UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Análisis de sentimientos en el Corpus SENTCOVID en español: modelos y técnicas de procesamiento de lenguaje natural, Machine Learning y Deep Learning

T E S I N A

QUE PARA OBTENER EL TÍTULO DE:

A C T U A R I O

P R E S E N T A

JUAN CARLOS BARAJAS ALARCÓN

TUTORA:

DRA. HELENA MONTSERRAT GÓMEZ ADORNO

2024

Jurado asignado:

1.- Act. Francisco Sánchez Villarreal

2.- M. en E.I.O. José Salvador Zamora Muñoz

3.- Dra. Helena Montserrat Gómez Adorno

4.- Ph. D. José Antonio Perusquía Cortés

5.- Dr. Gonzalo Pérez de la Cruz

Tutora:

Dra. Helena M. Gómez Adorno

Análisis de sentimientos en el Corpus SENTCOVID en español: modelos y técnicas de Procesamiento de Lenguaje Natural, Machine Learning y Deep Learning.

Tesina de Servicio Social

Juan Carlos Barajas Alarcón
Licenciatura en Actuaría
Facultad de Ciencias
Universidad Nacional Autónoma de México

Resumen

Este trabajo es un estudio comparativo que aborda la creciente importancia del análisis de sentimientos en el procesamiento del lenguaje natural. Iniciamos con una introducción que contextualiza la relevancia de comprender las emociones expresadas en el lenguaje digital, seguida por objetivos claramente definidos para guiar nuestra investigación. En el marco teórico, exploramos los conceptos esenciales del procesamiento del lenguaje natural, machine learning y análisis de sentimientos, estableciendo una base teórica sólida. El estado del arte revisa prácticas actuales, desde la recolección de datos hasta modelos avanzados de deep learning. En la metodología, describimos el enfoque utilizado para el procesamiento de datos, extracción de características y la evaluación de herramientas y modelos. Los resultados y análisis presentan hallazgos clave, y las conclusiones resumen las contribuciones y ofrecen recomendaciones para futuras investigaciones. Este estudio proporciona una guía integral para investigadores y profesionales, destacando mejores prácticas y herramientas efectivas en el análisis de sentimientos a través del uso de herramientas de PLN y modelos de machine learning.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El análisis de sentimientos, como área de estudio en el campo del Procesamiento del Lenguaje Natural (PLN) y Machine Learning, ha adquirido una relevancia más que significativa en la última década. La capacidad de comprender y analizar opiniones, emociones y actitudes expresadas a través del lenguaje escrito ha abierto un abanico de aplicaciones en campos diversos como la inteligencia empresarial, el análisis de redes sociales, la atención al cliente, entre otros. El Procesamiento del Lenguaje Natural, por su parte, es un campo enmarcado dentro del área de la inteligencia artificial, cuyo objetivo fundamental es facilitar la comunicación entre personas y computadoras mediante el uso de protocolos como los lenguajes naturales. Este estudio comparativo busca explorar y evaluar las herramientas de PLN y los modelos de Machine Learning más relevantes para el análisis de sentimientos, con el objetivo de proporcionar una visión holística de los enfoques actuales y su aplicabilidad en contextos reales.

A lo largo de este trabajo, se abordan definiciones fundamentales de PLN y Machine Learning, se exploran las técnicas de recolección y etiquetado de datos, así como las herramientas y modelos más comúnmente empleados en el análisis de sentimientos. Además, se lleva a cabo un detallado análisis comparativo de los enfoques presentados, con el propósito de identificar fortalezas, debilidades y posibles áreas de desarrollo en este campo en constante evolución.
Con esta investigación, se persigue no solo brindar una comprensión integral y un panorama general del estado del arte actual en el análisis de sentimientos, sino también promover el desarrollo y la implementación efectiva de estas herramientas en aplicaciones prácticas.

1.1. Motivación y Relevancia

En el contexto actual de la era digital, la inteligencia artificial y su constante desarrollo han adquirido gran interés y una importancia significativa en numerosos campos profesionales, uno de ellos es sin duda la ciencia actuarial, ya que esta demanda una comprensión aguda de diferentes áreas de la matemática, como el álgebra lineal, la probabilidad y principalmente la estadística, las cuales son disciplinas clave para el estudio relacionado con el análisis de información con grandes conjuntos de datos, elemento crucial para el auge que ha vivido en los últimos años este campo de estudio y que en la actualidad hemos podido ver su gran potencial en las nuevas inteligencias tales como GPT-1, las cuales utilizan las herramientas de lenguaje y se combinan con el uso de potentes redes neuronales y de un alto poder de recursos computacionales logrando así resultados sorprendentes, los cuales alimentan la motivación en utilizar estos conocimientos para profundizar en el estudio de estas técnicas. El desarrollo

futuro de estos modelos recae en gran parte en el desarrollo de su análisis estadístico y probabilístico y como consecuencia de esto, es un área con gran potencial donde el profesional de la actuaría puede desarrollarse y contribuir a la sociedad.

Este trabajo reviste una significativa relevancia en el contexto donde la interacción humana a través de la tecnología en línea ha alcanzado proporciones masivas, por tal razón, comprender y analizar los sentimientos expresados en el lenguaje digital se ha vuelto de gran importancia. La capacidad de extraer conocimientos significativos de grandes volúmenes de datos textuales es crucial en diversos campos, desde el análisis de opiniones en redes sociales hasta la toma de decisiones empresariales basadas en la retroalimentación del cliente. Este estudio se posiciona en la vanguardia de la investigación al explorar y comparar herramientas de PLN y modelos de machine learning, proporcionando una comprensión más profunda de las tecnologías disponibles y sus aplicaciones en el análisis de sentimientos. La relevancia de este trabajo radica en su contribución al avance de métodos efectivos para interpretar y comprender la complejidad de las emociones expresadas en el lenguaje humano, con implicaciones directas en áreas tan diversas como el marketing, la atención al cliente y la toma de decisiones estratégicas.

1.1.1. Relación de la Actuaría y la Inteligencia Artificial:

La disciplina de la actuaría se basa en gran medida en el análisis probabilístico y estadístico para evaluar y administrar riesgos en una variedad de campos, desde seguros y finanzas hasta planificación y predicción de eventos futuros como planes de retiro. En este contexto, la inteligencia artificial (IA) y, en particular, el Machine Learning, han surgido como herramientas poderosas que complementan y potencian las capacidades analíticas de los profesionales de la actuaría. La intersección de la actuaría y la inteligencia artificial representa un punto crítico de evolución en la profesión, donde el uso de algoritmos y modelos predictivos abre nuevas oportunidades para comprender, predecir y mitigar riesgos en escenarios más complejos.

La estadística es un pilar fundamental en la formación de los actuarios, ya que les proporciona las herramientas para comprender la incertidumbre y la variabilidad inherentes a los fenómenos que intentan modelar y predecir. Ahora, con el auge de la ciencia computacional y del Machine Learning, los actuarios tienen a su disposición una serie de técnicas y enfoques que les permiten abordar y modelar datos de manera más dinámica y productiva. Los modelos de Machine Learning pueden revelar patrones y relaciones en los datos que podrían haber pasado desapercibidos con enfoques tradicionales, permitiendo así una comprensión más profunda de los riesgos y la incertidumbre asociados con los eventos futuros que se estudian en la profesión.

El Machine Learning es especialmente relevante para el actuario debido a su capacidad para desarrollar modelos predictivos precisos a partir de conjuntos de datos complejos y de gran escala. En el ámbito del seguro, por ejemplo, los actuarios pueden emplear algoritmos de machine learning para evaluar el riesgo de siniestros, determinar primas y clasificar asegurados en segmentos de riesgo más precisos. Del mismo modo, en finanzas, la aplicación de estos modelos puede ser invaluable para pronosticar tendencias del mercado, detectar anomalías en transacciones financieras y gestionar carteras de inversión de manera más efectiva.

La integración de la inteligencia artificial y el Machine Learning en la práctica actuarial presenta un cambio fundamental en la forma en que se abordan los problemas tradicionales y en la capacidad de los actuarios para brindar soluciones más precisas y contextuali-

zadas en un entorno empresarial dinámico. Sin embargo, es importante enfatizar que, si bien estas herramientas tecnológicas ofrecen grandes ventajas, también plantean desafíos significativos en términos de ética, interpretabilidad de modelos y comprensión de sus limitaciones, aspectos que deben ser cuidadosamente considerados en el contexto de la profesión.

En resumen, la relación entre la actuaría, la estadística, la inteligencia artificial y en particular el Machine Learning, representa una sinergia poderosa que puede ampliar significativamente la capacidad de los actuarios para comprender y gestionar riesgos en escenarios complejos y dinámicos, aportando valor a sus decisiones y contribuyendo al desarrollo continuo de la disciplina en el contexto de la era digital.

1.2. Objetivos

Este apartado tiene como principal propósito establecer claramente las metas específicas que se persiguen a lo largo de este trabajo con la realización del estudio comparativo. Los objetivos en general pueden formularse de la siguiente manera:

- Comprender el Contexto: Proporcionar un análisis detallado del contexto actual del Procesamiento del Lenguaje Natural y el análisis de sentimientos a través del estudio del trabajo relacionado, identificando las tendencias, desafíos y avances más relevantes en el campo.

- Definir Objetivos de Investigación: Formular objetivos de investigación claros y específicos que guíen la exploración de las herramientas de Procesamiento de Lenguaje Natural y los modelos de Machine Learning que serán propuestos, centrándose en su aplicabilidad y eficacia en el análisis de sentimientos.

- Evaluar las Herramientas y Modelos: Realizar una evaluación exhaustiva de las diversas herramientas de Procesamiento de Lenguaje Natural y los modelos de Machine Learning utilizados para el análisis de sentimientos propuesto, destacando sus características, funcionamiento, ventajas y limitaciones.

- Identificar Fortalezas y Debilidades de las Prácticas Actuales: Identificar y destacar las mejores prácticas en el uso de herramientas de Procesamiento de Lenguaje Natural y modelos de Machine Learning para el análisis de sentimientos, con el objetivo de proporcionar pautas útiles para interesados, investigadores y profesionales en el campo.

- Proponer Recomendaciones: Proporcionar recomendaciones fundamentadas basadas en la comparación de herramientas y modelos, con el objetivo de orientar futuras investigaciones, propuestas y aplicaciones prácticas en el ámbito del análisis de sentimientos.

Estos objetivos contribuirán a la definición de la estructura general del trabajo y establecerán las bases para abordar de manera sistemática la investigación comparativa, de manera que puedan ser respondidas claramente las preguntas que surgen al establecer estos objetivos y el contexto del trabajo y la forma en que abordamos el análisis de sentimientos.

1.3. Organización del trabajo

La organización del trabajo y del estudio en general, con base en el índice proporcionado, será estructurada y presentada de la siguiente manera en cinco secciones sin contar la presente:

- Marco Teórico: En este capítulo se dará una explicación de los conceptos clave en el procesamiento del lenguaje natural para establecer la base teórica de los conceptos que se abordan en el resto del estudio, para posteriormente explorar los fundamentos teóricos del machine learning y algunas de sus aplicaciones en conjunto y su relación con el análisis de sentimientos. Revisión de los conceptos fundamentales en el análisis de sentimientos, estableciendo así un marco conceptual para la comprensión de este texto.
- Estado del Arte: Este capítulo describe de forma detallada las prácticas utilizadas en la recopilación y etiquetado de datos para la elaboración de un corpus (conjunto de datos que se utiliza para el entrenamiento de modelos de lenguaje) y una revisión de las herramientas y técnicas actuales en el procesamiento del lenguaje natural para efectuar el análisis de sentimientos. Examen de modelos de machine learning utilizados en el análisis de sentimientos. Exploración de modelos de deep learning en el contexto del análisis de sentimientos.
- Metodología Utilizada: Descripción de las prácticas utilizadas para procesar los datos recolectados. Detalles sobre la extracción de características relevantes para el análisis de sentimientos. Explicación de la utilización de modelos pre-entrenados en el contexto del estudio. Métodos utilizados para la predicción y el análisis comparativo de las herramientas y modelos evaluados.
- Resultados y Análisis de Errores: Presentación de los resultados obtenidos a través de la evaluación de herramientas y modelos. Análisis de los errores obtenidos en las predicciones de los modelos, de manera que se identifiquen los principales errores analíticamente y con un enfoque estadístico. Se examinan los fallos para comprender sus causas subyacentes y evaluar posibles mejoras en los algoritmos utilizados.
- Conclusiones: Resumen de los hallazgos clave y conclusiones derivadas del estudio comparativo, presentacion de propuestas para el trabajo futuro.

Esta estructura proporciona una guía clara para la presentación sistemática de la investigación, desde la introducción hasta las conclusiones, permitiendo a los lectores seguir lógicamente el desarrollo del estudio y comprender sus contribuciones en cada fase.

Capítulo 2

Marco Teorico

En este capítulo se exponen las principales definiciones y se describen los conceptos fundamentales que se utilizarán en el desarrollo de este estudio, ademas de revisar algunos resultados obtenidos en trabajos relacionoados, de forma que se brinde un contexto actualizado para la comprension y reforzamiento teorico de los temas abordados a lo largo de este trabajo.

2.1. Procesamiento de Lenguaje Natural

El Procesamiento del Lenguaje Natural, comúnmente conocido por sus siglas en inglés NLP (Natural Language Processing), es una rama interdisciplinaria de la inteligencia artificial que combina la ciencia de la computación y la lingüística computacional, y se centra en las interacciones entre las máquinas y el lenguaje humano (Jurafsky & Martin, 2019) [21]. Podemos decir entonces, que es un campo que se encuentra en la intersección de diversas disciplinas y busca replicar la capacidad de comprensión y producción del lenguaje humano en el software, y esta habilidad es profundamente compleja, ya que el lenguaje humano es variado y cambia constantemente con el paso del tiempo. El uso de estas metodologías y tecnologías que permiten a las computadoras interpretar, entender y generar lenguaje natural de una manera que sea tanto significativa como útil es un tema de amplio interes, debido a que se intenta llenar la brecha entre la comunicación humana y la comprensión computacional, y por esa razón es útil para múltiples funciones prácticas, tales como sistemas que permitan que las personas interactúen mas a fondo con la tecnología utilizando lenguaje cotidiano e incluso hasta lenguaje corporal o de señas, de forma que facilite para ellos el uso de dispositivos y aplicaciones sin necesidad de conocimientos especializados en computacion o programacion. El objetivo principal es entonces, permitir que las máquinas comprendan, interpreten y generen lenguaje humano de manera eficiente y efectiva.

El PLN es un campo vasto y diverso que engloba varios subcampos, como el análisis de sentimientos, la generación de lenguaje natural, la traducción automática, y la extracción de información, entre otros. Cada uno de estos subcampos implica diferentes técnicas y enfoques para analizar el lenguaje humano. (Chowdhury, 2003) [8]. En este titulo podemos ver como hace casi dos décadas se vislumbraban sus extensas aplicaciones e impactos en distintos sectores, los cuales hoy en dia existen y son de uso común, algunas de ellas incluyen:

- Asistente de voz y chatbots: Programas que utilizan el PLN para comprender y responder a las consultas de los usuarios en lenguaje natural.

- Traducción automática: Herramientas como Google Translate emplean PLN para proporcionar traducciones de texto o voz de un idioma a otro.
- Análisis de sentimiento: Empresas utilizan PLN para analizar las opiniones y sentimientos de los clientes a partir de comentarios en redes sociales o feedbacks, permitiéndoles mejorar sus productos y servicios.
- Extracción de información y resumen automático: El PLN se utiliza para escanear grandes volúmenes de texto y extraer la información importante o resumir contenido.
- Reconocimiento de voz: Tecnologías como Siri de Apple o Alexa de Amazon reconocen y procesan la voz humana gracias al PLN, facilitando la interacción con la tecnología mediante comandos verbales.

El PLN ha experimentado un auge considerable gracias a los avances en Machine Learning y Deep Learning, su aplicabilidad se ha extendido en varias áreas como la inteligencia empresarial, la minería de opiniones, atención al cliente, análisis de datos de salud, entre otros. (Hirschberg & Manning, 2015) [18]. En ellos se han alcanzado logros notables, ya que algunos algoritmos han conseguido niveles considerables de precisión en la traducción de idiomas, otros han mejorado la accesibilidad para personas con discapacidad mediante tecnologías de asistencia y otros han permitido la creación de interfaces de usuario más intuitivas y humanas. Además, tambien ha tenido un impacto significativo en el campo de la salud, donde facilita la interpretación de notas médicas y ayuda en la categorización y análisis de grandes volúmenes de textos clínicos. Esto ha sido gracias a los grandes avances en la inteligencia artificial, en particular, en técnicas de deep learning y redes neuronales, desarrollando la capacidad de máquinas para procesar lenguaje y ofreciendo soluciones más sofisticadas y humanas, tales como modelos conversacionales que sostienen diálogos complejos y contextualizados, análisis de textos para detección temprana de enfermedades como Alzheimer y otras herramientas de aprendizaje que se adaptan al estilo y nivel de comprensión del usuario. [18]

En resumen, el PLN está en rapida y constante evolucion, superando continuamente los límites de lo que creiamos que las máquinas eran capaces de entender, de procesar y de como pueden interactuar con los seres humanos, acercandose cada vez más a una tecnología que emula la forma natural en que los humanos se comunican y procesan la información.

2.1.1. Conceptos clave

En PLN existe una muy amplia gama de conceptos y terminos con los que facilmente podria no estarse muy familiardo, ya que es una extensa area de estudio en pleno auge, sin embargo, abordamos aqui algunos de los mas fundamentales y que seran frecuentemente utilizados:

Corpus: el cual se define como un conjunto estructurado de textos o datos lingüísticos que se recopilan y se utilizan para el entrenamiento y la evaluación de los modelos de PLN (McEnery & Hardie, 2012) [30]. Puede abarcar una amplia variedad de fuentes, como libros, artículos, conversaciones, sitios web o redes sociales y su propósito principal es proporcionar una base de datos representativa del lenguaje en cuestión, permitiendo analizar patrones, entrenar modelos y realizar diversas tareas como extracción de información, traducción automática, análisis de sentimientos, entre otros. El tamaño y la diversidad que pueda tener un corpus también son factores importantes para garantizar que los modelos entrenados cumplan con

ser robustos y generalizables.

Tokenizacion: el cual bajo este contexto se refiere al proceso de dividir el texto en unidades más pequeñas, generalmente palabras o frases (Kumar & Jain, 2020). [24]. Es decir es un proceso en el cual se dividen los textos en tokens, los cuales, tal como se menciona, pueden ser palabras o hasta oraciones completas y dichos tokens son usados en PLN como las unidades basicas de datos.

Steamming y ***Lemmatizacion*****:** el primero es un proceso que busca reducir las palabras a sus formas base eliminando sufijos, para agrupar palabras con formas similares (por ejemplo, las palabras 'corriendo' o 'corre' se reduciran a la raíz común 'corr') y el segundo es un proceso en donde las palabras se reducen a sus formas conocidas como lemas, para simplificar el análisis y la comparación de palabras relacionadas (por ejemplo, las formas verbales 'corriendo' y 'corre' se reducirían al lema 'correr'). (Manning, C. D., & Schütze (2008). [29]. Aunque el stemming puede generar raíces que no son palabras reales, su aplicación puede ayudar a simplificar el análisis y mejorar la recuperación de información mientras que lematizar es útil para normalizar palabras y facilitar la identificación de relaciones y similitudes semánticas entre términos.

2.1.2. Modelos de Lenguaje

En PLN, los modelos de lenguaje son la herramienta fundamental en la tarea de comprender y generar texto, desempeñan un papel esencial al predecir la probabilidad de aparición de una palabra en una secuencia dada, considerando las palabras que la preceden en dicha secuencia (Jurafsky & Martin, 2019)[21]. La función de los modelos de lenguaje es capturar patrones y dependencias semánticas en el texto, lo que permite inferir la estructura y significado inherente a expresiones lingüísticas. Un aspecto importante es la noción de *probabilidad condicional*, este concepto se refiere a la probabilidad de que una palabra ocurra dado el contexto proporcionado por las palabras previas en secuencia. Los modelos de lenguaje emplean técnicas avanzadas, como redes neuronales recurrentes, para aprender estas probabilidades condicionales y, por ende, tratar de capturar la coherencia y cohesión del lenguaje natural. Además, es crucial comprender la noción de 'entrenamiento' de los modelos, la cual se refiere a la fase en que estos se alimentan de grandes conjuntos de datos, ajustando sus parámetros internos para maximizar la verosimilitud de la secuencia observada (Vasawani, 2017)[39]. Este proceso de ajuste permite a los modelos internalizar las estructuras gramaticales y semánticas del idioma, lo que facilita su capacidad para generar texto coherente y relevante.

En síntesis, los modelos de lenguaje en el PLN constituyen herramientas poderosas para la comprensión y producción automatizada de texto, destacando por su capacidad para modelar y anticipar patrones en el uso del lenguaje natural. La exploración detallada de estos aspectos fundamentales proporciona una base sólida para comprender el funcionamiento y la relevancia de los modelos de lenguaje en el ámbito del Procesamiento del Lenguaje Natural.

- **Modelos de Aprendizaje Tradicionales:** Los mas simples y ampliamente utilizados, son los modelos clasicos de aprendizaje estadistico, que se basan en estadísticas de ocurrencia de palabras en un corpus de entrenamiento. Métodos como la cadena de Markov y el modelo de n-gramas son ejemplos de enfoques estadísticos clásicos utilizados para modelar la probabilidad de secuencias de palabras (Manning, 2015) [18].

- **Modelos de Aprendizaje Profundo:** Con los avances en el aprendizaje profundo, los modelos de lenguaje basados en redes neuronales han ganado prominencia. Los modelos de lenguaje neuronales, como las Redes Neuronales Recurrentes (RNN), las Redes Neuronales Convolusionales (RNC) y las Redes Neuronales de Transformers, a pesar de requerir un mayor poder computacional, han demostrado ser más eficaces para aprender patrones complejos y representaciones semánticas en grandes conjuntos de datos textuales (Vaswani, 2017) [39].

- **Embeddings de Palabras:** Una técnica esencial y muy eficaz en los modelos de lenguaje de aprendizaje profundo es la creación de embeddings de palabras. Estos embeddings representan palabras como vectores densos de alguna cierta dimension en un espacio semántico, los cuales actuan capturando relaciones semánticas y sintácticas entre las palabras (Mikolov, 2013)[32].

- **Transferencia de Aprendizaje:** Muchos modelos de lenguaje modernos aprovechan la transferencia de aprendizaje, pre-entrenando modelos en grandes corpus de texto y luego afinándolos para tareas específicas. Esto permite a los modelos aprender representaciones lingüísticas generales que pueden adaptarse a tareas específicas con conjuntos de datos más pequeños (Howard & Ruder, 2018) [19].

- **Modelos de Generación de Texto:** Además de la comprensión del lenguaje, los modelos de lenguaje también se utilizan para la generación de texto. Los modelos generativos, como GPT (Generative Pre-trained Transformer) y BERT (Bidirectional Encoder Representations from Transformers), han demostrado habilidades notables para generar texto coherente y que además es contextualmente relevante (Radford, 2018; Devlin, 2019) [37] [13].

La evolución de los modelos de lenguaje refleja el continuo avance en la capacidad de las máquinas para entender y producir texto, siendo un componente esencial en diversas aplicaciones de PLN, desde la traducción automática hasta el análisis de sentimientos.

2.2. Machine Learning

El Machine Learning, o aprendizaje automático, es un campo de la inteligencia artificial que equipa a las computadoras con la habilidad de aprender y mejorar de forma autonoma a partir de la experiencia y se basa en la construcción de sistemas que pueden aprender de los datos que se le presenten en lugar de seguir solamente instrucciones explícitas predefinidas, es decir, se centra en el desarrollo de algoritmos y modelos estadísticos que permitan a las máquinas mejorar su rendimiento a través de la experiencia obtenida al entrenar datos. (Mitchell, 1997) [33]. La podemos definir tambien como la capacidad computacional de aplicar y mejorar el conocimiento adquirido de manera independiente y progresiva. Los algoritmos de aprendizaje generan modelos matemáticos basados en datos de muestra o 'entrenamiento' que les permiten tomar decisiones sin estar específicamente orientados a través de código tradicional (Kotsiantis, 2007) [23]. Su objetivo es desarrollar técnicas que permitan a los sistemas informáticos utilizar la información existente para descubrir patrones

y tendencias, tomar decisiones, y predecir futuros comportamientos de forma autónoma a traves del analisis de grandes volúmenes de datos con mínima intervención humana. En el contexto de la actuaria, el machine learning también es particularmente relevante debido al gran contexto estadistico que conlleva y la utilidad que representa por su capacidad para manejar y modelar grandes conjuntos de datos y testar múltiples hipótesis de manera eficiente.

Esta disciplina se encuentra en la intersección de la estadística, la inteligencia artificial y la ciencia de datos y juega un papel crucial en la construcción de aplicaciones adaptables a nuevas circunstancias, desde aplicaciones diarias como el filtrado de correos basura y recomendaciones personalizadas en servicios de streaming, hasta usos más complejos en diagnósticos médicos, el aprendizaje automático impulsa la eficiencia y la precisión a través de diversas industrias. Las aplicaciones del Machine Learning son extensas y se extienden por casi todas las áreas imaginables: desde la atención sanitaria, donde ayuda en el diagnóstico y personalización de tratamientos, hasta el sector financiero, mejorando el análisis de crédito y la detección de fraudes. En el marketing digital, personaliza la experiencia del usuario; en la logística, optimiza las rutas de distribución; y en la robótica, permite a los robots aprender de sus interacciones con el entorno.

El machine learning ha logrado hitos impresionantes, incluyendo sistemas capaces de vencer a campeones humanos en juegos complejos como el Go, diagnósticos automatizados de imágenes médicas con precisión a la par o superior a la de los expertos humanos, y vehículos autónomos que pueden conducirse en las calles del mundo real. Estos logros son resultado de la capacidad de los modelos de aprender y mejorar sus conocimientos continuamente (Bishop, 2006) [6].
Por ende, el machine learning se ha convertido en un componente esencial en la toma de decisiones basada en datos debido a su habilidad para procesar y aprender de grandes conjuntos de datos complejos y variantes. Con habilidades para descubrir información valiosa y suministrar predicciones precisas, los algoritmos de aprendizaje automático están cambiando la forma en que interactuamos y comprendemos el mundo, mejorando notablemente la toma de decisiones en las organizaciones y la calidad de vida en general.

El reconocimiento de patrones y el análisis de regresión son otros aspectos fundamentales, estos algoritmos les brindan a los científicos y a los profesionales la capacidad de identificar patrones y regularidades en los datos que no serían discernibles por humanos, proporcionando una mayor comprensión de datos y a menudo ideas valiosas y accionables. (Bishop, 2006) [6].

2.2.1. Conceptos Clave:

Ahora se definen y describen algunos conceptos importantes que son mencionados frecuentemente en el estudio del Machine Learning, Deep Learning y a lo largo de este estudio:

Aprendizaje Supervisado: refiriendose a algoritmos que aprenden de un conjunto de datos etiquetados para hacer predicciones o decisiones sin ser explícitamente programados para realizar la tarea (Kotsiantis, 2007) [23]

Aprendizaje No Supervisado: Este enfoque aborda problemas y el uso de técnicas para encontrar estructuras ocultas donde no se dispone de datos etiquetados. Algoritmos como el clustering se utilizan para identificar patrones y agrupar datos de manera que elementos similares estén en el mismo grupo.(Aggarwal & Reddy, 2014) [2]

Supervisado vs. No Supervisado: En el aprendizaje supervisado, los algoritmos se entrenan utilizando un conjunto de datos etiquetado, donde se conoce la respuesta correcta para cada ejemplo. Por otro lado, el aprendizaje no supervisado implica el uso de datos no etiquetados y busca identificar patrones o estructuras sin conocer de antemano las respuestas correctas.

Aprendizaje Profundo: El aprendizaje profundo, o deep learning, es una rama del machine learning que utiliza redes neuronales profundas para modelar patrones. En PLN, los modelos de aprendizaje profundo, como BERT y GPT, han demostrado un rendimiento sobresaliente en tareas como el análisis de sentimientos y la generación de texto. (LeCun et al., 2015) [25]

Algoritmos de Clasificación: Dentro del aprendizaje supervisado, los algoritmos de clasificación son esenciales en PLN. Estos algoritmos asignan una etiqueta a una entrada, como la categorización de un correo electrónico como spam o no spam. (Kotsiantis, 2007) [23]

Redes Neuronales: En el contexto del PLN, las redes neuronales juegan un papel crucial. Estas estructuras, inspiradas en el funcionamiento del cerebro humano, son capaces de aprender representaciones complejas y realizar tareas complejas, como el procesamiento del lenguaje natural. (LeCun et al., 2015) [25]

2.2.2. Aplicaciones Relevantes en la actualidad:

El Machine Learning se ha integrado profundamente en diversos sectores, transformando procesos y habilitando avances que antes eran inconcebibles. A continuación, se presentan dos de las aplicaciones más influyentes y los trabajos relacionados con el aprendizaje automático, destacando los alcances y contribuciones significativas en estos campos.

- **Diagnósticos Médicos:** Una de las aplicaciones más prometedoras e impactantes del Machine Learning está en el campo de la medicina diagnóstica. Los sistemas de aprendizaje automático han demostrado ser excepcionalmente eficaces en la interpretación de imágenes médicas para la detección y el diagnóstico precoz de enfermedades. (David Esteban, 2015) [14][15]. En este trabajo se desarrollaron algoritmos de Deep Learning que superaron a los radiólogos humanos en la detección de referencias patológicas en imágenes de rayos X. También (LeCun, 2015) [25] demostraron la eficacia de las Redes Neuronales Convolucionales (CNNs por sus siglas en inglés) en la clasificación de imágenes médicas, estableciendo un importante precedente para futuras investigaciones.

- **Sistemas de Recomendación:** El impacto del Machine Learning en los sistemas de recomendación ha revolucionado la forma en que los consumidores encuentran y eligen productos en línea. Un trabajo pionero fue el del Premio Netflix, donde (Bennett y Lanning, 2007) [4] describieron el enfoque que su equipo utilizó para mejorar el algoritmo de recomendaciones de Netflix utilizando técnicas de Filtering. Estos sistemas utilizan algoritmos para filtrar y predecir las preferencias de los usuarios basándose en su historial anterior y patrones de datos agregados. La vasta aplicación de estos sistemas se puede ver en plataformas de comercio electrónico como Amazon y servicios de streaming como Spotify, donde se personaliza la experiencia del usuario para aumentar la satisfacción y retención del cliente (Koren, Bell, & Volinsky, 2009) [22]

Estos ejemplos ilustran cómo el ML no solo está potenciando la innovación tecnológica sino también catalizando cambios significativos en la asistencia sanitaria y en la experiencia del consumidor. Los trabajos relacionados ofrecen una visión de las potencialidades y retos futuros, así como un marco para la evolución continua del aprendizaje automático y su aplicación en ámbitos prácticos.

2.3. Análisis de Sentimientos

El análisis de sentimientos, también conocido como minería de opiniones, se refiere al uso de distintas técnicas de PLN, análisis de texto y machine learning para identificar, extraer y cuantificar información subjetiva de diversas fuentes de texto (Pang & Lee, 2008)[34]. En otras palabras, busca determinar la actitud, las emociones o las opiniones de un autor sobre un determinado tema o contexto en un texto escrito empleando métodos estadísticos y de aprendizaje automático, transforma el texto subjetivo en información estructurada y mensurable para su estudio posterior. En particular se trata de un campo dentro del PLN que apunta específicamente a interpretar y clasificar las opiniones en texto, al estudiar cómo se articulan las emociones y los puntos de vista, el análisis de sentimientos comprende no solo si un sentimiento es positivo, negativo o neutral, sino también su intensidad y la confianza generada en esa clasificación. (Liu, 2012)[27]. En general nos sirve para monitorear y comprender la percepción pública de una marca, producto, servicio o algun tema en general, y permite a las organizaciones capturar una retroalimentación en tiempo real y contestar a las demandas y preocupaciones de los clientes con una rapidez sin precedentes, posibilitando actuar de manera pro activa frente a las tendencias y cambios en la opinión pública. El objetivo es extraer información de opinión de los datos textuales, pero satisfacer esta necesidad implica abordar cómo se pueden identificar y categorizar las opiniones de manera efectiva, lo que implica enfrentar desafíos técnicos y contextuales en la interpretación del lenguaje.

Inicialmente, el análisis de sentimientos se utilizó principalmente en las empresas para el análisis de opiniones de los consumidores en reseñas de productos y servicios en la web, sin embargo, con el auge de las redes sociales, este ha encontrado una amplia gama de aplicaciones en campos como el análisis de medios sociales, la gestión de la reputación de la marca y el seguimiento de las tendencias del mercado, entre otros.(Pang & Lee, 2008)[34]. Más allá del ámbito empresarial, el análisis de sentimientos también se aplica en áreas como la política (para entender la reacción del público ante campañas o debates), el monitoreo de medios de comunicación (para analizar la respuesta emocional a las noticias), la salud mental (para detectar señales de depresión o ansiedad en los textos escritos), y hasta en la seguridad pública (para identificar posibles amenazas o comportamientos peligrosos. (Kumar, 2012)[24]. En el campo de la actuaria, también se utiliza en algunas areas como el análisis de riesgos y la toma de decisiones basada en datos.

Esta tarea plantea desafíos en la interpretación adecuada del lenguaje humano, debido a la ambigüedad, la ironía, el sarcasmo y la variabilidad cultural y contextual de los textos. Aún así, ha mostrado ser valioso a la hora de captar y entender las opiniones y emociones humanas representadas en los datos de texto y ha alcanzado varias metas notables, como el desarrollo de sistemas que pueden detectar el tono emocional de los clientes con un nivel de precisión que antes se consideraba imposible. (Pang & Lee, 2008)[34]. También ha ayudado en la generación de modelos predictivos de comportamiento del consumidor, en la comprensión de

las dinámicas sociales en plataformas en línea y en la mejora de la interacción entre humanos y máquinas. Además de eso, ha contribuido significativamente en el tratamiento y monitoreo de la salud mental, al analizar patrones en el discurso que podrían indicar cambios en el estado emocional de una persona.(Liu, 2012)[27].

Los avances metodológicos y tecnológicos en el análisis de sentimientos continúan expandiendo sus límites empleando técnicas cada vez más sofisticadas, como deep learning y modelos de lenguaje pre-entrenados, los cuales pueden abordar con mayor precisión las complejidades del lenguaje, como la ironía y el sarcasmo. Sin embargo, persisten desafíos relacionados con la contextualización cultural y el entendimiento multidimensional de las emociones, los cuales son objeto de investigación continua en el campo.

2.3.1. Conceptos Clave:

Dentro del análisis de sentimientos, algunos de los aspectos mas destacados a definir antes de abordarlos en el trabajo son algunos tales como Polaridad, que se refiere a identificar si la tonalidad del texto es positiva, negativa o neutra; Subjetividad, que se refiere a qué tan subjetiva u objetiva es la información contenida en el texto

Polaridad: La polaridad se refiere a la dirección de la emoción expresada en un texto, clasificándose comúnmente en positiva, negativa o neutra. La identificación de la polaridad es fundamental para comprender la actitud general de un texto hacia un tema específico. (Liu, 2012)[27].

Subjetividad: La subjetividad se relaciona con el grado en que un texto expresa opiniones, creencias o sentimientos en lugar de información objetiva y factual. Los enfoques de análisis de sentimientos suelen considerar la subjetividad para comprender la subjetividad inherente en las expresiones humanas. (Liu, 2012)[27].

Técnicas de Preprocesamiento: Dada la complejidad del lenguaje y la variabilidad en las expresiones, las técnicas de preprocesamiento son cruciales en el análisis de sentimientos. Esto incluye la tokenización, eliminación de stop words, y la lematización para simplificar y normalizar el texto antes de la análisis (Kumar, 2012)[24]..

Enfoques Supervisados y No Supervisados: El análisis de sentimientos puede abordarse tanto de manera supervisada como no supervisada. Los enfoques supervisados utilizan conjuntos de datos etiquetados para entrenar modelos de clasificación, mientras que los enfoques no supervisados se basan en la identificación de patrones y agrupamientos en datos no etiquetados. (Kotsiantis, 2007) [23].

2.3.2. Modelos y Enfoques:

El análisis de sentimientos a través del el aprendizaje automático abarca una diversidad de modelos y enfoques que han surgido para abordar esta tarea cada uno con sus propias fortalezas y debilidades, las cuales son el principal objeto de estudio en este trabajo comparativo. En esta sección, se describen y exploran tres categorías fundamentales: los modelos basados en reglas (lexicons), los basados en aprendizaje supervisado de machine learning y los modelos basados en transformers.

- **Modelos Basados en Reglas (Lexicons):**

Los modelos basados en reglas utilizan lexicons o diccionarios predefinidos que asignan a cada palabra una polaridad, generalmente positiva, negativa o neutra. Estos lexicons contienen términos previamente etiquetados con sus respectivas polaridades y se utilizan para evaluar el sentimiento de un texto sumando o promediando las polaridades de las palabras que lo componen. Este enfoque es eficiente y fácil de implementar, pero puede ser limitado por la subjetividad inherente en la asignación de polaridades a las palabras. Además, no captura la complejidad semántica y contextual de las expresiones. Los modelos basados en reglas, también conocidos como lexicons, se centran en la asignación automática de etiquetas de sentimiento a palabras o frases basándose en diccionarios preestablecidos de palabras con polaridades asociadas. Estos enfoques confían en reglas semánticas y lingüísticas predefinidas para determinar la orientación sentimental de una palabra en un contexto dado. (Liu, 2012)[27].

- **Modelos Basados en Aprendizaje Supervisado (Machine Learning):**

Los modelos basados en aprendizaje supervisado utilizan algoritmos de clasificación entrenados en conjuntos de datos etiquetados para predecir la polaridad de nuevos textos. Estos algoritmos aprenden patrones y relaciones entre las características del texto y las etiquetas de sentimiento durante la fase de entrenamiento. Este enfoque es más flexible y capaz de manejar la complejidad semántica, pero depende en gran medida de la calidad y cantidad de datos de entrenamiento. Además, su desempeño puede verse afectado por la variabilidad en el estilo de expresión y la subjetividad inherente al sentimiento humano. Los enfoques basados en aprendizaje supervisado de machine learning se centran en entrenar modelos para clasificar automáticamente el sentimiento asociado con un texto o frase determinada. Estos modelos se entrenan utilizando conjuntos de datos etiquetados previamente con polaridades de sentimientos, lo que permite a los algoritmos aprender patrones y características que se correlacionan con expresiones de sentimientos específicos. (Pang & Lee, 2008)[34].

- **Modelos Basados en Transformers:**

Los modelos basados en transformers, como BERT y GPT, han revolucionado el análisis de sentimientos. Estos modelos utilizan arquitecturas de atención y aprendizaje profundo para capturar las relaciones contextuales y complejas en el texto, logrando un entendimiento más profundo de las expresiones sentimentales. La fortaleza de los modelos basados en transformers radica en su capacidad para contextualizar el lenguaje, capturando matices y relaciones entre palabras en un contexto específico. Sin embargo, su desafío principal radica en los recursos computacionales necesarios para entrenar y utilizar estos modelos a gran escala. Los modelos basados en transformers, como el popular modelo BERT (Bidirectional Encoder Representations Transformers), representan la vanguardia en el análisis de sentimientos. Estos modelos utilizan arquitecturas de redes neuronales de auto-atención para capturar correlaciones más complejas y contextuales en el lenguaje, lo que permite capturar relaciones sintácticas y semánticas más profundas. (Devlin, 2019) [13]. (Radford, 2018)[37]

Estos tres enfoques proporcionan una visión completa de las estrategias utilizadas para interpretar las emociones en el lenguaje humano, desde métodos más simples y basados en reglas hasta modelos de vanguardia basados en transformers. Estos enfoques ofrecen un espectro diverso de métodos para abordar el desafío del análisis de sentimientos, cada uno con sus propias ventajas y consideraciones. Su comprensión y aplicación efectiva en contextos específicos requiere una evaluación cuidadosa de sus capacidades y limitaciones en relación con los requisitos del problema de análisis de sentimientos en cuestión.

Capítulo 3

Estado del arte

En 2020, el programa MIOPERS[1] a cargo de la doctora Helena Gómez Adorno [2] del *Departamento de Ingeniería en Sistemas Computacionales* del *Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas* respondió a la iniciativa de la UNAM de desarrollar modelos para el análisis y visualización de información que apoyen estrategias y toma de decisiones especialmente durante la etapa de confinamiento y estableció el objetivo de crear un sistema automático de vigilancia en la opinión publica en mensajes de texto extraídos de redes sociales acerca del tema de la pandemia por COVID-19 (principalmente recopilando mensajes de la red social *Twitter*, mejor conocidos como *tweets*) con dos principales motivaciones para iniciar dicho trabajo: a) evaluar el comportamiento, el estado de ánimo y la popularidad de las personas de las medidas dadas por el gobierno y b) monitorear a los usuarios con posibles síntomas. Esta iniciativa, que abarca tres años (2020-2022), permitió una recopilación de muchos tweets relacionados. Esto facilitó el estudio de léxico, menciones y hashtags relacionados con el tema, que a su vez sirvieron como base para estudiar otros temas importantes de PNL, como el análisis de sentimientos. La plataforma de MIOPERS mostró dicha información para su consulta utilizando técnicas de Procesamiento de Lenguaje Natural y Machine Learning con el fin de monitorear síntomas y emociones de forma diaria y mostrar la información de forma automática en un sitio web con diseño interactivo. Para este objetivo, era necesario contar con conjuntos de datos (*corpus*) especificos para cada tipo de análisis, que permitieran el estudio y entrenamiento de modelos de clasificación. Para la construcción del corpus de análisis de sentimientos y emociones, se determino que era necesario un conjunto de tweets etiquetados manualmente en 3 categorías (positivo, negativo, neutro) y en 6 categorías (ira, miedo, placer, tristeza, sorpresa, asco) para el análisis de emociones.

Además de la creación de un corpus propio, se realizo también una búsqueda de otros corpus previamente creados por alguna otra fuente, que se adecuaran a nuestras necesidades y que estuvieran disponibles de forma abierta. De esta forma, fue posible hacer uso del corpus "gold-standard.[el]aborado para el análisis de sentimientos y proporcionado por el TASS [3], el cual etiquetaba en tres categorías y no solo en dos, como en muchos casos puede encontrarse este tipo de recurso, y contaba con 9000 tweets. También se dispuso del corpus '*Sentmex-Neg´* creado en 2020 por el *Instituto de Ingeniería Lingüística*, el cual de igual forma esta etiquetado en 3 categorías y es útil para el análisis de sentimientos, aunque este no fue creado para dicha tarea, ya que su objetivo era el estudio de la semántica de negaciones lingüísticas.

[1] miopers.unam.mx
[2] https://g.co/kgs/UCHW5ex
[3] Taller de Análisis Semántico en la SEPLN

En 2021 trabajando en conjunto con el *Instituto de Ingeniería* se consiguió la creación de los recursos necesarios para las tareas de interés en el proyecto, uno de ellos fue el denominado '*Corpus SentCovid*' el cual consiste de 4,986 tweets que fueron etiquetados manualmente cada uno por un conjunto de 5 etiquetadores, en donde la metodología se describe a continuación:

3.1. Construcción del Corpus

3.1.1. Recolección de tweets:

Los tweets se descargaron utilizando la API de Twitter a través de un código que recopila los mensajes y los almacena en un servidor. Usando un cron [4] se realizó el análisis automático diario (Sentimientos, Síntomas, Conteos/Estadísticas) de los tweets encontrados en la base de datos. Los resultados de los distintos análisis se almacenan en un formato JSON y se almacena en otra base de datos. La página web desarrollada consulta la base de datos de resultados de análisis y los muestra en el varios formatos mencionados antes. A partir del 1 de abril de 2020, el grupo comenzó a recopilar tweets acumulando en total un poco más de 4,000,000 y almacenando estos en un servidor instalado para el proyecto.

Para el corpus se determinó no realizar ningún tipo de preprocesamiento en los tweets, es decir, no hubo cambiamos en los datos tales como corrección de faltas de ortografía o de truncamiento en el tamaño. Además, se consideraron solamente tweets con alguna información textual, no se tomaron en cuenta los tweets solo con contenidos audiovisuales o emojis. Sin embargo, no se omitieron los tweets que tenían negaciones sintácticas en conjunto con emojis, contenido audiovisual, menciones o hashtags. A veces, tal El contenido nos brindó información adicional o sustancial para comprender y estudiar la negación en el tweet, y se determino que dejarlos fuera seria una perdida de información. Finalmente, también incluimos tweets que fueron respuestas o retweets, es decir, el tipo y la forma del tweet no importaron para el proceso de extracción y anotación.

3.1.2. Protocolo de etiquetado:

A partir de la descripción de las ocho emociones primarias -ira, miedo, tristeza, asco, sorpresa, antipatía- ipación, confianza y alegría, descritas por el psicólogo Robert Plutchik [36], se creó una anotación guía que especifica cómo se deben etiquetar los tweets y bajo qué criterios. Se categorizarón los sentimientos en tres clases: positivos, negativos y neutrales. Se realizó un proceso de etiquetado manual para etiquetar cada uno de los tweets del corpus. El proceso involucró a 5 anotadores, dos de ellos etiquetaron los tweets sin ninguna orientación, es decir, asignaron una etiqueta según su perspectiva, mientras que el resto utilizaron la anotación guía.

Etiquetas Positivas:

Los tweets marcados con etiqueta positiva son utilizados para identificar a los que comunican y expresan sentimientos y emociones como alegria/confianza. Los tweets positivos estan caracterizados por satisfacer criterios tales como son:

[4] Cron es un administrador de tareas de Linux que permite ejecutar comandos en un momento determinado.

1. Mensajes donde predomina un mensaje de bienestar o placer:

    - *'No se ustedes pero yo he sido muy feliz durante esta cuarentena'.*
    - *'jaja Ana me acaba de alegrar la cuarentena'.*

2. Mensajes que permiten cultivar fortalezas y virtudes personales y conducen a la felicidad:

    - *'Desde que inició la cuarentena le ando dando duro al ejercicio y a la dieta'.*
    - *'Algunos están estresados, preocupado yo digo #GraciasCuarentena porq me ha hecho valorar tanto, porque la salud es primero y solo se valora cuando se pierde'.*

Etiquetas Negativas:
Los tweets marcados con etiqueta negativa son utilizados para identificar a los que comunican y expresan sentimientos y emociones como ira, miedo o tristeza. Los tweets negativos estan caracterizados por satisfacer criterios tales como son:

1. Mensajes que expresan un sentimiento implacentero o violento

    - *'Oigan me duele el pecho medio raro ¿Es síntoma de covid o ya me volví loca?'.*

2. Mensajes que expresan una barrera para conseguir un objetivo o que requieren crear un plan para resolver alguna situacion

    - *'El número de contagios de Covid-19 en México puede ser hasta 50 veces más que los reportados: Julio Frenk.'*
    - *'La contingencia continua y las necesidades son cada vez en más familias.'*

Etiquetas Neutras:
Los tweets marcados con etiquetas neutrales se utilizan para identificar tweets que no comunican ninguna emoción/sentimiento. Los tweets neutrales se caracterizan por:

1. No comunican un mensaje en especifico:

    - *'@ExpansionMx El virus no tiene nada que ver. Lo que sí, es el Gobierno!.'*

2. Mensajes que expresan alguna clase de duda, sin atacar a algo o a alguien:

    - *'No se supone que suspendieron parquímetros por contingencia? @ecoParq @Claudiashein.'*

3.1.3. Resultados del proceso de etiquetado:

Inicialmente se considero crear un corpus de 5000 tweets bajo los criterios de ser extraidos solo de la república mexicana y que en su contenido estuvieran relacionados al tema de la pandemia utilizando los filtros de palabras clave, sin embargo se decidió descartar algunos tweets que estaban en otros idiomas, por lo que se eliminaron 134 tweets de nuestro corpus inicial. Posteriormente en el proceso de etiquetado con los 4.866 tweets restantes, se decidió eliminar también aquellos tweets que contenían menos de tres palabras ya que se considero que estos no contenían información suficiente para permitir a los evaluadores asignar una etiqueta, por lo que nuestro corpus final termino con la cantidad de 4.800 tweets

Para evaluar qué tan bien se definió la tarea de etiquetado, utilizamos la puntuación del Acuerdo entre anotadores; para definir esto, utilizamos la medida estadística Kappa de Cohen. El coeficiente kappa de Cohen (κ) se utiliza con frecuencia para probar la confiabilidad entre evaluadores. La medición del grado en que los recolectores de datos (evaluadores) asignan la misma puntuación a la misma variable se denomina confiabilidad entre evaluadores. [31].

Para el proceso, se contó con 5 anotadores, de los cuales a dos de ellos se les pidió etiquetar los tweets sin utilizar la guía, solamente de acuerdo a su criterio y al resto de ellos se les pidió seguir el criterio de la guía, de esta forma los anotadores que no utilizaron la guía presentaron un acuerdo de Cohen de 0,178, un acuerdo bastante bajo. Por otro lado, los anotadores que utilizaron la guía presentaron un acuerdo de Cohen de 0,4369, un acuerdo ciertamente moderado.

La tabla 3.1 que se presenta a continuación muestra el porcentaje de acuerdo y la puntuación (κ) de Cohen para cada par de anotadores que no utilizaron la guía. A diferencia de esto, la tabla 3.2 que se presenta a continuación muestra el porcentaje de acuerdo y la puntuación (κ) de Cohen para cada par de anotadores que utilizaron la guía.

Tabla 3.1: Puntuación de acuerdo de los anotadores de la clasificación de sentimientos sin guía.

Par de anotadores	1&2
Acuerdo	41 %
Puntuación κ	0.1785

Tabla 3.2: Puntuación de acuerdo de los anotadores de la clasificación de sentimientos con guía.

Par de anotadores	1&2	1&3	3&4
Acuerdo	61 %	70 %	62 %
Puntuación κ	0.3945	0.5547	0.3716

Debido a la naturaleza del problema y al proceso de etiquetado, se decidió utilizar el 'kappa de Fleiss' para la concordancia de los tres anotadores que utilizaron una guía. Este análisis dio como resultado un acuerdo general de 0,4369 que refleja un acuerdo moderado de anotadores.

La alineación permitió identificar aquellas etiquetas con mayor confiabilidad, es decir, al tener tres etiquetas independientes, en los casos en los que había desacuerdo pudimos buscar acuerdo en dos de tres para fijar la etiqueta repetida como la definitiva en el corpus final. Al finalizar el proceso descrito anteriormente algunos tweets no tenían etiqueta asignada ya que los tres anotadores tenían un desacuerdo total, por esta razón fue necesario que los anotadores entre todos decidieran la etiqueta final para los tweets sin acuerdo.

Por lo tanto nuestro corpus final, el cual como se menciono antes cuenta con 4,800 tweets, fue etiquetado con 1834 (38.21 %) que contienen sentimientos negativos, 1,126 (23.46 %) que contienen sentimientos positivos y 1840 (38.33 %) con sentimientos neutrales. Finalmente, la tabla 3.3 que se presenta a continuación muestra las estadísticas generales calculadas a partir del recuento de palabras en cada tweet de nuestro corpus.

Tabla 3.3: Estadísticas calculadas del recuento de palabras

	Positivo	Negativo	Neutral
Numero promedio de palabras	22.85	26.39	20.97
Desviación Estándar	12.69	15.46	13.59
Varianza	161.14	239.02	184.65
Máximo numero de palabras	59	339	88
Total de palabras	25,729	48,398	38,580
Conteo de tweets	1,126	1,834	1,840

De esta manera se construyo el corpus para análisis de sentimientos necesario para el entrenamiento y pruebas de los modelos que se describen en las siguientes secciones. Vale la pena mencionar que el tamaño del corpus es aceptable para la realización de experimentos con distintas técnicas en el proyecto MIOPERS, sin embargo, dista de tener el tamaño suficiente para entrenar modelos mas ambiciosos como los que buscan algunas empresas de tecnología, donde se tiene mayor cantidad de recursos para la construcción de corpus mayor tamaño.

3.2. Herramientas y técnicas de PLN

En esta sección se describen a detalle algunas herramientas disponibles, marcos de trabajo y técnicas de preprocesamiento populares dentro del Procesamiento de Lenguaje Natural en la actualidad que fueron las elegidas para llevar a cabo los experimentos y pruebas comparativas que son el objetivo de este estudio. Por ejemplo, se describen las técnicas propuestas para realizar el proceso de tokenizacion (división) de los tweets, que como se menciono antes, se puede llevar a cabo palabra por palabra o un conjunto de estas, con el objetivo de modelar frases, oraciones y otras estructuras gramaticales como negaciones o afirmaciones, las cuales resulten mas significativas para los modelos al momento de realizar la decisión en una predicción.

También presentaremos el uso de algunas bibliotecas con diversos enfoques en cuanto a técnicas de PLN que realizan de forma automática y sencilla la predicción del sentimiento, con el mínimo preprocesamiento y sin necesidad de tener ningún conocimiento previo en PLN o Machine Learning, y aunque estos no consiguen grandes resultados en sus predicciones, nos es de gran utilidad entender su funcionamiento y tomarlos como punto de comparación.

3.2.1. Bibliotecas

NLTK (Natural Language Toolkit): Un Vistazo Profundo

El Natural Language Toolkit (*NLTK)* es una biblioteca de Python diseñada para facilitar la investigación y desarrollo en el procesamiento del lenguaje natural. Desde su creación, se ha convertido en una herramienta fundamental para estudiantes, investigadores y profesionales en el campo del PLN. A continuación, exploramos su origen, utilidad y aplicaciones destacadas.

NLTK fue desarrollado por Steven Bird y Edward Loper en la Universidad de Pennsylvania a principios de la década de 2000[5]. Fue concebido como una plataforma abierta y extensible que brindara acceso a recursos lingüísticos y herramientas de PLN para la comunidad académica y la industria. Su enfoque en la educación y la investigación lo ha convertido en un recurso invaluable para aquellos que buscan comprender y aplicar técnicas de PLN. Ofrece

una amplia gama de módulos y recursos para abordar diversas tareas, desde tokenización y análisis gramatical hasta traducción automática. Otras de sus características incluyen:

- Análisis Gramatical y Parsing: Incluye módulos para realizar análisis gramatical, construir árboles sintácticos y realizar parsing (segmentación) y modelado de frases.
- Recursos Lingüísticos: Ofrece acceso a una variedad de recursos lingüísticos, como corpus y lexicones, que son esenciales para el entrenamiento y la evaluación de modelos de PLN.
- Clasificación de Texto: Facilita la implementación de modelos de clasificación de texto y análisis de sentimientos mediante algoritmos de aprendizaje supervisado.
- NLTK Corpora: Incluye una amplia colección de corpora, que son conjuntos de datos lingüísticos, utilizados para entrenar y evaluar modelos. Ejemplos incluyen corpus de texto, etiquetado POS (part-of-speech), y más.
- Herramientas de Traducción Automática: Ofrece módulos para trabajar con sistemas de traducción automática, lo que lo hace valioso en aplicaciones multilingues.

En conclusión *NLTK* ha desempeñado un papel crucial en la popularización y avance del PLN al proporcionar una base sólida para la investigación y el desarrollo. Su utilidad va más allá de las aulas académicas y se extiende a la industria, donde sigue siendo una elección común para aquellos que buscan una herramienta robusta y versátil para el PLN.

Gensim: Explorando la Potencia del PLN

Gensim es una biblioteca de código abierto para PLN desarrollada por Radim Řehůřek y se lanzó por primera vez en 2009 [40] y ha evolucionado para convertirse en una herramienta fundamental en el ámbito de la minería de texto. El objetivo principal de *Gensim* es proporcionar implementaciones eficientes de algoritmos, como *Word2Vec* y *Doc2Vec*, así como herramientas para la comparación de documentos y el modelado de tópicos. *Gensim* ha ganado popularidad debido a su simplicidad de uso y su eficiencia en la implementación de algoritmos clave. Algunas de sus características y utilidades más destacadas incluyen:

- Word Embeddings: *Gensim* ofrece implementaciones eficientes de modelos como *Word2Vec*, que permite la creación de representaciones vectoriales de palabras. Estas representaciones capturan relaciones semánticas entre palabras y son valiosas en tareas de análisis de texto. Además de *Word2Vec*, *Gensim* proporciona una implementación de *Doc2Vec*, que extiende el concepto a la representación vectorial de documentos completos, incorporando información contextual y semántica más amplia.
- Similaridad de Documentos: Permite calcular la similitud entre documentos utilizando modelos entrenados. Esto es útil en la recuperación de información, clasificación de documentos y otras tareas relacionadas.
- Modelado de Tópicos: *Gensim* incluye herramientas para realizar el modelado de tópicos en conjuntos de documentos. Esto es esencial para descubrir patrones temáticos y estructuras subyacentes en grandes colecciones de texto.

En conclusion, *Gensim* ha demostrado ser una herramienta versátil y robusta en el campo del PLN, facilitando el acceso a algoritmos eficientes y útiles para diversas tareas. Ya sea para modelado de tópicos, análisis de sentimientos o comparación de documentos, *Gensim* ha dejado una huella significativa en la comunidad de procesamiento del lenguaje natural.

spaCy: Transformando el Procesamiento del Lenguaje Natural

Esta biblioteca de código abierto fue desarrollada por *Explosion AI*, una empresa con sede en Berlín y fue lanzada en 2015, desde entonces se ha convertido en una herramienta esencial en la caja de herramientas de muchos profesionales del PLN [3]. spaCy está diseñada para ser rápida, eficiente y fácil de usar, y se ha ganado una reputación por su rendimiento en el procesamiento de grandes volúmenes de texto. *spaCy* es conocida por su eficiencia y su enfoque en la simplicidad. Algunas de sus características incluyen:

- Reconocimiento de Entidades Nombradas (NER): Ofrece capacidades robustas de reconocimiento de entidades nombradas, identificando automáticamente nombres de personas, organizaciones, ubicaciones, fechas, entre otros.
- Procesamiento de Documentos Grandes: *spaCy* es conocida por su velocidad y eficiencia en el procesamiento de grandes volúmenes de texto, haciéndola ideal para aplicaciones que implican análisis de corpus extensos.
- Desarrollo de Chatbots: La capacidad de *spaCy* para analizar el significado y la estructura de las oraciones es fundamental en el desarrollo de chatbots para comprender y responder de manera más precisa a las consultas de los usuarios.

En conclusión, *spaCy* se ha convertido en una elección popular en la comunidad de PLN debido a su velocidad, precisión y facilidad de uso. Con sus características avanzadas y su enfoque en el rendimiento, *spaCy* ha demostrado ser una herramienta valiosa para una amplia gama de aplicaciones en el procesamiento del lenguaje natural.

Tabla 3.4: Comparación de NLTK, Gensim y spaCy

	NLTK	**Gensim**	**spaCy**
Funcionalidad	Amplia funcionalidad para tareas básicas de PLN (tokenización, análisis gramatical, etc.)	Especializado en modelos de vectores de palabras (Word2Vec, Doc2Vec) y modelado de tópicos	Eficiente análisis gramatical, reconocimiento de entidades nombradas (NER), y embebido de vectores de palabras
	Soporte para una variedad de tareas de procesamiento del lenguaje natural	Modelos eficientes para la representación semántica de palabras	Modelos de aprendizaje profundo para tareas específicas como NER
Variedad de Herramientas	Amplia variedad de herramientas y recursos lingüísticos	Enfoque específico en modelos de vectores de palabras y modelado de tópicos	Herramientas eficientes para análisis gramatical y reconocimiento de entidades
Desempeño	Puede ser más lento en comparación con *Gensim* y *spaCy* en algunas tareas	Rápido y eficiente, especialmente en modelos Word2Vec	Conocido por su velocidad en el procesamiento de grandes volúmenes de texto
Facilidad de uso	Interfaz fácil de usar, especialmente para principiantes en PLN	Intuitivo para implementar modelos de vectores de palabras y modelado de tópicos	Diseñado para ser fácil de usar y rápido de implementar

3.2.2. Técnicas propuestas:

Una vez seleccionadas las herramientas y analizadas sus características, se tiene un vistazo de los recursos disponibles para posteriormente proponer y elegir las técnicas de preprocesamiento mas adecuadas para cada tipo de modelo propuesto, permitiendo el mejor desempeño posible de cada uno de estos. Las técnicas propuestas son relativamente simples y distan de ser de las mas avanzadas en PLN, como la identificación automática de verbos y sustantivos (POS Tagging) o el reconocimiento de entidades, ya que estas requieren un estudio e implementación mas profundo, y el propósito de este estudio se concentra mas en los distintos modelos de clasificación, sin embargo a continuación se describen algunas técnicas de amplio uso en la actualidad y que son las que utilizamos para nuestro corpus:

Tokenización Simple (Palabra a Palabra): La tokenización simple es una etapa crucial en el procesamiento del lenguaje natural, se encarga de dividir el texto en cuestión en unidades más pequeñas llamadas tokens, los tokens pueden ser palabras, subpalabras o incluso caracteres, dependiendo de la técnica utilizada, la división del texto implica tomar en cuenta unidades basadas en espacios en blanco o signos de puntuación y cada palabra se trata como un token individual. Aunque es una técnica básica y simple, sigue siendo ampliamente utilizada debido a su simplicidad y eficacia en muchos escenarios.

Tokenización por N-gramas: La tokenización por medio de n-gramas es una técnica que se basa en dividir el texto en secuencias contiguas de n elementos, que pueden ser palabras, caracteres o subpalabras. Los n-gramas capturan las relaciones de proximidad entre las unidades del texto y son particularmente útiles para preservar información contextual. Aquí se describen dos variantes comunes de la tokenización por medio de n-gramas:

- Unigramas: Cada palabra se trata como un token individual. Esta es la forma más simple de tokenización por n-gramas y es equivalente a la tokenización palabra a palabra.
- Bigramas: En esta variante, el texto se divide en secuencias de dos palabras consecutivas. Esto permite capturar ciertos patrones de co-ocurrencia y mejora la representación contextual.
- Trigramas: Similar a bigramas, pero en este caso, las secuencias consisten en tres palabras consecutivas. Esta variante aumenta el contexto considerado en la representación.

(*Ventanas Deslizantes: Otra aproximación a la tokenización por n-gramas implica el uso de ventanas deslizantes. En lugar de dividir el texto en n elementos consecutivos, una ventana de tamaño n se desliza a lo largo del texto. Esto permite una sobreposición de información entre tokens consecutivos y captura una gama más amplia de relaciones contextuales.*)

La elección del tamaño de n en la tokenización por n-gramas dependerá de la naturaleza específica del análisis de sentimientos y de la longitud típica de las expresiones en el conjunto de datos. La tokenización por n-gramas es especialmente valiosa en tareas donde la información contextual y las relaciones entre palabras desempeñan un papel crítico, como en el análisis de sentimientos en textos más extensos.

Embeddings de Palabras: Los embeddings de palabras son representaciones vectoriales que capturan las relaciones semánticas entre ellas. Los embeddings han revolucionado la

manera en que las palabras son representadas en el ámbito del PLN. En lugar de representar palabras como entidades aisladas (tokens individuales), los embeddings asignan vectores numéricos a palabras que capturan similitudes semánticas y relaciones contextuales. Este enfoque ha demostrado ser especialmente efectivo en tareas de análisis de sentimientos, donde comprender la semántica subyacente de las palabras es esencial para una clasificación precisa.

Existen técnicas populares en la generación de embeddings, los cuales asignan a cada palabra un vector en un espacio multidimensional, de modo que palabras similares están más cercanas entre sí en ese espacio, y con esto se consigue una propuesta que no solo considera la estructura gramatical, sino también el significado contextual de las palabras. Uno de los métodos más influyentes en la creación de embeddings de palabras es *Word2Vec*, introducido por (Mikolov, 2013) [32], que utiliza modelos de aprendizaje no supervisado para aprender representaciones vectoriales de palabras a partir de grandes cantidades de texto. La idea es utilizar *KeyedVectors* (vector con clave) la cual es una estructura para almacenar y manipular vectores de palabras que pueden ser identificados por una clave única, como el nombre de la palabra. Por ejemplo, cuando se utiliza Word2Vec para entrenar un modelo de embeddings de palabras, cada palabra en el vocabulario se asigna a un vector denso en un espacio de características continuas. Estos vectores se pueden almacenar en una estructura de KeyedVector, donde cada palabra es una clave que se mapea a su vector correspondiente. Estos vectores capturan relaciones semánticas y contextuales, lo que resulta en una representación densa y distribuida de palabras.

Modelado de Frases: El modelado de frases implica la representación de secuencias completas de palabras como vectores, este es un enfoque particularmente interesante ya que permite la abstracción de frases u oraciones completas en una representación numérica, técnicas como Doc2Vec son ejemplo de herramientas que capturan la información contextual y el significado de una oración completa. El modelado de frases representa una evolución significativa en el campo del análisis de sentimientos, permitiendo la captura de relaciones más complejas y contextuales en el lenguaje que a diferencia de los enfoques tradicionales que consideran palabras individuales, el modelado de frases se centra en entender la estructura y la interconexión de las palabras en una secuencia. Métodos como la atención contextual, las redes neuronales recurrentes y las redes neuronales de transformadores (BERT) han demostrado ser eficaces en la representación semántica de frases, permitiendo un análisis de sentimientos más preciso y contextualizado.

Uno de los hitos en el modelado de frases es la introducción de BERT (Bidirectional Encoder Representations from Transformers). BERT, desarrollado por Google, utiliza una arquitectura de transformer que tiene en cuenta el contexto bidireccional en el que una palabra se encuentra en una oración. Esta capacidad para comprender el contexto permite a BERT capturar matices semánticos y relaciones más ricas, elevando el rendimiento en tareas de análisis de sentimientos. Su éxito ha influido en numerosos desarrollos subsiguientes en el campo del procesamiento del lenguaje natural (Devlin, 2019) [13].

3.2.3. Modelos basados en Reglas (Lexicones):

Los modelos basados en reglas representan un enfoque clásico y transparente para abordar el análisis de sentimientos. A diferencia de los modelos de aprendizaje automático, que aprenden patrones automáticamente a partir de datos etiquetados, los modelos basados en reglas

dependen de reglas explícitas diseñadas manualmente. Estas reglas actúan como directrices que indican cómo clasificar el sentimiento de un texto según ciertas características lingüísticas.

Sus principales enfoques son por ejemplo el de Lexicones de Sentimientos, los modelos basados en reglas a menudo utilizan lexicones, que son listas de palabras asociadas con sentimientos positivos, negativos o neutrales, donde la presencia y frecuencia de estas palabras en un texto pueden influir en la clasificación final. Otro enfoque es el de Expresiones Regulares, que se emplean para identificar patrones específicos en el texto que pueden estar relacionados con emociones, por ejemplo, la presencia de emojis, signos de exclamación, o patrones específicos de negación pueden influir en la clasificación. Algunos modelos de reglas consideran también la estructura sintáctica y semántica de las oraciones para inferir el tono y orientación emocional y pueden tener reglas específicas para manejar negaciones, intensificadores, entre otros.

Análisis de Sentimientos con VADER de NLTK:

VADER (Valence Aware Dictionary and sEntiment Reasoner) es una herramienta de análisis de sentimientos diseñada específicamente para el procesamiento del lenguaje natural. Desarrollado como parte del kit de herramientas NLTK, *VADER* se destaca por su capacidad para reconocer la polaridad emocional en el texto, evaluando no solo palabras individuales sino también expresiones y contextos complejos. *VADER* fue creado por C.J. Hutto y Eric Gilbert en 2014 [20], con el objetivo de superar algunos desafíos comunes en el análisis de sentimientos, como la identificación de sarcasmo y la comprensión de las emociones en contextos informales, como las redes sociales. Su enfoque se basa en un conjunto de reglas y un diccionario previamente etiquetado con puntuaciones de valencia y polaridad para palabras y expresiones. *VADER* se ha convertido en una herramienta popular para el análisis de sentimientos por su facilidad de uso y eficacia en contextos variados. Sus características incluyen:

- Análisis de Polaridad: *VADER* evalúa la polaridad asignando puntuaciones a palabras y expresiones, permitiendo la determinación de si una declaración es positiva, negativa o neutra.
- Consideración del Contexto: A diferencia de muchos enfoques que tratan palabras de manera aislada, *VADER*tiene en cuenta el impacto del contexto en la evaluación de la polaridad, lo que lo hace más robusto en situaciones de lenguaje coloquial y ambiguo.
- Manejo de Intensidad: *VADER*no solo clasifica la polaridad, sino que también asigna puntuaciones de intensidad, lo que permite distinguir entre declaraciones fuertemente positivas o negativas y aquellas más moderadas.

*VADER*se ha integrado ampliamente en diversas aplicaciones, desde la evaluación de comentarios en redes sociales hasta el análisis de reseñas de productos. Su capacidad para lidiar con el lenguaje informal y expresar emociones le ha otorgado una posición destacada en el análisis de sentimientos en entornos donde la comunicación es variada y dinámica.

Análisis de Sentimientos con TextBlob

TextBlob es una biblioteca de PLN para Python que ofrece una interfaz sencilla para realizar diversas tareas lingüísticas, incluido el análisis de sentimientos. Diseñada sobre *NLTK* y otras bibliotecas, *TextBlob* es un modelo de procesamiento de lenguaje natural basado en reglas y en un enfoque de lexicón, utiliza reglas gramaticales y un lexicón predefinido para realizar

tareas como tokenización, análisis sintáctico, clasificación de texto y extracción de entidades. Simplifica la implementación de tareas complejas, proporcionando funciones predefinidas para análisis sintáctico, extracción de entidades y, específicamente, análisis de sentimientos.

TextBlob fue desarrollado por Steven Loria [28] como un proyecto de código abierto. Su objetivo principal es hacer que las tareas de procesamiento del lenguaje natural sean más accesibles para los desarrolladores, ofreciendo una interfaz fácil de usar y funcionalidades poderosas. *TextBlob* utiliza modelos previamente entrenados y bases de datos léxicas para realizar análisis de sentimientos de manera efectiva. *TextBlob* se destaca por su simplicidad y versatilidad. Algunas de sus características notables incluyen:

- Análisis de Sentimientos: *TextBlob* proporciona una función de análisis de sentimientos que devuelve la polaridad (positiva, negativa o neutra) y subjetividad de un texto.
- Interfaz Amigable: Su interfaz fácil de usar permite a los desarrolladores realizar análisis de sentimientos con solo unas pocas líneas de código, haciéndolo ideal para proyectos rápidos y aplicaciones prototipo.
- Personalización: Aunque utiliza modelos preentrenados, *TextBlob* permite a los usuarios personalizar los modelos y las bases de datos léxicas según sus necesidades específicas.

TextBlob ha encontrado aplicación en diversas áreas, desde la evaluación de comentarios en redes sociales y análisis de reseñas de productos hasta la clasificación automática de documentos. Su enfoque fácil de usar y su capacidad para proporcionar resultados razonables hacen que sea una elección popular para aquellos que desean integrar rápidamente análisis de sentimientos en sus proyectos de procesamiento del lenguaje natural.

3.3. Modelos de Machine Learning

En esta sección, exploramos los modelos de aprendizaje automático propuestos y utilizados para nuestro análisis de sentimientos, sus características, fortalezas y debilidades. Estos modelos representan la etapa esencial de nuestro estudio ya que permiten automatizar la tarea de comprender, entrenar y clasificar las emociones, sentimientos y opiniones expresadas en el texto del tweet. El objetivo de comparar entre estos modelos se extiende a experimentar con diversas técnicas de aprendizaje supervisado y también algunas de aprendizaje no supervisado, aunque nos enfocaremos mas en el supervisado, destacando su aplicación específica en la detección y clasificación de sentimientos en el lenguaje humano.

Abordamos desde algunos algoritmos clásicos hasta enfoques mas avanzados y recientes que forman los cimientos de lo que es el aprendizaje profundo, analizando cómo estos modelos interpretan y generalizan las complejidades del lenguaje para capturar la carga emocional en las expresiones escritas, teniendo como desafió principal capturar la escénica del lenguaje humano en el texto, como podria ser la ironía o el sarcasmo. Además, examinaremos la importancia de la elección de características (o variables del modelo), la optimización de hiperparámetros y la evaluación de rendimiento para construir modelos de análisis de sentimientos eficaces y precisos. Esta sección brindará una visión detallada de la riqueza y diversidad de los modelos de Machine Learning utilizados para el análisis de sentimientos, ofreciendo una perspectiva completa desde métodos tradicionales hasta enfoques de vanguardia.

3.3.1. Bibliotecas

Scikit-learn: Forjando Caminos en el Aprendizaje Automático

Scikit-learn [5] es una biblioteca de aprendizaje automático de código abierto para Python que se originó en 2007 como un proyecto de *Google Summer of Code*, desarrollado por David Cournapeau [10], *scikit-learn* ha crecido y madurado, convirtiéndose en una herramienta esencial en el ámbito del aprendizaje automático y la minería de datos. Su nombre "scikit-efleja su naturaleza de ser parte de la familia de herramientas "sci-kit"de software científico en Python. Esta herramienta ha desarrollado grandes contribuciones para la investigación en aprendizaje automático y aprendizaje estadístico ya que cuenta con una amplia comunidad de desarrolladores que aportan continuamente a su desarrollo y actualización. *Scikit-learn* se beneficia de una activa comunidad de desarrolladores y científicos de datos que contribuyen con mejoras, nuevas funciones y tutoriales. La biblioteca se mantiene actualizada con las últimas tendencias en aprendizaje automático y sigue siendo una elección popular tanto para principiantes como para expertos en la materia.

Scikit-learn se ha convertido en la biblioteca de referencia para muchos científicos de datos y practicantes de aprendizaje automático debido a su simplicidad, eficiencia y versatilidad. Algunas de sus características y utilidades más destacadas incluyen:

- Algoritmos de Aprendizaje Automático: *scikit-learn* proporciona implementaciones eficientes de una amplia gama de algoritmos de aprendizaje supervisado y no supervisado, incluyendo clasificación, regresión, agrupación y reducción de dimensión.
- Selección de Modelos y Ajuste de Hiperparámetros: *scikit-learn* incluye herramientas para realizar selección de modelos y ajuste de hiperparámetros mediante validación cruzada y búsqueda de cuadrícula.
- Evaluación de Modelos: Proporciona funciones para evaluar el rendimiento de modelos mediante métricas como precisión, recall, F1-score y curvas de características operativas del receptor (ROC).
- Análisis de Componentes Principales (PCA): *scikit-learn* facilita la implementación de técnicas de reducción de dimensión como PCA, que es crucial para la visualización y comprensión de datos de alta dimensión.
- Aprendizaje No Supervisado: En tareas de aprendizaje no supervisado, como la detección de anomalías y la exploración de datos, *scikit-learn* ofrece herramientas eficaces.

En conclusión, *Scikit-learn* ha desempeñado un papel fundamental en la democratización del aprendizaje automático, brindando a los desarrolladores y científicos de datos herramientas poderosas y fáciles de usar. Su amplia adopción y contribuciones continuas de la comunidad de código abierto son testamentos de su impacto duradero en la disciplina del aprendizaje automático.

[5] scikit-learn Documentation: https://scikit-learn.org/stable/documentation.html

3.3.2. Modelos propuestos

Scikit-learn trae consigo una amplia variedad de modelos, cada uno con distintas variantes y enfoques que permiten un estudio comparativo profundo en el que resaltan sus fortalezas y debilidades de cada uno o hasta incluso el uso en conjunto de algunos de ellos para potenciarse de forma combinada. A continuación se enlistan los modelos propuestos para este estudio, se analizan sus características, ventajas, desventajas, requerimientos, entre otros.

Regresión Logística:

Un modelo lineal generalizado utilizado para la clasificación que se usa ampliamente en aplicaciones de aprendizaje automático porque funciona bien con datos grandes y dispersos, por lo que tiene un uso especial en tareas de minería de texto. El efecto de aplicar la regresión logística es comprimir la salida a un conjunto discreto de valores que corresponden a un valor de respuesta categórico. Como la salida $\hat{y}$ es la probabilidad de que la instancia de entrada pertenezca a una clase determinada, utilizamos un modelo binario '*uno frente al resto*' para cada clase que se interpreta como la probabilidad de ser o no estar dentro de la clase, por lo tanto creará tres clasificadores binarios [*'neg', 'neu', 'pos'*] que entrenamos con los siguientes parámetros usando el modelo *scikit-learn*:

Parámetro:	Penalización	C	Multi clase	Solver
Dominio:	'l1', 'l2'	(0.1,100)	'ovr'	'lbfgs', 'newton-cg'

1. *Regularización:* Usamos penalización *L2*[6] en los coeficientes estimados (como regresión Ridge), que se puede controlar usando el parámetro *'C'*, los valores altos corresponden a una menor regularización, el modelo intentaría ajustarse a los datos de entrenamiento lo mejor posible, mientras que con valores bajos se esforzaría más en encontrar coeficientes cercanos a cero incluso si el modelo se ajusta a los datos de entrenamiento un poco peor.

2. *Multi-clase:* El algoritmo de entrenamiento usa el esquema 'uno contra el resto' si la opción multi_class está configurada en 'ovr', y usa la pérdida de entropía cruzada si la opción multi_class La opción está configurada en 'multinomial'.

3. *Solucionador:* Algoritmo a usar en el problema de optimización, solo los solucionadores 'newton-cg', 'sag' y 'lbfgs' admiten la regularización con penalty L2 y formulación primaria como seleccionamos para la penalización

4. *Formulación:* Preferimos la Primal cuando las instancias de datos de entrenamiento son mayores que el número de características y Dual para otros casos. (Dual solo se implementa para penalización L2 con solucionador 'liblinear')

Clasificador Naive Bayes:

El clasificador Naïve Bayes (NB) es un clasificador probabilístico basado en el teorema de Bayes para distribuciones a priori, es decir, encuentra sus fundamentos en la probabilidad condicional con supuestos independientes entre los características y encuentra aplicaciones en problemas como filtrado de spam, clasificación de texto, sistema de recomendación híbrido, entre algunos otros. El clasificador NB ha demostrado ser óptimo y eficiente en muchas tareas de clasificación de textos de aprendizaje automático [26] (especialmente con

[6]Esto es usando la distancia euclidiana habitual al calcular la norma.

supuestos de independencia entre las etiquetas de los documentos). Categoriza los documentos basándose en la técnica de n-gramas, por lo que agrupa o categoriza la polaridad de cada oración presente en los documentos. Para implementarlo, utilizamos el modelo *Bag of Words* (del que hablaremos mas adelante) como modelo de características en el que obtuvieron el mejor resultado, al igual que los algoritmos de Support Vector Machines y Regresión Logística.

La etiqueta predicha $\hat{y}$ es la variable respuesta que maximiza este cálculo de probabilidad de Y dado X. El clasificador multinomial Naive Bayes (que asume distribución multinomial) es adecuado para clasificación con características discretas (por ejemplo, recuentos de palabras para clasificación de texto) y lo consideraremos usando los siguientes parámetros:

Parámetro:	Alpha	Force_alpha	Fit_prior	Class_prior
Dominio:	(0,1)	'true', 'false'	'true'	'(0.33,0.44,0.21)'

1. *Alpha:* Se probará con el parámetro de suavizado aditivo *(Laplace/Lidstone)* con valor predeterminado de 1. (alpha=0 y force_alpha=true significa que no hay suavizado).

2. *Force_ alpha:* Si se establece 'Falso' y alfa es menor que 1e-10, se establecerá alfa en 1e-10. Si es "Verdadero", alfa permanecerá sin cambios. (Esto puede causar errores numéricos si alfa está demasiado cerca de 0).

3. *Fit_ prior:* Ya sea para aprender las probabilidades previas de la clase o no. Si es falso, se utilizará un previo uniforme.

4. *Class_ prior:* Las probabilidades de clase previa del modelo. Si no se especifica, estos antecedentes se ajustan solo de acuerdo con la frecuencia de los datos, lo que resulta útil para una distribución de clases desequilibrada.

Maquinas de Soporte Vectorial

Recientemente, ha habido un creciente interés por las máquinas de soporte vectorial, ya que varios estudios han demostrado que el SVM supera a otros algoritmos de clasificación. [9]. El problema SVM (primal) es encontrar la superficie de decisión que maximiza el margen entre los puntos de datos de las clases. En el caso lineal, este clasificador recompensa la cantidad de separación entre clases aplicando una función de signo para producir una salida categórica, por lo tanto, manejaremos clases múltiples creando clasificadores binarios lineales únicos para cada clase. Una vez definido este criterio como regla de decisión definimos los límites de decisión para cada clasificador y su respectivo margen de clasificación, así entre todos los clasificadores, el mejor de ellos es el que tiene mayor margen de separación entre puntos y lo llamamos a este *máquina de soporte vectorial lineal.*

La ventaja de este método es que pequeños cambios en los datos de un documento en particular no cambiarían la etiqueta que asignará el clasificador, por lo que es más resistente al ruido o perturbaciones que están comúnmente presentes. Además, sigue siendo eficaz en los casos en los que el número de características es mayor que el número de instancias de datos y solo utiliza un subconjunto de puntos de entrenamiento en la función de decisión, por lo que es eficiente en memoria. Elegimos *LinearSVM* en *scikit-learn* en lugar de SVM implementado en términos de liblinear en lugar de libsvm, el cual tiene más flexibilidad en la elección de penalizaciones y funciones de pérdida y debería escalar mejor a una gran cantidad

de muestras y vamos a probar con los siguientes parámetros:

Parámetro:	Penalización	C	Gamma	Multi clase	Kernel
Dominio:	'L1', 'L2'	(0.1,100)	'scale'	'ovr'	'linear', 'poly'

1. *Regularización*: Aplicamos una penalización *L2* con *'C'*=1 igual que en la regresión logística, para medir qué tan importante es que algunos documentos individuales estén etiquetados correctamente, valores más pequeños de 'C' (más regularización) significan más tolerancia a errores en documentos individuales.

2. *Kernel*: Los límites de decisión lineales generalmente funcionan mejor con datos de texto, pero se probará usando kernel polinómico para intentar límites más complejos.

3. *Multi-clase:* Decidimos preferir aprender menos clasificadores, por eso se prefiere uno contra resto a uno contra uno que producirá más clasificadores.

3.4. Modelos de Deep Learning

En esta sección, exploramos un paso mas adelante con los modelos de aprendizaje profundo para nuestro análisis de sentimientos, y de igual forma que en los de aprendizaje automático, analizamos sus características, fortalezas y debilidades. Estos modelos representan un paso mas avanzado en nuestro estudio ya que permiten entrenar y clasificar al igual que en los modelos anteriores, pero de una forma mas poderosa aunque a la vez mas exigente en cuanto a requerimientos computacionales, tamaño de los datos, entre otros. El objetivo propuesto al explorar estos modelos se extiende a experimentar, dentro del alcance posible, la potencia de estos algoritmos y mostrar la diferencia en el desempeño contra los algoritmos tradicionales.

Abordaremos algunos algoritmos de amplio uso como lo son las redes neuronales y buscamos experimentar también con enfoques mas recientes como lo son algunos modelos pre-entrenados en español los cuales en la actualidad han mostrado obtener excelentes resultados en tareas de clasificación de procesamiento de lenguaje natural y en los cuales existe una extensa comunidad que enfoca sus esfuerzos en el continuo desarrollo de este tipo de modelos, analizamos sus requerimientos y áreas de oportunidad en contraste con la eficiencia que estos han demostrado.

3.4.1. Bibliotecas

PyTorch: Iluminando el Camino en el Aprendizaje Profundo

PyTorch[7] es una biblioteca de aprendizaje profundo de código abierto, que ha dejado una marca significativa en el mundo del Machine Learning y la investigación en inteligencia artificial. Desarrollada por el grupo de investigación de inteligencia artificial de Facebook, *PyTorch* se ha convertido en una herramienta esencial para investigadores y profesionales debido a su flexibilidad y capacidad para facilitar la implementación de modelos complejos. *PyTorch* fue lanzada en octubre de 2016 [38] y ha evolucionado rápidamente gracias a su comunidad activa de desarrolladores. Su enfoque y estructura han atraído a una amplia gama de usuarios, desde investigadores académicos hasta ingenieros en la industria, algunas de sus características:

[7]PyTorch Official Documentation: https://pytorch.org/docs/stable/index.html

- Aprendizaje Dinámico y Definición de Grafos: A diferencia de algunas bibliotecas de aprendizaje profundo, *PyTorch* adopta un enfoque dinámico para definir y modificar grafos computacionales durante el tiempo de ejecución. Esto brinda una mayor flexibilidad para experimentación y desarrollo iterativo.
- Soporte para Redes Neuronales Profundas: *PyTorch* facilita la construcción y entrenamiento de modelos de redes neuronales profundas, incluyendo arquitecturas complejas como redes neuronales convolucionales (CNN) y redes neuronales recurrentes (RNN).
- Comunidad Activa y Recursos de Investigacion: La comunidad de *PyTorch* es vibrante y ofrece una amplia gama de recursos educativos, tutoriales y documentación, lo que facilita la adopción y el aprendizaje para nuevos usuarios.

PyTorch ha encontrado aplicaciones en diversas áreas de investigación y es una opción popular en la comunidad académica debido a su capacidad para experimentación rápida e implementación sencilla de nuevas ideas en modelos. En el campo de la visión por computadora se utiliza extensamente para construir y entrenar modelos para tareas como clasificación de imágenes, detección de objetos y segmentación semántica. También ofrece herramientas y modelos preentrenados, y esto lo vuelve una elección popular en distintos proyectos de PLN.

3.4.2. Redes Neuronales en Análisis de Sentimientos y PLN

Las redes neuronales han emergido como pilares fundamentales en PLN, en particular, en análisis de sentimientos, estas arquitecturas avanzadas permiten capturar relaciones complejas en datos textuales, lo que resulta crucial para comprender subjetividad y emocionalidad en el lenguaje humano. Las redes neuronales son ampliamente utilizadas para clasificación de sentimientos, donde se entrenan para asignar etiquetas a fragmentos de texto. Modelos como LSTM (Long Short-Term Memory) y GRU (Gated Recurrent Unit) en RNN, así como variantes de CNN, han demostrado éxito y también son aplicadas para el análisis más detallado de emociones, modelos como BERT han mostrado capturar matices emocionales y contextuales, permitiendo una mejor representación de la carga emocional en el lenguaje. [17].

En un panorama donde el lenguaje es intrínsecamente ambiguo y contextual, destacan por su capacidad para aprender patrones y representaciones de manera automática, adaptándose a la diversidad y complejidad del discurso. Su potencia permite una comprensión más profunda y contextual de las expresiones en texto. La sutileza y complejidad del lenguaje son fundamentales, las capacidades adaptativas de estas arquitecturas proporcionan una base sólida para abordar tareas desafiantes. Aunque como mencionamos, su mejora en el desempeño esta sujeta a mayor requerimiento computacional y a mayor cantidad de datos, es la mejor alternativa disponible en la actualidad y por lo tanto es imposible ignorarlas dentro de este estudio, de esta forma, nos interesan los siguientes tipos de redes neuronales:

- Redes Neuronales Recurrentes (*RNN*): Las *RNN* son aptas para secuencias de datos, y las convierte en una elección natural para tareas de procesamiento de texto donde el orden de palabras es crucial. Su capacidad para mantener información de estados anteriores las hace adecuadas para capturar dependencias a corto plazo en texto [39].
- Redes Neuronales Convolucionales (*CNN*): Las *CNN*, conocidas por su eficacia en la visión por computadora, también han demostrado ser útiles en tareas de PLN. Se aplican ventanas de convolución sobre secuencias de palabras para capturar patrones y características relevantes, proporcionando una visión localizada del contexto.

- Redes Neuronales de Transformers: Los modelos basados en transformers, como BERT (Bidirectional Encoder Representations from Transformers), han revolucionado el PLN. Estos modelos atienden a la totalidad del contexto, permitiendo una comprensión profunda de las relaciones semánticas y contextuales. Su atención no restringida a la secuencia de entrada ha superado barreras en la captura de dependencias a largo plazo [12].

3.4.3. Modelos propuestos

Perceptrón Multicapa:

Las redes neuronales han formado la base del reciente resurgimiento del aprendizaje profundo, logrando impresionantes resultados de vanguardia en tareas que van desde la clasificación de objetos en imágenes hasta la traducción automática rápida y precisa. [39] Uno de los métodos más simples de redes neuronales es el Perceptrónes multicapa, abreviado como *MLP*, es similar a la regresión logística pero va un paso más allá al agregar 'unidades ocultas' contenidas en 'capas ocultas', donde cada una de estas calcula una función no lineal de las variables, esta función se llama *función de activación* y consideramos algunas de estas funciones 'con forma de S' al entrenar los datos. El resultado de agregar estas capas ocultas adicionales implica calcular una suma ponderada inicial diferente de los valores de características para cada unidad oculta, lo cual es más costoso desde el punto de vista computacional.

Existe un peso entre cada entrada y cada unidad y otro entre cada unidad y la salida, usar esto y combinar funciones de activación nos permite aprender funciones más complejas para realizar predicciones más precisas, pero también significa que hay más pesos y coeficientes para estimar en la etapa de entrenamiento para la predicción. (Se necesitan más datos de entrenamiento y más cálculos para aprender en comparación con los modelos lineales).

Parametro:	Hidden_Layer_Sizes	Solver	Activation fun.
Dominio:	[100], [100,100], [100,100,100]	'lbfgs', 'adam'	'logistic', 'tanh', 'relu'

1. *Tamaño de capas ocultas:* Una lista con un elemento para cada capa oculta, que proporciona el número de unidades ocultas para esa capa. Pasamos dos valores de 100. (dos capas ocultas, 100 unidades por capa)

2. *Función de activación:* La función de activación de la capa oculta podría ser la función sigmoidea logística, la función tan hiperbólica o la función unitaria lineal rectificada.

3. *Solucionador:* Para la optimizacion, podemos usar 'lbfgs', un optimizador de la familia de métodos cuasi-Newton, 'sgd' se refiere al descenso de gradiente estocástico, 'adam' un optimizador basado en gradiente estocástico propuesto por Kingma, Diederik [39]. El solucionador predeterminado 'adam' funciona bien en conjuntos de datos relativamente grandes en tiempo de entrenamiento y validación.

3.4.4. Pysentimiento: Una alternativa poderosa y sencilla

PySentimiento emerge en 2021 desarrollada por Mario García como una biblioteca de codigo abierto especializada en análisis de sentimientos para Python, simplificando la tarea de explorar las complejidades emocionales dentro del lenguaje natural [16]. Esta biblioteca se erige como un recurso valioso para aquellos que buscan abordar de manera eficiente y precisa el análisis de sentimientos en sus proyectos de PLN. *PySentimiento* ofrece una interfaz fácil de usar y

herramientas avanzadas que permiten a los investigadores y desarrolladores aplicar modelos de Machine Learning y Deep Learning para el análisis de sentimientos, además de soportar múltiples lenguajes y adaptarse a diferentes necesidades y contextos de análisis. Esta biblioteca ofrece un conjunto de funciones potentes que facilitan el análisis de sentimientos en texto de manera intuitiva. Algunas de sus características destacadas incluyen:

- Múltiples Modelos de Sentimientos: *PySentimiento* incorpora diversos modelos de análisis de sentimientos, permitiendo a los usuarios elegir el enfoque que mejor se adapte a sus necesidades específicas.
- Soporte Multilingüe: *PySentimiento* extiende su utilidad más allá del inglés, ofreciendo soporte para varios idiomas y permitiendo análisis en contextos multilingües.
- Análisis de texto especifico de Redes Sociales: *PySentimiento* se puede emplear para evaluar la polaridad de comentarios y publicaciones en plataformas de redes sociales, proporcionando información valiosa sobre las opiniones y emociones de los usuarios.
- Monitoreo de Reputación en Línea: Empresas y marcas pueden utilizar *PySentimiento* para analizar la percepción en línea, identificando tendencias y evaluando la reputación en función de las interacciones y comentarios en la web.
- Análisis de Reseñas de Productos: En el ámbito del comercio electrónico, *PySentimiento* facilita el análisis de las reseñas de productos, ayudando a comprender la satisfacción del cliente y detectar áreas de mejora.

Contribuciones y Comunidad: *PySentimiento* se beneficia de una comunidad activa de desarrolladores y usuarios que contribuyen a su mejora continua. Las actualizaciones regulares y la participación de la comunidad aseguran que la biblioteca siga siendo relevante y efectiva en el dinámico campo del análisis de sentimientos.

Capítulo 4

Metodología

Una vez seleccionadas las herramientas para el estudio y las técnicas propuestas bajo distintos enfoques, procedemos con manos a la obra hacia la fase de experimentación, donde comenzamos buscando un preprocesamiento del texto para los tweets del corpus, esto es, aplicar métodos de corrección en las palabras, eliminar signos de puntuación o caracteres especiales, entre otros, con la finalidad de extraer de la forma mas limpia posible las palabras dentro del corpus para posteriormente construir de estas, una representación numérica que sea útil para los modelos de clasificación. Esta etapa es crucial dentro del Procesamiento de Lenguaje, ya que los tweets contienen distintos elementos como pueden ser hashtags, menciones o emojis y buscamos sacar el mejor provecho posible de estos elementos para extraer el contexto del sentimiento en el que han sido utilizados. Seguidamente pasaremos a la parte de extracción de características, donde el objetivo es construir el mejor vocabulario de palabras de nuestro corpus, el cual sera la base para la matriz de diseño que utilizan los modelos de aprendizaje, la pregunta que se busca responder es cual de los distintos tipos de representación de características propuestos (o combinación de estos) funcionan mejor para nuestros modelos.

Finalmente, una vez seleccionado un método de extracción de características, buscaremos también la mejor configuración de hyperparametros en cada uno de los modelos utilizando distintas mallas de valores para mostrar los resultados de cada uno y poder visualizar una comparativa entre cada uno de los métodos tradicionales y los basados en reglas o transformers.

4.1. Procesamiento de Texto

Tal como se mencionó antes, el siguiente paso es procesar los tweets en bruto (tal como se recolectaron) y convertirlos en datos que podamos usar para la clasificación. Primero debemos tener en cuenta que al realizar un preprocesamiento de texto, este dependerá tanto del tipo de texto como de la tarea que se pretenda realizar, por esta razón, es importante considerar el método adecuado para nuestro problema. Entonces, al realizar análisis de sentimientos, es importante un procesamiento para capturar las emociones de las palabras, y bajo este enfoque nos centramos en procedimientos específicos para construir un vocabulario de las palabras.

En esta sección, se hace uso de las bibliotecas *Nltk, Gensim y sPacy* para llevar a cabo las tareas necesarias de procesamiento de texto, esto incluye la eliminación de dígitos, la separación de palabras según patrones, la normalización de palabras, el ajuste de tokens especiales, la transcripción de emojis, lematización e incluso la eliminación de palabras vacías. Con el

objetivo de obtener los tokens más limpios posibles hechos de palabras individuales y representados como cadenas UTF-8, los cuales constituirán nuestro vocabulario de todos palabras únicas (puede que nos interese encontrar las más frecuentes) en el corpus, hemos construido un proceso secuencial, que sigue los pasos siguientes:

Tareas básicas de procesamiento

1. **Eliminar dígitos, espacios en blanco dobles y saltos de línea:** Extraer información de dígitos presenta un desafío ya que pueden representar diferentes cosas como magnitudes, fechas, direcciones, etc. En consecuencia, los dígitos se omiten intencionalmente.

2. **Separar palabras encontrando patrones:** Los tweets contienen errores ortográficos o escritura en camello en los hashtags. Entonces, identificamos patrones comunes (como puntos antes de una letra mayúscula o símbolos como '#' o '¿') y los separamos usando expresiones regulares.

3. **Normalizar palabras:** Transformamos el texto a minúsculas y eliminamos signos de puntuación y otros caracteres especiales. Si hubiésemos hecho esto antes del paso anterior no se habría dividido correctamente. Además, los caracteres repetidos consecutivos (normalmente utilizados para reír) se minimizaron a dos repeticiones para evitar la formación de nuevos símbolos para palabras que ya estaban presentes en el vocabulario.

4. **Identificar tokens especiales:** Los tweets suelen contener menciones, enlaces web e imágenes. Así, identificamos objetos comunes con etiquetas generales. Por ejemplo, las menciones a los usuarios estaban empaquetadas con el token 'usuario' y los enlaces web con el token 'url'.

Tareas enfocadas a extraer contexto

1. **Transcribir emojis a palabras:** Convertimos emojis en palabras, posicionándolos correctamente dentro del tweet usando el módulo Python emoji[1]. Todos estos están normalizados a minúsculas.

2. **Lematizar:** Dada una palabra, podemos encontrar su forma de diccionario (o lema). Este proceso se realizó utilizando la biblioteca *sPacy*. Esto nos permitió reducir el tamaño del vocabulario al evitar múltiples tokens para diferentes inflexiones de la misma palabra.

3. **Remover *Stop words*:** Las palabras comunes que no contienen información semántica, llamadas "palabras vacías", se eliminan para reducir el tamaño del vocabulario.

A continuación mostramos los resultados del procesamiento de los tweets utilizando el método descrito. La tabla 4.1 muestra el tweet original sin procesar en la primera columna y el texto procesado en la segunda columna. Después del proceso, todo el texto queda en minúsculas, los enlaces se cambian a un token especial denominado 'url', mientras que los usuarios mencionados se reemplazan por el token especial 'usuarios'. Además, los emojis se convierten en palabras que los describen y se evita la repetición consecutiva de algunos elementos comunes como risas, a un máximo de 2 repeticiones.

[1] `https://pypi.org/project/emoji/`

Tweet Original	**Tweet Procesado**
#CuarentenaNacional #CDMX consulta:https://t.co/TcjustEg	cuarentena nacional cdmx consulta url
Buen díaa!!!#ConCaféEnMano para alegrar la mañana #EnCasa	buen dia con cafe en mano para alegrar la mañana en casa
@CONANPmx@GobiernoMX lleno de gente en Av. Tenorio	usuario lleno de gente en av tenorio
Marcarle a mi preciosita en momento de crisis. 🥺 🥺 🥺	marcarle preciosa momento crisis cara por favor
Uff 😢 🦠 🦠 🦠 #QuedateEnCasa 🏠 #Coahuila #Mexico 😥	uf cara triste alivio microbio quedar casa jardin coahuila mexico
#SNTEsalud ⚕ ⚠ ALERTA alto contagio en los mochis	sntesalud simbolo medicina advertencia alerta alto contagiar mochis

Tabla 4.1: Resultados del procesamiento de tweets

Es importante remarcar que este procesamiento es un proceso secuencial, es decir los pasos mencionados anteriormente están en el orden en el que fueron llevados a cabo en el corpus, de no respetar este orden, el procesamiento no funcionaria de forma óptima, ya que separaría incorrectamente ciertos patrones lo cual impediría que se reconozcan correctamente algunos elementos como enlaces web o hashtags.

4.2. Extracción de características (variables del modelo)

Con el texto procesado, lo siguiente fue experimentar con diferentes representaciones numéricas de los datos para la etapa de entrenamiento. Se comparan algunas técnicas simples de extracción de características (variables) para construir el vocabulario de todas las palabras presentes en el corpus. Ademas se prueba el uso de modelos pre-entrenados para PLN, los cuales proporcionan un enfoque más complejo aunque mas efectivo para extraer características.

4.2.1. Enfoque *Bag of Words* (BoW)

El enfoque BoW es el mas simple, cada token del texto corresponde a una dimensión (característica) determinada, suponiendo una representación vectorial. Cada token en el texto tendrá un peso que puede estar dado por el número de apariciones de una palabra en el texto (Term Frecuency) o multiplicando el número de apariciones de una palabra en todo el corpus por las apariciones de una palabra en el texto (Term frequency-Inverse Document Frequency).

1. **Term Frequency**: Para transformar tweets en representaciones vectoriales, utilizamos *Scikit-learn*, la cual tiene las herramientas para ajustar los datos y construir el vocabulario, de esta manera se obtiene una matriz de frecuencias que se usa en los modelos de clasificación. El tamaño del vector de cada tweet es igual al tamaño del vocabulario del corpus y, como se mencionó antes, el valor de cada dimensión es el número de apariciones de una palabra en el tweet. Se extraen diferentes conjuntos de características modificando los siguientes parámetros:

- *n_ gram_ range:* Nos permite especificar si cada token está conformado por palabras singulares o por n-gramas [2] de palabras (esto con el objetivo de intentar preservar el orden local de las palabras, lo cual conlleva el costo de incrementar considerablemente el numero de características).
- *min_ df:* Especifica el número mínimo de veces que debe aparecer una palabra en todo el corpus para ser considerada parte del vocabulario. Existen muchas palabras dentro del corpus que tienen pocas apariciones, por lo que pequeñas variaciones de este parámetro aumentan considerablemente el tamaño del vocabulario.
- *stop_ words:* Estas palabras son consideradas 'palabras vacías', es decir, que no cuentan con significado relevante para la clasificación, por lo que se busca probar si son útiles al estar presentes o ausentes en el entrenamiento. Utilizamos la lista de palabras vacías proporcionada por la biblioteca Nltk, que consta de 313 palabras.

Una gran desventaja es que se pierde el orden de ocurrencia de las palabras, ya que creamos un vector de tokens en orden aleatorio, el uso de n-gramas pretende preservar un poco de este orden. La tabla 4.2 muestra en la última columna el tamaño del vocabulario extraído modificando los parámetros mencionados anteriormente. La primera columna indica si el texto está lematizado o no, la segunda columna indica si se eliminan las palabras vacías, la tercera columna indica el rango de n_gram y la cuarta columna muestra el valor de la frecuencia mínima del documento.

Texto	Stopwords	N-gramas	mindf	Tamaño de Vocabulario
Normalizado	Si	1_grama	2	4343
			3	2797
		2_grama	2	10674
			3	5919
	No	1_grama	2	4198
			3	2664
		2_grama	2	6456
			3	3614
Lematizado	Si	1_grama	2	3663
			3	2422
		2_grama	2	10213
			3	5812
	No	1_grama	2	3605
			3	2369
		2_grama	2	6533
			3	3629

Tabla 4.2: Tamaño del vocabulario para diferentes parámetros

2. **Term Frequency - Inverse Document Frequency:** Para mejorar nuestro modelo básico de BoW, ahora debemos pensar en términos comunes dentro del corpus que no aporten información sobre la polaridad del tweet ya que están relacionados con el tema. La ponderación de frecuencia de términos considera que todas las palabras tienen la

[2] n-grama es una secuencia continua de n elementos de una secuencia dada de texto

misma importancia, esto se puede mejorar teniendo en cuenta el significado de cada término en el corpus. Para darle un peso a cada término, calculamos los valores de Tf-Idf, usando *Scikit-learn*.

Una puntuación alta de Tf-Idf indica que una palabra está presente en el tweet pero no se encuentra en muchos otros tweets del corpus. Sin embargo, un valor bajo de Tf-Idf implica que la palabra se utiliza con frecuencia en la mayoría de los tweets. Este proceso enfatiza las palabras que son importantes para el tweet. Usamos *Scikit-learn* para ajustar los datos y encontrar los valores tf-idf, los cuales aumentan proporcionalmente al número de veces que aparece una palabra en un determinado documento (tweet) y disminuye con la cantidad de documentos en el corpus que contienen la palabra. La tabla 4.3 muestra términos con los valores Tf-Idf más pequeños y más grandes. Esto, usando tokenización normalizada de Bigramas con palabras vacías incluidas y mindf=2:

Valores Tf-Idf	**Características**
Menores	'buen lunes', 'app', 'inicio semana', 'inmediato', 'oms', 'periodico hoy'
	'alto contagio', 'ganar seguidor', 'calidad', 'amlolujo'
Mayores	'financiero', 'muerte covid', 'lugar', 'movil', 'movilidad', 'dar positivo'
	'lopez', 'muerte', 'cuidarte profesional', 'gracia'

Tabla 4.3: Características con menores y mayores valores Tf-Idf.

4.2.2. Embeddings de palabras y Modelado de frases

Como se definió antes, los embeddings (incrustaciones) de palabras es una técnica para representar palabras como vectores. Partiendo del supuesto de que palabras con significados similares suelen aparecer en contextos similares, procedemos a experimentar con representaciones vectoriales de las palabras de nuestro corpus, buscando obtener predicciones mas precisas usando vectores pre-entrenados y además con la ventaja adicional de reducir el numero de características mientras preservamos el contexto.

1. **Word2Vec:** La biblioteca *Gensim* nos proporciona este algoritmo popular utilizado para entrenar modelos de embeddings de palabras a partir de grandes cantidades de texto sin etiquetas (o cargar modelos pre-entrenados), en el cual tenemos dos enfoques disponibles; CBOW (Continuous Bag of Words) y Skip-gram, ambas técnicas aprenden pesos que actúan como representaciones vectoriales de palabras. CBOW predice la palabra actual a partir de un ventana de palabras de contexto circundantes, mientras que Skip-gram predice las palabras de contexto circundantes dada la palabra actual y hemos decidido utilizar Skip-gram ya que funciona mejor para esta tarea.

   Debido a que nuestro corpus no es lo suficientemente grande como para entrenar de manera óptima un modelo propio, hemos optado por hacer uso de vectores pre-entrenados, para esto hemos usado los vectores pre-entrenados de *Spanish Billion Word Corpus* construidos por Cristian Cardellino [3] que contiene mil millones de palabras en español y es una fuente común para modelos pre-entrenados al cual se puede acceder de forma abierta. De esta forma, como se planeo inicialmente, se crearon 'Keyed Vectors' de tamaño 300

[3] https://crscardellino.ar/SBWCE/

(dimensión) con el nombre de la palabra como clave para buscar relaciones semánticas y contextuales entre las palabras del corpus, la tabla 4.4 muestra los resultados de una prueba simple de las relaciones mas relevantes que encuentra nuestro algoritmo cuando se usa la palabra 'contagio', primero utilizando vectores entrenados con nuestro conjunto de datos y otra usando los vectores pre-entrenados, donde las diferencias son notables.

Entrenados en el corpus	**Vectores pre-entrenados**
('fallecimiento', 0.7339)	('contagiosas', 0.7951)
('repunte', 0.7279)	('contagiosa', 0.7761)
('disparar', 0.7241)	('contagioso', 0.7540)
('elevar', 0.6851)	('infección', 0.7088)
('cuatro', 0.6850)	('contagiado', 0.7037)
('oms', 0.6832)	('contagió', 0.6875)
('curva', 0.6649)	('infeccioso', 0.6703)
('sumar', 0.6634)	('infecciones', 0.6689)
('tasa', 0.6600)	('infecciosas', 0.6661)
('advertencia', 0.6574)	('infecciosos', 0.6647)

Tabla 4.4: Palabras relacionadas a contagio utilizando vectores entrenados en el corpus y los vectores pre-entrenados

2. **Doc2Vec:** Con *Gensim* podemos optar también por un algoritmo Doc2Vec que usa la misma lógica que Word2Vec, pero aplicada a nivel de documento (tweet), cada documento se representa como un vector denso en un espacio de características continuas y esto nos proporciona un enfoque diferente al de predecir con palabras individuales, el cual nos servirá como base principal para experimentar con técnicas de modelado de frases. Para utilizar este algoritmo haremos uso nuevamente de los vectores pre-entrenados del Spanish Billion Word Corpus utilizando Keyed Vectors de tamaño 300, con el nombre de palabra como clave de la misma forma que con vectores de palabras.

   Cada documento es asignado a un vector único, representado por una columna en una matriz D y cada palabra también está asignada a un vector único, representado por una columna en otra matriz W, el vector de documento y los vectores de palabra se promedian o concatenan para predecir la siguiente palabra en un cierto contexto, esto actúa como una memoria que recuerda lo que falta en el presente contexto o el tema del párrafo y a este enfoque lo llamaremos modelo de memoria distribuida, según Le y Mikolov (2013) [32] y probaremos comparar los resultados de este modelo contra el enfoque simple de Bolsa de Palabras Distribuida.

3. **Modelado de frases:** Una vez que se tienen los embeddings de palabra y documento obtenidos con Word2Vec y Doc2Vec, somos capaces de llevar a cabo un intento mas audaz por preservar el contexto valiéndonos de la similaridad de las palabras a través de un modelado de frases, para el cual utilizaremos los enfoques mencionados antes y que describimos de la siguiente forma:

   a) Bolsa de Palabras Distribuida: Este es el modelo análogo al modelo Skip-gram en Word2Vec. Los vectores de documento se obtienen entrenando una red neuronal en la tarea de predecir una distribución de probabilidad de palabras en

un documento dada una palabra muestreada aleatoriamente.

b) Memoria Distribuida: Este es el modelo análogo al modelo CBOW en Word2vec. Los vectores de documento se obtienen entrenando una red neuronal en la tarea de inferir una palabra central basada en palabras de contexto y un documento de contexto, manejaremos este modelo usando métodos de promedio y concatenación:

- *Memoria Distribuida Media:* Los vectores de documento y palabras se promedian para predecir la palabra objetivo. Luego, la media de los vectores se utiliza como características para la predicción.

- *Memoria Distribuida Concatenada:* Los vectores de documento y palabra se concatenan para predecir la palabra objetivo. Luego, los vectores se utilizan como características para la predicción.

La biblioteca *Gensim* también cuenta con funcionalidad para la detección de frases, la cual fue construida de forma similar a la representación de n-gramas, pero en lugar de obtener todos los n-gramas deslizando una ventana a través del tweet, esta detecta secuencias de palabras de uso frecuente y las une en un solo token. La idea se resume en que el modelo aprenda la representación vectorial de frases que tienen un significado que no sea una simple composición de palabras individuales, de esta manera, podemos formar frases razonables sin aumentar demasiado el tamaño del vocabulario, como lo presentaron Mikolov y Chen en 2013). [32]

Construimos modelos de 2 y 3 gramas para detectar y combinar frases de dos y tres palabras de uso frecuente dentro del corpus. Luego de obtener el corpus con las frases, hicimos el mismo proceso Doc2Vec aplicado previamente a los tokens unigrama. Así, presentamos los resultados para cada uno, utilizando tanto la Bolsa Distribuida de Palabras (DBOW) como la Memoria Distribuida (DM), así como su combinación.

La tabla 4.5 muestra las frases detectadas por el algoritmo de detección de frases de *Gensim* en un tweet determinado. Podemos ver que unigrama produce 13 tokens, mientras que 2-gramas y 3-gramas producen 10 y 9 tokens, respectivamente. Como podemos ver, frases como *'no es justo'* y *'quedate en casa'* se convierten en trigramas pero el resto de los tokens permanecen como unigramas. Esto se debe a que el algoritmo sólo extrae los n-gramas más significativos y con esto medimos el impacto de las diferencias entre los tokens producidos en el clasificador.

	Deteccion de frases
Unigrama	['@usuario', 'por', 'su', 'trabajo', 'no', 'es', 'justo', 'para', 'los', 'demas', 'quedate', 'en', 'casa']
Bigrama	['@usuario', 'por', 'su trabajo', 'no es', 'justo', 'para', 'los', 'demas', 'quedate', 'en casa']
Trigrama	['@usuario', 'por', 'su', 'trabajo', 'no es justo', 'para', 'los', 'demas', 'quedate en casa']

Tabla 4.5: Tokens producidos por cada modelo.

4.3. Modelos BERT pre-entrenados en Español:

Los modelos BERT se consideran modelos de última generación para diversas tareas de PLN que implican representación de texto. BERT posee el importante beneficio de apoyar el aprendizaje por transferencia. Estos modelos de lenguaje han sido sometidos a una capacitación exhaustiva durante varios días en máquinas robustas sobre una gran cantidad de texto de plataformas como Wikipedia y sitios web de noticias [12]. Luego, un modelo previamente entrenado se puede ajustar para alinearlo con nuestra tarea de clasificación específica. En los últimos años ha habido avances considerables en los modelos BERT previamente entrenados [13]. Debido a su uso cada vez mayor, se han lanzado muchas variantes de estos (con parámetros reducidos) para acelerar los tiempos de entrenamiento e inferencia. Sin embargo, las versiones de estos (Ej. *DistilBERT*) para idiomas distintos del inglés aún son escasas.

Usamos algunas variantes de los modelos BERT en español. Estos son particularmente útiles ya que han sido creados para tareas de PLN con análisis de alta dimensión. Los modelos españoles son difíciles de conseguir y, cuando están disponibles, con frecuencia se desarrollan utilizando importantes conjuntos de datos y recursos propios. Como resultado, los algoritmos y técnicas relevantes están restringidos a las grandes corporaciones tecnológicas. Sin embargo, existen algunos recursos disponibles de forma abierta, a los que podemos echar mano para experimentar con sus vectores pre-entrenados, ejemplos de estos modelos son:

1. BERTIN: Serie de modelos basados en BERT en español, el centro de modelos actual apunta a lo mejor de los modelos basados en RoBERTa entrenados desde cero en la parte española de mC4 usando Flax[4][1].

2. RoBERTuito: Un modelo de lenguaje para contenido generado por usuarios en español, entrenado siguiendo las pautas de RoBERTa en 500 millones de tweets. RoBERTuito viene con y sin caja [35].

3. BETO: Otro modelo entrenado en un corpus grande en español que tiene un tamaño similar a un BERT-Base y fue entrenado con la técnica de enmascaramiento de palabras completas, que supera a algunos otros modelos [7].

En la tabla podemos visualizar la forma en que se produce la tokenizacion con un fragmento de tweet utilizando el modelo BETO, el cual reconoce de forma correcta la mayoria de las palabras y las asigna a un token numerico, a excepcion de las que estan marcadas con un '#'.

Frase	"bueno pero es no le pidas demasiado mejor preguntenle de la fuerza moral de su patron"
Tokens	['bueno', 'pero', 'es', 'no', 'le', 'pidas', 'demasiado', 'mejor', 'pregunte', '##n', '##le', 'de', 'la', 'fuerza', 'moral', 'de', 'su', 'patr', '##on']
Tokens numéricos	[1491, 1195, 1028, 1054, 1165, 28903, 2668, 1544, 16216, 30959, 1080, 1009, 1032, 3193, 8003, 1009, 1069, 5102, 1022]

Tabla 4.6: Tokenización de tweet con vectores pre-entrenados BETO

[4]Flax: una biblioteca de redes neuronales ecosistema para JAX diseñado para la flexibilidad

4.4. Bibliotecas de '*Caja Negra*':

1. **Nltk Vader:** Debido a que *Vader* no cuenta con lexicón en español, nuestra alternativa fue utilizar el *SentiSense Lexicón* [11] que contiene una lista de palabras clasificadas según su connotación emocional e información sobre intensidad de la emoción que transmite cada palabra. La tabla 4.7 ilustra el funcionamiento de *Vader* con frases que expresan diferente polaridad, donde la columna 'compound' es la suma normalizada de las puntuaciones de cada palabra y un [-1] indica un sentimiento negativo, mientras que [+1] indica uno positivo. Las columnas 'neg', 'neu' y 'pos' indican la probabilidad de que el texto pertenezca a la clase negativa, neutral o positiva, respectivamente y de esta forma utilizamos la herramienta para ejecutar la predicción de todo nuestro conjunto de datos.

Input	**'neg'**	**'neu'**	**'pos'**	**'compound'**
hoy es un pésimo día	0.779	0.221	0.0	-0.5461
hoy es un mal día	0.646	0.354	0.0	-0.7424
hoy es un día cualquiera	0.123	0.637	0.24	0.231
hoy es un gran día	0.0	0.408	0.592	0.5404
hoy es un excelente día	0.0	0.294	0.706	0.8633

Tabla 4.7: Resultados de Vader en frases con distinta polaridad .

2. **Textblob:** Esta biblioteca cuenta con un traductor integrado para distintos idiomas, y debido a que usamos estas librerías con fines meramente comparativo para medir el desempeño de los modelos anteriores, se decidio utilizar el traductor en lugar de sustituir el lexicón. El analizador de *TextBlob* devuelve dos resultados; El valor de polaridad pertenece a [-1, 1], donde -1 indica un sentimiento negativo y +1 uno positivo. El valor de subjetividad oscila entre 0 y 1, siendo 0 indica un texto objetivo, mientras que 1 representa un texto subjetivo. La tabla 4.8 muestra el funcionamiento de *TextBlob* con diferentes oraciones en español donde previamente estas frases fueron traducidas a ingles.

Input	**'polarity'**	**'subjectivity'**
Este teléfono tiene una pantalla de excelente resolución, además es muy rápido	0.63	0.89
Este teléfono tiene una pantalla de alta resolución, además es rápido	0.18	0.57
Este teléfono es lo máximo, lo adoro <3 :D	1.0	1.0
Este teléfono no me gusta :(	-0.75	1.0

Tabla 4.8: Resultados de Textblob en distintas oraciones.

3. **Pysentimento:** Para el caso de esta herramienta, debido a que cuenta con un preprocesador de tweets especialmente adecuado para la clasificación de tweets con modelos basados en transformadores, haremos uso de este en lugar de nuestro proceso anterior y utilizaremos el conjunto de datos resultante para proceder directamente a la predicción, la cual al igual que *Vader* nos muestra la probabilidad de que el tweet pertenezca a cada una de las clases y en este caso la etiqueta predicha del tweet.

Capítulo 5

Pruebas y Resultados experimentales

Con la metodología definida, ahora es posible realizar experimentos con el objetivo de mejorar el desempeño en la predicción de los modelos, comenzando por dividir el corpus en un conjunto de entrenamiento y uno de pruebas, separados aleatoriamente en aproximadamente 3.600 para entrenamiento y 1.200 para pruebas (75 %-25 %) y utilizamos la misma semilla aleatoria para preservar la partición en diferentes experimentos. La tabla 5.1 muestra la distribución de etiquetas en cada conjunto, donde vemos que ambos mantienen una distribución similar.

(Seed=37) [1]	**Negativo**	**Neutral**	**Positivo**
Entrenamiento	33.642 %	44.934 %	21.422 %
Prueba	34.899 %	44.38 %	20.713 %

Tabla 5.1: Distribución de clases al ser divididas

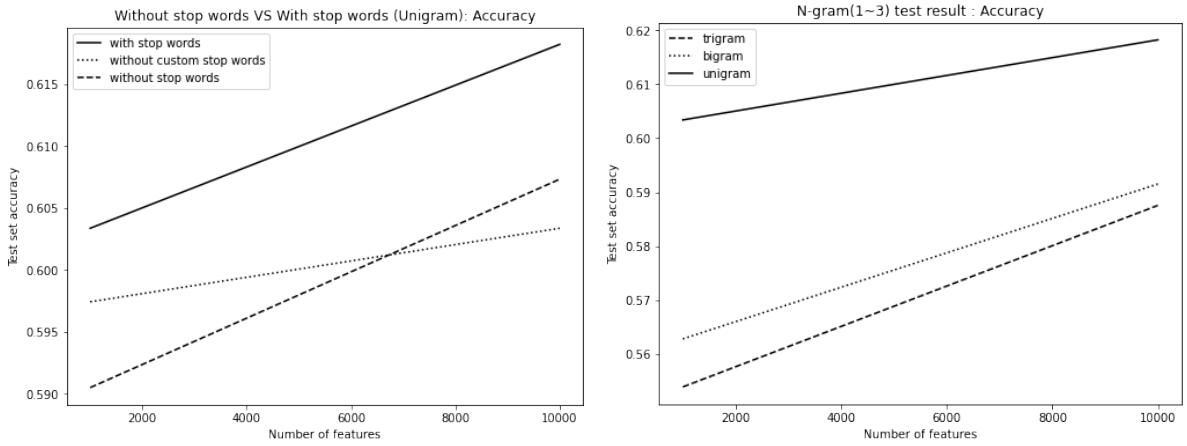
Al comparar algoritmos, es conveniente utilizar una líneamiento base que nos proporcione un punto de referencia para comparar el desempeño. Consideramos la *Regla Cero* (ZeroR), que puede interpretarse como que un clasificador aleatorio simplemente predice la clase mayoritaria, aunque este no tiene poder de previsibilidad, es útil para determinar una guía. Como se muestra en la tabla, la clase mayoritaria es neutro con un 44 %, lo que significa que si un clasificador hubiera predicho neutral para cada dato de prueba, obtendría un 44 % de desempeño.

Se decidió comenzar experimentando con los modelos BoW en pruebas para el mejor tamaño de vocabulario y en pruebas para de reducir la dimensión. Posteriormente se llevaron a cabo pruebas usando modelos Doc2Vec, para buscar una forma de modelar las frases con los n-gramas de una manera diferente, con el objetivo final de comparar si el uso de incrustaciones de palabras puede funcionar mejor para extraer las características (o si tiene algún tipo de ventaja, como el costo computacional) para los modelos de aprendizaje.

5.1. Experimento para el tamaño de Vocabulario:

El objetivo es encontrar el número de características que serían adecuadas para ajustarse mejor a los modelos y buscar cómo establecer un criterio simple para elegir características. Un intento de preservar una parte del contexto perdido al usar un enfoque como el BoW se realizó considerando el uso de *n-gramas* y eliminando las palabras vacías comunes (y un conjunto de algunas de ellas con la frecuencia de uso como criterio). Se procedió probando con diferente numero de n-gramas y palabras vacías para después compararlos utilizando un modelo

de regresión logística simple (con parámetros por defecto) que nos muestre los resultados al predecir en el conjunto de prueba para distintos tamaños de vocabulario. La figura 5.1 muestra los resultados de probar con hasta 10,000 características para medir el *accuracy* en el conjunto de prueba al intentar a) el uso de las stop word y b) el uso de unigramas, bigramas y trigramas.



(a) **(b)**

Figura 5.1: Test Accuracy para distinto numero de características

Podemos ver de forma general que cuantas más características incluimos, mejor predicciones obtenemos, luego, de manera un poco inesperada, eliminar las palabras vacías no resulta útil para el modelo a pesar de que esas palabras no aportan identificación semántica. A continuación, muestra que usar unigramas funciona mejor a medida que aumentan las características que usar bigramas o trigramas y esto significa que estas no logran capturar el contexto deseado, tokens de solo una palabra parecen funcionar mejor. De manera similar utilizar la técnica de 'tf-idf' no logró mostrar ninguna mejora aparente al hacer los mismos experimentos que usando termino de frecuencias simple (no mostramos aquí el gráfico para evitar sobrecargar el papel) esto nos lleva a utilizar el modelo sin restricciones (9546 tokens de una sola palabra, incluidas todas las palabras y sin considerar las frecuencias inversas de los documentos.

5.2. Experimento para reducción de dimensión:

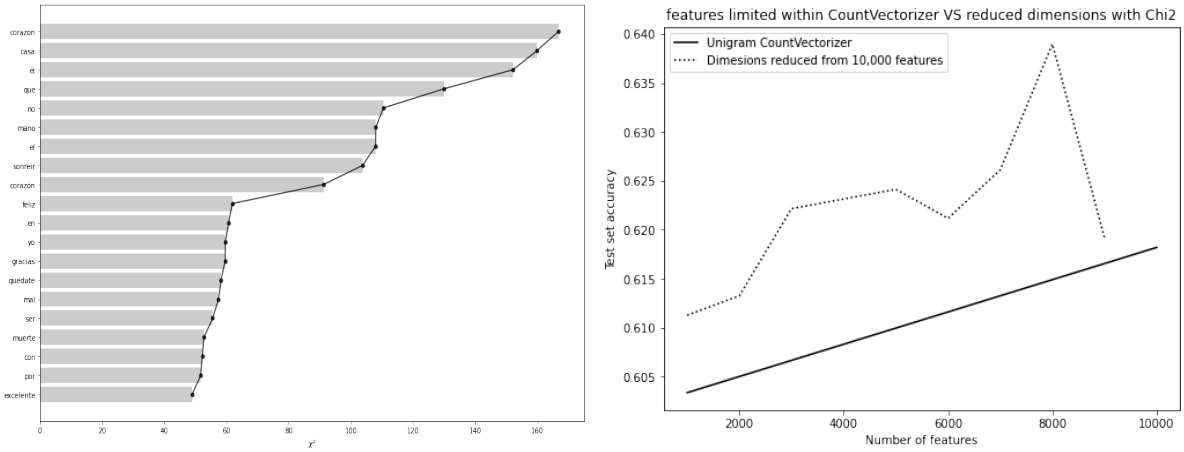
Ahora que los experimentos anteriores bajo el enfoque BoW nos han mostrado que es conveniente utilizar todas las palabras posibles dentro del corpus, nos encontramos que al extraer las características, quedaremos con una matriz de diseño de muy alta dimensión, lo cual podría resultar en algunas complicaciones como podrían ser el costo computacional en la etapa de entrenamiento, por lo que nuestros esfuerzos se dirigen en esta ocasión a abordar este problema. En *Scikit-learn*, disponemos de algunos métodos para reducir el numero de características en matrices dispersas [2] como es nuestro caso (ya que un modelo de Análisis de Componentes Principales no funciona en este tipo de matrices) y algunos otros para matrices densas como los producidos por los modelos que trabajan con embeddings como Doc2Vec, por lo que es de nuestro interés probar algunos de estos métodos con fines comparativos.

[2]Una matriz dispersa es la que tiene ceros en su mayoría

5.2.1. Criterio χ^2 para selección de características

La estadística χ^2 mide el grado de dependencia entre una característica (aquí, un término dentro de un tweet) y una clase (el sentimiento del tweet, ya sea positivo o negativo). A través de una tabla de contingencia, que muestra la distribución de frecuencias, vemos la relación entre un término dentro de un tweet y la clase a la que pertenece el tweet.

Inicialmente, evaluamos este método utilizando el modelo BoW, para esto fue necesario transformar los datos de entrenamiento en vectores de frecuencias, seguido del cálculo de las estadísticas χ^2 entre cada característica y clase. Esta puntuación ayuda a seleccionar la cantidad de características con los valores más altos en relación con las clases. Posteriormente, utilizamos la estadística χ^2 para determinar qué características eran útiles y luego presentamos nuestros hallazgos gráficamente para mostrar qué características de las palabras eran importantes para la predicción. Para visualizar esto, en la Figura 5.2 en el apartado a) se muestran las 20 características mas relevantes seleccionadas con χ^2 entre las cuales algunas nos resultan inesperadas ya que podríamos considerarlas como "palabras vacías"pero el modelo parece encontrarlas útiles en la predicción. Luego reducimos las dimensiones a diferentes números de características y evaluamos el desempeño en la predicción según el conjunto de prueba.



(a) **(b)**

Figura 5.2: Resultados seleccionando características usando χ^2

En a Figura 5.2 en el apartado (b) calculamos el test accuracy para diferentes números de características. Donde se muestra que se obtiene una mayor precisión al seleccionar alrededor de 8000 características basadas en el criterio chi2 en lugar del modelo BoW sin restricciones, pero a pesar de este pequeño aumento en la precisión, en realidad no se está consiguiendo el objetivo principal que era reducir el tamaño de la dimensión, menos características no necesariamente están obteniendo mejores resultados, pero al menos podemos ver que con 3000 chi2 características seleccionadas estaríamos obteniendo un resultado ligeramente superior al obtenido con más de 9000 por el otro modelo. Estos resultados son ilustrativos pero no los que esperábamos conseguir, por lo tanto buscamos otra manera de reducir la dimensión.

5.2.2. Descomposición de Valores Singulares Truncada

Otro método que ofrece *Scikit-learn* para reducir la dimensión en matrices dispersas es la descomposición de valores singulares. Al contrario del método PCA, este estimador no centraliza los datos antes de calcular la descomposición del valor singular, esto significa que puede trabajar con matrices dispersas de manera eficiente, particularmente en el recuento de términos con matrices tf-idf devueltas por los 'vectorizadores' que hemos utilizado. En SVD, la matriz término-documento se descompone en tres matrices: U, σ y V, y se conservan los k valores singulares superiores y sus columnas correspondientes de U y V. Estos vectores singulares retenidos representan efectivamente las características más importantes de los datos del texto. Posteriormente, estos vectores se pueden utilizar como un nuevo conjunto de características reducido de n componentes y con los cuales podemos entrenar modelos.

Al utilizar SVD con esta biblioteca se admiten dos algoritmos: un solucionador SVD aleatorio rápido y un algoritmo 'naive' que utiliza ARPACK[3] como solucionador propio en X * X.T o X.T * X, el que sea más eficiente. Como resultado de aplicar esto, podremos obtener cualquier número deseado de componentes de las características y ver la varianza que logran explicarme cada uno de ellos para luego elegir la menor cantidad posible, dependiendo de su desempeño.

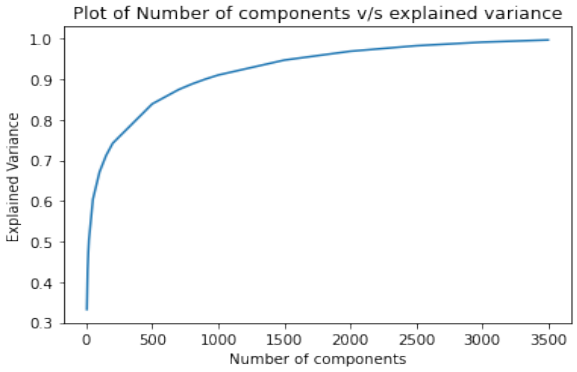


Figura 5.3: a)

n_componentes	accuracy
1000	63.12 %
1500	64.79 %
2000	**65.76 %**
2500	65.51 %
3000	64.83 %

Tabla 5.2: b)

En la Fig. 5.3 Vemos que con al menos 1000 componentes ya se está explicando más del 90 % de la varianza, lo cual es una reducción bastante considerable pero debemos probar si se obtienen buenas predicciones con estas nuevas características, entonces probamos la precisión como lo hemos estado haciendo en experimentos anteriores y podemos ver en la Tabla 5.2 que los mejores resultados se obtienen cuando se usan aproximadamente 2000 componentes, por supuesto, podríamos discutir y analizar más sobre seleccionar el número de componentes a utilizar para la clasificación, pero por ahora lo haremos de esta forma sencilla.

5.3. Experimentos con emnbeddings de palabras

Con el objetivo de sacar el mayor provecho de las representaciones vectoriales por palabra y por documento propuestas, obtuvimos los embeddings de nuestro corpus usando vectores entrenados como se menciono antes, es decir, utilizando los vectores que obtuvimos del *Spanish Billion Word Corpora*. Usando estos keyed vectors de tamaño 300 pudimos representar un

[3] arpack

tweet como un vector más precisamente a través de un modelo Doc2Vec.

A continuación, realizamos un intento por identificar y modelar las frases presentes dentro del corpus, para esto, lo primero fue construir las representaciones 1-grama, 2-grama y 3-grama de los tokens de todos los tweets, pero esta vez con la ayuda de una herramienta proporcionada por la biblioteca *Gensim* llamada 'TaggedDocument' la cual aprende el documento y los embeddings de documento a través de los métodos de memoria distribuida (DM) y la bolsa de palabras distribuida (DBOW) de la forma que describimos anteriormente. Específicamente, para DM utilizamos los algoritmos de entrenamiento alternativos de concatenación de memoria distribuida (DMC) y media de memoria distribuida (DMM) para la generación de vectores de documentos. DMC mejora el modelo DM al concatenar el vector del documento con el promedio de los vectores de palabras de contexto, con el objetivo de capturar tanto la semántica general del documento como el contexto de palabras específicas. Por otro lado, DMM simplifica este proceso promediando directamente los vectores de palabras de contexto sin concatenación. Una vez hecho esto, los comparamos evaluando la precisión en el conjunto de prueba. Estos métodos se probaron tanto por separado como en combinación.

En la Tabla 5.3 podemos ver que los mejores resultados se dan al combinar los modelos DBOW y DMM. Aunque estos resultados no son mejores que los que hemos obtenido hasta ahora, es notable que la representación usando bigramas y trigramas fuera cada vez más efectiva. Potencialmente, esta puede ser una dirección a seguir explorando.

	1-grama	**2-grama**	**3-grama**	**Mejor modelo**
DBOW	60.642 %	59.934 %	60.422 %	60.642 %
DMC	56.893 %	54.387 %	55.713 %	56.893 %
DMM	59.641 %	58.935 %	57.253 %	59.641 %
DBOW+DMC	61.927 %	61.234 %	62.422 %	62.422 %
DBOW+DMM	63.185 %	62.617 %	63.373 %	**63.373** %

Tabla 5.3: Test accuracy para Doc2Vec con modelado de frases

5.4. Búsqueda de Hiperparametros

Habiendo obtenido representaciones del corpus adecuadas para aprender modelos, cambiamos nuestro enfoque a encontrar los mejores hiperparámetros, como la regularización de los coeficientes estimados y el tipo de penalización aplicada. Para hacer esto, probamos diferentes valores potenciales usando una búsqueda sobre una malla predefinida (*Grid Search*), en conjunto con esto realizamos un método de validación cruzada en la búsqueda sobre la malla dividido en 10 bloques a través de un 'k-pliegue' estratificado para identificar los valores óptimos dentro del rango que examinamos en la malla para cada algoritmo de clasificación, de esta forma es posible tener un criterio para la selección del modelo y la optimización de los hiperparámetros que obtengan el mejor rendimiento y precisión en la etapa de predicción de nuevos tweets. Las decisiones al momento de seleccionar se basaron en la optimización de la precisión (accuracy)

Para la regresión logística, exploramos un rango de intensidad de regularización (C) que abarca desde 10^{-5} hasta 10^{2} (o de 0,00001 a 100) en una escala logarítmica. Para los solucionadores[4], consideramos {'newton-cg', 'lbfgs', 'liblinear'}, y para especificar la penalización experimentamos con los valores {'none', 'l1', 'l2', 'elasticnet '}. En el modelo Naïve Bayes se probó con el parámetro de suavizado alfa variando veinte valores de 0 a 1, con ajuste tanto verdadero como falso. Para la implementación de SVM, utilizamos el mismo conjunto de valores de intensidad de regularización que los utilizados en regresión logística. Además, realizamos pruebas utilizando núcleos {'poly', 'rbf', 'sigmoid' } con sus coeficientes predeterminados. Por último, al crear nuestra arquitectura de red neuronal, exploramos de 1 a 5 tamaños de capas ocultas. También probó diferentes funciones de activación, incluidas {'relu', 'tanh', 'logistic'}, y consideró múltiples solucionadores como {'lbfgs', 'sgd', 'adam'}.

En la Tabla 5.4, los parámetros elegidos para cada algoritmo de aprendizaje supervisado se presentan como los óptimos, utilizando el modelo sin restricciones BoW como conjunto de características. Lamentablemente, hay poca variación en el rendimiento cuando se emplean varios valores o tipos de penalización, y solo se muestran ligeras mejoras. Esta observación también es válida para el parámetro de suavizado alfa dentro del modelo Naive Bayes.

]

GridSearchCV (CV=10)		
Modelo	**Hiperparametros probados**	**Valor Óptimo**
Regresión Logística	C :[1e-5, 1e-4, 1e-3, 0.01, 0.1, 1, 10, 1e3], [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9] solver :['newton-cg', 'lbfgs', 'liblinear'] penalty :['none', 'l1', 'l2', 'elasticnet']	C : 0.7, solver : 'lbfgs', penalty : 'l2'
Multinomial Naive Bayes	alpha : 20 valores de 0 a 1 fit_prior : ['true', 'false'] class_prior : [None, [0.35, 0.4, 0.25]]	alpha : 0.85, force_alpha : 'true', class_prior : None
SVM	C : [1e-5, 1e-4, 1e-3, 0.01, 0.1, 1, 10, 1e3] kernel : ['linear', 'poly', 'rbf', 'sigmoid'] gamma : ['scale']	C : 1, gamma : 'scale', kernel : 'poly'
Perceptrón Multicapa	Capas : de 1 a 5 Solver : ['lbfgs', 'sgd', 'adam'] Activación : ['relu', 'tanh', 'logistic']	Capas : 2, Solver : 'adam', Activación : 'relu'

Tabla 5.4: Configuraciones óptimas seleccionadas para cada modelo en función de la optimización del accuuracy mediante grid search y validación cruzada.

[4]algoritmo que encuentra los valores de los parámetros del modelo que mejor se ajustan a los datos.

5.5. Modelos finales y Resultados

Finalmente, ajustamos los algoritmos de aprendizaje supervisado utilizando los hiperparámetros encontrados con la búsqueda en cuadrícula utilizando como características las representaciones BoW reducidas y sin restricciones, así como el Doc2Vec con DBOW+DMM usando '3-gramas'.

La tabla 5.5 informa los resultados de los modelos de clasificación obtenidos en términos de exactitud, precisión, recuperación y puntuación micro F1 utilizando una validación cruzada de 10 veces. Además, los resultados de ajustar un modelo KNN adicional con k = 5 (que no estima coeficientes) solo se incluyeron como punto de referencia para el rendimiento de los modelos que analizamos. Este modelo no paramétrico se incluyó con fines de análisis empírico. Aunque no podemos afirmar que los modelos paramétricos sean más eficaces, las primeras observaciones parecen indicar que tienen más éxito en esta tarea.

BoW sin restricciones					
	KNN	SVM	MNB	Logistic	MLP
Accuracy	59.75 %	62.31 %	62.03 %	64.26 %	63.51 %
Precision	60.08 %	62.25 %	63.24 %	64.39 %	63.92 %
Recall	59.76 %	62.31 %	62.03 %	64.25 %	63.51 %
F1-score	58.78 %	61.96 %	60.78 %	63.61 %	62.59 %
BoW reducido con DVS					
	KNN	SVM	MNB	Logistic	MLP
Accuracy	60.88 %	64.98 %	64.19 %	**67.12** %	68.84 %
Precision	61.40 %	64.91 %	62.27 %	66.91 %	67.25 %
Recall	60.88 %	64.98 %	64.19 %	67.12 %	68.84 %
F1-score	59.92 %	64.02 %	62.98 %	*66.24* %	67.90 %
Doc2Vec con DBOW+DMM					
	KNN	SVM	MNB	Logistic	MLP
Accuracy	56.47 %	62.56 %	62.96 %	63.68 %	64.31 %
Precision	54.76 %	63.81 %	63.54 %	61.94 %	60.81 %
Recall	56.47 %	62.56 %	62.96 %	63.68 %	62.31 %
F1-score	42.93 %	57.59 %	63.88 %	61.55 %	62.11 %

Tabla 5.5: Resultados obtenidos al entrenar los algoritmos de clasificación en los tres conjuntos de características.

Para las bibliotecas de caja negra, no hay necesidad de transformaciones matriciales ni división de datos; la evaluación se realiza sobre el conjunto completo porque no es necesario entrenar un algoritmo. Solo necesitamos preprocesar el corpus. Esto es especialmente importante para bibliotecas como *VADER*, que necesitan hacer coincidir la mayor cantidad de palabras posible para obtener mejores predicciones. Lo mismo aplica para *PySentimiento* ya que no se recomienda utilizar palabras lematizadas (de hecho, para esta biblioteca es más recomendable el procesamiento bajo). Vader tiene un coste computacional bajo, mientras que *PySentimiento* requiere un poco de tiempo para calcular los resultados, pero la ventaja es que no se necesitan

conocimientos previos.

La tabla 5.6 muestra los resultados del análisis de sentimiento utilizando las bibliotecas de caja negra. Podemos ver una diferencia sustancial en todas las métricas de rendimiento entre *TextBlob, VADER y Pysentimiento*. Dado que *TextBlob* se basa en un enfoque basado en patrones y léxico y *VADER*, que está diseñado para textos de redes sociales, utiliza un sistema basado en reglas con un léxico de sentimientos para capturar tanto la polaridad como la intensidad; Ambos no logran captar los matices del idioma español. *PySentimiento* está diseñado específicamente para textos en español. Emplea modelos de aprendizaje automático entrenados en grandes corpus para clasificar sentimientos y detectar emociones, ofreciendo un enfoque más matizado y consciente del contexto en comparación con los métodos basados en reglas de *TextBlob* y *VADER*, particularmente en el dominio del idioma español.

	TextBlob	**Nltk Vader**	**PySentimiento**
accuracy	51.23 %	58.07 %	**68.89** %
precision	55.45 %	58.60 %	72.20 %
recall	51.23 %	57.19 %	52.81 %
F1-score	52.92 %	56.42 %	60.38 %

Tabla 5.6: Resultados obtenidos por las bibliotecas de caja negra

La tabla 5.7 muestra la evaluación de los modelos BERT españoles, BETO, BerTin-base y roBERTa. Todos ellos habían sido formados previamente en español, este último específicamente para redes sociales (robertuito). Empleamos sus pesos previamente entrenados para tokenizar el texto utilizando los tokenizadores correspondientes para cada modelo. Además, utilizamos la biblioteca *PyTorch* con un tamaño de datos por lotes de 16 y probamos con el modelo BoW sin restricciones, ya que los modelos previamente entrenados han demostrado funcionar mejor con todas las palabras presentes en el tweet.

	BETO-uncased	**roBERTa-sentiment**	**BerTin-base**
training accuracy	96.20 %	97.54 %	96.91 %
validation accuracy	**73.26** %	71.88 %	72.14 %
validation loss	0.3945	0.2847	0.2141

Tabla 5.7: Resultados de clasificación de los modelos BERT en español

Hemos definido una función de pérdida basada en Entropía Cruzada Categórica que mide la discrepancia entre la distribución de probabilidad predicha y la distribución de probabilidad real de las etiquetas, de esta manera podemos calcular la pérdida en cada iteración del bucle, luego la pérdida calculada es se utiliza para realizar retropropagación de gradiente para ajustar los parámetros del modelo durante el entrenamiento. También se definió un abandono para regularizar el modelo y evitar el sobreajuste y evitar que las neuronas se vuelvan demasiado dependientes unas de otras durante el entrenamiento, de esta manera, se crea una capa de abandono que especifica la probabilidad de que una neurona se desactive durante el entrenamiento. el entrenamiento. En este caso, se establece en 0,3, lo que significa que cada neurona tiene un 30 % de probabilidad de ser desactivada durante cada paso de entrenamiento.

El ajuste fino de los modelos previamente entrenados produjo los mejores resultados. Sin embargo, cuantas más épocas usamos para el entrenamiento, más sobreajuste observamos, como se puede ver en la Tabla 5.8, donde mostramos la precisión en los conjuntos de entrenamiento y prueba cuando entrenamos en 3, 5 y 10. épocas. Este aumento en la varianza no es tan fácil de interpretar como el error de entrenamiento de modelos anteriores, donde podemos observar mejor las tasas de error.

Épocas	Train accuracy	Test accuracy	Validation Loss
3	92.89 %	70.33 %	0.4554
5	96.20 %	**73.26** %	0.3945
10	97.12 %	72.76 %	0.3161

Tabla 5.8: Resultados de incrementar épocas utilizando BETO

Finalmente, la Tabla 5.9 compara las mejores puntuaciones de precisión de varios modelos de sentimiento evaluados en nuestro estudio. En particular, la tabla muestra que los modelos basados en BERT funcionan mejor que los modelos basados en reglas o de aprendizaje automático. Estos niveles de precisión incluso se logran con la biblioteca *Pysentimiento* lista para usar. Entre ellos, BETO-uncased es el modelo más preciso, con una puntuación de precisión del 73,26 %. Este desempeño de los modelos basados en BERT puede deberse al hecho de que el modelo logra una comprensión profunda del contexto y los matices del discurso en español mexicano, adquirida a través de un entrenamiento previo extenso en una variedad de corpus de textos. Esto les permite capturar las expresiones de sentimiento y estructuras lingüísticas complejas de los tweets relacionados con COVID-19. Por el contrario, los modelos basados en reglas como *Vader* y *TextBlob* exhiben las puntuaciones de precisión más bajas en nuestro análisis, lo que refleja su limitación para adaptarse a la estructura y el dominio lingüístico específicos.

Modelo	**accuracy**
TextBlob	51.23 %
Nltk Vader	58.07 %
Pysentimiento	68.89 %
SVM	64.89 %
Naive Bayes	62.22 %
Regresión Logística	67.12 %
MLP	68.84 %
BETO-uncased	**73.26 %**
roBERTa-sentiment	71.88 %
BerTin-base	72.14 %

Tabla 5.9: Resumen del desempeño en accuracy de diferentes modelos de análisis de sentimiento en el corpus SENT-COVID.

5.6. Análisis de Errores

Los modelos previamente entrenados han dado los mejores resultados como era de esperar, pero podemos ver que presentan algunas desventajas como el gran costo computacional, el tiempo de entrenamiento y el sobreajuste que este presenta conforme más épocas se utilizan, este aumento de la varianza es menos interpretativo que el error de entrenamiento de los modelos tradicionales, donde podemos tomar una mejor visión de las tasas de error. Debemos entonces echar un vistazo a los términos del error (residuales) para poder explorar y comprender mejor la varianza presente en las predicciones, esto nos lleva a identificar ciertos aspectos y areas de oportunidad que en un análisis profundo permitan mejorar los resultados, pero consideraremos un breve análisis y comentarios al respecto para una mayor investigación.

Un análisis de errores es una práctica muy valiosa para comprender mejor el rendimiento de los modelos de aprendizaje automático, comenzamos tomando las predicciones de los modelos de regresión logística y perceptrón multicapa (ya que estos fueron los que obtuvieron los mejores resultados) para ver sus tasas de error en los conjuntos de entrenamiento y prueba, estas medidas se usan para evaluar el rendimiento del modelo durante el entrenamiento y evaluación respectivamente.

Modelo	**Train Error Rate**	**Test Error Rate**
Regresión Logística	19.51 %	34.12 %
Perceptrón Multicapa	16.17 %	32.62 %

La tasa de error de entrenamiento indica la tasa de error del modelo en el conjunto de datos de entrenamiento, es el error que el modelo comete al predecir las etiquetas de las muestras que ya ha visto, una tasa de error baja sugiere que el modelo puede ajustarse bien a los datos de entrenamiento, mientras que la tasa de error de prueba es análoga pero en el conjunto de datos de prueba y esta mide la capacidad de generalización del modelo, es decir, su capacidad para realizar predicciones precisas en datos que no ha visto antes.

5.6.1. Matrices de Confusión

Las matrices de confusión nos ayudan a evaluar el rendimiento de los modelos y proporcionan una representación visual de la precisión en las predicciones y de cómo el modelo clasifica las clases de manera correcta o incorrecta. Las filas representan las clases reales, mientras que las columnas representan las clases predichas por el modelo, cada valor de la matriz muestra el número (o proporción) de tweets que pertenecen a una determinada combinación de clase real y predicha, lo que nos brinda una rápida identificación de los aciertos y errores del modelo. Las matrices de confusión tienen además la capacidad para proporcionar información detallada sobre el rendimiento del modelo en las diferentes clases, ya que también permiten calcular medidas de rendimiento como la precisión, la sensibilidad y la especificidad, las matrices de confusión permiten identificar patrones de errores específicos, como la confusión entre clases similares o la presencia de clases desbalanceadas.

En las figuras 5.4 y 5.5 analizamos las matrices de confusión obtenidas para nuestros modelos de regresión logística y perceptrón multicapa, las primeras son utilizando el enfoque no restringido de características, mientras que en las dos subsecuentes son utilizando reducción

de dimensión mediante descomposición de valores singulares.

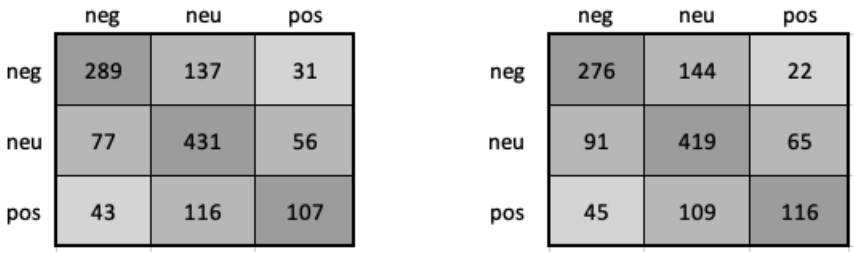


Figura 5.4: Matrices de modelos LR y MLP con BoW no restringido

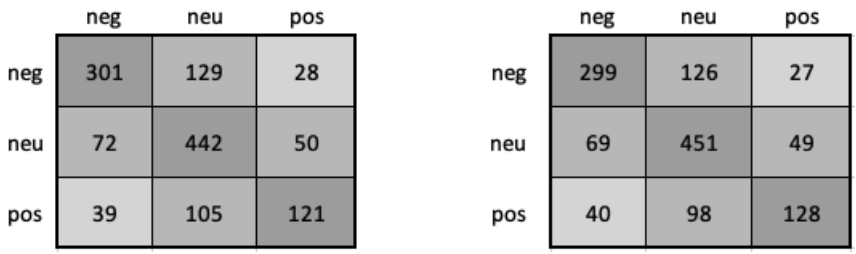


Figura 5.5: Matrices de modelos LR y MLP con DVS

Las matrices nos muestran que el modelo tiene dificultades para identificar la clase neutral, ya que en ambos modelos clasifico como neutro mas de cien tweets positivos y mas de cien negativos. También pudimos observar que suele confundir con relativa frecuencia un tweet neutro con alguno entre negativo y positivo, pero pocas veces confundió uno positivo con uno negativo o viceversa, lo cual es un buen indicador, por lo tanto, identificamos la dificultad para predecir correctamente los tweets neutros como la mayor área de oportunidad presente.

5.6.2. Análisis de residuales

El análisis de errores utilizando residuales proporciona una comprensión más profunda de la variabilidad en las predicciones de los modelos y ayuda a identificar áreas para mejorar su rendimiento en futuras iteraciones. Cuando se trabaja con clases categóricas, calcular residuales directamente como la diferencia entre el valor real y el valor predicho no es adecuado, ya que estas clases no representan valores numéricos continuos, en lugar de eso, se realiza el análisis de residuales considerando la probabilidad de predicción para cada clase y para ello, existen técnicas como el análisis de residuales de Pearson, el cual es una medida de la discrepancia entre los datos observados y las predicciones de un modelo estadístico, los residuales de Pearson representan la discrepancia entre las probabilidades observadas y las predichas para cada clase.

Para llevar a cabo este análisis se uso como referencia solo los resultados del modelo de regresión logística, calculando con el las probabilidades de pertenencia a cada clase en cada uno de los tweets del conjunto de datos de prueba y posteriormente calculando los residuales de Pearson para cada clase, estos residuales se calculan como la diferencia entre valores observados y valores predichos, dividida por la raíz cuadrada del valor predicho multiplicado por uno menos el valor predicho. Este tipo de análisis es útil para evaluar la calidad del ajuste

de un modelo, si los residuales de Pearson son grandes en magnitud, puede indicar que el modelo no se ajusta bien a los datos, por otro lado, si los residuales de Pearson son pequeños, puede indicar que el modelo se ajusta bien a los datos.

Una vez obtenidos los residuales, la Figura 5.6 muestra una gráfica de violin donde podemos visualizar la distribución de los residuales, esto permite identificar patrones en los errores de predicción y entender mejor la varianza en las predicciones.

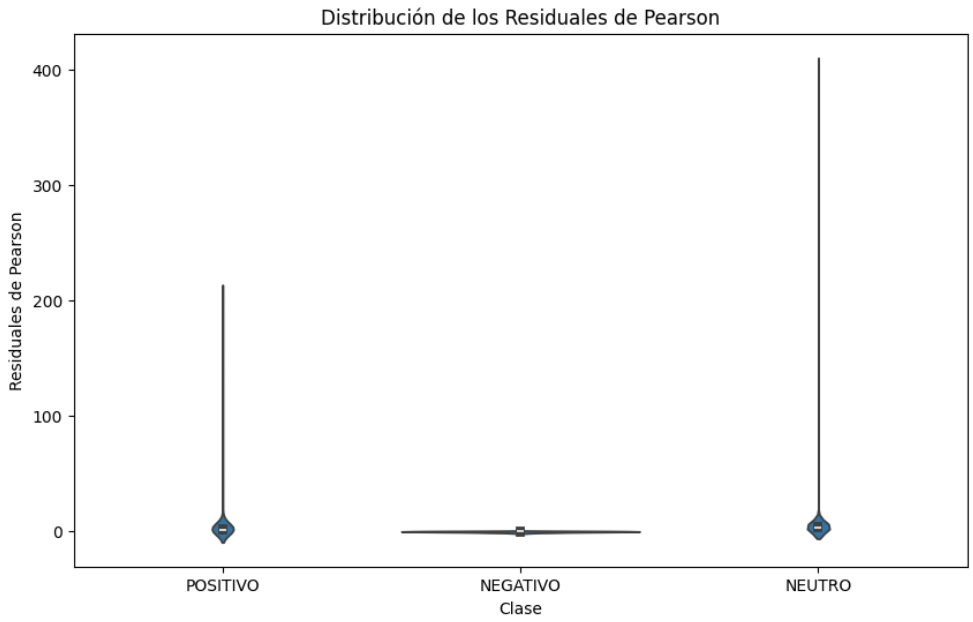


Figura 5.6: Residuales de Pearson para cada clase.

Si los residuales para una clase están simétricamente distribuidos alrededor de cero, esto indica que el modelo está haciendo un buen trabajo al predecir. Los residuos positivos y negativos se cancelan entre sí, lo que indica que el modelo no está sistemáticamente sobrestimando o subestimando la probabilidad de esa clase. Por otro lado si los residuales de una clase están sesgados hacia los valores positivos o negativos, indica que el modelo está sistemáticamente sobrestimando o subestimando la probabilidad de esa clase. Un sesgo positivo indica que el modelo está subestimando la probabilidad, mientras que un sesgo negativo indica que el modelo está sobrestimando la probabilidad. Finalmente si los residuos tienen muchos valores extremos, esto indica que el modelo está teniendo dificultades para predecir esa clase, los valores extremos pueden ser causados por outliers en los datos o por una clase que es intrínsecamente difícil de predecir.

En nuestro caso los residuos para la clase negativa están aplanados con una línea horizontal en cero, esto indica que el modelo está haciendo un buen trabajo al predecir esa clase y significa que hay una pequeña discrepancia entre las probabilidades observadas y predichas. Al mismo tiempo vemos lineas verticales en las clases positiva y neutra, esto indica que los residuos varían y que hay una gran discrepancia entre las probabilidades observadas y las predichas, por lo tanto sugiere que el modelo no está haciendo predicciones precisas.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones finales

El presente estudio ha explorado diversos enfoques y modelos en el análisis de sentimientos, desde métodos tradicionales hasta técnicas avanzadas basadas en aprendizaje profundo. A partir de los experimentos realizados y los resultados obtenidos, se pueden extraer varias conclusiones que nos permitieron contestar las preguntas de investigación planteadas al inicio.

Podemos concluir que a pesar del tamaño relativamente pequeño de nuestro conjunto de datos, se obtuvieron resultados alentadores, que ademas permitieron indentificar las fortalezas y debilidades de los modelos propuestos. De esta forma pudimos comprobar que los modelos pre entrenados fueron los que lograron los mejores resultados en la prediccion, aunque estos requirieron un alto costo computacional, mientras que los modelos tradicionales potenciados por las tecnicas de seleccion de caracteristicas (como DVS) obtuvieron resultados no tan distantes a un costo computacional mucho menor, sin embargo los modelos basados en reglas resultaron ser los menos eficientes, posiblemente debido a que es necesario contar un lexicon mas grande y mas trabajado, ya que los utilizados en este trabajo no cumplian con esa caracteristica. Por otro lado, tambien se llego a la conclusion que las tecnicas de embeddings de palabras y modelado de frases no resultaron ser lo suficientemente efectivas para superar los resultados de los modelos tradicionales (tanto con BoW no restringido como con seleccion de caracteristicas) a pesar que estos supusieron tambien ser mas costosos en cuanto a recursos.

Otro aspecto importante a destacar de los resultados obtenidos, es que a los modelos tienen mayor dificultad para encontrar la clase correcta debido a que un tweet es un texto muy pequeño, esto en contraste con los resultados obtenidos en algunos de los trabajos relacionados donde se utilizaba otro tipo de texto, como lo son reseñas de productos, los cuales son muestras de texto mas extensas y que proporcionan mas caracteristicas al momento de la clasificacion. Esto mismo pudimos verlo en nuestros experimentos para la seleccion del vocabulario, donde a pesar de nuestros intentos por usar tfidf, remover palabras vacias, entre otros, los modelos mostraron ser mas efectivos cuantas mas carcteristicas eran incluidas, es decir a mayor cantidad de datos de entrenamiento. Misma conclusion obtuvimos al probar el modelo perceptron, mejores resultados fueron obtenidos a medida que se incrementaba el numero de capas, lo que se intepreta como mayor costo computacional. Por lo que podemos concluir sin mucha sorpresa, que la forma mas simple de mejorar las predicciones, es contar con mas recursos como lo son mayor cantidad de datos de entrenamiento y mayor poder computacional.

6.2. Trabajo futuro

A partir de los resultados obtenidos y las limitaciones identificadas en el presente estudio, se abre la puerta a exploraciones y en diferentes aspectos, algunas áreas potenciales para el trabajo futuro que se plantean son las siguientes:

- **Ampliación del corpus:** Los resultados demostraron que a mayor cantidad de datos, en el entrenamiento, mayor precisión en las pruebas, el tamaño del corpus SENTCOVID utilizado, aunque adecuado para los experimentos iniciales, podría mejorarse con una mayor cantidad de datos. Aumentar el corpus, tanto en volumen como en diversidad temática, permitiría entrenar modelos más robustos y con mejor capacidad de generalización. Esto incluiría la incorporación de mas tweets, mejor seleccionados en contenido, lo que enriquecería nuestro análisis de sentimientos en un contexto general.

- **Combinación de análisis de sentimientos con otras técnicas de PLN:** La combinación de este análisis con otras técnicas de PLN no exploradas en este estudio, como el análisis de temas o la extracción de entidades nombradas, podría proporcionar una visión más rica y completa del significado del discurso público en torno al tema de interés.

- **Mejora del análisis de errores y residuales:** El análisis de errores realizado en este estudio ha permitido identificar áreas clave en las que los modelos tienen dificultades para hacer predicciones correctas, especialmente en la clasificación de sentimientos ambiguos o irónicos. Sin embargo, el análisis de residuales podría ser profundizado en investigaciones futuras mediante la implementación de técnicas más avanzadas que permitan una mejor interpretación de los fallos. Una posible línea de trabajo futuro sería explorar métodos más detallados de análisis de errores, como la descomposición de errores por tipo de sentimiento (positivos, negativos, neutrales) y por estructuras lingüísticas complejas, tales como el sarcasmo, la ironía y las negaciones. Esto podría lograrse mediante la aplicación de técnicas de análisis de errores de clase, que permitirían observar en qué clases específicas (emociones o polaridades) los modelos tienden a cometer más errores.

- **Incorporación de etiquetas de emoción:** Actualmente, el análisis se realizó utilizando tres categorías principales (positivo, negativo y neutral). Sin embargo, el análisis emocional podría refinarse incorporando categorías adicionales de emociones como sorpresa, disgusto, ira o miedo. Modelos de aprendizaje profundo preentrenados en conjuntos de datos con etiquetas emocionales más detalladas permitirían un análisis más granular que permitiria un analisis de sentimientos mas preciso.

- **Uso de técnicas avanzadas de ajuste de hiperparámetros:** En los experimentos realizados, el ajuste de hiperparámetros se basó en técnicas clásicas como la búsqueda grid search o búsqueda aleatoria. En trabajos futuros, podrían implementarse técnicas más avanzadas como la optimización bayesiana o el uso de algoritmos genéticos para mejorar la selección de hiperparámetros. Esto podría aumentar la precisión de los modelos y y adicionalmente también reducir el tiempo de entrenamiento.

Bibliografía

[1] Bertin: Serie de modelos basados en bert en español. `https://huggingface.co/bertin-project`, 2021. El centro de modelos actual apunta a lo mejor de los modelos basados en RoBERTa entrenados desde cero en la parte española de mC4 usando Flax.

[2] C. C. Aggarwal and C. K. Reddy. *Data Clustering: Algorithms and Applications*. Chapman and Hall/CRC, 2014.

[3] E. AI. spacy: Industrial-strength natural language processing in python. `https://spacy.io`, 2015. Accessed: 2024-06-01.

[4] J. Bennett and S. Lanning. The netflix prize. *Proceedings of KDD Cup and Workshop*, 2007.

[5] S. Bird and E. Loper. Nltk: Natural language toolkit, 2000. Desarrollado en la Universidad de Pennsylvania a principios de la década de 2000.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[7] J. Canete, G. Chaperon, R. Fuentes, and J. Pérez. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*, 2020.

[8] G. G. Chowdhury. *Natural Language Processing*, volume 37. Annual Review of Information Science and Technology, 2003.

[9] F. Colas, N. Courty, S. Rakotomamonjy, and J. Atif. A comparison between svm and some older classification algorithms in a machine learning task. *Proceedings of the International Conference on Machine Learning and Data Mining in Pattern Recognition*, 4:18–32, 2006.

[10] D. Cournapeau. Scikit-learn: Machine learning in python. `https://scikit-learn.org/`, 2007. Accessed: 2024-09-10.

[11] J. C. de Albornoz, L. Plaza, and P. Gervás. SentiSense: An easily scalable concept-based affective lexicon for sentiment analysis. In N. Calzolari, K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3562–3567, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[14] C. Esteban, A. Lundervold, and S. Aase. Diagnóstico asistido por computadora en radiología: donde nos encontramos y adónde vamos. *Radiología Académica*, 22(4):506–510, 2015.

[15] D. Esteban. *Aplicaciones del Machine Learning en Medicina Diagnóstica*. Editorial Médica, 2015.

[16] M. García. Pysentimiento: Análisis de sentimientos simplificado para python, 2021. Versión 0.3.0.

[17] Y. Goldberg. *Neural network methods for natural language processing*, volume 10. Morgan & Claypool Publishers, 2017.

[18] J. Hirschberg and C. D. Manning. Advances in natural language processing. *Science*, 349(6245):261–266, 2015.

[19] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

[20] C. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.

[21] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson, 3rd edition, 2019.

[22] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 8:30–37, 2009.

[23] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31(3):249–268, 2007.

[24] A. Kumar and P. Jain. *Natural Language Processing: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2020.

[25] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[26] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. *European Conference on Machine Learning*, pages 4–15, 1998.

[27] B. Liu. *Sentiment Analysis and Opinion Mining*, volume 5 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers, 2012.

[28] S. Loria. Textblob: Simplified text processing. `https://textblob.readthedocs.io/en/dev/`, 2018. Accedido: 2024-06-01.

[29] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[30] T. McEnery and A. Hardie. Corpora. In P. Baker and T. McEnery, editors, *A Glossary of Corpus Linguistics*, pages 7–28. Edinburgh University Press, 2012.

[31] M. L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(144):276–282, 2012.

[32] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[33] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[34] B. Pang and L. Lee. *Opinion Mining and Sentiment Analysis*, volume 2 of *Foundations and Trends in Information Retrieval*. Now Publishers Inc., 2008.

[35] N. d. A. Perez. Robertuito: Un modelo de lenguaje para contenido generado por usuarios en español, entrenado siguiendo las pautas de roberta en 500 millones de tweets. *Nombre del Journal*, volumen del journal, si aplica(número del journal, si aplica):páginas del artículo, si aplica, 2021. RoBERTuito viene con y sin caja.

[36] R. Plutchik. *The Emotions*. University Press of America, 1991.

[37] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018. Technical report, OpenAI.

[38] P. D. Team. Pytorch: An open source machine learning framework. `https://pytorch.org/`, 2016. Accessed: 2024-06-01.

[39] A. Vaswani and et al. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017.

[40] R. Řehůřek. Gensim: Topic modelling for humans. `https://radimrehurek.com/gensim/`, 2009. First released 2009.