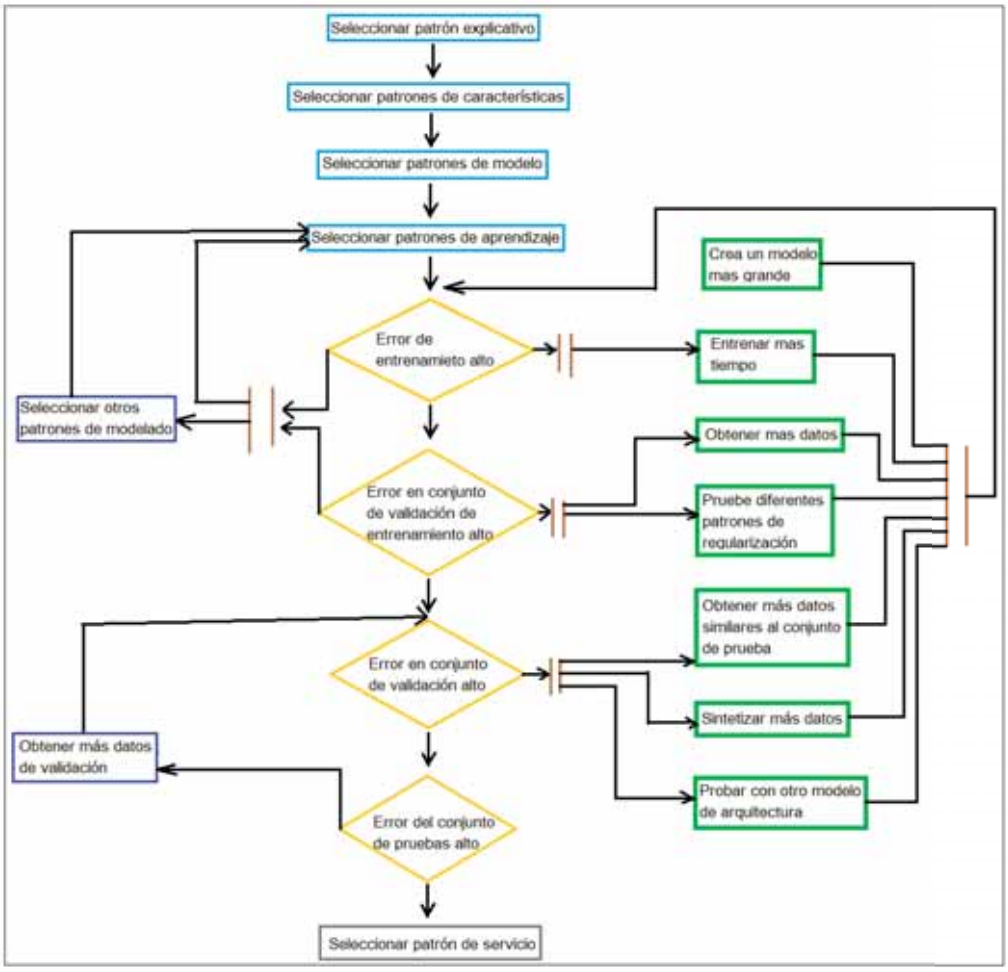


Figura 1.1: Diagrama de la metodología de aprendizaje automático adaptado por Carlos Enrique Perez (2016).

1.7. Cronograma de actividades

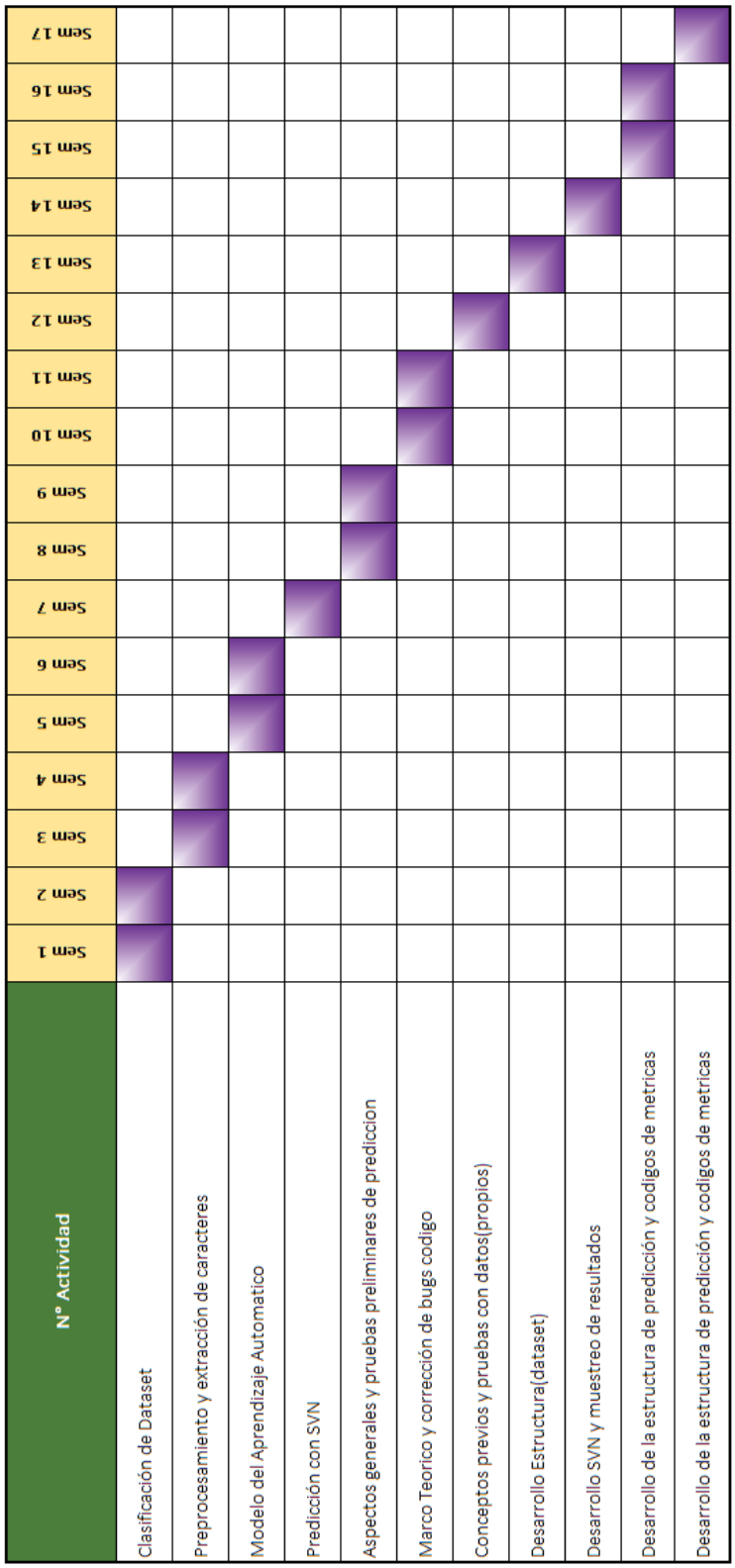


Figura 1.2: Cronograma de actividades, task(tarea), formato de periodos por semanas.

Parte II

MARCO TEÓRICO

Capítulo 2

Marco teórico

2.1. Marco Teórico

2.1.1. Mineria de Opinión

La minería de opinión, es el campo de estudio que analiza la opinión de la gente, sentimientos, evaluaciones, apreciaciones, aptitudes y emisiones a través de entidades tales como productos, servicios, organizaciones, individuales, problemas, eventos, temas y sus atributos. Esto representa un largo espacio donde hay muchos nombres y se resaltan diferentes tareas por ejemplo: extracción de opinión; sin embargo no todos están dentro de análisis de sentimientos, este último es mas comúnmente utilizado en procesamiento del lenguaje natural, análisis de texto, lingüística computacional que estudia los efectos del estado de información. El análisis de sentimiento es ampliamente usado también en la extracción de información para determinar si una opinión expresada es un sentimiento positivo o negativo Zhao et al. (2010). Aunque el Procesamiento del Lenguaje natural tiene una larga historia, pocos estudios han invertido esfuerzos en el análisis de opinión y sentimientos antes del 2000. Desde entonces ya este campo de investigación es mas activo.

2.1.2. Aprendizaje Automático

Aprendizaje automático, es un tipo nuevo de tecnología muy utilizada hoy en día, vamos a definir esto formalmente tomando como referencia algunos autores quienes intentaron definir de alguna manera, para esto debemos remontarnos a los inicios. Hasta ahora no hay

un definición aceptada como general y única, pero Basta con entender la idea práctica y el aporte de esta nueva tecnología. Debemos indicar también que incluso algunos entre aficionados y expertos indican que no hay una definición exacta al respecto debido a la gran cantidad de tareas y tentativamente observamos que existen numerosos métodos y campos de estudio relacionados y multidisciplinarios.

Una definición muy útil es la de Arthur Samuel(pionero de Inteligencia Artificial y profesor de Stanford), quien nos dio una definición de lo que es Machine Learning(aprendizaje automático en español). El lo definió como el campo de estudio que le dá a las computadoras la habilidad de aprender sin ser explícitamente enseñado, lo que quiere decir es que las computadoras pueden ser capaces de aprender sin darles una definición en cada instante.

Ciertamente esta definición muy generalizada de Samuel, no fue aceptada hasta que en los años 50 se lanzo a la fama debido a que Arthur programó un juego por computadora para jugar damas(similar al ajedrez con tablero bicolor) y lo que sorprendió fue que el algoritmo fue capaz de aprender cuales son las posiciones que se pueden jugar dándole la mejor posibilidad de ganar o tener un mejor resultado, y cuales son las posiciones que no debería tomar, pues le darán como perdidas. Este mismo juego en estas condiciones no sería muy asertivo e increíble para su época, sin la posibilidad de aprendizaje, así que lo que Samuel empleó en estos algoritmos fue la capacidad de que el algoritmo practicará contra él y así predijera su error para mejorar su nivel de juego, eventualmente el juego mismo pudo adquirir experiencia y aprender en cada una de los encuentros que tuvo, incluso podía hacerlo consigo mismo dándole la suficiente experiencia como para poder vencer a Samuel mismo, esto abrió muchas posibilidades debido a que la capacidad computacional brinda la posibilidad de juego indefinidos y en unidades de tiempo muy rápidos, lo que un humano tardaría, incluso años.

La definición anterior, de alguna forma es mas generalizada; una definición mucho mas reciente ofrecida por Mitchell (2019) y referenciada por Andrew Ng(profesor de la universidad de Stanford y lider del team Google Brain), dice que un problema de aprendizaje bien planteado es definido como: un programa de computadora que aprende de la experiencia E con respecto a alguna tarea T y alguna medida de rendimiento P y si su rendimiento en la tarea T como medida por P mejora con la experiencia E. Este mismo aplicado otra vez a nuestro ejemplo del algoritmo que juega damas seria: La experiencia E vendría a ser la experiencia de tener que jugar por sí mismo decenas de miles de juegos con e objetivo de mejorar su jugada. Así mismo la tarea T para este mismo seria la de jugar damas y la medida de rendimiento P seria la probabilidad de ganar el siguiente juego de damas contra su nuevo oponente.

2.1.3. Tipos de Aprendizaje Automático

Hoy en día la inteligencia artificial(IA) está en su mejor época, debido a la gran mayoría de tecnologías emergentes que ahora poseen módulos que les permiten interactuar de forma más autónoma. Dentro de las múltiples áreas que comprende la inteligencia artificial, se encuentra la de aprendizaje automático o Machine Learning (ML) por su traducción en inglés.

Los algoritmos de ML pretenden que las computadoras aprendan a tomar decisiones sin la necesidad de ser programadas explícitamente. Es por ello que hoy en día podemos escuchar acerca de autos de conducción autónoma, agentes virtuales de atención al cliente (chatbots), sistemas de recomendación y recolección de datos (Netflix, Google, Facebook).

El ML esta involucrado en muchas tareas y por la cantidad de tareas y relaciones con otras áreas de estudio fue necesario clasificarlos de tal manera que pueda ser mas especifica el campo de estudio como parte del desarrollo de este campo de estudio, esta se dividen dependiendo de las necesidades del problema, el ambiente en el que se van a desenvolver y los factores que afectarán en la toma de decisiones, podemos encontrar distintos tipos de algoritmos de aprendizaje, entre los cuales vamos a hablar de 3 de ellos: supervisado, no supervisado y por refuerzo (basado en el sitio web Medium G.).

2.1.3.1. Aprendizaje supervisado (Supervised ML)

En los algoritmos de aprendizaje supervisado se genera un modelo predictivo, basado en datos de entrada y salida. La palabra clave "supervisado" viene de la idea de tener un conjunto de datos previamente etiquetado y clasificado, es decir, tener un conjunto de muestra el cual ya se sabe a qué grupo, valor o categoría pertenecen los ejemplos. Con este grupo de datos, el cual llamamos datos de entrenamiento, se realiza el ajuste al modelo inicial planteado. De esta forma es como el algoritmo va "aprendiendo" a clasificar las muestras de entrada comparando el resultado del modelo, y la etiqueta real de la muestra, realizando las compensaciones respectivas al modelo de acuerdo a cada error en la estimación del resultado. Por ejemplo, el aprendizaje supervisado ha sido utilizado para la programación de vehículos autónomos. Algunos métodos y algoritmos que podemos implementar son los siguientes:

- K vecinos más próximos (K-nearest neighbors).
- Redes neuronales artificiales (Artificial neural networks).

- **Máquinas de vectores de soporte** (Vector Support Machines).
- Clasificador Bayesiano ingenuo (Naïve Bayes classifier).
- Árboles de decisión (Decision trees)
- Regresión logística (Logistic regression).

Figura 2.1: Ejemplo de como funciona el tipo aprendizaje supervisado

El aprendizaje automático supervisado se clasifica en dos tipos descritos a continuación:

- **Regresión:** El aprendizaje automático supervisado por regresión tiene como resultado un número especifico. Si las etiquetas suelen ser un valor numérico, mediante las variables de las carácteristicas, se pueden obtener dígitos como dato resultante.
- **Clasificación:** El aprendizaje automático supervisado por clasificación el algoritmo encuentra diferentes patrones y tiene por objetivo clasificar los elementos en diferentes grupos.

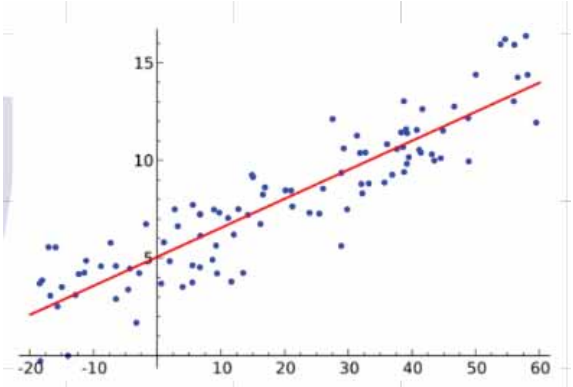


Figura 2.2: Gráfico de como funciona el aprendizaje supervisado de regresión

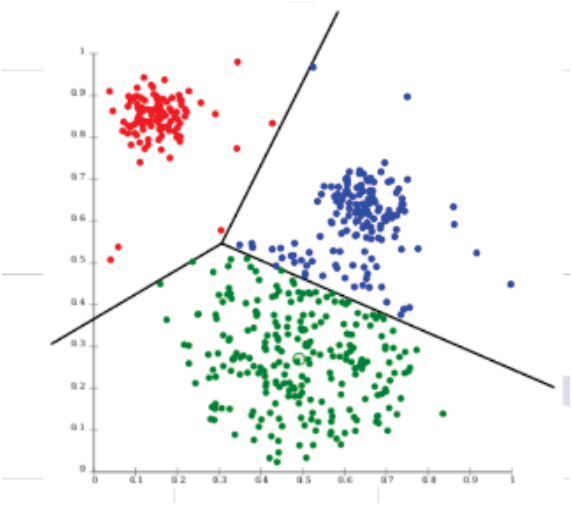


Figura 2.3: Gráfico de como funciona el aprendizaje supervisado de clasificación

2.1.3.2. Aprendizaje no supervisado (Unsupervised ML)

Los algoritmos de aprendizaje no supervisado trabajan de forma muy similar a los supervisados, con la diferencia de que éstos solo ajustan su modelo predictivo tomando en cuenta los datos de entrada, sin importar los de salida. Es decir, a diferencia del supervisado, los datos de entrada no están clasificados ni etiquetados, y no son necesarias estas características para entrenar el modelo. Dentro de este tipo de algoritmos, el agrupamiento o clustering en inglés, es el más utilizado, ya que particiona los datos en grupos que posean características similares entre sí. Una aplicación de estos métodos es la compresión de imágenes. Entre los principales algoritmos de tipo no supervisado destacan:

- K-medias (K-means).

- Mezcla de Gaussianas (Gaussian mixtures).
- Agrupamiento jerárquico (Hierarchical clustering).
- Mapas auto-organizados (Self-organizing maps).

2.1.3.3. Aprendizaje por refuerzo (Reinforcement learning)

Los algoritmos de aprendizaje por refuerzo definen modelos y funciones enfocadas en maximizar una medida de "recompensas", basados en "acciones" y al ambiente en el que el agente inteligente se desempeñará. Este algoritmo es el más apegado a la psicología conductista de los humanos, ya que es un modelo acción-recompensa, que busca que el algoritmo se ajuste a la mejor "recompensa" dada por el ambiente, y sus acciones por tomar están sujetas a estas recompensas. Este tipo de métodos pueden usarse para hacer que los robots aprendan a realizar diferentes tareas. Entre los algoritmos más utilizados podemos nombrar:

- Programación dinámica (Dynamic programming).
- Q-learning.
- State–action–reward–state–action (SARSA).

2.1.4. Procesamiento del Lenguaje Natural

Es un campo de las ciencias de la computación, inteligencia artificial y lingüística que estudia las interacciones entre las computadoras y el lenguaje humano. El NLP se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas y máquinas por medio del lenguaje natural, es decir, de las lenguas del mundo. El NLP no trata de la comunicación por medio de lenguas naturales de una forma abstracta, sino de diseñar mecanismos para comunicarse que sean eficaces computacionalmente que se puedan realizar por medio de programas que ejecuten o simulen la comunicación. Los modelos aplicados se enfocan no solo a la comprensión del lenguaje de por sí, sino a aspectos generales cognitivos humanos y a la organización de la memoria. El lenguaje natural sirve solo de medio para estudiar estos fenómenos. Hasta la década de 1980, la mayoría de los sistemas de NLP se basaban en un complejo conjunto de reglas diseñadas a mano. A partir de finales de 1980, sin embargo, hubo una revolución en NLP con la introducción de algoritmos de aprendizaje automático para el procesamiento del lenguaje.

2.1.4.1. Aplicaciones de NLP

Es fácil deducir que el NLP es una disciplina que cuenta con un alto potencial y múltiples posibilidades prácticas, tantas como los lenguajes naturales poseen. Su cometido es mejorar y hacer eficaz la comunicación entre personas y computadores. Por esta razón, cualquier área asociada al lenguaje y a las relaciones entre humanos y máquinas se puede ver afectada positivamente por el NLP. Aunque sus aplicaciones son innumerables y el único límite existente es la propia imaginación, algunas de las más populares e importantes son las siguientes :

- **Recuperación de la información:** El objetivo de este tipo de sistemas en la búsqueda y obtención de grupos de documentos electrónicos a partir de un conjunto de palabras clave proporcionadas por el usuario. Los documentos devueltos normalmente se ordenan en base a algún tipo de atributo que mide su relevancia dentro del resultado global. Estos sistemas son en los que basan su funcionamiento los buscadores de contenidos de Internet y representan la primera aplicación implantada masivamente dentro del mundo de las Tecnologías de la Información. Algunos ejemplos populares podrían ser el servicio Google Search o Microsoft Bing Sobrino Sande.

- **Traducción automática de textos:** Una de las aplicaciones paradigmáticas de los sistemas de NLP es la traducción automática entre múltiples lenguajes naturales. Esta labor es tácitamente imposible para un humano debido a la dificultad que existe a la hora de encontrar personas que conozcan decenas de lenguas distintas o la sola combinación de lenguas sobre las que queremos hacer la traducción (por ejemplo, griego y camboyano simultáneamente). Los sistemas actuales de traducción automática utilizan un enfoque basado en mediciones estadísticas y relaciones entre textos a partir de un entrenamiento previo con cientos o miles de textos. Aunque las traducciones no siempre son perfectas y no pueden sustituir a humanos en textos complejos en los que se requiera alta fiabilidad, sí son aceptables para tareas como traducir un mensaje de una red social, una página web o una reseña en una página de alquiler de coches. Uno de los ejemplos más significativos de este tipo de sistemas es el traductor Google Translate tal y como lo indica el autor Sobrino Sande.

- **Reconocimiento del habla:** Este tipo de sistemas permiten a las personas interactuar con los ordenadores u otros dispositivos electrónicos como teléfonos inteligentes o automóviles mediante el uso de un lenguaje natural y por medio de la voz. En los últimos tiempos se han hecho muy populares los llamados “asistentes virtuales” como Cortana,

Siri o Google Assistant. Éstos pueden realizar acciones como enviar un correo electrónico, gestionar el calendario de citas o incluso realizar una compra usando para ello solo comandos de voz. Otras aplicaciones que hacen uso de estos sistemas de NLP son los servicios telefónicos de atención al cliente. Estos servicios permiten realizar diferentes gestiones sin necesidad de un interlocutor humano, siendo muy habituales en actividades de banca y telecomunicaciones. Virtualmente todo dispositivo electrónico podría en el futuro poseer un sistema de reconocimiento del habla de manera que sus funciones pudiesen ser controladas mediante comandos de voz. Por tanto, el campo del reconocimiento de la voz humana se postula como uno de los más importantes debido a la popularización de este tipo de sistemas.

- **Extracción de la información:** Este tipo de tareas consiste en analizar textos o mensajes con el objetivo de capturar y extraer automáticamente aquella información considerada de interés. Una aplicación habitual es el escaneo de documentos escritos en algún lenguaje natural y para después volcar la información extraída a una base de datos de manera automática. Estos documentos pueden ser anuncios por palabras, artículos de prensa, informes de carácter científico, etc., y los datos a extraer, nombres de personas, organizaciones, teléfonos, fechas, valores monetarios u otros. El proceso de extracción de la información es básico para poder clasificar documentos, resumirlos o relacionarlos entre sí.

- **Análisis de sentimientos:** Como se verá en cada capitulo de este trabajo, el análisis de sentimientos ofrece la posibilidad de conocer automáticamente cuál es la opinión que una persona tiene sobre un determinado tema a partir de las ideas expresadas en un texto. Este sentimiento u opinión es una valoración cualitativa o cuantitativa acerca de un producto, servicio, persona o cualquier otro tipo de entidad. El poder extraer de manera automática esta información permite la creación de poderosas herramientas que facilitarán conocer cuál es el sentir de las personas en cada momento en relación a diferentes objetos de estudio(Sobrino Sande).

2.1.5. Análisis de sentimientos

En este apartado se expondrá el concepto de análisis de sentimientos mediante su definición y un breve repaso por su historia, nombrando también algunos de los autores y trabajos más relevantes. Como se verá a continuación, el análisis de sentimientos es un área de

investigación en pleno auge cuyo esplendor se debe a una serie de factores muy determinados. En este capítulo también se indicarán las aplicaciones más importantes que tiene este campo en la vida real, los diferentes niveles de análisis de sentimientos que se pueden ejecutar sobre textos escritos, las tareas necesarias para su realización y una definición formal sobre el concepto de opinión. Para finalizar esta sección se explicarán cuáles son las dificultades más comunes a las que se enfrenta el análisis de sentimientos.

2.1.5.1. Introducción

El Análisis de Sentimientos (AS o SA por sus siglas del inglés Sentiment Analysis) es un campo de investigación dentro del NLP que trata de extraer de manera automática y mediante técnicas computacionales información subjetiva expresada en el texto de un documento dado y acerca de un determinado tema. De esta forma, mediante el análisis de sentimientos podremos saber si un texto presenta connotaciones positivas o negativas. Una definición ampliamente extendida de este concepto es la ofrecida por los investigadores Pang y Lee en Pang et al. (2008) y que define el análisis de sentimientos como:

"Tratamiento computacional de opiniones, sentimientos y subjetividad en textos."

Esta definición es la más aceptada por la comunidad de investigadores, pero debido a su generalidad otros autores como Cambria y Hussain Cambria et al. (2012) han definido el análisis de sentimientos de la siguiente manera:

"Conjunto de técnicas computacionales para la extracción, clasificación, comprensión y evaluación de opiniones expresadas en fuentes publicadas en Internet, comentarios en portales web y en otros contenidos generados por usuarios."

Se puede observar que la segunda definición es mucho más concreta que la primera y sólo hace referencia a las opiniones, dejando fuera del alcance de estudio a los sentimientos y a la subjetividad. Es posible que dejar fuera a los sentimientos sea un error puesto que muchas veces las opiniones están fundamentadas y emanan de los sentimientos de quien las expresa, pero como indica E. Martínez en (Martínez Cámara, 2016), sí es un acierto no hacer referencia a la subjetividad ya que las opiniones se pueden encontrar en oraciones subjetivas y también objetivas. En cualquier caso, ambas definiciones son útiles y válidas para comprender en qué consiste el análisis de sentimientos.

Aunque la historia del análisis de sentimientos pertenece sin ninguna duda al siglo XXI, existen algunos trabajos desarrollados mucho antes considerados como precursores de este campo de investigación. Uno de ellos es (Carbonell, 1979) en donde se propone un modelo computacional que permite representar el pensamiento subjetivo de las personas, tratando de entender su ideología y su personalidad a través de la subjetividad que contienen sus textos escritos. Pocos años después, en (Wilks Bien, 1984), se presenta un estudio sobre las creencias que tiene un sujeto sobre otro en base al conocimiento que tienen de ambos por separado. Estos dos estudios, aunque relacionados, no avanzaron hacia lo que hoy se conoce como análisis de sentimientos, sino hacia otros campos de investigación como la interpretación de metáforas, los puntos de vista, el afecto y otras áreas relacionadas.

La verdadera explosión de trabajos de investigación del análisis de sentimientos se produce a partir de 2001 y su número ha ido incrementándose de manera exponencial con el paso de los años. En (Pang Lee, 2008) se atribuye este progresivo interés a tres factores:

- La popularización de los métodos de aprendizaje automático y su uso dentro de las diferentes áreas del NLP.

- La disponibilidad de datos con los que se puede entrenar a los sistemas de aprendizaje automático provenientes principalmente de internet y de su capacidad para generar ingentes cantidades de información, en especial a partir de la aparición de la llamada Web 2.0.

- El creciente interés por explotar esta información por parte de organizaciones y empresas debido a las posibilidades que ofrece el poder obtener automáticamente una valoración por parte de las personas acerca de productos, servicios o personas concretas.

2.1.6. Cross validation

La validación cruzada o cross validation es probablemente una de los mas importantes técnicas que un data scientist usa cuando siempre hay una necesidad para validar la estabilidad de un modelo de ML y medir cuan bien generaliza nueva data. Debe asegurarse de que el modelo tenga la mayoría de los patrones de los datos correctos, y que no detecte demasiado el ruido, o en otras palabras, su bajo sesgo y varianza. En este apartado describiremos y presentamos el concepto de validación cruzada.

2.1.6.1. Concepto

Es un modelo de técnicas de validación para evaluar cómo los resultados del análisis estadístico (modelo) se generalizarán a un conjunto de datos independiente. Se utiliza principalmente en entornos donde el objetivo es una predicción, y uno desea estimar con qué precisión un modelo predictivo se desempeñará en la práctica.

El objetivo de la validación cruzada es definir un conjunto de datos para probar el modelo en la fase de entrenamiento (es decir, el conjunto de datos de validación) para limitar problemas como el sobre ajuste, la falta de ajuste y obtener una idea de cómo el modelo se generalizará a un conjunto de datos independiente. Es importante que la validación y el conjunto de capacitación se extraigan de la misma distribución, de lo contrario, empeorarían las cosas. Acontinuación listamos las prioridades del desarrollo de este método.

- La validación nos ayuda a evaluar la calidad de nuestro modelo.
- La validación nos ayudará en seleccionar el mejor modelo para datos reales.
- La validación nos permite evitar el sobre entrenamiento(Overfitting) y el bajo entrenamiento(underfitting)

2.1.7. Pre Procesamiento de texto

2.1.7.1. Tokenización de texto

En el campo del procesamiento del lenguaje natural para realizar un análisis detallado de cada elemento en este caso "textos", debemos hacer un análisis detallado en el elemento principal que son las palabras, para este precedimiento necesitamos dividir una frase u oración en palabras y almacenarlas para su mejor ańalisis sin perder el orden o estructura dentro del contexto, y lo mas importante la relación y orden que lleva dentro del texto. Una definición mas detallada nos dice que un token es una instancia de un texto dado".

Ejemplo: Templada y riente (como lo son las del otoño en la muy graciosa ciudad de Buenos Aires) resplandecía la mañana de aquel veintiocho de abril.

Hay 25 palabras, pero en verdad si contamos a "Buenos-Aires" como una sola palabra (ya que es un nombre propio) hay: 24 => 24 tokens, Pero tipos hay: 21 ya que las palabras:

"de" y "la" se repiten. si pasáramos a sus lemas (palabras que referencian al mismo concepto común) las palabras de la oración entonces los tipos serían: 18 ya que "de" y "del" tienen el mismo lema y "la", "lo" y "las" también comparten un mismo lema.

Algoritmo de tokenización La forma más sencilla de Tokenización es identificar las palabras separando los caracteres alfabéticos de los demás, en particular del carácter espacio: " ". A continuación voy a mostrar una forma muy sencilla de Tokenizar un texto en Linux utilizando el comando: "tr", sin embargo esta forma solo vale en ingles ya que "tr" no soporta codificaciones de caracteres que usen más de 1 byte como por ejemplo Unicode.

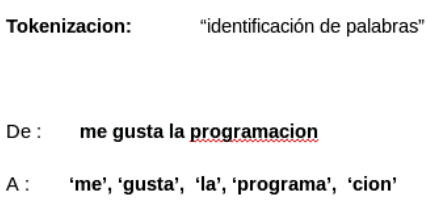


Figura 2.4: La figura muestra un ejemplo simple de tokenizacion, notese que aqui programa es una palabra raiz.

```
def tokenize(text):
    # remove non letters
    text = ''.join([c for c in text if c not in non_words])
    # tokenize
    tokens =  word_tokenize(text)

    # stem
    try:
        stems = stem_tokens(tokens, stemmer)
    except Exception as e:
        print(e)
        print(text)
        stems = ['']
    return stems
```

Figura 2.5: La figura muestra el codigo implementado que se utiliza para tokenización

2.1.7.2. Lematización y Radicación

Este procedimiento es sumamente importante en el procesamiento de lenguaje natural debido a que aporta una forma útil de resumir la cantidad de procesamiento y aporta un valor importante en la extracción de información a partir de un texto, este paso a la vez es muy útil en diversas tareas, sobre todo en las tareas de clasificación o recuperación de información a

partir de un corpus.

El proceso de lematización, radicación o steaming conocido en en el ingles, es un paso que podemos ver como la forma de reducir contenido que no aporta mucho a nuestro análisis, con lo cual estaríamos simplificando muchas operaciones de calculo computación y hacer el manejo del mismo mucho mas rápido, pero que conjuntamente con el paso de la tokenización aportan un valor extra en la significancia, debido selección de los tokens(generalmente palabras) mas importante y significativos, esto procedimiento se hace junto a la limpieza de palabras y representa un paso primordial en el análisis de texto proveniente de medios sociales, debido a que estos medios están llenos de ruido, la limpieza mas conocida es la de stopwords, eliminación de links, urls, hashtags. Para entender mejor este concepto vamos a definir en los siguientes párrafos.

Lematización: Es un proceso lingüístico que consiste en: Dada una forma flexionada (es decir, en plural, en femenino, conjugada, etc), hallar el lema correspondiente. El lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra. Es decir, el lema de una palabra es la palabra que nos encontraríamos como entrada en un diccionario tradicional: singular para sustantivos, masculino singular para adjetivos, infinitivo para verbos. Por ejemplo, decir es el lema de dije, pero también de diré o dijéramos; guapo es el lema de guapas; mesa es el lema de mesas.

La lematización puede realizarse automáticamente mediante programas de análisis morfológico. Hay diversos grados de lematización posible: Podemos hacer una lematización puramente morfológica, o bien hacer una lematización sintáctica que tenga en cuenta el contexto en el que aparece la palabra. Por ejemplo, en un análisis morfológico la palabra ama tendría dos lemas: el sustantivo ama y el verbo amar. Sin embargo, en un contexto sintáctico (es decir, en una oración), podemos desambiguarlo y optar por un único lema. Así, en El ama de llaves abrió la puerta, ama es sustantivo, mientras que en María ama a Pedro, ama es del verbo amar. Para poder hacer este tipo de lematización es necesario, por lo tanto, hacer un análisis sintáctico.

Steamming : Se llama steamming o radicación al procedimiento de convertir palabras en raíces, estas raíces son la parte invariable de las palabras relacionadas sobre todo por su forma. De cierta manera se parece a la lematización, pero los resultados (la raíces o steams) no son necesariamente palabras de un idioma, esto lo podemos definir por ejemplo al solo escoger palabras de 4 alfabetos, o también podemos conseguir esta raíz"de la palabra a través de la

eliminación cruda de los sufijos, afijos o elementos extremos de la palabra sin importar el orden y análisis morfológico de las palabras, normalmente para obtener una palabra base del mismo se eliminan los caracteres extremos.

El steamming es mucho más rápido desde el punto de vista de procesamiento que la lematización, debido a que no toma tiempo ni hace el análisis morfológico de la palabras. También tiene como ventaja que reconoce relaciones entre palabras de distinta clase, podría reconocer por ejemplo, que *picante* y *picar* tiene como raiz *pic*, esto mismo hace flexible y potente este método, y en otras palabras el steamming puede reducir el numero de elementos de que forman nuestro texto sin la necesidad de un análisis exhaustivo aportando reducción y flexibilidad a la hora de definir nuestra raíz de tokens y eso es en muchos casos lo que se busca. Por otro lado una desventaja que presenta este método del steamming es que sus algoritmos pueden recortar demasiado las palabras con la finalidad de encontrar la raíz y encontrar relaciones entre palabras que realmente no existen(*oversteamming*), también pueden suceder que deje raíces demasiado extensas o específicas y que tengamos mas bien un déficit de raíces (*understeamming*), en cuyo caso las palabras deberían convertirse en una misma raíz, no lo hacen, en cuyo caso no hay mucho por hacer para solucionar el mismo, pero que sin embargo el steamming sigue siendo un método fácil y rápido de implementar para llegar a ayudar en el pre procesamiento de textos que sigue siendo muy útil hoy en día.

2.1.7.3. Vectorización

Este proceso consiste en generar una matriz esparsa a partir de una lista de palabras o dicho de otro modo consiste en tranformar cada lista de palabras en un vector cuya representacion es el espacio euclideo, donde cada columna es una caracteristica(Una matriz cuya mayoría de elementos son ceros).

Las caracterısticas son principalmente palabras del vocabulario extraıdo de la tiranización. De esta forma, podemos considerar cada palabra del vocabulario como una columna, o podemos obtener representaciones mas detalladas si consideramos como posibles caracterısticas secuencias de palabras, llamadas n-gramas, que han ocurrido en el texto. Los n-gramas pueden ser secuencias de una palabra (unigramas), de dos palabras (bigramas), de tres palabras (trigramas), y ası sucesivamente .En el ejemplo de código siguiente podemos ver como nuestros tweets de ejemplo generan un vocabulario con unigramas y bigramas, con los cuales luego podremos representar nuestros datos.

[Colector tweets]

```
['#beatles', '#beatles mira', '#blackbird', '#blackbird
#beatles', '#davegrohl', '#davegrohl paso',
'#leonardodicaprio', '#leonardodicaprio #oscars',
'#oscars', ..., 'paso', 'paso #oscars', 'spotlight',
'spotlight oscar', 'tocando', 'tocando #blackbird']
```

Cada tweet se representa en una fila, y los valores que toma en cada columna están determinados por la ocurrencia de las palabras de cada columna en el tweet. De esa forma, la celda i,j de la matriz tendrá el valor de la ocurrencia del n-grama representado en la columna i en el tweet j.

La forma más simple de asignar valores a las celdas es con valores binarios: 1 si el n-grama representado por la columna i ocurre en el tweet j, y 0 si no ocurre. Teniendo un vocabulario grande, es de esperar que la mayor parte de n-gramas no ocurra en un tweet. Por lo tanto las matrices serán ralas, es decir, con gran cantidad de 0. A esta forma de representar los tweets la identificaremos como bin o binarización. La figura 4 nos muestra como podríamos representar los tweets de la figura 2 utilizando el vocabulario definido en el siguiente figura.

[Colector tweets]

```
[[0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0],
[1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1],
[0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0]]
```

donde el vocabulario asociado a las columnas es: [Colector tweets]

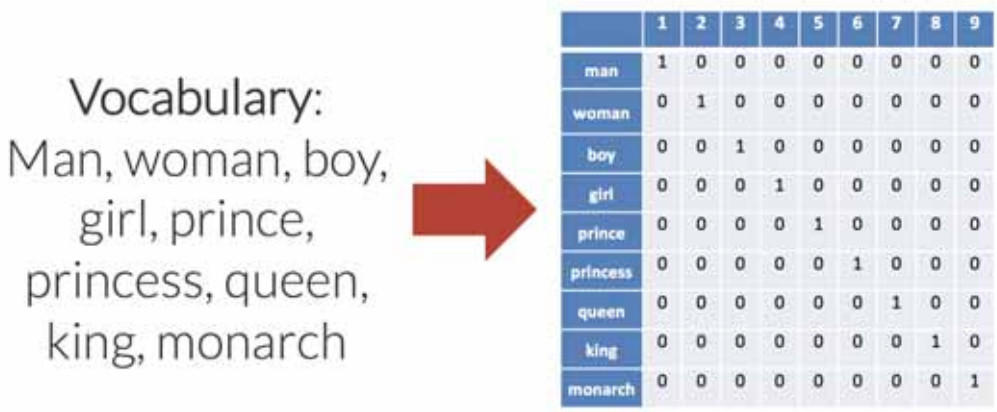
```
['#beatles', '#blackbird', '#davegrohl',
'#leonardodicaprio', '#oscars', 'actores', 'favoritos',
'guion', 'http://ow.ly/YUmHr', 'http://ow.ly/YUmRu',
'mejor', 'merecias', 'mira', 'mis', 'original', 'oscar',
'paso', 'spotlight', 'tocando']
```

Al momento de vectorizar los documentos, podemos elegir entre diferentes opciones que nos permiten reducir el volumen del corpus y obtener los resultados más relevantes. Una palabra

que aparece en más de la mitad de los tweets es redundante y no brinda mucha información para diferenciar un tweet de otro. Para esto es útil definir un umbral de frecuencia para determinar qué caracterıstica deben ser incluidas en la matriz, de tal forma que aquellas caracterıstica que son muy frecuentes sean ignoradas, ya que pueden ser consideradas stop words propias del corpus. Tampoco debemos incluir aquellas caracterısticas que son poco frecuentes ya que no proveen generalizaciones sobre tendencias en los textos. Identificaremos con min df y max df a las cotas inferior y superior, respectivamente, del umbral de frecuencia, describiendo ası las palabras que descartamos porque ocurren menos veces que min df o más veces que max df .

2.1.7.4. Word embedding

Es el nombre colectivo de un conjunto de técnicas de aprendizaje de características y modelos de lenguaje en el Procesamiento del Lenguaje Natural donde las palabras o frases del vocabulario se asignan a vectores de números reales. Conceptualmente, implica una incrustación matemática desde un espacio con una dimensión por palabra a un espacio vectorial continuo con una dimensión mucho más baja, en palabras mas sencillas es la conversión de texto a un modelo matemático en el cual podemos aplicar modelos matemáticos para el ajuste de curvas con aprendizaje automático.



	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Figura 2.6: Ejemplo de un word embedding de palabras a una matriz binaria: óptima forma de representación embedding Sayinath and Murphy (2013)

2.1.8. Social Media dataset

Hoy en dia usar medios sociales son una fuente al que recurrimos para poder expresar nuestra opinión, estos medios sociales se volvieron hoy en día una fuente de fácil uso y accesible para poder expresar nuestra opinión casi de manera anónima y con una vasta cantidad de información surgen una cantidad de data sin analizar con un rica fuente de información para ser explorada y para ser analizada por especialistas en el campo, es por eso que en este proyecto se aprovecha las redes sociales, especificamente twitter como medio publico de opinión, en el cual podemos encontrar una gran cantidad de párrafos pequeños (textos).

Figura 2.7: Ejemplo de publicaciones en twitter .

Una vez obtenida el data set para el proceso de entrenamiento tenemos una forma de etiquetado pues necesitamos que nuestro algoritmo de aprendizaje automático pueda aprender y ajustar sus parámetros para luego poder predecir automáticamente para ello necesitamos un dataset etiquedato como se muestra abajo.

content	polarity	polarity_bin
@marodriguezb Gracias MAR	P	1
Off pensando en el regalito Sinde, la que se va de la SGAE cuando se van sus corruptos. Intento no sacar conclusiones (lo intento)	N+	0
Conozco a alguien q es adicto al drama! Ja ja ja te suena d algo!	P+	1
Toca @crackoviadeTV3 . Grabación dl especial Navideño...Mari crismas!	P+	1
Buen día todos! Lo primero mandar un abrazo grande a Miguel y a su familia @libertadmontes Hoy podría ser un día para la grandeza humana.	P+	1

Figura 2.8: Ejemplo de dataset en twitter: Con su etiqueta de análisis de sentimiento

2.2. Redes Neuronales

Las redes neuronales artificiales (también conocidas como sistemas conexionistas) son un modelo computacional vagamente inspirado en el comportamiento observado en su homólogo biológico . Consiste en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitir señales. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida Matich (2001).

Cada neurona está conectada con otras a través de unos enlaces. En estos enlaces el valor de salida de la neurona anterior es multiplicado por un valor de peso. Estos pesos en los enlaces pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Del mismo modo, a la salida de la neurona, puede existir una función limitadora o umbral, que modifica el valor resultado o impone un límite que se debe sobrepasar antes de propagarse a otra neurona. Esta función se conoce como función de activación.

Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional. Para realizar este aprendizaje automático, normalmente, se intenta minimizar una función de pérdida que evalúa la red en su total. Los valores de los pesos de las neuronas se van actualizando, buscando reducir el valor de la función de pérdida. Este proceso se realiza mediante la propagación hacia atrás.

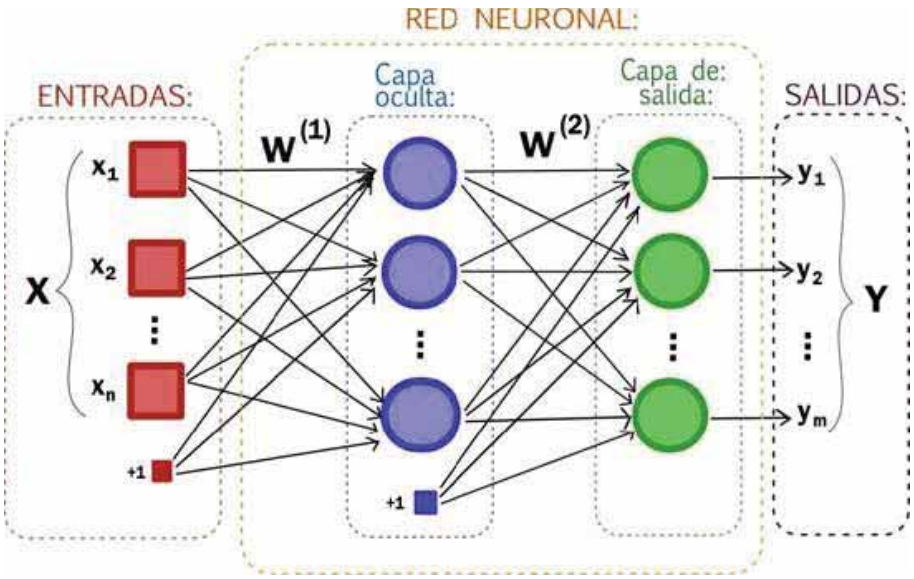


Figura 2.9: Diagrama general de redes neuronales: imagen extraida de : Burnett. (2019).

2.3. Maquinas de Soporte Vectorial (SVM)

Las Maquinas de sorporte vectorial(SVM) del ingles support vector machine fue introducido por Vapnik y sus colegas por los años 90 con la finalidad de proponer una solución al problemas de clasificacion de conjuntos binarios. El problema de la clasificación binaria fue luego ampliado y entonces las maquinas de soporte vectorial pasaron de clasificadores binarios a la regresion y actualmente se utilizan en múltiples tareas y problemas como son: regresión, agrupamiento y multiclasificación. Del mismo modo en el que evolucionò SVM también se introdujo en otras áreas debido a que presentaba similitudes en los datos y problemas así que en años posteriores las Maquinas de Soporte Vectorial son aplicadas en múltiples disciplinas de la inteligencia artificial, los ejemplos mas relevantes los encontramos en visión artificial, procesamiento de lenguage Natural, análisis de series temporales e incluso se uso exitosamente en la clasificacion de proteinas.

2.3.1. La tarea de SVMs

Dentro de las tareas de clasificación, las máquinas de soporte vectorial pertenecen a la catogoria de los clasificadores lineales, puesto que inducen separadores(rectas) lineales o hiperplanos, ya sea en el espacio original de los ejemplos de entrada, si son separables o casi separables(con ruido) o en un espacio transformado (espacio de características), si los ejemplos(datos) no son separables linealmente en el espacio original. Como se verá mas adelante, la busqueda de esta linea o hiperplano de separacion en estos espacios transformados, que normalmente son de muy alta dimension, se hará de forma implicita utilizando los *kernel*.

Mientras que la mayoría de métodos de aprendizaje se centran en minimizar los errores cometidos por el modelo generado a partir de los ejemplos de entrenamiento(error empírico), la inducción en los SVMs radica en la minimización del denominado *riegso estructural*. La idea es seleccionar un hiperplano de separación que equidista de los ejemplos más cercanos de cada clase para que de esta forma conseguir lo que se denomina, un *margen más* a cada lado del hiperplano. A la hora de definir el hiperplano, solo se consideran los ejemplos de entrenamiento de cada clase que caen justo en la frontera de dichos margenes. Estos ejemplos reciben el nombre de *vectores de soporte* . Desde un punto de vista práctico, el hiperplano de separación ha demostrado tener una buena capacidad de generalización, que evita el problema de sobre ajuste en el entrenamiento.

2.3.2. Tipos de SVM

Existen varios problemas que afrontan las maquinas de soporte vectorial dentro de las mas importantes esta la de clasificación con conjuntos perfectamente separables SVM hard margen, casi-separables(conjuntos separables con ruido) SVM soft margen, y los problemas de clasificación no separable linealmente que se conoce como SVM kernelizado.

2.3.3. SVM hard-margen

Este tipo de SVM son utilizados para problemas de clasificación binaria en los que inicialmente se sabe que son perfectamente clasificables o que podemos decir que existen dos grupos completamente separables, pero que son necesarios el uso de un método clasificador debido a la gran cantidad de datos y el SVM con un margen duro es aplicado. Para esto vamos a definir de la siguiente forma:

Dado un conjunto separable de ejemplos agrupados con $S = (x_1, y_1), ;(x_n, y_n)$ donde $x_i \in \mathbb{R}^d$ e $y_i \in +1, -1$, se puede definir un hiperplano de separación tal y como se aprecia en la figura 2.10 como una función lineal que es capas de separar sin error el conjunto, el cual esta definido como sigue.

$$D(x) = (w_1 x_1 + \dot{+} w_d x_d) + b =< w, x > +b \tag{2.1}$$

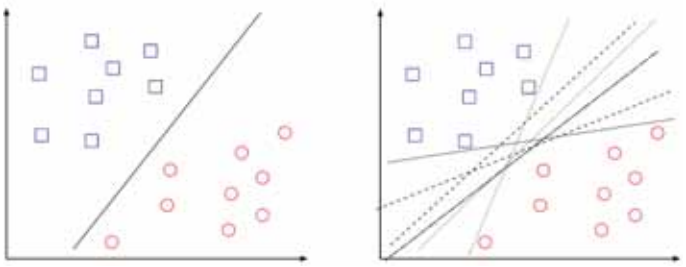


Figura 2.10: Figura con ejemplos de clasificación binaria separables, izquierdo: con un hiperplano(recta) de separación. derecha: multiples hiper planos de separación .

de la última ecuación podemos inferir que w y b son coeficientes reales, el hiperplano de separación cumplirá las siguientes restricciones para todo x_i del conjunto de ejemplos:

$$< w, x_i > +b \geq 0 \quad \text{si} \quad y_i = +1 \tag{2.2}$$

$$< w, x_i > +b \leq 0 \quad \text{si} \quad y_i = +1, i = 1, \ldots, n \tag{2.3}$$

o la forma $yD(x) \geq 0 \quad i = 1, \ldots, n$ Tal y como se puede deducir fácilmente de la figura anterior, el hiper planos que permite separar los ejemplos no es único, es decir existen infinitos hiper planos separables, representados por todos aquellos hiper planos que son capaces de cumplir las restricciones impuestas por cualquiera de las expresiones equivalentes(ecuaciones anteriores). Al tener muchos hiper planos posibles, para elegir uno óptimo. Para esto primero vamos a definir el concepto de *margen* de una hiper plano de separación, denotado por τ: como la mínima distancia entre dicho hiper plano y el ejemplo mas cercano de cualquiera de las dos clases como se aprecia en la figura 2.11. Sabiendo el margen, un hiper plano es denominado óptimo si *margen* es de tamaño máximo figura 2.11 b

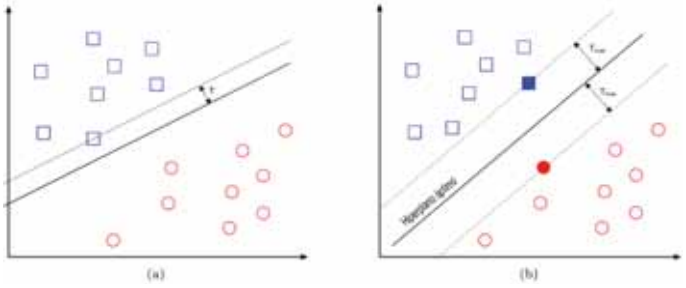


Figura 2.11: Margen de hiper plano de separación, a) hiperplano de separación no óptimo, b) hiperplano óptimo y margen máximo .

Se aplica un razonamiento similar en el caso de suponer que la distancia del hiper plano óptimo al ejemplo más cercano de la clase -1 fuese menor que la correspondiente al ejemplo más cercano de la clase $+1$. Por geometría, se sabe que la distancia entre un hiper plano de separación $D(x)$ y un ejemplo x' viene dado por

$$\frac{|D(x')|}{||w||} \tag{2.4}$$

Siendo $|.|$ el operador absoluto, $||.||$ el operador norma de un vector y w el vector que junto con el parámetro b, define el hiper plano D(x) y que ademas tiene la propiedad perpendicular al hiper plano considerado. Haciendo uso de las dos últimas expresiones anteriores

todos los ejemplos de entrenamiento cumplirán que:

$$\frac{|D(x')|}{||w||} \geq \gamma, \qquad i = 0, \ldots, n \tag{2.5}$$

la escala del producto γ y la normal de w se fija , de forma arbitraria a al unidad es decir.

$$\gamma||w|| = 1 \tag{2.6}$$

Llegando a la conclusión final de que aumentar el margen es equivalente a disminuir la norma de w , ya que la expresión anterior se puede expresar como

$$\gamma = \frac{1}{||w||} \tag{2.7}$$

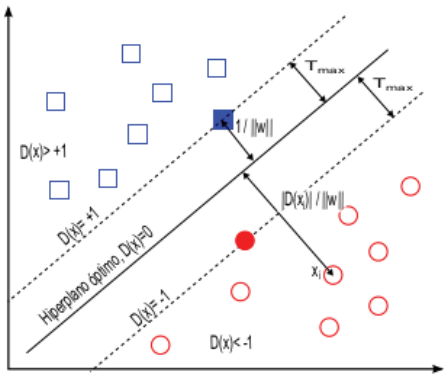


Figura 2.12: La distancia de cualquier ejemplo ,x , al hiper plano de separacion optimo viene dada por la expresión $\frac{|D(x')|}{||w||}$. En particular, si dicho ejemplo pertenece al conjunto de vectores soporte (identificados por siluetas sólidas), la distancia a dicho hiper plano será siempre $\frac{1}{||w||}$, Además, los vectores soporte aplicados a la función de decisión siempre cumplen que $|D(x)| = 1$.

Por lo tanto de acuerdo a la defición un hiperplano de separación óptimo es figura 2.12 es aquel que posee un margen máximo y por lo tanto un valor mínimo de $||w||$, y además, esta sujeto a la restricción dada por la ecuación 2.5 junto con el criterio expresado en 2.6, es decir:

$$y_i D(x) \geq 1, \qquad i = 1, \ldots, n \tag{2.8}$$

o lo que es lo mismo

$$y_i \left(< w, x_i > + b\right) \geq 1, \qquad i = 1, \ldots, n \tag{2.9}$$

El concepto de margen máximo está relacionado directamente con la capacidad de generalización del hiperplano de separación, de tal forma que, a mayor margen, mayor distancia de separación

existirá entre las dos clases. Los ejemplos que están situados a ambos lados del hiperplano óptimo y que definen el margen o, lo que es lo mismo, aquellos para los que la restricción 2.9 es una igualdad, reciben el nombre de *vectores soporte* ver la figura 2.12. Puesto que estos ejemplos son los más cercanos al hiperplano de separación, serán los más difíciles de clasificar y, por tanto, deberían ser los únicos ejemplos a considerar a la hora de construir dicho hiperplano. De hecho, se demostrará más adelante, en esta misma sección, que el hiperplano de separación óptimo se define sólo a partir de estos vectores. La búsqueda del hiperplano óptimo para el caso separable puede ser formalizado como el problema de encontrar el valor de w y b que minimiza el funcional $f(x) = ||x||$ sujeto a las restricciones del la ecuacion 2.5 o de forma equivalente(Observese que es equivalente minimizar $f(w) = ||w||$ o el funcional propuesto en 2.10 El proceso de minimización de este funcional equivalente, en lugar del original, permitirá simplificar la notación posterior,obteniendo expresiones más compactas).

$$\min \quad f(w) = \frac{1}{2}||w||^2 = \frac{1}{2} < w, w > \tag{2.10}$$

$$\text{s.a.} y_i\,(< w, x_i > +b) - 1 \geq 1, \qquad i = 1, \ldots, n \tag{2.11}$$

Este problema de optimización con restricciones corresponde a un problema de programación cuadrático y es abordable mediante la teoría de la optimización. Dicha teoría establece que un problema de optimización, denominado primal, tiene una forma dual si la función a optimizar y las restricciones son funciones estrictamente convexas. En estas circunstancias, resolver el problema dual permite obtener la solución del problema primal.

Así, puede demostrarse que el problema de optimización dado por la ecuación 2.10 satisface el criterio de convexidad, por tanto, tiene un dual. En estas condiciones, se pueden enumerar los siguientes pasos encaminados a resolver el problema primal:

En el primer paso , se construye un problema de optimizacion sin restricciones utilizando la funcion lagranjiana(el signo menos del sumando es por que las restricciones estan expresadas $g(x) \geq 0$ en vez de $g(x) \leq 0$ en 2.10).

$$L(w, b, \alpha) = \frac{1}{2} < w, w > - \sum_{i=1}^{n} \alpha_i [y_i(< w, x_i >) - 1] \tag{2.12}$$

donde los $\alpha_i \geq 0$ son los denominados multiplicadores de lagrange. El segundo paso

consiste en aplicar las condiciones de Karush-Kuhn-Tucker, tambien conocidad como condiciones KKT:

$$\frac{\partial L(w*, b*, \alpha)}{\partial w} = w* - \sum_{i=1}^{n} \alpha_i y_i x_i = 0, \quad i = 1, \ldots, n \tag{2.13}$$

$$\frac{\partial L(w*, b*, \alpha)}{\partial w} = \sum_{i=1}^{n} \alpha_i y_i x_i = 0, \quad i = 1, \ldots, n \tag{2.14}$$

$$\alpha_i[1 - y_i(< w*, x_i > + b*)] = 0, \quad i = 1, \ldots, n \tag{2.15}$$

las restricciones representadas por 2.13 corresponden al resultado de aplciar la primera condicion de KKT, y la expresadas en el final , al resultado de aplicar la denominada condición complementaria(segunda condición de KKT), se expresa los primeros parámetros de w y b en términos de α_i:

$$w* = \sum_{i=1}^{n} \alpha_i y_i x_i, \quad i = 1, \ldots, n \tag{2.16}$$

y a establecerán las restricciones adicionales para los coeficientes α_i:

$$\sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \ldots, n$$

con las nuevas relaciones obtenidas, se construirá el problema dual, así bastará usar la ecuación 2.16, para expresar la función lagrangiana solo en función de α_i, antes de ello, se puede reescribir la ecuación 2.12 como :

$$L(w, b, \alpha) = \frac{1}{2} < w, w > - \sum_{i=1}^{n} \alpha_i y_i < w, x_I > - b \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i \tag{2.17}$$

Teniendo encuenta que la segunda condicion anterior, el tercer término de la expresión seria nulo , y la sustitución de los coeficientes α_i en dicha expresión resulta ser:

$$L(\alpha) = \frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} \right) \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x_j} \right) - \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} \right) \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x_j} \right) + \sum_{i=1}^{n} \alpha_i \tag{2.18}$$

$$L(\alpha) = -\frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} \right) \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x_j} \right) + \sum_{i=1}^{n} \alpha_i \tag{2.19}$$

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_i y_i y_j < x_i, y_j > \tag{2.20}$$

Entonces hemos tranformado el problema de minimización primal ecuación 2.10, en el problema dual, consiste en maximizar la ecuacion anterior, sujeto a las restricciones de la ecuacion 2.17 junto a las asociadas originalmente a los multiplicadores de lagrange:

$$\text{máx} \quad L(\alpha) = \sum_{i=1}^{n} i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j < x_i, x_j > \tag{2.21}$$

$$\text{s.a.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \alpha_i \geq 0, \quad i = 1, \ldots, n \tag{2.22}$$

Al igual que el problema primal, este problema es abordable mediante técnicas estándar de programación cuadrática. Sin embargo, como se puede comprobar, el tamaño del problema de optimización dual escala con el número de muestras, lo hace con la dimensionalidad, d . n. Por tanto, aquí radica la ventaja del problema dual, es decir, el coste computacional asociado a su resolución es factible incluso para problemas con un número muy alto de dimensiones.

La solución del problema dual, $\alpha*$, nos permitirá obtener la solución del problema primal. Para ello, bastará sustituir dicha solución en la expresión 2.16 y, nornalmente, sustituir el resultado así obtenido en 2.1, es decir:

$$D(x) = \sum_{i=1}^{n} \alpha * y_i < x, x_i > + b* \tag{2.23}$$

Volviendo a las restricciones de KKT, al aplicar la restricciones de la segunda expresión KKT se puede afirmar que $\alpha_i \geq 0$ entonces el segundo factor de la parte izquierda de dicha expresión tendrá que ser cero y, por tanto

$$y_i(< w*, x_i > + b*) = 1 \tag{2.24}$$

Es decir, el correspondiente ejemplo,(x_i, y_i), en 2.24 satisface la correspondiente restricción del problema primal 2.10, pero considerando el caso "igual que". Por definición, los ejemplos que satisfacen las restricciones expresadas en (12), considerando el caso "igual que", son los vectores soporte y, por consiguiente, se puede afirmar que sólo los ejemplos que tengan asociado un $\alpha_i > 0$ serán vectores soporte. De este resultado, también puede afirmarse que el hiperplano de separación 2.23 se construirá como una combinación lineal de sólo los vectores soporte del conjunto de ejemplos, ya que el resto de ejemplos tendrán asociado un $\alpha_j = 0$. Para

que la definición del hiper-plano 2.23 sea completa, es preciso determinar el valor del parámetro $b*$. Su valor se calcula despejando $b*$ de 2.24:

$$b* = y_v s - < w*, x_v s > \tag{2.25}$$

donde $(x_v s, y_v s)$ representa la tupla de cualquier vector soporte, junto con su valor de clase, es α i distinto de cero. En la práctica, es más robusto obtener el valor de $b*$ promediando a partir de todos los vectores soporte, $N_v s$, es decir, la tupla de cualquier ejemplo que tenga asociado la expresión 2.25 se transforma en:

$$b* = \frac{1}{N_v s} \sum_{i=1}^{N_v s} (y_v s - < w*, x_v s >) \tag{2.26}$$

Finalmente haciendo uso de las restricciones adicionales de KKT, en 2.25, o en 2.26, permitira tambien calcular el valor de $b*$ en funcion de la solución del problema dual. Observese que tanto la optimización de la expresión 2.21 como la evaluación de 2.23, dependen del producto escalar de los vectores ejmplos. Esta propiedad se utilizará mas tarde para calcular hiper planos de separación óptimos en espacios transformados de alta dimensionalidad.

2.3.4. SVM para clasificación binaria de ejemplos no separables linealmente

En conceptos anteriores vimos una breve introducción a SVM y se ha mostrado que los hiper planos de separación son buenos clasificadores cuando los ejemplos son perfectamente separables o casi perfectamente separables. También se vio que el proceso de búsqueda de los parámetros que definen dichos hiperplanos se puede hacer independientemente de la dimensionalidad del problema a resolver. Así, si ésta es baja, basta con resolver directamente el problema de optimización primal asociado. En cambio, si la dimensionalidad es muy alta, basta con transformar el problema primal en su problema dual equivalente y resolver este último. Sin embargo, hasta ahora, se ha asumido la idea de que los ejemplos eran separables o casi separables y, por tanto, los hiper planos se definían como funciones lineales en el espacio x de los ejemplos. En esta sección se describirá cómo usar de forma eficiente conjuntos de funciones base, no lineales, para definir espacios transformados de alta dimensionalidad y cómo buscar hiper planos de separación óptimos en dichos espacios transformados. A cada uno de estos espacios se le denomina características , para diferenciarlo del espacio de ejemplos de entrada (espacio- x).

Sea $\Phi : \mathbb{X} \rightarrow \mathcal{F}$ la funcion de transformación que hace corresponder cada vector de entrada $\mathbb{X}$ con un punto en el espacio de características $\mathcal{F}$, donde $\Phi(x) = [\phi_1(x, \ldots, \phi_m(x))]$ y $\phi_i(x), \quad i = 1, \ldots, m$, tal que $\phi_i(x)$ es una función no lineal. La idea entonces es construir un hiper plano de separación lineal en este nuevo espacio. La frontera de decisión lineal obtenida en el espacio de características se transformará en una frontera de decisión no lineal en el espacio original de entradas, tal y como se puede apreciar en la figura 2.13.

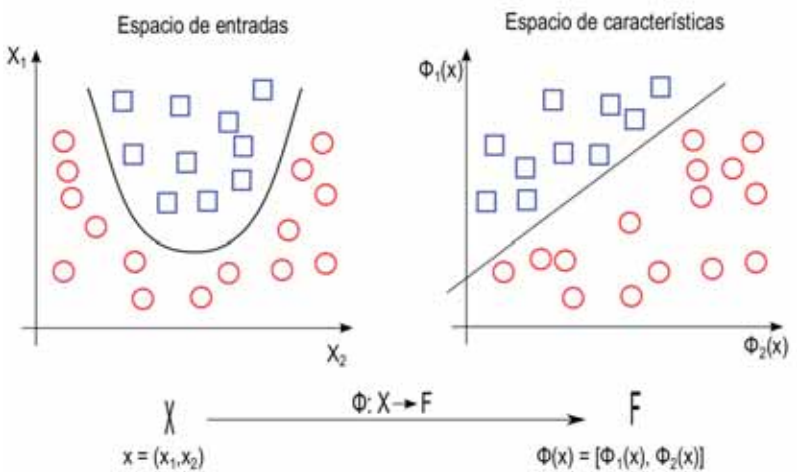


Figura 2.13: izquierdo: espacio de entrada (original) y a la derecha el espacio transformado(espacio de características).

En este contexto la función de decisión de la ecuación 2.1, en el espacio de características será data de la siguiente manera.

$$D(x) = (w_1\phi_1(x)+, \ldots, +w_m, \phi_m(x)) =< w, \phi(x) > \tag{2.27}$$

$$D(x) = \sum_{i=1}^{n} \alpha_i^* y_i < x, x_i > +b^* \tag{2.28}$$

Podemos notar que no consideramos b dentro del término, por que esto puede ser representado e incluye en la función base de funciones de transformación osea $\phi_1(x)$=1 por definición. Podemos también apreciar el mismo en su forma dual, la función de decisión de obtiene transformando convenientemente la expresión de la frontera de decisión en la expresión:

$$D(x) = \sum_{i=1}^{n} \alpha_i^* y_i K(x, x_i) \tag{2.29}$$

de donde la expresion $iK(x, x')$ se le denomina funcion **Kernel**. Por definición, una función kernel es una función $K : \mathbb{X}\mathbb{X} \rightarrow \mathbb{R}$, que asigna a cada par de elementos del espacio de entrada

, $\mathbb{X}$, un valor real correspondiente al producto escalar de las imágenes de dichos elementos en un espacio $\mathcal{F}$ (espacio de características), es decir :

$$K(x, x') =< \Phi(x), \Phi(x) >= (\phi_1(x)\phi_1(x')+, \ldots, +\phi_m(x)\phi_m(x')) \tag{2.30}$$

donde $\Phi : \mathbb{X} \to \mathcal{F}$. Por tanto, una función kernel puede sustituir convenientemente el producto escalar en 2.28, asi dado el conjunto de funcion base $\Phi = \{\phi_1(x), \ldots, \phi_m(x)\}$ el problema a resolver en la funcion dual sigue siendo el de encontrar el valor de los parametros $\alpha_i^*, i = 1, \ldots, n$ que optimiza el problema dual que representamos ahora como:

$$\text{máx} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) \tag{2.31}$$

$$\text{s.a.} \sum_{i=1}^{n} \alpha_i y_i = 0 \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \tag{2.32}$$

$$\tag{2.33}$$

Bueno ahora ya estamos casi en la disposicion descriptiva de metodología necesaria para resolver un problema de clasificación de ejemplos no separables linealmente. Concretamente, la función de decisión dada por la expresión 2.29, donde el valor de los parámetros $\alpha_i, i = 1, \ldots, n$ se obtendrán como solución al problema de optimización cuadrática dado por la última expresión. Conocidos el conjunto de ejemplos de entrenamiento $(x_i, y) = 1, \ldots, n$ el kernel K y el parámetro de regularización C. Actualmente, no existe una forma teórica de encontrar el valor de C . Sólo existe la heurística de usar un valor suficientemente grande (recuerde que $C =$ ínf para el caso linealmente separable).

Ejemplo práctico : A modo de ejemplo ahora supongamos el caso de entrada de dos vectores de dos dimensiones $X = (x_1, x_2)$ y el conjunto de funciones base formado por todos los polinomios de grado tres, es decir:

$$\phi_1(x_1, x_2) = 1 \quad \phi_2(x_1, x_2) = x_1 \quad \phi_3(x_1, x_2) = x_2 \tag{2.34}$$

$$\phi_4(x_1, x_2) = x_1 x_2 \quad \phi_5(x_1, x_2) = x_1^2 \quad \phi_6(x_1, x_2) = x_2^2 \tag{2.35}$$

$$\phi_7(x_1, x_2) = x_1^2 x_2 \quad \phi_5(x_1, x_2) = x_1 x_2^2 \quad \phi_9(x_1, x_2) = x_2^3 \tag{2.36}$$

$$\phi_1 0(x_1, x_2) = x_2^3 \tag{2.37}$$

En este caso, cada entrada de dos dimensiones es transformada en un espacio de características de diez dimensiones. La idea es entonces buscar un hiper plano en el espacio de características que sea capaz de separar los ejemplos. La frontera de decisión lineal asociada a dicho hiper plano se transformará en un límite de decisión polinomial de grado tres en el espacio de entradas. Obsérvese también que si, en este ejemplo, un problema de tan solo dos dimensiones se transforma en uno de diez dimensiones, un pequeño aumento en la dimensionalidad del espacio de entrada puede provocar un gran aumento en la dimensionalidad del espacio de características. En el caso límite, existen incluso espacios de características de dimensión infinita. Es por esta razón por la que, ahora, el problema de optimización se expresa sólo en su forma dual, ya que, como se ha visto en las dos secciones anteriores, la solución de este problema no depende de la dimensionalidad del espacio sino de la cardinalidad del conjunto de vectores soporte.

Si la transformación del espacio de entradas al espacio de características puede definirse a partir de un conjunto infinito de funciones base, surge la pregunta de cómo transformar los ejemplos de entrada, de dimension finita, en otro espacio de dimensión infinita. El siguiente teorema responde a esta pregunta

Teorema de Aronszajn :- Para cualquier funcion $K : \mathbb{X}x\mathbb{X} \rightarrow \mathbb{R}$ que sea simetrica(una funcion es simetrida si $K(x, x') = K(x', x), \forall x', x, \in \mathbb{X}$) y semi definida positiva, existe un espacio de Hilbert y una funcion $\phi : \mathbb{X} \rightarrow \mathcal{F}$ tal que

$$K(x, x') =< \Phi(x), \Phi(x') > \quad \forall x, x' \in \mathbb{X}$$

Una consecuencia importante de este teorema es que para construir una función kernel no es necesario hacerlo a partir de un conjunto de funciones base, $\Phi(x) = (\phi_1(x), \ldots, \phi_m(x))$, simplemente basta definir una función que cumpla las dos condiciones del teorema. Por tanto, para evaluar una función kernel no se necesitará conocer dicho conjunto de funciones base y, aún conocido éste, tampoco sería necesario realizar explícitamente el cálculo del producto escalar correspondiente, es decir, será suficiente con evaluar dicha función. En definitiva, para resolver el problema dual, no sólo no se necesita conocer el conjunto de funciones base de transformación, sino que tampoco es necesario conocer las coordenadas de los ejemplos transformados en el espacio de características. Sólo se necesitará conocer la forma funcional del kernel correspondiente,$K : \mathbb{X}x\mathbb{X} \rightarrow \mathbb{R}$,aún cuando este pudiera estar asociado a un conjunto infinito de funciones base.

Ejemplos de Funcion Kernel:

- Kernel Lineal

$$K(x, x') =< x, x' >$$

- Kernel Polinomico de grado p:

$$K_p(x, x') = [\gamma < x, x' > +\tau]^p \tag{2.38}$$

- Kernel Gaussiano:

$$K(x, x') = \exp(-\gamma||x - x'||^2), \gamma > 0 \tag{2.39}$$

- Kernel Sigmoidal :

$$K(x, x') = \tanh(\gamma < x, x' > +\tau)$$

2.3.5. Herramientas para el uso de machine learning

Cuando se quiere depurar correos, hacer reconocimiento facial, predicción en un gran conjunto de datos o reconocimiento de patrones, Machine Learniing es lo que se tiene que utilizar. La proliferación de software de código abierto hace que machine learning sea cada vez menos difícil de implementar tanto para una maquina como para una gran escala. Las siguientes herramientas de código abierto incluyen librerías para Python, R, C++, Java, Scala, Clojure, JavaScript y Go.

Python se ha convertido en el lenguaje de programación para matemáticas, ciencia y estadística ya sea por lo fácil que es adoptarlo y el número de librerías disponibles. Scikit-learn lidera al estar construido en las mejores librerías de Python (NumPy, SciPy y Matplotlib) para matemática y ciencias. La librería esta disponible bajo licencia BSD, por lo tanto es completamente abierto y reutilizable (según el blog de machine lerning referido en la cita 4.

2.4. Herramientas de programación a usar

Para el desarrollo de este proyecto vamos hacer uso de diferentes herramientas y por convención y la tarea de identificar opiniones vamos hacer uso de herramientas de programación como python(lenguaje de programación) con librerías de este mismo, como pandas, sklearn, NLTK, pytorch, tensorflow, para poder usar estos hacemos uso, de anaconda y para el dataset usamos el tipo de formato .csv muy ampliamente utilizado en procesamiento de lenguaje natural basado en texto.

Python Python es un interprete con orientación a objetos, es también un lenguaje de programación de alto nivel con uso de semántica dinámica. su alto nivel de estructura construida, combinado con el tipo dinámico que usa a si mismos el direccionamiento dinámico lo hacen muy atractivo para una aplicación de desarrollo rápido , también para el uso de `script` or a modo de *glue language* osea como lenguaje de programación para desarrollar, testear y unir proyectos de desarrollo de software y conectar componentes existentes. La simplicidad de python es que es fácil de aprender con respecto a la sintaxis, pues esta enfatiza la lectura y ademas reduce el costo de modularidad en un codigo reusable. El interprete de python y la extensa libreria estandar estan disponibles en codigo fuente y tambien posee un version binaria sin mucha carga para casi todas las plataformas y es libremente distribuida.

A menudo la comunidad de aficionados y desarroladores lo usan debido al incremento de productividad que este provee, ya que no posee un paso de compilación, el proceso de edicion y el test de pruebas es increiblemente rápido. Asi mismo encontrar errores en los programas realizados en Python es fácil, y el error de mensaje por el segmento en memoria, nunca causará una fallo, pues no existe un método de compilación. En vez de eso cuando el interprete descubre un error, este lanza un excepcion(tipo de alerta de conflicto). Cuando el programa no detecta la excepción, el intérprete imprime un seguimiento de la pila. Un depurador de nivel fuente permite la inspección de variables locales y globales, la evaluación de expresiones arbitrarias, el establecimiento de puntos de interrupción, el paso a través del código línea por línea, etc. El depurador está escrito en Python, testificando el poder introspectivo de Python. Por otro lado, a menudo la forma más rápida de depurar un programa es agregar algunas declaraciones de impresión a la fuente: el ciclo rápido de edición, prueba y depuración hace que este enfoque simple sea muy efectivo.

Scikit-learn Scikit-learn es una biblioteca en Python que proporciona muchos algoritmos de aprendizaje supervisados y no supervisado. Se basa en algunas de las tecnologías con las que quizás ya estés familiarizado, como NumPy, pandas y Matplotlib.

La funcionalidad que proporciona scikit-learn incluye:

- Regresión, incluida la regresión lineal y logística.
- Clasificación, incluidos los vecinos K más cercanos
- Agrupación, incluidos K-Means y K-Means ++

- Selección de modelo
- Pre procesamiento, incluida la normalización Min-Ma.

Formalmente hablando scikit-learn es una biblioteca de aprendizaje automático de software libre para el lenguaje de programación Python. Cuenta con varios algoritmos de clasificación, regresión y agrupación que incluyen Máquinas de Vectores de Soporte, bosques aleatorios, aumento de gradiente, k-means y DBSCAN, y está diseñado para interoperar con las bibliotecas numéricas y científicas de Python NumPy y SciPy(AdaptaciÓn conceptual el sitio web proveido por wikipedia.

NLTK El Kit de herramientas de lenguaje natural (NLTK) es una plataforma utilizada para crear programas de Python que funcionan con datos de lenguaje humano para aplicar en el procesamiento estadístico de lenguaje natural (NLP).Este también contiene bibliotecas de procesamiento de texto para tokenización, análisis, clasificación, derivación, etiquetado y razonamiento semántico. También incluye demostraciones gráficas y conjuntos de datos de muestra, además de un libro que explica los principios detrás de las tareas subyacentes de procesamiento del lenguaje que admite NLTK.

Otra definicion de Natural Language Toolkit es una biblioteca de código abierto para el lenguaje de programación Python originalmente escrito por Steven Bird, Edward Loper y Ewan Klein para su uso en desarrollo y educación. Viene con una guía práctica que presenta temas en lingüística computacional, así como los fundamentos de programación para Python, lo que lo hace adecuado para lingüistas que no tienen un conocimiento profundo en programación, ingenieros e investigadores que necesitan profundizar en lingüística computacional, estudiantes y educadores. NLTK incluye más de 50 corpus y fuentes léxicas como Penn Treebank Corpus, Open Multilingual Wordnet, Problem Report Corpus y Lin's Dependency Thesaurus.

Numpy y Pandas Numpy es una biblioteca de Python de código abierto que se utiliza para computacion científica y proporciona una serie de características que permiten que un programador de Python trabaje con matrices de alto rendimiento. Además, pandas es un paquete para la manipulación de datos que utiliza los objetos DataFrame de R (así como diferentes paquetes R) en un entorno Python. Tanto NumPy como pandas a menudo se usan juntos, ya que la biblioteca de pandas depende en gran medida de la matriz NumPy para la implementación de objetos de datos de pandas y comparte muchas de sus características. Además, pandas se

basa en la funcionalidad proporcionada por NumPy. Ambas bibliotecas pertenecen a lo que se conoce como la Pila SciPy, un conjunto de bibliotecas Python utilizadas para la computacion científica. La distribución Anaconda Scientific Python de Continuum Analytics instala pandas y NumPy como parte de la instalación predeterminada.

Colaboratory Colaboratory es un entorno libre de Jupyter, que no requiere una configuracion o instalación y este se ejecuta enteramente en la nube. El también llamado colab es ideal sobre todo si usted quiere mostrar o hacer un modo de presentación de su código en python con el cual se puede ejecutar y mostrar los resultados de cada paso de desarrollo, esto te ayuda a mejorar tu codigo en python y al tener la facilidad de incorporar todas las librerías para trabajar con Machine Learning y también todas las librerías disponibles para python desde las version 2.7 hacia arriba, esto hace posible poder hacer un entorno de presentación usando librerías y de Machine Learning como Pytorch, Keras, TensorFlow, openCV, etc. Tanto Colab como Jupyter se editan sobre el navegador haciendo esto muy accesible para cualquier usuario que quiera involucrarse con el aprendizaje de ML, así cualquier persona puede crear un archivo de Colab, actualizar, almacenarlo en su repositorio de google, compartirlo; además colab puede ser almacenado en github haciendo un entorno compatible y fácil de usar. Una de las grandes ventajas que este posee quizá son los recursos que te ofrece la plataforma de google cloud, pues tienes una gran capacidad para ejecutar proyectos grandes sobre estos y hacerlos de forma rápida. Por otro lado una desventaja apreciable es que al ser un producto de google tenemos que crearnos una cuenta de google para tener suficiente acceso y además al depender de un entorno en la nube esta sujeto a la velocidad de nuestra red y también a la sesión que se maneje pues esta plataforma intenta siempre reducir sus recursos a lo que estemos usando, administrando y muchas veces quitando recurso si no lo necesita o se tiene una baja señal de conexión a internet. Finalmente esta demas decir que tenemos que estar activos en la sesión para que esto funcione.

Anaconda Es una distribución gratuita y de código abierto de los lenguajes de programación Python y R para computación científica (ciencia de datos, aplicaciones de aprendizaje automático, procesamiento de datos a gran escala, análisis predictivo, etc.), cuyo objetivo es simplificar la administración de paquetes y despliegue. Las versiones del paquete son administradas por el sistema de gestión de paquetes Conda. La distribución de Anaconda es utilizada por más de 13 millones de usuarios e incluye más de 1400 paquetes populares de ciencia de datos adecuados para Windows, Linux y MacOS(de acuerdo a la pagina oficial de Anaconda). .

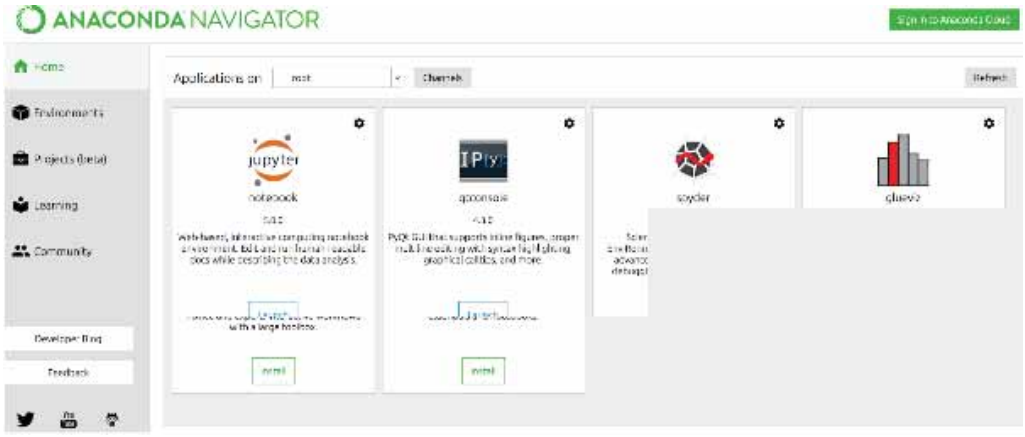


Figura 2.14: Imagen de interfaz de uso e instalación anaconda .

CSV file: Los archivos CSV (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal: Chile, Perú, Argentina, España, Brasil...) y las filas por saltos de línea.

El formato CSV es muy sencillo y no indica un juego de caracteres concreto, ni cómo van situados los bytes, ni el formato para el salto de línea. Estos puntos deben indicarse muchas veces al abrir el archivo, por ejemplo, con una hoja de cálculo. El formato CSV no está estandarizado. La idea básica de separar los campos con una coma es muy clara, pero se vuelve complicada cuando el valor del campo también contienen comillas dobles o saltos de línea. Las implementaciones de CSV pueden no manejar esos datos, o usar comillas de otra clase para envolver el campo. Pero esto no resuelve el problema: algunos campos también necesitan embeber estas comillas, así que las implementaciones de CSV pueden incluir carácteres o secuencias de escape Morales and Santos (2012).

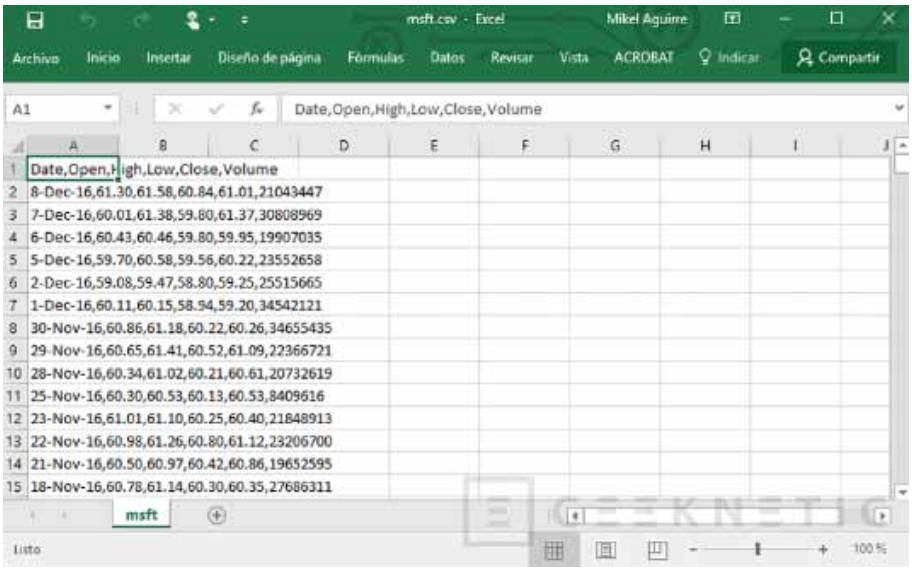


Figura 2.15: Archivo csv cargardo en excel .

Parte III

DESARROLLO DE LA ARQUITECTURA

Capítulo 3

Desarrollo de la arquitectura

En este capitulo será estructurado en tres partes para su mejor compresión acerca del desarrollo de un analizador automático masivo de comentarios (datos) en twitter para identificación de opinión. Así, en el capitulo 3.1 vamos a ver el pre-procesamiento de texto así como la extracción de características, luego en el capitulo 3.2 describimos los métodos necesarios para la clasificación, y finalmente en el capitulo 3.3 explicamos el post procesamiento y el almacenamiento de los datos.

Para tener un mejor preambulo vamos a mostrar un diagrama de pasos que vamos a seguir para el desarrollo de este proyecto teniendo como entrada un conjunto de datos como mostramos a continuación, el cual hace en síntesis un resumen de nuestro pipeline.

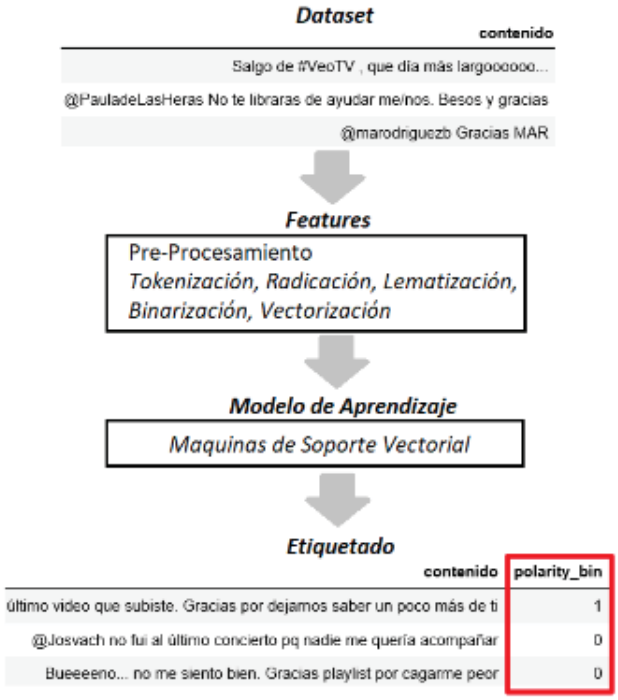


Figura 3.1: Diagrama general del desarrollo de la estructura, el cuadro rojo resalta el resultado de predicción.

3.1. Pre procesamiento y Extracción de Características

Este es el punto de partida no solo de un sistema de identificación de opiniones sino cualquier sistema de procesamiento de lenguaje natural, análisis de sentimientos, minería de opinión, entre otros. Aclarando que esta tarea pertenece a un algoritmo de aprendizaje automático, vamos a continuación detallar como se realiza la captura de información y el tratamiento de la data obtenida de twitter antes de utilizar un algoritmo de aprendizaje automático, así que primero tenemos que hacer un estudio y desarrollo de la información necesaria para nuestra tarea. Por otro lado muchos trabajos relacionados con esta tarea usan técnicas numerosas en este campo desde analizadores sintáctico, semánticos, diccionarios, entre otros métodos, pero según el estado de arte actual para el análisis de opiniones en el idioma inglés podemos concluir que los mas usados siguen un pipeline(pasos para realizar un modelo o desarrollo de una investigación) en el que se hace uso de eliminación de palabras con poca significancia *stop words* así como el uso de relación de palabras por su significado convirtiendo para ello cada palabra a su significado y relación con otros *steaming* y finalmente estas palabras son convertidas en un conjunto de vectores para ser almacenados y extraer las características como veremos en esta sección.

3.1.0.1. TASS DataSet

El conjunto de datos TASS es un corpus de textos (principalmente tweets) en español etiquetados para tareas relacionadas con el análisis de sentimientos. Se divide en varios sub conjuntos creados para las diversas tareas propuestas en las diferentes ediciones a lo largo de los años. Toda la información sobre estos conjuntos de datos se puede encontrar en el sitio web de TASS en ttp://www.sepln.org/workshops/tass

El conjunto de datos se divulga de forma voluntaria por las personas interesadas. El conjunto de datos contiene los datos reales, así como cualquier trabajo derivado, productos o servicios basados en la totalidad o parte de los datos. Todos los datos contenidos en el conjunto de datos se ha recopilado y procesado de acuerdo con las leyes vigentes en España.

La información del TASS es recopilada en un archivo de formato XML, ya que es una colección de tweets. Para el tratado de dicha información realizamos un proceso de conversión del archivo de formato XML a CSV(extensión utilizada por la libreria Pandas de Python) para su mejor manejo de la información como muestra la figura 3.2. Así también procedemos a filtrar los campos que necesitamos para la preparación del corpus inicial, en nuestro caso los campos: Content, Polarity y agreement.

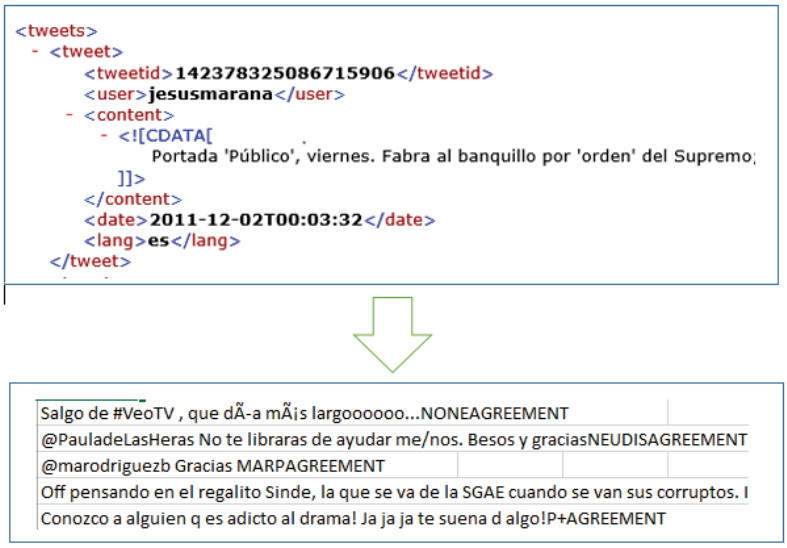


Figura 3.2: En la figura se la conversión de un archivo xml a csv.

3.1.0.2. Captura y Colección de tweets para el test

En esta parte vamos a ver como podemos recolectar y capturar nuestra propia información de texto (tweets) de la red social para poder aplicarlo mas adelante en nuestro análisis masivo de datos y así poder determinar si un conjunto de tweets catpurados tiene opinión positiva o negativa. Esto se realizó de manera masiva para lo cual se hizo uso de librerias de python que nos permiten acceder al data treaming de twitter.

Con la finalidad de que esta data (corpus) se de en un ámbito local referido este último a nuestro país, Perú, se hace uso del famoso sistema de coordenadas geográfico para localización de un punto geográfico (latitud, longitud). Para nuestra tarea especifica de delimitar una región alrededor de una zona, hacemos uso de dos puntos geográficos para poder formar un cuadrado y poder capturar la información de una zona seleccionada como se ve en la figura 3.3.

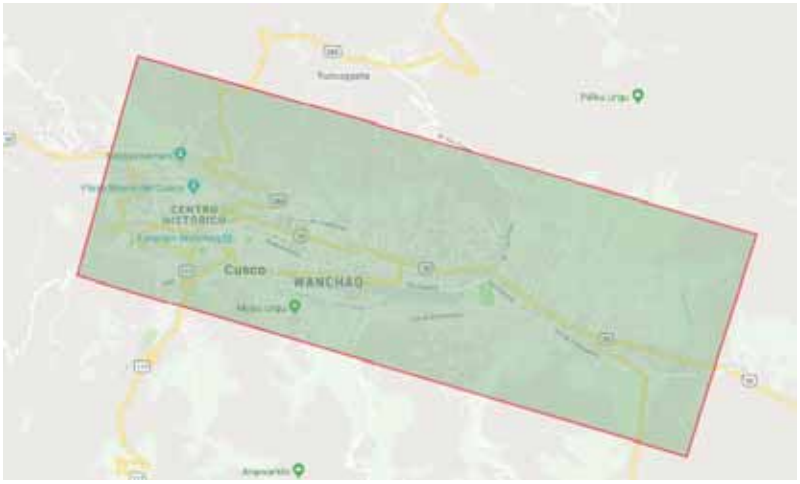


Figura 3.3: En la figura se puede ver la región cuadrada que engloba a la ciudad del Cusco.

Una vez establecida la región verde como se aprecia en la figura mostrada se deduce que es la zona para la recolección de nuestra data, teniendo cada esquina del rectángulo como un punto de coordenadas geográfico con elementos de latitud y longitud. Con cada esquina del cuadrilátero como punto de coordenada se puede recolectar haciendo uso de tweepy y el acceso a la data de la red social llegamos a la conclusión que solo necesitamos dos coordenadas para formar un cuadrilátero debido a que tenemos latitud y longitud para cada punto que si prolongamos tenemos dos coordenadas en el plano con dos vectores que unidos a otro punto se forma un cuadrilátero, de esta manera podemos obtener tweets de una región (cuadrilátero) establecida, para nuestro caso de estudio se seleccionó también la ciudad de Lima como se aprecia en la figura 3.4.

Figura 3.4: Figura con la región cuadrilátero rodeando la ciudad de Lima - Perú.

A continuación vamos a ver la implementación en Python de este procedimiento, para esto se utilizo tweetpy y la libreria JSON de python que decodifica el archivo json ofrecido por la red social.

[Colector tweets]

```
# al reves [lat, long,lat, long]
cusco = [ -72.013327, -13.552415, -71.870900, -13.523662]
lima = [-77.156443, -12.315642, -76.937431, -11.893478]

file =  open('tweets.txt', 'a')

class listener(StreamListener):

    def on_data(self, data):
        # Twitter returna json necesitamos decodificar esto
        try:
            decoded = json.loads(data)
        except Exception as e:
            print (e) # no queremos escuchar el excep
            return True
        #print(decoded)
```

```
17          if decoded.get('geo') is not None:
18              location = decoded.get('geo').get('coordinates')
19              print("here")
20          else:
21              #location = '[,]'
22              location = decoded.get('user').get('location')
23          text = decoded['text'].replace('\n',' ')
24          user = '@' + decoded.get('user').get('screen_name')
25          created = decoded.get('created_at')
26          tweet = '%s|%s|%s|%s\n' % (user,location,created,text)
27
28         # guardamos en archivo tweet
29          file.write(tweet)
30          return True
```

3.1.1. Pre-Procesamiento

Este es el punto de partida para el procesamiento y tratado de la información antes de poder entrenar nuestro modelo o usar un clasificador específico para la tarea del modelamiento. Así, en este punto vamos a detallar todos los pasos necesarios para llevar acabo esta tarea, el objetivo de esta fase es limpiar y normalizar, organizar y ordenar nuestros datos para evitar que algunos de estos datos puedan influir de manera negativa en el resultado final de nuestra tarea de identificación de opiniones. Este paso es importante cuando se habla de procesamiento de información en redes sociales por que es habitual y muy frecuente encontrar mensaje con contenido que tienen faltas ortográficas, caracteres repetidos, una mezcla de mayúsculas y minúsculas y en este caso concreto de twitter uso mayoritario de abreviaturas, los famosos hash tags y jergas que son usados como una forma de escribir mayor contenido en espacios reducidos.

A continuación para nuestra tarea en particular vamos a realizar estos pasos para mejorar y limpiar nuestra información al momento del test, debido a que deseamos aplicar nuestro algoritmo en un reconocimiento de opinión para tweets en Perú. Una vez ya guardada la información en un formato de archivo pasamos a usar librerías de python para primero cargar la información que en los siguientes puntos vamos a detallar.

3.1.1.1. Supresión de palabras *stopwords*

Una vez seleccionado y con nuestra data disponible en un formato csv a continuación procedemos a revisar un poco el contenido de nuestra data y pudimos apreciar que nuestra información de contenido de texto presentó mucho ruido, esto debido a que los usuarios no tienen mucho cuidado al momento de escribir en un red social y estos microblogs no controlan el uso del lenguaje escrito en una forma estándar, así que tenemos que eliminar, reemplazar y hacer una supresión de palabras.

Para llevar a cabo esta tarea primero necesitamos un listado de palabras conocidas como *stop words*, estas son un listado de palabras que no son necesarias para el análisis de la información, debido a que muchos de ellos no tienen significancia por si mismos tal es el caso con los muy conocidos artículos, conectores, preposiciones. Además en este procedimiento es generalmente usado en el pre-procemiento y se entiende como el procedimiento de convertir la data en un contenido que la computadora pueda interpretar o entender y una de las mayores formas para conseguir esto es filtrar las palabras menos útiles.

Para la implementación de este procedimiento en python se utilizo la librería *NLTK*(Natural Lenguaje ToolKit), que es un paquete de python para el uso de herramientas para el procesamiento del lenguaje natural que nos facilito el uso y la definición propia de los términos; el paquete contiene una lista de stop words conocida como diccionario con el cual podemos filtrar nuestro data(tweet), a continuación mostramos el procedimiento para extraer esto con python.

[Stop Words]

```
1 import nltk
2 from nltk.corpus import stopwords
3  spanish_stopwords = stopwords.words('spanish')
4 # print spanish stopwords
5 print(spanish_stopwords)
6
7  # OUTPUT
8  '''
```

```
'de', 'la', 'que', 'el', 'en', 'y', 'a', 'los', 'del', 'se', 'las',
    'por', 'un', 'para', 'con', 'no', 'una', 'su', 'al', 'lo', 'como',
    'más', 'pero', 'sus', 'le', 'ya', 'o', 'este', 'sí', 'porque',
    'esta', 'entre', 'cuando', 'muy', 'sin', 'sobre', 'también', 'me',
    'hasta', 'hay', 'donde', 'quien', 'desde', 'todo', 'nos',
    'durante', 'todos', 'uno', 'les', 'ni', 'contra', 'otros', 'ese',
    'eso', 'ante', 'ellos', 'e', 'esto', 'mí', 'antes', 'algunos',
    'qué', 'unos', 'yo', 'otro', 'otras', 'otra', 'él', 'tanto', 'esa',
    'estos', 'mucho', 'quienes', 'nada', 'muchos', 'cual', 'poco',
    'ella', 'estar', 'estas', 'algunas', 'algo', 'nosotros', 'mi',
    'mis', 'tú', 'te', 'ti', 'tu', 'tus', 'ellas', 'nosotras',
    'vosotros', 'vosotras'...'
'''
```

Despues de cargar la lista de palabras procedemos a filtrar los stopwords, pero antes de realizar el proceso de filtrado realizamos la tokenización de los comentarios de los tweets, así tambien hacer otros pasos como podemos apreciar aqui en el algoritmo 3.1.1.5 la lista mostrada como ejemplo del output(salida) son todas minúsculas, así que al momento de filtrar las palabras también se tiene que hacer la conversión a minúsculas. Así como la conversión en minúsculas existen también otros pasos adicionales de limpieza que casi no son persibidos pero tiene una efecto al momento del pre-procesamiento aunque sean cortos, acontinuación mencionamos algunos:

Normalización de Mayúscula y Minúscula.- Aunque el termino *normalización* es referido al proceso de conversión de texto difícil de interpretar mas comúnmente relacionado con jergas y abreviaturas a un texto con correcta gramática y de mejor compresión, por motivos de similitud de términos decidimos llamarlo normalización debido a que haremos también una conversión de palabras mayúsculas a minúsculas, aunque esta tarea no parece necesaria en aprendizaje automático se sugiere trabajar en minúsculas, por ello existen librerias como nltk que provee listas de palabras en minúsculas haciendo diferencia entre mayúsculas y minúsculas. Entonces en este paso convertimos palabras como “PROGRAMA” en “programa” independientemente del carácter.

Tratamiento de duplicidad de Caracteres .- En las redes sociales es habitual escribir texto con contenido repetido tales como : “que rico” y “queee ricooooo”, estos ejemplos quizá tengan el mismo significado pero están escritas de manera distinta y expresan un grado de énfasis mayor en el segundo caso, pero para nuestra tarea no consideramos el nivel de énfasis, así que todas las palabras repetidas se remplazaron con su equivalente estándar.

Números, enlaces y hasgtags .- Este último también es necesario, en este paso debido a que estamos tomando contenido de texto escrito por usuarios de un medio real y dentro de estas redes sociales tenemos contenido numérico los que tenemos que eliminar debido a que no presenta mucha importancia con respecto al análisis de sentimiento que esta mas relacionado a las palabras. Así como se eliminó los números, de igual manera desestimaremos a los links(enlaces, urls) por que no presenta mucho contenido de palabras con respecto a la opinión, quizás sea una fuente externa de respaldo o alguna otra información adicional con respecto a la opinión del usuario. Con respecto a los hastag, se tomo en consideración que estos solo son una abreviación que por lo general representan algún tema de interes en la cual se va a comentar mas no una forma simplificada de referirse a un concepto o bien fueron usados para asignar o enlazar un contenido u opinión de tweeter con otro comentado en la red social (a menudo usado en tweeter **hastags**).

3.1.1.2. *Tokenización*

La tokenización en NLP es normalmente el proceso de separar la cadena de entrada, frases en lenguaje natural, en las distintas palabras que componen la frase. Puede ser simplemente un separador que trocee la frase cada vez que se encuentre un espacio.

```
>>> 'of escapades demonstrating the adage that what is good for the goose'.split()
['of', 'escapades', 'demonstrating', 'the', 'adage', 'that', 'what', 'is', 'good',
 ↪  'for', 'the', 'goose']
>>>
```

Figura 3.5: Tokenización sencilla mediante la separación de espacios.

La tokenización en su sentido más amplio y general es el procedimiento del análisis léxico para reconocer lexemas del flujo de entrada con la que se obtiene un flujo de salida compuesto por tokens.

Token .- Denominamos token al conjunto de cadenas de caracteres con significado mínimo.

Patrón - El patrón es una regla que describe el conjunto de caracteres que cuadran un token. El concepto de "cuadrar" se denomina en inglés como match, y también se puede traducir como concordancia o correspondencia.

Lexema .-Un lexema es una secuencia concreta de caracteres que se corresponde con el patrón de un token determinado

3.1.1.3. Radicación *steamming*

Otro de los pasos para seguir en el pre-procesamiento es aquella por la cual debemos reducir las palabras a su forma mas básica pero sin tomar en cuenta la forma gramatical y de una forma mas directa, debido a un orden gramatical en los textos se hace uso de estas variaciones de los significados de las palabras. Un ejemplo de este problema es generalmente el de la variación de la forma como: organizar, a otras formas como organización, *organismos*, entre otros, estos además presentan un significado parecido por lo que es necesario trabajar con la forma básica en ves de las variaciones de esta. Esta forma de inferencia es muy útil para reducir el tamaño de texto y también extraemos la información esencial del contenido y su significancia. así por ejemplo:

Los carros de los niños son de diferentes colores. La oracion anterior será transformada a una forma como: *carro niños diferente color*. La manera mas común de desarrollar este método en Procesamiento de Lenguaje Natural es eliminar los sufijos para quedarse con la raíz de la palabra y su significado más primitivo.

3.1.1.4. Lematización

De la misma forma como vimos en el paso anterior (steamming), en este paso nos dedicamos a la simplificación de palabras derivadas pero tomando en cuenta la forma gramatical, para ello nos fijamos en la forma de las palabras, dicho de otro modo este procedimiento es la reducción de la palabra en su forma básica gramatical al cual se le llama "lema", el lema o raíz de una palabra es la forma base de una palabra así que si tomamos la palabras escribiendo la forma base es *escribir*. Otro aspecto a considerar y por el cual este paso es diferente al anterior,

es que a diferencia del steaming que reduce las palabras de manera directa sin tomar en cuenta la raíz o lema de la palabra, aplicando para esta reducción por sufijos en infijos, por los cuales se llega a eliminar y reducir la palabra. Sin embargo en lematización se fija en la forma de la raíz y una reducción mas fijada en la palabra principal.

Para nuestro proyecto se considero oportuno trabajar con las dos operaciones de reducción de palabras debido a la gran importancia de fijarse en el modelo raíz y la forma mas rápida y complementaria para llegar a obtener las reducción usando steaming. En síntesis la lematización es la conversión de la palabra en su forma simple o su **lema**, el lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra.

En el siguiente código se muestra el uso de lematización y radicación utilizando la libreria NLTK. [Lematización y Radicación]

```
1 from nltk.stem import SnowballStemmer
2 from nltk.tokenize import word_tokenize
3
4 stemmer = SnowballStemmer('spanish')
5
6 def stem_tokens(tokens, stemmer):
7     stemmed = []
8     for item in tokens:
9         stemmed.append(stemmer.stem(item))
10    return stemmed
```

Figura 3.6: Ejemplo de conversión a través de la radicación y lematización

3.1.1.5. Binarización de Etiquetas

Uno de los procedimientos adicionales, pero no por eso menos importantes es el paso de binarización de los labels(etiquetas). Este procedimiento es necesario cuando se tiene labels

que no son tan manejables como los serian el formato binario 1 y 0, el cual reduce y hace mas simple el manejo y la comparación de variables.

Para nuestra proyecto de tesis, tomamos en cuenta esta parte debido a los datos de etiqueta proveídos por el data set (TASS), el cual presenta un NEU, POS, NEG en su etiquetado asi que tenemos que manejar strings para su manipulación y debido a que estos presenta un carga adicional en la comparación y este proceso se realizará de manera repetitiva durante el entrenamiento y la evaluación de nuestros algoritmos decidimos convertirlo a una forma binaria tomando en consideración la parte NEU(neutra) también como datos válidos, para lo cual eliminamos los neutros y consideramos los positivos como 1 y los negativos como 0 así que de aquí en adelante se trabaja con unos y ceros como en la forma binaria.

La implementación de este procedimiento es intuitivo para ello nosotros consideramos la columna con los labels (N,N-, P+, P, NEU) los cuales vamos a convertirlos en “uno” y “cero” como se puede ver en el ejemplo.

[Binarizacion]

```
# vamos unicamente tomar los tweets  que son diferentes NEU
tweets_corpus = tweets_corpus[tweets_corpus.polarity != 'NEU']

# dandole el nuevo nombre a la fila
tweets_corpus['polarity_bin'] = 0
# todos los P, y los  P+ son = 1
tweets_corpus.polarity_bin[tweets_corpus.polarity.isin(['P', 'P+'])] =
  ↪  1
tweets_corpus.polarity_bin.value_counts(normalize=True)
```

3.1.2. Extracción de Características

La extracción de características es uno de los procedimientos mas importantes al momento de la identificación de opiniones y de texto en general, este procedimiento corresponde la conversión de texto en números normalmente alcanzado por algoritmos y procedimientos llamados como embeddings,

3.1.2.1. Vectorización de Palabras

Este paso es uno de los pasos fundamentales en el procesamiento de lenguaje natural debido a que involucra la conversión de nuestro input(texto) en una representación mucho mas accesible para los procesadores, en otras palabras esta es como una forma de transformación a un lenguaje en el que podemos manipular y hacer un procesamiento, pues así analizaremos solo las palabras y buscaríamos una representación para estos sería una tarea enorme, así que convertir nuestro texto en un representación matemática es una buena oportunidad de poder reducir no solo el procesamiento sino también nos facilita la manipulación y representación de los mismos. Existen muchos de estos métodos actualmente desde los que transforman un párrafo completo hasta los que solo lo hacen a través de un carácter, mediante representación contextual y así podemos enumerar la gran variedad de estos. Para nuestro trabajo de tesis, tomamos en consideración uno de los mas conocidos y el más fácil de implementar y conceptualmente hablando no involucra mucha teoría abstracta desde el punto de visto general por supuesto, este método llamado **One hot encoding** que consiste en generar una matriz sparsa(muchos ceros) transformando cada lista de palabras en un vector, donde la matriz tiene dimensiones tan grandes como los tiene el vocabulario de palabras que tiene todo el dataset, estas palabras son únicas. Cada palabra considerada como única tiene una dimensión y va ser representada por 1 donde esta aparece y con '0' donde no aparece. Finalmente obtendremos un resultado con un lista muy grande(tipo matriz) matriz esparsa, pero que no captura la relación de información entre las palabras como se muestra en la figura siguiente 3.7 con la representación de un ejemplo de *One hot encoding* para el conjunto de ciudades y paises.

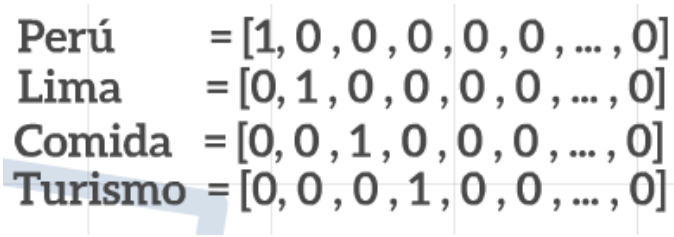


Figura 3.7: Ejemplo de la representación de una matriz esparsa.

En este último ejemplo podemos ver como estamos asociando un representación de diccionario para el conjunto de palabras, para luego crear una matriz tan grande como palabras tenga nuestro diccionario, así, el tamaño de la lista de listas (en otras palabras la matriz) va ir incrementando con mas cantidad de datos. En este ejemplo podemos apreciar que los uno '1' no se repiten en la misma columna lo que indica que estos son diferentes de acuerdo a la

representación `Count Vectorizer` siempre presentará una diferencia entre estas palabras y una asociacion.

Para la implementación de este método, en nuestro trabajo de tesis descidimos otra vez usar la herramienta de python que nos permita hacer una implementación muy sencilla y eficiente; afortunadamente existe una clase de nuestra librería ya conocida hasta este punto sklearn. Para ello primero haremos este procedimiento haciendo un llamado de la clase CountVectorizer desde la librería de sklearn y pues como esto pertenece a *features Extraction* (extracción de características) pues esta misma librerías posee una enorme cantidad de features para la extracción esto depende de la tarea a realizar y al algoritmo de clasificación, así que como estamos en una tarea de NLP, se extrae de la extensión text como se puede ver en el código 3.1.2.1, a continuación creamos la clase instanciando a esta *CountVectorizer*, que por motivos de descripción mostramos un ejemplo pequeño de vectorizacion de nuestro algoritmo que luego se usara masivamente para todos nuestro input (corpus de entrada tweeter), después de haber creado la clase lo que se procede a hacer es la creacion de nuestro vocabulario respectivo con el cual vamos a crear una matriz con las características vectorizadas, así que para este ejemplo en particular se crea esto mismo con el método fit, y como se puede apreciar se tiene una salida con los parámetros establecidos por defecto y una configuración por defecto como el tokenizer=None que para nuestra tarea se realizo para tokens de palabras así para este trabajo de tesis se considero como tokenizer=tokenize, otro parámetro a considerar ya que estamos en una tarea que se realizará para el idioma español, entonces hemos considerado también los stop words ya mencionados anteriormente en el español así que configuramos este parámetro también para la vectorizacion. Finalmente este paso de la vectorización se consigue con la transformacion de nuestro texto en un conjunto de número los cuales estan representados en una matriz, en este ejemplo que mostramos podemos notar que se usa vectorizador.transform(texto), esto mismo hace evidencia de la transformación y como se puede apreciar en la parte final del código se muestra una salida de lista conteniendo números unos, los cuales son una representación de nuestro ejemplo.

[Count Vectorizer]

```
# importamos primero la libreria de encodificacion One hot
from sklearn.feature_extraction.text import CountVectorizer

# creamos un corpus de ejemplo
corpus = ['Esto es una prueba de evaluacion de un texto en python']
```

```
6
7  # creamos la clase one hot
8  vectorizer = CountVectorizer()
9
10 # // luego hacemos la transformacion con el metodo
11 # // one hote encodding
12
13 # aqui primero creamos el vocabulario
14 X = vectorizer.fit(corpus)
15
16 # salida del codigo anterior con sus setting
17 '''
18 CountVectorizer(analyzer='word', binary=False, decode_error='strict',
19                 dtype=<class 'numpy.int64'>, encoding='utf-8',
   ↪  input='content',
20                 lowercase=True, max_df=1.0, max_features=None,
   ↪  min_df=1,
21                 ngram_range=(1, 1), preprocessor=None,
   ↪  stop_words=None,
22                 strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
23                 tokenizer=None, vocabulary=None)
24 '''
25 # mostramos el vocabulario (comprimido)
26  print(vectorizador.vocabulary_)
27
28 # salida de vocabulario para el ejemplo
29  '''
30 {'esto': 3, 'es': 2, 'una': 8, 'prueba': 5, 'de': 0, 'evaluacion': 4,
   ↪  'un': 9, 'texto': 7, 'en': 1, 'python': 6}
31 '''
32 # encodificar el documento
33 vector = vectorizador.transform(texto)
34 # mostramos el tamaño de nuestro vector
35 print(vector.shape)
```

```
'''
(1,10)
'''
print(type(vector))
'''
<class 'scipy.sparse.csr.csr_matrix'>
'''
print(vector.toarray())
'''
[[2 1 1 1 1 1 1 1 1 1]]
'''
```

Adicionalmente como resumen mostramos que este paso muy importante para el procesamiento de texto se puede resumir en tres paso fundamentales :

- Crear la instancia de la clase countVectorizer.
- Llamado de la función fit, con la finalidad de aprender el vocabulario necesario de uno o mas documentos(inputs texto).
- Llamado de la función **transform** en uno o más documentos según sea necesario para codificar cada uno como un vector.

En este proyecto de tesis, haciendo uso de este mismo procedimiento pero con la finalidad de utilizar para nuestro corpus en español, Perú y además para trabajar con la configuración establecida y el clasificador SVM planteamos el mismo, pero creamos nuestra clase vectorizer 3.1.2.1 con unos parámetros sobreescritos como el tokenizer y el stop_words en español, además podemos notar que el analizador esta en *word*, pues esto mismo nos permite realizar el override de los parámetros de la clase countVectorizer, aquí también podemos hacer uso de los unigramas y configuramos previamente con el tamaño maximo del vector de features a 1000, esto efectivamente esta realizado con una conversión previa de nuestro texto a minúscula , tokenizado y todo lo que involucra el pre procesamiento.

[class Vectorizer]

```
vectorizer = CountVectorizer(
    analyzer = 'word',
```

```
    tokenizer = tokenize,
    lowercase = True,
    stop_words = spanish_stopwords,
    min_df = 50,
    max_df = 1.9,
    ngram_range=(1, 1),
    max_features=1000
)
```

3.1.3. Definición de Validación cruzada y GridSearch para Selección del Modelo

Por otro lado en un típico proceso de machine learning que envuelve entrenar diferentes modelos en un dataset y seleccionar uno con el performance(rendimiento). Sin embargo, evaluar la performance de un algoritmo no es siempre una tarea directa de realizar, hay varios factores que involucran y que pueden ayudarnos a determinar que algoritmo rinde mejor. Uno de tales factores es el rendimiento de la set de validación cruzada y otro de los factores es la elección de parámetros para el algoritmo de aprendizaje automático.

En este trabajo de tesis nos enfocamos entonces en estos dos factores que involucran la elección del mejor algoritmos de aprendizaje automático para nuestro dataset, así, primero vamos a mostrar la importancia de la validación cruzada (*cross-validation*), por que es necesario estos y como logramos implementarlo mediante la libreria python `scikit_learn` de python. Mas adelante mostraremos el algoritmo `Grid_Search` y ver como esto puede se usado para seleccionar automáticamente el mejor parámetro para un algoritmo.

3.1.3.1. Validación cruzada(cross-validation)

Normalmente en un proceso de ML, el dataset es divido, por lo mismo también optamos a dividir nuestro corpus(Dataset), esto se realiza en dos grupos uno para realizar el entrenamiento(*trainning*) y el set de prueba(*test*); el set de entrenamiento es entonces usado para entrenar el modelo y el conjunto de test para evaluar el rendimiento de nuestro modelo, no obstante este enfoque podría conducir al problemas de la varianza. En otras palabras podría presentarse el escenario donde nuestro accuracy varié ligeramente en diferentes test realizados,

usando el mismo set de data y el mismo algoritmo, a esta varianza es referido el problema de varianza.

Numerosos estudios han mostrado la importancia de este problema debido a que podría presentar un problema en la puesta en práctica de nuestro algoritmo de aprendizaje automático, esto mas en desarrollo, una plausible solución para esto es el ya vastante usado *K-Fold Cross-Validation* para evaluar el rendimiento donde **K** es cualquier número entero. El calculo de *K-Flod cross-validation* es de forma directa y como su nombre lo denota es la elección de k pliegos(*fold*), entonces nosotros dividimos la data en **K** pliegues o folds. Una vez divido el Dataset en k folds, un set de $k-1$ son usados para el entrenamiento y un fold se guarda para el test y así el algoritmo es entrenado y probado k veces, cada vez un nuevo set es usado para el test y el resto es guardado para el entrenamiento, finalmente el resultado de K-Fold Cross-Validation es el promedio de los resultados obtenidos por cada set.

Ahora vamos a mostrar este concepto asumiendo $K = 5$ como ejemplo: supongamos que queremos realizar 5-fold cross validation. para hacer este proceso, lo primero es dividir la data en 5 sets, para esto nosotros apropiadamente hemos denotado como: SET A, SET B, SET C, SET D y SET E. Debido a que k es 5, entonces el algoritmo es entrenado y probado k veces. Entonces en el primer pliegue tenemos: el conjunto de set desde el SET A hasta el SET D son usados para el entrenamiento mientras que el SET E para el test como se puede apreciar en la figura. En el segundo pliegue (fold): el SET A, SET B, SET C y SET E son usados para el entrenamiento y el SET D es usado para el test, y asi sucesivamente este proceso continua hata que cada SET es usado por lo menos una vez para el entrenamiento y una vez para el test. El resultado final es el promedio de los resultados obtenidos usando todos los folds. Esta es la forma en que el cross-validation ataca el problema de la varianza, usando una desviacion estandar de los resultados obtenidos de cada fold, con esto nosotros pudimos encontrar la varianza sobre todo el resultado.

Cross validation con scikit_learn .- El primer paso para la implementación del algoritmo cross-validation en python es tener listo el modelo de clasificación que vamos a usar, para ello tenemos que definir esto mismo antes de pasar al uso de validación cruzada y como casi todos los algoritmos usados en machine learning o data science nosotros solo tenemos que importar el método de la librería cross_val_score, el método puede ser usado de `sklearn model_selection`, así este método nos retornará el promedio del accuracy para todos los folds. sin embargo este mismo también tiene parametros para la configuracion básica y obviamente alimentamos con el

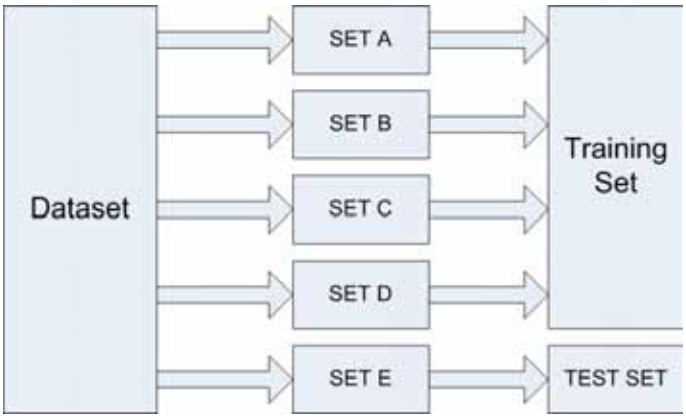


Figura 3.8: Ejemplo de la division de pliegues (k fold) para hacer la validacion cruzada.

método clasificador, dataset, y el número de folds en cv; esto mismo es *classifier, x_train, y_train, cv=5* el primer parámetro es el clasificador , el segundo y tercero corresponder al dataset osea los features y el label correspondiente, y finalmente como ya lo mencionamos el cv muestra en numero de folds K= 5, el código ejemplo es mostrado acontinaucion.

La implementacion de este metodo es muy accesible si tomamos como base y uso una libreria como scikit-learn que tiene base en python para uso de la colección de algoritmos para el Machine learning, así que para este trabajo de tesis se utilizó `cross_val_score` de *scikit-learn*, para esto necesitamos tener primero nuestro modelo de entrenamiento bien definido osea, dicho en otras palabras debemos tener nuestro modelo de clasificación definido y apropiadamente configurado, además como el ejemplo anterior vamos a necesitar el conjunto de entradas en este caso vamos a tener una entrada de features(características) que para este trabajo de tesis se opto por la vectorizacion y los labels o etiquetas que tenemos ya definidos e hicimos una apropiada configuración en el pre procesamiento y creamos un campo de polaridad, también tal y como nuestro el ejemplo anterior, nosotros optamos el uso de $k = 5$ con lo que nuestra data será muestreada y evaluada con la configuración de la figura 3.8. Para realizar este método nos valimos de la librería muy extendida en python scikit-learn que realmente facilita el uso tal y como se suele hacer en data science a continuación mostramos el algoritmo muy general para realizar esto, asumiendo que los detalles menores de conversión de features y las funciones externas no son tan relevantes para este punto nos centramos en mostrar la validación cruzada:

[Cross Validation]

```
1 # primero: llamada a la funcion cross_val_score de scikit learn
2
```

```
from sklearn.model_selection import cross_val_score
# creamos la instancia al metodo con parametros
# model : clasificador definido anteriormente "linearSVC"
# x_train: corpus de texts features
# y_train: polaridad (labels)
# cv : K=5 fold
scores = cross_val_score(
    model,
    corpus_data_features_nd[0:len(tweets_corpus)],
    y=tweets_corpus.polarity_bin,
    scoring='roc_auc',
    cv=5
    )
# mostramos promedio score
scores.mean()
```

3.1.3.2. Grid Search y Selección de parámetros

Otro problema que hemos comentado para la elección de nuestro mejor modelo de Aprendizaje automático técnicamente hablando del algoritmo de ML, son los parámetros que mejor se ajustan para alcanzar un efectivo accuracy; así tanto en Data Science como en Machine Learning los modelos suelen tener dos tipos parámetros: el primer tipo de parámetros son los parámetros aprendidos a través de un modelo de aprendizaje para nuestro caso en particular podemos identificar estos como: vect_max_df, min, vect_ngram_range (unigramas, bigramas), numero maximo de interacciones, loss, mientras que el segundo tipo de parámetros son los *hiper parámetros* que pasamos al modelo de machine learning para este caso tenemos por ejemplo: el kernel, n_jobs= , scoring, estos parámetros ya están de antemano definidas por nosotros tomando en cuenta los trabajos relacionados en el estado de arte actual para este tipo de tareas y también consideramos el mejor método que se ajusta aun buen accuracy, en promedio, pero estos son ajustados(*tunning* termino en ingles) pero no son propiamente aprendidos por el modelo sino que son externamente suministrados en este caso en particular, introducidos por nosotros de acuerdo a nuestra metodología de investigación 3.2.1

Dentro de este sección vamos a explicar la implementación en la ultima parte de Grid

Search que antes debidamente especificamos el clasificador usado SVM, asi como el tipo de kernel y los demas hiper parametros necesarios. Bueno, nosotros primero hicimos un configuracion aleatoria de los valores para los hiper parámetros como se suele trabajar en machine learning y ver que parámetros resulta tener mejor performance, el cual llamamos “prueba y error” , sin embargo este tipo de método es un método exhaustivo, es decir es un proceso largo y muchas veces agotador con el uso de recurso computacional. También no es un trabajo fácil comparar los diferentes algoritmos con sus respectivas configuraciones de sus **hiper parámetros** debido justamente a que un algoritmo con una configuración diferente en un solo parámetro de variación pequeña podría tener un mejor performance(rendimiento) que otro; y si uno de los parámetros varia, quiza este mismo algoritmo tenga un performance mucho mas bajo que antes. Finalmente, en vez de una elección muy aleatoria y variada de acuerdo al criterio del investigador no es muy recomendable y puede presentar error. En vez de este enfoque decidimos enfocarnos en una selección desarrollada por un algoritmo que automáticamente realize este proceso y encuentre los mejores parámetros de un modelo particular que hace el trabajo al cual se le conoce como grid search que vamos a desarrollar la implementación a continuación.

[parametros]

```
# mostrar los mejores paremetros
grid_search.best_params_
'''
{'cls__C': 0.5,
 'cls__loss': 'squared_hinge',
 'cls__max_iter': 500,
 'vect__max_df': 1.9,
 'vect__max_features': 1000,
 'vect__min_df': 10,
 'vect__ngram_range': (1, 1)}
'''
```

3.1.3.3. Grid Search con Scikit-Learn

A continuación vamos a mostrar la implementación del algoritmo grid search centrándonos en el desarrollo de nuestra tarea de investigación usando una configuración y librerías relacionadas al procesamiento del lenguaje natural y el modelo de clasificación propuesto es

decir SVM, bueno para realizar este proceso primero debemos ejecutar el modelo seleccionado anteriormente, por que como ya lo mencionamos anteriormente el algoritmo *Grid Search* es aplicado a un modelo. Para implementar este método necesitamos primero importar la clase GridSearchCV de la librería sklearn.model_selection.

Una vez importado la clase, el primer paso que se realiza es la creación de un diccionario de todos los parámetros y su correspondiente conjunto de valores inicializados aleatoriamente. El nombre de los items del diccionario corresponden a los nombres de los parámetros correspondientes con sus respectivos valores, todo esto detallado anteriormente podemos ver en el siguiente código 3.1.3.3, que muestra los parámetros y configuración usada.

[Grid Search param]

```
from sklearn.model_selection import GridSearchCV

vectorizer = CountVectorizer(
                analyzer = 'word',
                tokenizer = tokenize,
                lowercase = True,
                stop_words = spanish_stopwords)

pipeline = Pipeline([
    ('vect', vectorizer),
    ('cls', LinearSVC()),
])


parameters = {
    'vect__max_df': (0.5, 1.9),
    'vect__min_df': (10, 20,50),
    'vect__max_features': (500, 1000),
    'vect__ngram_range': ((1, 1), (1, 2)),  # unigrams or bigrams
    'cls__C': (0.2, 0.5, 0.7), # penalidad para el error
```

```
    'cls__loss': ('hinge', 'squared_hinge'), # hinge: svm
    ↪  loss(standar): hinge: hinge^2
    'cls__max_iter': (500, 1000) # numero maximo de iter.
}
```

El código mostrado debemos considerar que creamos un diccionario parameters, para nuestro ejemplo en particular con siete parámetros. Los valores de estos parámetros que queremos probar son pasados en la lista, por ejemplo en el código de arriba son referidos al tamaño de los vectores definidos para el vectorizer, con un tamaño maximo de 500 a 1000 , mínimo de 10,20, ó 50; los otros parámetros que vamos a probar son por ejemplo si usamos unigramas o bigramas los cuales están denotados en el parámetro vect_ngram_range con $((1,1),(1,2))$ otro parámetro importante para definir el nivel de penalidad para el error impuesto inicialmente es dada con tres opciones en C como 0.2, 0.5 y 0.7, otros parámetros para probar y ver con cuales tenemos mejores resultados son el **loss** aquí nosotros ingresamos como parámetros *'hinge', 'squared_hinge'* el primer parámetro (hinge) es referido al los aplicado de manera general en el SVM, esto se utiliza para la clasificación de "margen máximo"(el margen maximo esta definido como $l(y) = \text{máx}(0, 1 - t * y)$), especialmente para las máquinas de soporte vectorial y el squared es referido a [2]. Finalmente podemos encontrar que pusimos una iteración para el ajuste de entre 500 a 1000 iteraciones como máximo número.

El algoritmo `Grid_Search` básicamente prueba todas las posibles combinaciones de los parámetros(valores) previamente configurados y nos va retornar la mejor combinación con la cual obtenemos un mejor resultado, así para nuestro ejemplo tenemos de acuerdo a nuestra entrada una combinación de $2x3x2x2x3x2x2 = 252$ posibilidades, debido a esto el algoritmo suele ser un poco lento y puede también consumir los recursos de nuestro procesador además la validacion cruzada también incrementa el tiempo de ejecución y por ende la complejidad del algoritmo.

Una vez creado el diccionario de parámetros con los valores posibles configurados el siguiente paso es la creacion de una instancia de la clase `GridSearch`, nosotros le pasamos los valores de la instancia o clasificador, que básicamente es el algoritmo que queremos ejecutar y del cual obtendremos los mejores parámetros con el que tenemos el mejor accuracy, el `scoring` es el hiper parámetro de la métrica, el `cv` corresponde al numero de folds que ya mensionamos anteriormente, en este trabajo de tesis se obtuvo una configuracion que acontinuacion mostramos

[params results]

```
# mostrar los mejores paremetros
grid_search.best_params_

'''
{'cls__C': 0.5,
 'cls__loss': 'squared_hinge',
 'cls__max_iter': 500,
 'vect__max_df': 1.9,
 'vect__max_features': 1000,
 'vect__min_df': 10,
 'vect__ngram_range': (1, 1)}
'''
```

este ultimo quiere decir que en la penalidad óptima es de 5, y lo óptimo usado para esta tarea es el cuadrático, además solo se requiere 500 iteraciones y es mejor unigramas con un tamaño de 1000 para el feature, por la anterior configuración también obtenemos un accuracy de entrenamiento junto al modelo de cross validation de 0.8102 que es un buen desempeño para nuestra tarea con lo que podemos usarlo para el desarrollo de una manera confiable con esta exactitud.

3.2. Aprendizaje Automático (*Machine Learning*)

Como hemos ya definido el campo de estudio del Aprendizaje Automático (del inglés ML), hoy en dia existen numerosos algoritmos relacionados a esto y entre los mas importantes destacan el SVM, Aritifitial Neuronal Networks (ANN), Naive Bayes (NB) de los cuales pudimos observar mediantes los antecedentes que el método de clasificación SVM obtiene un resultado considerable que sus similares anteriormente mencionados, a continuación detallamos la clasificación con las maquinas de soporte vectorial y una breve descripción de otro método implementado. Debemos indicar también que en este trabajo de tesis se optó por SVM debido a la tarea especifica centrada en opinión negativa y positiva, existiendo una polaridad binaria y como pudimos notar las Máquinas de Soporte Vectorial originalmente fueron desarrolladas para una clasificación binaria y no es de sorprender que funcione bien y tenga un mejor rendimiento en este tipo de tareas, ademas se resalta en el survey de analisis de sentimientos pagina 8 que indican por que las SVM son ampliamente y exitosamente usado en analisis de sentimientos con

un pequeño grupo atraido por el uso de ANN para el enfoque de aprendizaje de sentimientos Medhat et al. (2014).

3.2.1. Metodos de Clasificación

Los métodos de clasificación referidos a los algoritmos usados en el aprendizaje automático para clasificación de los datos, existen numerosos métodos para alcanzar lo mencionado. Uno de los trabajos previos muestra un analisis de los datos en analisis de sentimientos y hace una comparacion de los diferentes metodos existentes como SVM y otros usando la representación de Tf-idf().

	F1 / HIPOTESIS 1			F1 / HIPOTESIS 2						F1 / HIPOTESIS 3						
	NB	SVM	J48	NB	SVM	J48	NB	SVM	J48	NB	SVM	J48	NB	SVM	J48	PROMEDIO
UNIGRAMAS	71,6	82,9	71,8	71,9	77,9	68,1	75,7	81,3	73,8	63,6	61,3	63,5	66	71	57,1	70,50
BIGRAMAS	74,8	82,3	67,4	70,7	75,5	63,9	74,7	75	76,1	56	63,4	52,4	62,8	66,5	62,1	68,24
TRIGRAMAS	71,3	79,3	54,9	68	68,7	55,4	68,4	82,2	70,4	60,5	63,6	51,4	63,3	66,7	59,9	65,60
TF-IDF	71,6	82,9	71,8	71,9	77,9	68,1	75,8	81,2	73,8	63,5	61,3	63,5	65,9	71	57,1	70,49
TF-RFL	94,8	91,9	98	95,6	92,1	95,1	96,1	89,2	99,5	57,2	65,5	57,3	72,5	65,8	70,8	82,76

Figura 3.9: Tabla comparativa que muestra los diferentes metodos con uso de svm como una parte importante y sobresaliente VALDEBENITO (2014)

3.2.2. Máquinas de vectores de soporte(SVM)

El método de clasificación binaria mas utilizado en este campo son las maquinas de soporte vectorial, esta existe en diferentes formas lineal y no lineal. Una máquina de soporte vectorial es un clasificador comúnmente usado con tareas que involucran aprendizaje supervisado. Para nuestro caso en particular se hizo uso de este método debido a la tarea de clasificación binaria, clasificando en opinión positiva y negativa respectivamente.

Las máquinas de soporte vectorial harán el trabajo de clasificación para nuestro caso específicamente la tarea de clasificar opiniones positivas y negativas en las palabras, en términos geométricos el problema es resolver a cual de los lados positivo o negativo pertenece un texto así identificar una frontera de decisión lineal entre dos clases es intuitiva. Esta frontera para dividir nuestros textos lo vamos conseguir mediante una linéa de separación de hiper plano, tal y como el SVM(support Vector Machine) lo hace a través de una linea, esto además maximiza el espacio del hiper plano. Sin embargo, las SVM incluye una función llamada kernel, el cual

permite realizar separaciones para datos muy complejos en sus distribución y más difíciles de dividir. Así esta función y sus variaciones nos permiten realizar separaciones mas alcanzables de datos difíciles de separar, esto se consigue proyectando la información en un espacio de características de mayor dimensión. Para nuestro caso vamos hacer uso del kernel lineal que implementa `LinearSVC` pero con la variación bilineal, esto permite mas flexibilidad para escoger las penalidades y escalar en el ajuste del loss.

Para la implementación se utilizo la librería de sklearn, espeficicamente su extensión SVM con la variación `linearSVC` como se muestra a continuación.

[SVM]

```
from sklearn.svm import LinearSVC # SVM

model = LinearSVC(C=.2,
  ↪  loss='squared_hinge',max_iter=1000,multi_class='ovr',
              random_state=None,
              penalty='l2',
              tol=0.0001
)
```

3.2.3. Otros Métodos

Se pudo observar que los clasificadores que obtiene resultados considerables para el idioma inglés en esta misma tarea son las redes neuronales y el clasificador bayesiano llamado "Naive bayes", pero debido a que en este trabajo de tesis nos enfocamos a clasificar en dos grupos, siendo a si una tarea de clasificación binaria el método de SVM es el que tiene mejor resultado justamente por que este tiende a tener mejor resultado en clasificación binaria, esto mismo se experimentó en el desarrollo de identificación de opinión en redes sociales. Así las SVM ocupan una parte importante en identificación de opiniones tal y como se pudimos detallar en trabajos anteriores como el de Becerra (2017).

3.3. Pos Procesamiento y Almacenamiento de Datos

3.3.1. Post-Procesamiento

3.3.1.1. Validación (ROC)

Una vez tengamos nuestro modelo entrenado con los parámetros específicos ajustados y evaluados de manera que podamos afrontar problemas de la vida real, en nuestro caso específicamente tweets recolectados directamente de la red social. El siguiente paso para verificar si nuestro modelo realmente obtiene un buen porcentaje de rendimiento con respecto a las opiniones positivas y negativas, es usar la curva ROC como medida de precisión para poder ratificar evaluamos la curva ROC, que es el encargado de mostrar como nuestras predicciones son identificadas correctamente con respecto a los falsamente identificados. Este paso es vital en modelos de clasificación, por que nos permite determinar si un modelo realmente refleja su medida de rendimiento, así que debemos hacer un diagrama de la taza de TP frente a FP.

3.3.1.2. Predicción de polaridad

Eliminación de ruido Este procedimiento es realizado una vez hecho el entrenamiento, en este trabajo de tesis se realiza este paso como una forma de evaluar el modelo entrenado, para ello se utiliza el modelo entrenado o un modelo guardado previamente entrenado al cual en algoritmos y proyectos de aprendizaje automatico se le denomina *pre-trained*. Inicialmente se implemento la extraccion directa del texto (contenido tweet) para la alimentacion directa y la predicción, pero debido al ruido inmenso que se presenta en este tipo de dataset y mas aun recolectada por nuestro propio medio, en la region Perú, es necesario primero hacer una limpieza de ruido. La limpieza de ruido no es tan profundo como en el pre-procesamiento debido que directamente recolectamos tweets de la region Perú, se elimino y configuro algunos, para ello se implemento el método parse, el cual implementa la eliminacion de ruido: *URL, link videos, HashTag*, entre otros, para luego finalmente pasarlo a un count vectorizer y convertir nuestro texto en números y finalmente poder predecir.

Acontinuacion se muestra el proceso de eliminacion de ruido y la implementacion del algoritmo al que denominamos **parse** en python.

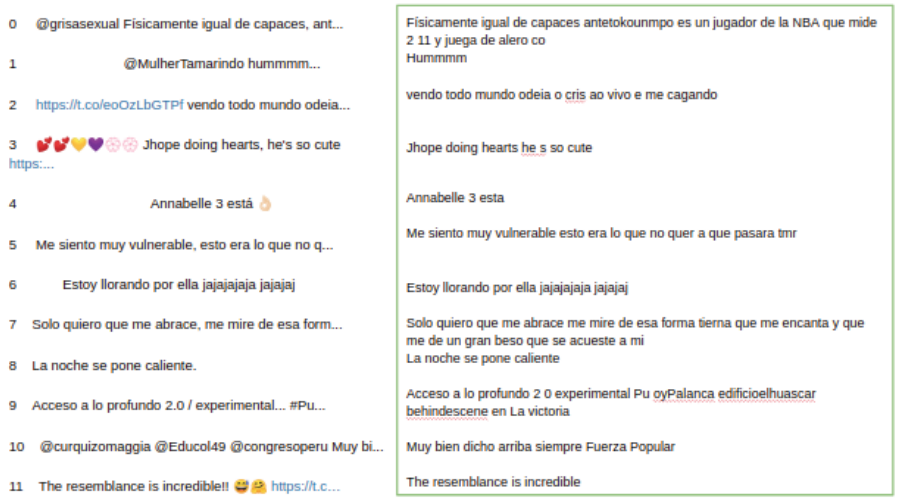
[parse]

```
# importamos las librerias necesarias para el algoritmo
import pandas as pd
import numpy as np
# cargamos el archivo recolectado
tweets_raw = pd.read_csv('tweets_test.csv', index_col=0)
# convertimos los tweets
test_tweets = []
for tweet in tweets_test:
    x = tweet
    t = ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z
    ↪  \t])|(\w+:\/\/\S+)"," ",x).split())
    test_tweets.append(t)
    print(t)

content = pd.DataFrame(test_tweets)
# guardamos el texto_test e
content.to_csv('texto_test.csv')
```

La primera parte a afrontar es la eliminación de ursl y links, numero y Hashtags, este conjunto de datos es cargado como se puede apreciar en la figura 3.10, una vez aplicado el algoritmo tenemos un nuevo texto el cual se aprecia con menos ruido en la figura (lado derecho) esto ultimo sera alimentado directamente a nuestro embeding y se predice la polaridad de cada contenido de texto.

Predicción : En esta etapa vamos a trabajar y desarrollar el proceso de prediccion de los contenidos de twitter obtenidos para nuestro caso en particular con la recoleccion de nuestro propio contenido de twitter ver figura 3.11. Para llevar a cabo este procedimiento vamos a primero pasar como entrada(input) el contenido de texto (publicacion twitter) y acontinuacion vamos a usar el modelo propiamente entrenado y listo para hacer la prediccion masiva, para esto nosotros hemos hecho uso de una lista almacenada de datos en formato csv el cual contiene un cantidad de tweets recolectados al cual le llamamos lista de contenido y le pasamos a nuestro modelo con el manipulador de datos pandas. Finalmente la salida de nuestra prediccion sera un etiquetado de nuestra lista de contenidos de twitter con un etiqueta “1” o “0” el primero para identificar si un contenido es positivo y el segundo para identificar si el contenido de twitter es



0	@grisasexual Físicamente igual de capaces, ant...	Físicamente igual de capaces antetokounmpo es un jugador de la NBA que mide 2 11 y juega de alero co
1	@MulherTamarindo hummmm...	Hummmm
2	https://t.co/eoOzLbGTPf vendo todo mundo odeia...	vendo todo mundo odeia o cris ao vivo e me cagando
3	💕💕💛💜🌸🌸 Jhope doing hearts, he's so cute https:...	Jhope doing hearts he s so cute
4	Annabelle 3 está 👌	Annabelle 3 esta
5	Me siento muy vulnerable, esto era lo que no q...	Me siento muy vulnerable esto era lo que no quer a que pasara tmr
6	Estoy llorando por ella jajajajaja jajajaj	Estoy llorando por ella jajajajaja jajajaj
7	Solo quiero que me abrace, me mire de esa form...	Solo quiero que me abrace me mire de esa forma tierna que me encanta y que me de un gran beso que se acueste a mi
8	La noche se pone caliente.	La noche se pone caliente
9	Acceso a lo profundo 2.0 / experimental... #Pu...	Acceso a lo profundo 2 0 experimental Pu oyPalanca edificioelhuascar behindescene en La victoria
10	@curquizomaggia @Educol49 @congresoperu Muy bi...	Muy bien dicho arriba siempre Fuerza Popular
11	The resemblance is incredible!! 😂🤗 https://t.c...	The resemblance is incredible

Figura 3.10: El lado izquierdo se muestra tweets recolectados y con mucho ruido de escritura. La imagen derecha muestra los mismos tweets con filtros de limpieza de ruido.

negativo. Para el desarrollo del algoritmo, para este punto solo se hace llamado primero a la clase

	contenido
15	perezv Haber brutosKi los cuellos blancos se cayo en la corte de Espa a Mamani es juzgado en fuero penal d
16	El problema es la calidad de la se al pero eso ya eso otro tema
17	No recordaba lo cagu de risa que es esta pel cula
18	Siempre a esta hora te persiguen los hubiera
19	xito asegurado

Figura 3.11: Ejemplo de contenido de tweets recolectados para la predicción.

pipeline, ademas creamos un campo para almacenar los valores que tendran cada contenido de twitter asi armamos nuestra lista de contenidos con sus respectivos etiquetado como se puede apreciar en la figura 3.12:

[Prediccion de Polaridad]

```
#  primero entrenamos y luego vamos a predecir
pipeline.fit(tweets_corpus.content, tweets_corpus.polarity_bin)
#hacemos el llamado a la clase pipeline y pasamos
# el contenido te tweets(lista)
tweets_collected['pred'] = pipeline.predict(tweets_collected.contenido)
```

El almacenamiento de los datos se realizó en un archivo `csv`, el cual es un formato para manejo de contenido textual y numérico, que permite la flexibilidad de almacenar una

	contenido	pred
15	perezv Haber brutosKi los cuellos blancos se cayo en la corte de Espa a Mamani es juzgado en fuero penal d	0
16	El problema es la calidad de la se al pero eso ya eso otro tema	0
17	No recordaba lo cagu de risa que es esta pel cula	1
18	Siempre a esta hora te persiguen los hubiera	1
19	xito asegurado	1

Figura 3.12: Ejemplo de contenido de tweets despues de la predicción.

inmensa cantidad de datos y es fácilmente accesible desde un entorno python, así que nosotros consideramos almacenar toda la cantidad de contenido de texto (tweets) en una columna llamada tweets y en otra columna su etiquetado llamado label. como se puede apreciar en el diagrama(diagrama de un ejemplo de datos), lo anterior se realizó para el entrenamiento y la evaluación.

También debemos indicar que usamos el formato `.xml` pues debido a que el conjunto de Dataset ofrecido de manera publica para Perú y en español esta proveeida por **TASS**, este formato cuenta con etiquetas que engloban el tweet con otras informaciones extras que se hacen uso para el análisis de sentimientos y otras tareas compartidas como se muestra en la figura(figura que muestra contenido xml original). Para este trabajo de tesis precisamos obtener como información relevante el contenido del tweet y la polaridad, debido a que estos son los datos ascensionales para nuestra tarea especÍfica, así la polaridad por convención para nuestro caso se denoto como label(etiqueta) y el contenido de tweets en otro campo y desestimamos otras informaciones adicionales como tiempo, fecha, tweetId, aspect, entre otros.

Capítulo 4

Resultados

4.1. Resultados Experimentales

Para realizar nuestra evaluación y ver la efectividad de nuestro modelado y además realizar nuestros experimentos vamos a describir el dataset o corpus que fue usado en este trabajo de tesis. A continuación vamos a ver las métricas utilizadas y los diferentes diagramas que muestran los resultados realizados; explicando en las secciones siguiente.

4.1.1. DataSet

El DataSet que vamos a usar en este trabajo de investigación se llama **TASS**, este hace referencia al Taller de Análisis Semántico del SEPLN (Sociedad Española para el Procesamiento del Lenguaje Natural) que desde el 2012 sea ha realizado con el motivo de la innovación en el Análisis de Sentimientos en Español. El TASS creo una base de datos con la cual se puede trabajar, desde entonces se ha mantenido como el principal medio para el desarrollo de esta tarea para el idioma español y además motiva a la comunidad científica el desarrollo de tareas que involucran el análisis de sentimiento para el idioma español.

El TASS tiene una gran variedad de bases de datos para la investigación de opiniones y análisis de sentimientos, así este taller tiene disponible una gran variedad de datos coleccionados cuidadosamente, cada año siguiendo un criterio riguroso para el uso y el desarrollo de análisis de sentimientos para el español, desde el 2017 este mismo evento aumento su colección de datos añadiendo variaciones del lenguaje hablado para el español, así ahora se tiene un Dataset para

español-Perú, español-costa rica, español-España, esta misma hace que nuestro trabajo de tesis sea viable y respaldado en una base sólida. Este DataSet es presentado en un archivo *xml* con una forma universal de colección de mensaje publicados en la red social Twitter con etiquetas de *polaridad* que indican la etiqueta positiva, negativa o neutra de cada tweet, además para el español Perú se tiene 1500 tweets para el entrenamiento y 1428 para el test como se muestra en la Tabla 4.2, además el **TASS** tiene contenido relacionado a la televisión y espectáculos, al medio Político y el Médico como muestra la Tabla 4.1.

Otro punto que consideramos en el manejo de las bases de datos es el tamaño de los datos a entrenar, como todos conocemos que un algoritmos de Aprendizaje automático necesita de una vasta cantidad de información para obtener un buen rendimiento y tal como podemos observar no se tiene un promedio tan alto para los datos de Perú, pues decidimos aumentar nuestro DataSet incluyéndolo en el DataSet ofrecido por el TASS con todos los temas convenientes, otro hecho que también se considero, es el hecho que por ejemplo para los temas generales (archivo DataSet) decidimos unir varios data set (data set orientado a Perú y varios data set orientado a un español neutro), obteniendo alrededor de 16 mil tweets; con el que pudimos obtener un mejor rendimiento.

Para hacer una análisis mas detallado decidimos hacer un ploteo y una análisis de datos, para ello usamos la librería `seaborn, matplotlib` con los cuales pudimos observar por ejemplo que nuestros datos están muy desvalanceados; obteniendo un 87.9 % de contenido relacionado a la opinión negativa, lo que hace que este conjunto de datos no sea tan equitativo con su similar positivo; así también podemos notar en la representación cuantitativa que un aproximado de 2400 tweets cuyo contenido esta clasificado como negativo y solo una cantidad menor de 500 tiene la clasificación como positivo. Debemos mencionar que estos datos son los que nos proporcionan el TASS, así que tuvimos que implementar formas para hacerlo mas equitativo, como se mencionó en pasos anteriores.

4.1.2. Métricas

A pesar que existen numeroras metricas para la evaluación tanto del modelo de entrenamiento y la precision del algoritmo al momento de la identificación, para nuestra tarea especifica de análisis de opiniones para el idioma español, de acuerdo a los trabajos relacionados en el estado de arte actual para el idioma ingles y para algunos trabajo iniciales para el español,

DATASET (Corpus)	File	Nº Tweets
TASS	General	7219
TASS	Social	1008
TASS	SocialTV	1773
TASS	Politico	784
TASS	GeneralTest	6212
TASS	Total	16996

Cuadro 4.1: Tabla: Muestra el contenido por archivo y temas en español.

DATASET (Corpus)	File	Nº Tweets
TASS Peru	Training	1000
TASS Peru	Development	500
TASS Peru	Test	1428
TASS Peru	Total	2928

Cuadro 4.2: Tabla: Contenido del Data Set para el Caso de estudio Peru.

se tiene una métrica que son efectivamente aplicados en este tipo de tareas y el mas conocido es denotado como **F1** que normalmente es conocido como "*f-score*", sin embargo existen otras metricas que ayudan en la evaluación que describiremos en los siguientes párrafos.

4.1.2.1. Precisión-Recall

Como hemos mensionado anteriormente la efectividad (del ingles **accuracy**) no es el único metodo para evaluar nuestro modelo(algoritmo) de Aprendizaje Áutomático, al igual que el accuracy existen dos métricas muy útiles tales como Precisión y Recall; estas dos métricas proveen una percepción exhaustiva dentro del rendimiento de un clasificador binario como el **SVM**.

Antes de definir lo que significa cada uno de estas métricas, primero vamos a desvelar la clave para aplicarlos finalmente en una ecuación que sera muy sencillo de aplicarlos en cualquier método de clasificación(algoritmo IA). Para esto primero necesitamos saber el concepto de una matriz de confusión, que básicamente es una medida de rendimiento basada en

las salidas(predicción) de nuestro método comparado con los resultados esperados de nuestro conjunto de evaluación(data test), esto es fácilmente apreciable en un cuadro de doble entrada, los cuales tienen valores relacionados a las predicciones y valores esperados como se muestra en la figura 4.3, en esta podemos observar que tenemos valores `TP, FP, FN, TN`, las cuales vamos a explicar acontinuación.

- **TP**: Este valor esta relacionado a la predicción *Positiva* y que efectivamente es lo esperado, que esta correctamente realizado.
- **FP**: Aquí también podemos ver un valor de predicción *Positiva*, pero que no es lo que esperamos(falso), con el que podemos concluir que es incorrecta.
- **TN**: Sin embargo en este valor la predicción es *Negativa* y además es lo que se espera que sea falso, por lo que esto es correcto.
- **FN**: Este valor por el contrario la predicción es *Negativa* pero que no es lo esperado(Falso), lo que indica que es incorrecta.

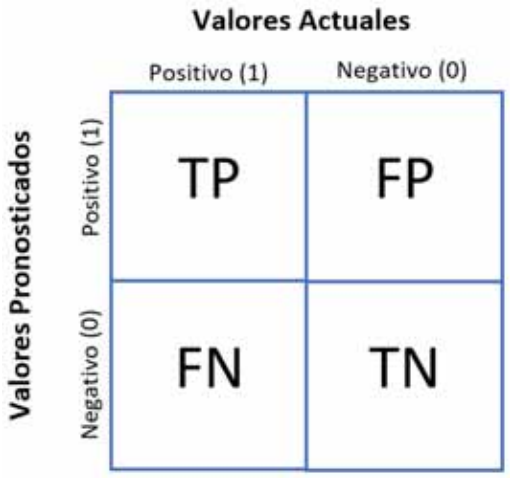
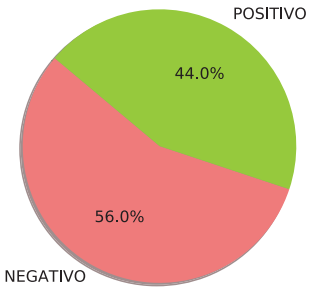


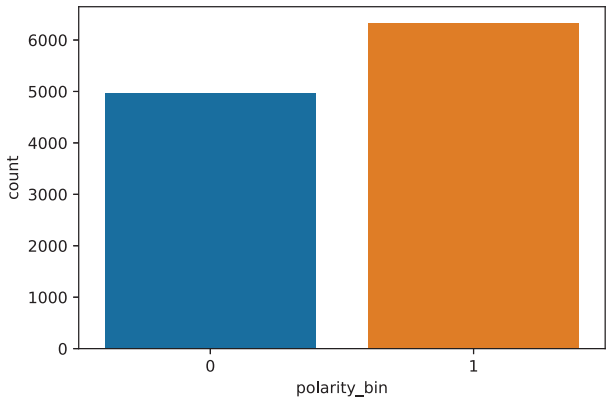
Figura 4.1: Matriz de confusion con los valores TP,TN, FP,FN los values(valores esperados), Prediction(valores de nuestro metodo).

Para concluir debemos resumir que los valores **TP, TN** deberían tener un valor cuantificable mucho mayor que los otros, debido a que estos son considerados como una correcta salida de nuestro metodo de clasificación, esto además se puede usar para evaluar métodos de Aprendizaje Áutomáticos que clasifican mas de solo dos grupos, extensivamente las matrices de confusión son muy usadas en Inteligencia artificial debido a la flexibilidad en el cálculo y su efectividad.

Para nuestro trabajo de tesis realizamos el test con una cantidad de datos de 11305 tweets, distribuidos en un 44 % como positivo y un 56 % como negativo, como se puede apreciar en la figura 4.2.



(a)



(b)

Figura 4.2: Polaridad de contenido del dataset

Por lo cual obtuvimos una matriz de confusión como se muestra en la figura 4.3.

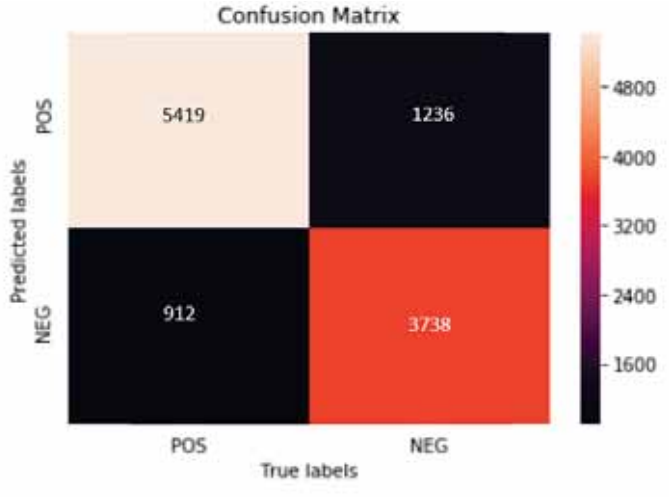


Figura 4.3: Matriz de confusión con los valores POS, NEG.

Precisión .- La precisión mide la exactitud de un modelo de aprendizaje, técnicamente mide la exactidud que rinde un algoritmo de Aprendizaje Áutomático, esto nos sirve para identificar a menudo si nuestro proyecto de inteligencia artificial realmente ha aprendido de una forma que podemos cuantificar su aprendizaje en un porcentaje. Matemáticamente esto se puede expresar como el cociente entre los valores **TP** y la suma de los valores **TP, FP**, como se puede observar en la ecuación siguiente.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

Sensibilidad Recall .- El recall mide la sensibilidad de un clasificador, dicho en otras palabras mide la proporción de los actuales casos positivos que son correctamente identificados, un ejemplo practico podria ser : el porcentaje de personas diagnosticadas correctamente con un caso clínico en un hospital. Asi esta sensibilidad o recall es calculado de la forma:

$$recall = \frac{TN}{TN + FP} \tag{4.2}$$

4.1.2.2. Métrica F

Dentro de las medidad de evuacion también tenemos la F, es cual hacer referencia a una mucho mas general debido a que este hace uso de las metricas anteriormente definidas, en otras palabras esta medida conocida como **measure F** que es un promedio armonico de la precisión y el recall, de una forma muy general podría compararse con el del accuracy.

4.1.2.3. Mean absolute error

El error absoluto medio o mean absolute error(ingles) es el promedio aritmético entre los valores originales(valores esperados) y los valores predecidos. Este valor nos da un acercamiento de cuan bien esta nuestra modelo de predicción, al hacer una diferencia directa con los valores de comprobación originales podemos tener una idea de lo cercano o lejano estamos de una acertividad, pero esto mismo nos impide ver si nuestro modelo tiene una predicción errónea o correcta,por lo que no se toma como medida general en algunos casos. esto esta definido como sigue:

$$Meanabsoluteerror = \frac{1}{N}\sum_{j=1}^{N}|y - \hat{y}| \tag{4.3}$$

Los resultados que obtuvimos para un test de datos,se muestran contención tanto en la table 4.3 que indicam primero el entrenamiento(trainning) con una cantidad de datos vastante grande para que nuestro algoritmo pueda aprender y un accuracy de 81 % de promedio con el valor **AUC**, el cual muestra y se califica como un buen desempeño para el desarrollo del mismo, asi también mostramos abajo el test con una cantidad de 673 tweets(contenido extraido de tweeter), además debemos indicar que para esto, ambos tanto como para el entrenamiento como para el test tuvimos que eliminar los casos en los que teniamos un valor denominado como *NONE,NEU* y considerar todo lo demás no sin antes hacer el paso de la polarizacion binaria, así entonces una vez hecho, el pre procesamiento de los datos procedimos a entrenarnos y finalmente hacer nuestra prueba obteniendo una f1 score considerable como se puede apreciar en la imagen 4.4 tenemos aun un largo campo para los cuales no tenemos un resultado preciso, por lo que debemeos mejorar en muchos caso, pero que para fines de investigacion en este tipo de algoritmos y además aplicados para nuestro dataSet caso Peru, pues debemos interpretar como un primer acercamiento efectivo en todos los pasos para determinar el reconocimiento o la idfentificacion masiva de nuestras opiniones posteadas en twitter.

DATASET (Corpus)	Nº Tweets
TASS Peru train	11305
TASS Peru test	673

Cuadro 4.3: Tabla: Contenido dataSet usado para el Caso de estudio Perú.

Uno de los casos que tuvimos que enfrentar en este mismo paso es la rigirnos sola-

mente la clasificación binaria, y debido a eso tuvismo que despreciar una enorme cantidad de información, asi que nuestra data llega a ser reducida tanto para el entrenamiento como para el test ademas de solo considerar dos tipos de datos identificables haciendo esto una clasificación binaria. debemos mencionar tambien que se esta realizando un test de manera masiva ingresando la cantidad de 673 contenido publicado en twitter en un solo instante de modo que hemos cumplido con los objetivos presentados el cual también puede ser afrontado mediante el uso de minibatch que es una técnica de división de la data en pequeños pedazos o bloques de data para así alimentarlos de una manera continua hasta terminar con todo el bloque, esta técnica reduce el procesamiento mejorando el rendimiento aunque el resultado sea casi siempre el mismo al final.

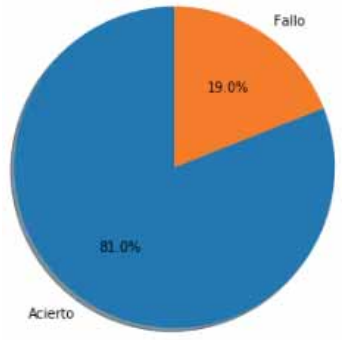


Figura 4.4: Diagrama de acertividad para nuestra tarea de identificación de opiniones caso Perú

4.2. Aportes de la Investigación

- En este trabajo de tesis se propone una nueva arquitectura para el desarrollo de la investigación usando el método de Aprendizaje Automático para la identificación de opiniones en twitter.
- Nuestro trabajo de tesis esta orientado a la implementación para el desarrollo de software, debido a que se encontró diferencias sustanciales que marcan entre una arquitectura de investigacion y pueda ser aplicado directamente a una aplicación de producción final.
- También durante la exploración de los datos se encontró, que no existen trabajos relacionados a este tema que se hayan hecho en la que abordamos con la recolección de los datos, la aplicación del método de Aprendizaje Automático y las pruebas con una DataSet real obtenido de twitter directamente para nuestro país.
- Debido al desvalance de la data mostrada y proveída por el TAS, y la cantidad de infor-

mación desvalanceada que se muestra en nuestros resultados 4.2, se propuso e implemento el uso de un método de balanceo de la data llamada Cross Validacion, encontrando que que mejora el desempeño del algoritmo de Aprendizaje automático, propiamente en el aprendizaje óptimo de las SVM.

- Para la Implementación se hizo una análisis exhaustivo para encontrar los parámetros de configuración así como la configuración de los hiper-parámetros necesarios para la implantación con el método de gridsearch.

- Finalmente este trabajo de tesis muestra un desarrollo e investigación con la implementación y necesaria documentación de los conceptos teóricos para la implementación de una identificador de opiniones pero haciendo una análisis masivo de las redes sociales como el caso de twitter.

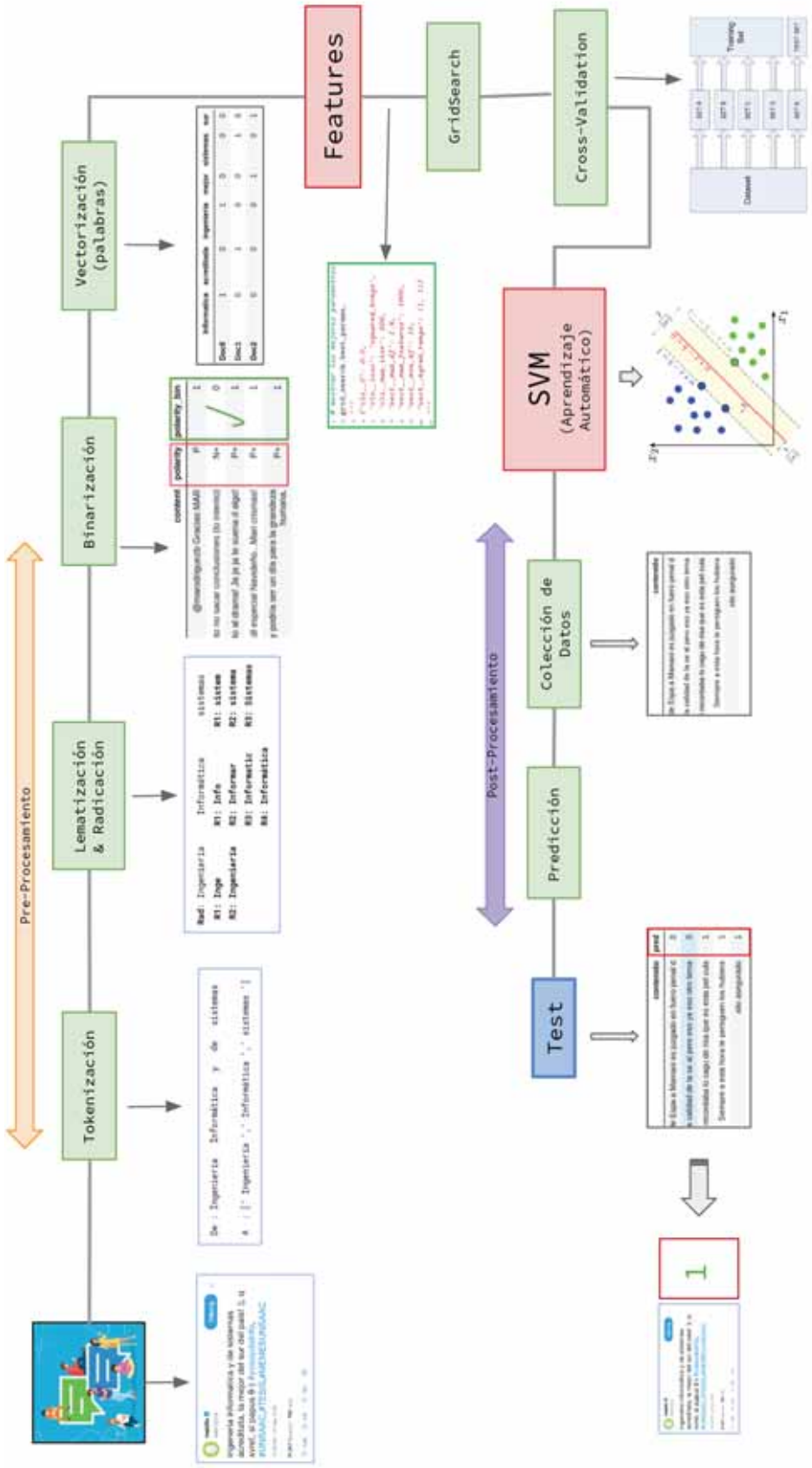


Figura 4.5: Diagrama: Arquitectura propuesta en el trabajo de tesis

Conclusiones

- Se implemento la arquitectura para identificar masivamente opiniones de las publicaciones en twitter mediante Máquinas de Soporte Vectorial para obtener como resultado una calificación positiva o negativa.

- Para el desarrollo del presente trabajo de tesis se eligió cuidadosamente el DataSet TASS de la asociación española para el Procesamiento de Lenguaje Natural (SEPLN) para el caso Perú, debido a que este provee un corpus de tweets recolectados rigurosamente bajo un estricto modelo de recolección de datos aceptado por la sociedad española para el procesamiento de lenguaje natural además es uno de lo más conocidos y ampliamente usado para la tarea de análisis de sentimientos para el español adicionalmente (SEPLN es el máximo ente en investigación de procesamiento de lenguaje natural para la comunidad en español).

- Sé implemento los módulos respectivos de Pre-Procesamiento de texto, extracción de características, aprendizaje automático y la manipulación de datos con librerías de python.

- Sé aplico el algoritmo SVM(Maquinas de Soporte Vectorial) de ML para la identificación de opiniones, debido a que este es el mas sobresalientes dentro del estado de arte actual(trabajos relacionados) y presenta mejor resultado para esta tarea de identificación de opinión.

- Sé hizo una análisis detallado para ver la configuración de nuestros parámetros que mejor se ajustan para una implementación a nivel de desarrollo que nos permitió tener una alta exactitud para la identificación de opiniones ya sea positiva o negativa, este paso es importante por que releva la elección de nuestro ajuste de valores iniciales y la elección de nuestros pasos con el error especifico, pues si se da una elección aleatoria no se llega a tener buen desempeño, esto además nos permitió conocer el mejor resultado en precisión, alcanzando este paso a través del análisis prueba y error y la elección aleatoria, pudiendo

elegir los mejores parámetros(funciones y métodos).

Recomendaciones y Trabajos Futuros

- Se recomienda tener un amplio conocimiento en Procesamiento de Lenguaje Natural debido a la complejidad que este campo en pleno auge pueda involucrar, además del concepto teórico que involucra el Aprendizaje Automático(Deep Learning) que recientemente tiene investigaciones en análisis de sentimientos.
- Se recomienda poder hacer un implementación de desarrollo web en una interface y se pueda aplicar automaticamente, pues debido al proceso que involucra la investigación obtamos por plantear el prototipo.
- Recomendamos aplicar algunos algoritmos en el pre procesamiento como: *normalización de texto* y uso de un diccionario de palabras positivas, que no se llego a implementar por motivos de tiempo; más suponemos que mejorarían nuestra data set en el pre procesamiento y por ende conseguir una mejor precisión en el entranamiento de nuestro modelo.
- Recomendamos probar con otros métodos de extraccion de features como transfer learning u otro método de extraccion de features, que quizá obtenga mejores resultados, que también no se pudo demostrar para fines de este trabajo.
- Recomendamos finalmente tomar como base el presente trabajo para el desarrollo de una tecnología con fines que ayuden a la exploración y percepción libre de opinión en nuestro país, con fines que estén dentro de lo ético.

Bibliografía

Becerra, M. (2017). Análisis de sentimientos en twitter: El bueno, el malo y el¿. In *XX Concurso de Trabajos Estudiantiles-JAIIO 46 (Córdoba, 2017)*.

Burnett., C. M. (2019). Esquema de red neuronal artificial. urlhttps://www.wikiwand.com/es/Red$_n euronal_a rtificial$.

Cambria, E., Havasi, C., et al. (2012). Senticnet 2: A semantic and affective resource for opinion mining and sentiment analysis. In *Twenty-Fifth International FLAIRS Conference*.

G., J. L. (2018). Tipos de aprendizaje automatico. urlhttps://medium.com/soldai/tipos-de-aprendizaje-autom %C3 %A1tico-6413e3c615e2.

Hernández Sampieri, R., Fernández Collado, C., et al. (2014). Metodología de la investigación. *Mac Graw Hill, México*.

Martínez, F. J. M. (2017). Análisis de sentimiento en twitter de las principales compañías del sector asegurador español.

Matich, D. J. (2001). Redes neuronales: Conceptos básicos y aplicaciones. *Universidad Tecnológica Nacional, México*.

Medhat, W., Hassan, A., et al. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Mitchell, T. M. (2019). profesor cornell mellon university, departamento machine learning. https://en.wikipedia.org/wiki/Tom$_{M.M}itchell.[Web; accedidoel17 - 12 - 2019]$.

Morales, K. and Santos, F. (2012). Extensión de archivo csv — valores separados por comas. urlhttps://www.excel-avanzado.com/1064/archivo-csv.html.

Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Pang, B., Lee, L., et al. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Perez, C. E. (2016). A development methodology for deep learning. urlhttps://medium.com/intuitionmachine/a-development-methodology-for-deep-learning-2ce515158bb7.

Sayinath and Murphy, A. (2013). Various optimisation techniques and their impact on generation of word embeddings. https://hackernoon.com/various-optimisation-techniques-and-their-impact-on-generation-of-word-embeddings-3480bd7ed54f. [Web; accedido el 02-06-2019].

Sobrino Sande, J. C. Análisis de sentimientos en twitter.

Tumasjan, A., Sprenger, T. O., et al. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Fourth international AAAI conference on weblogs and social media*.

Turney, P. D. (2002). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

VALDEBENITO, F. I. O. (2014). MinerÍa de opiniÓn y anÁlisis de sentimientos. 5.

Zhao, Y.-Y., Qin, B., et al. (2010). Sentiment analysis. *Journal of Software*, 21(8):1834–1848.