Pontificia Universidad Católica del Ecuador

Facultad de Ingeniería

TEMA:

Desarrollo de un modelo de análisis de sentimientos enfocado en la encuesta de evaluación docente de una institución educativa superior.

AUTOR:

JAIME ALEJANDRO GOVEA SOTOMAYOR

TRABAJO PREVIA A LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN SISTEMAS DE INFORMACIÓN MENCIÓN DATA SCIENCE

TUTOR: MGS. CHARLES EDISSON ESCOBAR TERAN

Quito, marzo – 2025

DEDICATORIA

Dedico este trabajo a mi familia, quiénes han sido mi apoyo por varios años y cuya ayuda ha sido fundamental para mi todo este tiempo, a mis amigos y amigas quiénes han estado ahí en todo momento y todas las personas que han generado un impacto en mi vida para ser la persona que soy ahora.

AGRADECIMIENTO

Quiero agradecer a las personas que me han apoyado durante el desarrollo del presente trabajo de titulación, a mi asesor de tesis y a todos los docentes del programa de maestría por su guía y conocimientos impartidos a lo largo de este trayecto. A mi familia por su apoyo a lo largo de estos años que me ha permitido llegar a esta posición. También a mis compañeros de trabajo quiénes me ayudaron con sus conocimientos en mis momentos de duda y a mis amigos y amigas quiénes fueron un punto de apoyo en los momentos difíciles.

RESUMEN

El presente trabajo de titulación aborda el desarrollo, evaluación y comparación de distintos modelos de análisis de sentimientos diseñados específicamente para procesar las respuestas abiertas en las encuestas de evaluación docente de una institución de educación superior. Los modelos desarrollados utilizaron los algoritmos de machine learning Naive Bayes y Random Forest con las técnicas de vectorización Bag of Words y Tf-Idf respectivamente. El tercer modelo se desarrolló utilizando la técnica de vectorización Word2Vec y una arquitectura de una red neuronal recurrente del tipo Long Short-Term Memory.

Se aplicó la misma metodología para el entrenamiento de los tres modelos, se realizó un proceso de limpieza, tokenización y lematización de los datos. Luego, se llevó a cabo la vectorización de los datos preprocesados para poder aplicar los algoritmos de clasificación, entrenar los modelos y finalmente evaluarlos sobre un conjunto de datos de prueba. Este enfoque de procesamiento de lenguaje natural posibilita la transformación de datos cualitativos en métricas cuantificables, clasificando automáticamente las opiniones de los estudiantes en tres categorías principales: positivas, negativas y neutras.

El objetivo principal de esta investigación es optimizar el proceso de evaluación docente, proporcionando a los coordinadores de carrera una herramienta que reduzca significativamente el tiempo dedicado al análisis de respuestas abiertas. Además, el modelo busca minimizar la subjetividad inherente a la interpretación manual de las evaluaciones, ofreciendo un método más sistemático y objetivo para la valoración del desempeño docente.

Los resultados obtenidos demuestran la viabilidad y efectividad del uso de modelos de clasificación para procesar grandes volúmenes de respuestas textuales, proporcionando análisis consistentes y objetivos que facilitan la labor administrativa en la evaluación del personal docente.

Palabras clave: *Análisis de sentimientos, evaluación docente, procesamiento de lenguaje natural, vectorización, machine learning, redes neuronales*

ABSTRACT

This thesis addresses the development, evaluation, and comparison of different sentiment analysis models specifically designed to process open-ended responses in teaching evaluation surveys at a higher education institution. The developed models used Naive Bayes and Random Forest machine learning algorithms with Bag of Words and Tf-Idf vectorization techniques respectively. The third model was developed using Word2Vec vectorization and a Long Short-Term Memory recurrent neural network architecture.

The same methodology was applied for training the three models, performing a process of cleaning, tokenization, and lemmatization of the data. Then, the vectorization of the preprocessed data was carried out to apply classification algorithms, train the models, and finally evaluate them on a test dataset. This natural language processing approach enables the transformation of qualitative data into quantifiable metrics, automatically classifying student opinions into three main categories: positive, negative, and neutral.

The main objective of this research is to optimize the teaching evaluation process, providing program coordinators with a tool that significantly reduces the time spent analyzing open-ended responses. Additionally, the model seeks to minimize the subjectivity inherent in manual interpretation of evaluations, offering a more systematic and objective method for assessing teaching performance.

The results obtained demonstrate the viability and effectiveness of using classification models to process large volumes of textual responses, providing consistent and objective analysis that facilitate administrative work in the evaluation of teaching staff.

Keywords: *Sentiment analysis, teacher evaluation, natural language processing, vectorization, machine learning, neural networks*

Tabla de contenido

Lista de tablas

Tabla de ilustraciones

CAPITULO I: INTRODUCCIÓN

1.1 Planteamiento del problema

En la institución de educación superior que fue objeto de estudio del presente trabajo de titulación se aplica la encuesta de evaluación docente como herramienta de medida de satisfacción de sus aproximadamente 18000 estudiantes. Este proceso se lleva a cabo 2 veces por semestre académico a través de sus plataformas digitales, dicha evaluación contiene en su mayoría preguntas de opción múltiple y una pregunta abierta al final donde el estudiante puede dar su retroalimentación sobre cada uno de sus docentes.

A pesar de que la encuesta de evaluación docente se lleva a cabo por medios digitales, las respuestas de preguntas abiertas no se pueden tabular de la misma manera que los resultados de preguntas de opción múltiple, ya que necesitan ser interpretadas individualmente. La revisión manual de los resultados de la encuesta es un trabajo largo y arduo debido al gran volumen de estudiantes que posee la institución y la periodicidad con las que se lleva a cabo la encuesta, esto causa que las autoridades pertinentes tarden en tomar acciones apropiadas con respecto a las insatisfacciones de los estudiantes con los docentes y en consecuencia se puede dar una deserción de los alumnos.

Además, como todo proceso manual se está sujeto a errores humanos y sesgos que pueden llevar a la toma de decisiones equivocada, por lo que, el uso de un modelo de análisis de sentimiento sería capaz de proveer un análisis mucho más rápido y objetivo, que permita a la institución tomar decisiones informadas en tiempos oportunos.

1.2 Objetivos

1.2.1 Objetivo General

Generar un modelo de análisis de sentimientos que permita clasificar automáticamente los resultados de las preguntas abiertas de la encuesta de evaluación docente de la institución de educación superior seleccionada, con la finalidad de facilitar la interpretación y comprensión de las respuestas cualitativas proporcionadas por los estudiantes.

1.2.2 Objetivos Específicos

- Recolectar las respuestas de la encuesta de evaluación docente.
- Realizar una limpieza y preprocesamiento de los datos recolectados.
- Entrenar y evaluar el modelo de análisis de sentimientos.

1.3. Justificación

La encuesta de evaluación docente es una herramienta de mucha utilidad para evaluar la satisfacción de los estudiantes con respecto a las clases que reciben, pero como lo indican Hajrizi y Pireva (2020), esto se vuelve un problema cuando tratamos con grandes volúmenes de datos, ya que se vuelve imposible o muy ineficiente que una persona se siente a analizar un enorme volumen de datos y llegar a la conclusión si el sentimiento general es positivo, neutro o negativo. Por lo que el presente trabajo propone automatizar este proceso mediante un modelo de análisis de sentimientos que sea capaz de analizar esta enorme cantidad de datos y clasificarlos en un tiempo oportuno de manera que sea sencillo de interpretar

1.4. Contextualización del tema u objeto

En la institución educativa superior se utiliza la encuesta de evaluación docente dos veces por semestre, la primera se realiza al terminar el primer parcial como una encuesta de alerta temprana, para tomar acciones de manera más rápida ante cualquier problema que pueda surgir con alguno de los docentes, la segunda encuesta se lleva a cabo al final del semestre, como evaluación del docente por parte de los estudiantes.

Estas encuestas cuentan con dos tipos de preguntas, de opción múltiple y preguntas abiertas, el primer tipo de pregunta es fácil de cuantificar y procesar al tratarse de valores numéricos, pero las preguntas abiertas son más complicadas de analizar, ya que no están estandarizadas, la respuesta de cada estudiante es diferente, cada una con distintas opiniones respecto a los docentes, esto conlleva a una mayor cantidad de esfuerzo para lograr interpretar la satisfacción general de los estudiantes.

Esperar que una persona o grupo de personas analicen estas respuestas de manera manual se vuelve un problema al tomar en cuenta que hay un aproximado

de 18000 estudiantes y la encuesta se lleva a cabo dos veces por semestre, como lo indican Hajrizi y Pireva (2020), se vuelve sumamente ineficiente realizarlo de forma manual a esta escala. Por lo que el uso de un modelo de análisis de sentimientos mejoraría este proceso al clasificar las respuestas de manera automatizada, liberando el tiempo del personal y facilitando su comprensión.

1.5 Alcance

En el presente trabajo de titulación se utilizarán técnicas de machine learning y Deep learning para entrenar varios modelos de análisis de sentimientos, evaluarlos y determinar cuál es el más eficiente. Los modelos estarán enfocados en la encuesta de evaluación docente de una institución educativa superior, la cual contiene un componente de preguntas abiertas al final de cada encuesta, estas respuestas serán utilizadas para el entrenamiento y prueba del modelo. El objetivo final de este trabajo es desarrollar un modelo que pueda clasificar las respuestas de los estudiantes en las categorías positivo, neutro y negativo, de manera que sea más sencillo cuantificar la percepción de los estudiantes ante el docente en un tiempo apropiado.

CAPITULO II: MARCO TEÓRICO Y CONCEPTUAL

2.1 Antecedentes o marco referencial

El análisis de sentimientos ha sido aplicado en la educación de varias maneras, como lo indica Bhalla (2022), algunos investigadores como Gutierrez et. al (2018) tomaron como muestra tres grupos de estudiantes de ingeniería y los encuestaron para recolectar sus datos, otros como Altrabsheh et. al (2013) propusieron realizar un web scrapping de redes sociales como Twitter y Facebook para entrenar un modelo de análisis de sentimientos acerca de las opiniones de los estudiantes con respecto a sus instituciones.

En el artículo desarrollado por Gutierrez et. al (2018) nos indican que su proyecto constó de tres fases, la primera fue la recolección y limpieza de datos, en donde encuestaron a 3 cursos de estudiantes de ingeniería en sistemas y obtuvieron 1040 comentarios, procedieron a limpiar los datos, eliminando los signos de puntuación y las palabras vacías para después asignar una puntuación numérica a cada comentario en el rango de -2 a 2 puntos, siendo entre -2 y -0.2 el sentimiento negativo, de 0.2 a 2 el sentimiento positivo y el resto fue considerado neutral. En la segunda etapa realizaron la extracción de características y en la tercera etapa procedieron a entrenar el modelo, tomaron dos tercios del data set como datos de entrenamiento y un tercio como datos de prueba, utilizaron el algoritmo Support Vector Machines y consiguieron una exactitud del 80%.

Por otro lado, en el trabajo llevado a cabo por Altrabsheh et. al (2013) propusieron un proyecto en el cual los estudiantes darían su opinión acerca de las clases y sus docentes a través de redes sociales, luego se recolectaría los datos mediante un web scrapper para ser limpiados, esto implica eliminar emoticones, signos de puntuación, palabras vacías y colocar todas las palabras en letras minúsculas, luego se entrenaría un modelo mediante técnicas combinadas de los algortimos Naive Bayes y Support Vector Machines, luego el resultado sería enviado al docente a través y una aplicación. Sin embargo, este proyecto se queda en la etapa teórica ya que no se desarrolló ni implementó ningún modelo, únicamente propone la arquitectura de como funcionaría el sistema en sí.

En el presente trabajo se pretende utilizar los resultados oficiales de la encuesta de evaluación docente de una institución educativa superior, lo que garantiza

tener un gran volumen de datos para entrenar el modelo, además de representar las opiniones de estudiantes de distintas carreras y diferentes rangos de edad, abarcando un rango más amplio de palabras que suelen utilizar los estudiantes y permitiendo que el modelo sea capaz de clasificar correctamente resultados que utilicen un léxico más variado.

2.2 Procesamiento de lenguaje natural

El procesamiento de lenguaje natural (NLP) es definido por Chowdhary (2020) como un área de la investigación que explora como las computadoras pueden entender y manipular lenguaje natural, tanto en forma de texto como hablado, para hacer cosas útiles. Sarkar (2016) señala que para entender lo que es NLP, es necesario entender que es el lenguaje natural y lo conceptualiza de la siguiente manera: “en términos simples, el lenguaje natural se desarrolla y evoluciona a partir de los humanos a través del uso natural y comunicación, en lugar de ser construido y creado artificialmente, como un lenguaje de programación de computadora”.

Existe una variedad de aplicaciones que se le puede dar al procesamiento de lenguaje natural, Sarkar (2016) menciona algunas de estas categorías en su libro, como traducción de idiomas, sistemas de reconocimiento de voz, sistemas de respuesta de preguntas, reconocimiento y resolución contextual, resumen de textos y clasificación de textos.

2.2.1 Traducción de idiomas

La traducción de idiomas es una de las aplicaciones más conocidas del NLP, Sarkar (2016) la define como “la técnica que facilita la traducción sintáctica, gramatical y semánticamente correcta entre pares de idiomas”. Aunque a simple vista la traducción de idiomas parezca simple como traducir palabras con un diccionario, muchas veces se pierde el significado gramático o contextual del texto entero, por lo que actualmente se utilizan técnicas estadísticas y lingüísticas para lograr una traducción adecuada.

2.2.2 Sistemas de reconocimiento de voz

Sarkar (2016) señala que esta es una de las aplicaciones más difíciles de procesamiento de lenguaje natural, ya que es muy rígida en cuanto a la entrada

de datos y cualquier desviación de la entrada esperada puede dar un resultado muy diferente al esperado, aunque este tipo de sistemas se han vuelto cada vez más comunes.

2.2.3 Sistemas de respuesta de preguntas (QAS)

Sarkar (2016) explica que los QAS se construyen sobre principios de NLP y recuperación de información, diseñados para proporcionar respuestas a preguntas formuladas en lenguaje natural. Estos sistemas requieren una extensa base de conocimientos y sistemas de consulta eficientes, lo que ha llevado a su desarrollo principalmente en dominios específicos como salud, comercio electrónico y servicio al cliente.

2.2.4 Reconocimiento y resolución contextual

Esta es una aplicación muy interesante y compleja del procesamiento de lenguaje natural, ya que se trata de que la computadora interprete el lenguaje tal como lo hace un humano, dependiendo del contexto y forma en que se utiliza la palabra, como lo menciona Sarkar (2016), queremos determinar el significado de una palabra en un contexto dado, usa un ejemplo con la palabra “book” en inglés, que puede ser utilizada para referenciar a un libro o como verbo de reservar un lugar, se puede encontrar casos como este en todos los lenguajes.

2.2.5 Resumen de textos

El objetivo de esta aplicación de NLP es tomar un cuerpo de texto extenso y tomar las ideas principales para así devolver un texto más conciso, como lo señala Sarkar (2016) “se trata de tomar el cuerpo de documentos de texto, que puede ser una colección de textos, párrafos u oraciones y reducir el contenido de manera apropiada para crear un resumen que contenga los puntos clave de la colección”. Además, nos indica dos tipos de resumen que se pueden llevar a cabo, como lo son el resumen genérico y el resumen basado en consultas, el genérico se refiere a realizar un resumen general del texto de entrada del sistema y el basado en consultas realiza un resumen con base en indicaciones que da el usuario sobre puntos específicos que quiere conocer del texto.

2.2.6 Categorización de textos

Sarkar (2016) define la categorización de texto como la tarea de asignar documentos a categorías o clases específicas basándose en su contenido. Esta aplicación puede implementarse utilizando técnicas de aprendizaje supervisado y no supervisado, y ha encontrado aplicaciones prácticas en filtros de spam y categorización de noticias.

2.2.7 Analítica de texto

Otra aplicación importante del procesamiento de lenguaje natural es la analítica de texto o minería de texto, Sakar (2016) nos dice lo siguiente:

> Es la metodología y proceso que se sigue para obtener información de calidad y accionable, así como insights de datos de texto. Esto involucra usar NLP, recuperación de información y técnicas de machine learning para analizar datos de texto no estructurados y convertirlos en datos estructurados para conseguir patrones e insights que pueden ser útiles para el usuario final.

También nos menciona algunas áreas en las que aplica la analítica de texto, como clasificación de textos, clustering de textos, resumen de textos, análisis de sentimientos, extracción y reconocimiento de entidades, análisis de similitud y modelado de relaciones.

2.3 Análisis de sentimientos

Shaik et. al. (2023) explican que el análisis de sentimientos es una aplicación de NLP que tiene como objetivo identificar las intenciones de una persona tras su reseña. Además, señalan que en el sector de la educación este tipo de análisis se puede utilizar para mejorar las técnicas de enseñanza en una institución.

Liu (2020) nos cuenta acerca de la historia del análisis de sentimientos, comenta que no existía investigación en el campo hasta después del año 2000, debido a que no existían registros digitales sobre texto de opiniones previo a este año, pero esto cambio con la popularidad del internet y las redes sociales, generando un flujo constante de datos que son necesarios para llevar a cabo investigación en el campo. Un aspecto clave de las redes sociales que ha ayudado a recolectar este

tipo de datos es la facilidad de interacción que le da a los usuarios, permitiéndoles expresarse mediante publicaciones como reseñas y blogs, incluso comentar sobre estas en las distintas plataformas existentes.

Además, Liu (2020) señala la importancia del análisis de sentimientos desde puntos de vista distintos, para las compañías es importante conocer la percepción de los clientes acerca de sus productos o servicios, para los gobiernos es importante conocer como la población general se siente con respecto a las políticas que implementa y desde la perspectiva del consumidor, le interesa saber la percepción de otros consumidores previamente a adquirir un producto o servicio para tomar una decisión más informada. El internet y las redes sociales han cambiado la forma en que estas entidades y personas consiguen las opiniones de otros con respecto a algo, antes preguntaban a sus amigos o familiares, las empresas realizaban encuestas y estudios de mercado, hoy en día recolectan y analizan las opiniones de redes sociales.

Existen múltiples métodos y técnicas que se pueden utilizar para desarrollar un modelo de análisis de sentimientos, Mao, Liu y Zhang (2024) nos indican que estos son: técnicas basadas en léxico, consiste en asignar un puntaje a cada palabra para así calcular un sentimiento, sin embargo, esta técnica enfrenta el desafío de que cada palabra puede tener un sentimiento distinto dependiendo del contexto en el que es usada. También señalan métodos basados en machine learning, se basa en separar el data set en subconjuntos de entrenamiento y de prueba para entrenar el modelo mediante una técnica de clasificación, como puede ser Naive Bayes, Support Vector Machines, Árboles de Decisión, entre otros, y posteriormente evaluar la efectividad del modelo utilizando el subconjunto de prueba. Además, nos muestran otro método basado en el Deep Learning, mencionan que se pueden usar técnicas como las Redes Neurales Convolucionales (CNN) y las Redes Neuronales Recurrentes (RNN) para entrenar un modelo de análisis de sentimientos.

También hay distintos niveles en los que se puede realizar un análisis de sentimientos, como lo explica Liu (2020), se puede analizar a nivel de documento, como su nombre lo indica, analiza un documento entero como lo puede ser una reseña de algún producto y determina la opinión asociada, este tipo de análisis es

limitado ya que requiere que el documento hable de una sola entidad y no varias. El siguiente nivel es el basado en oraciones, en este su principal función es distinguir entre oraciones que indican información factual y aquellas que indican una opinión. El último nivel es basado en aspectos o características, este tiene como objetivo determinar el sujeto de opinión y el sentimiento asociado a este, explica esto con un ejemplo, “a pesar del mal servicio, me encanto este restaurante”, en esta oración tenemos dos sujetos y dos sentimientos asociados, uno negativo hacia el servicio y otro positivo hacia el restaurante en general.

2.4 Algoritmos de Machine Learning

2.4.1 Naive Bayes

El algoritmo de clasificación Naive Bayes es uno de los más simples y por ende utilizados, como nos explica Webb (2010)

> Naive Bayes es un algoritmo de aprendizaje simple que utiliza la regla de Bayes junto con una fuerte suposición de que los atributos son condicionalmente independientes dada la clase. Aunque esta suposición de independencia a menudo se viola en la práctica, naïve Bayes frecuentemente ofrece una precisión de clasificación competitiva. Junto con su eficiencia computacional y muchas otras características deseables, esto lleva a que naïve Bayes sea ampliamente aplicado en la práctica.

También nos señala algunos puntos fuertes de este algoritmo, como su eficiencia computacional, su robustez en datasets con mucho ruido y altos volúmenes de datos faltantes.

2.4.2 Support Vector Machine

Noble (2006) nos habla acerca del algoritmo Support Vector Machine, es un algoritmo de aprendizaje supervisado utilizado para clasificar datos que se basa en el uso de hiperplanos para encontrar el margen máximo entre clases. Es mediante el uso de estos hiperplanos que mantiene la máxima distancia posible entre los puntos más cercanos de una clase.

Algunos puntos fuertes de este algoritmo son su capacidad para trabajar con datos complejos no lineales, su versatilidad con distintos tipos de datos y su sólida base teórica.

2.4.3 Árbol de Decisión

El algoritmo de árbol de decisión tiene un funcionamiento más sencillo de entender, como lo explican Kingsford y Salzberg (2008), “Los árboles de decisión son clasificadores que predicen etiquetas de clase mediante una serie de preguntas jerarquizadas sobre las características de los datos”. Es decir, el algoritmo divide los datos en distintas categorías en cada nivel jerárquico hasta llegar a un punto que puede asignarlo a una clase específica. Entre las ventajas de este algoritmo se encuentra su fácil interpretabilidad, su capacidad de trabajar con distintos tipos de variables y clasificación de datos multiclase.

2.4.4 Random Forest

Oshiro, et. Al. (2012) nos indica que el algoritmo de Random Forest es un algoritmo de aprendizaje supervisado basado en los árboles de decisión, ya que se construye con varios de estos y combina sus predicciones, se basa en dos conceptos principales, el bagging que consiste en crear mutliples datasets de entrenamiento mediante el muestreo aleatorio del dataset original y la selección aleatoria de características para cada uno de los árboles de decisión. Este algoritmo es robusto en contra del sobreajuste, tiene la capacidad de manejar datos de alta dimensionalidad y generalmente tiene buen rendimiento sin necesidad de modificar los hiperparámetros.

2.5 Redes neuronales

Las redes neuronales como tal tienen una larga historia, como nos cuenta Gerón (2019), el primer tipo de red neuronal fueron las Redes Neuronales Artificiales (ANN) en 1943 cuando fueron presentadas en un libro de cálculo por Warren McCulloch y Walter Pitts, presentaron un modelo matemático que intentaba explicar cómo las neuronas de un cerebro biológico podrían funcionar entre sí para realizar cálculos complejos. El desarrollo de estos modelos fue lento y tuvo un periodo donde se detuvo su investigación hasta los años 1980 cuando volvió el interés por este tipo de tecnologías, aunque pronto se desarrollarían otros

modelos como Support Vector Machines que en su momento tenían un rendimiento mejor.

Gerón (2019) nos explica de forma resumida el modelo propuesto por McCulloch y Pitts, en este modelo las neuronas tienen una entrada y una salida que están activas o inactivas, la salida dependía de la cantidad de entradas activas que tenía y seguía una lógica booleana, como se puede observar en la siguiente ilustración.

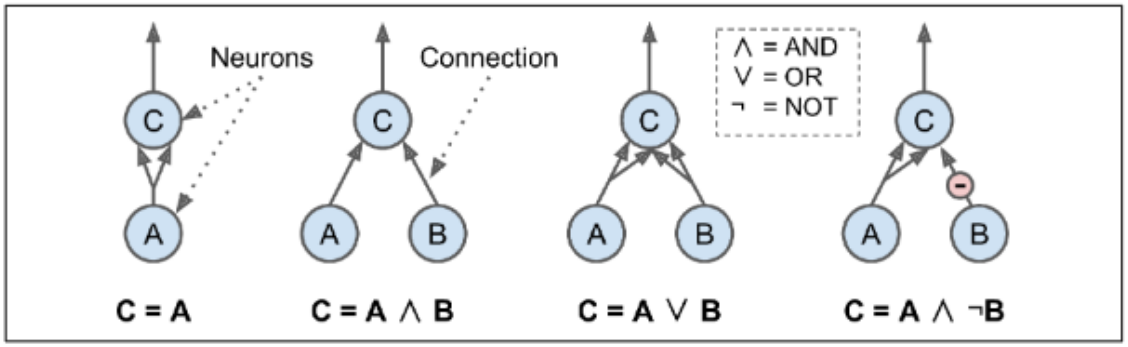

Ilustración 1. Modelo de McCulloch y Pitts (Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow)

Posteriormente, en el año 1957 hubo un avance significativo en el campo de las redes neuronales por Frank Rosenblatt, como lo cuenta Gerón (2019), se desarrolló el “perceptrón” que tomaba un tipo de neurona distinta llamada Linear Threshold Unit (LTU), en lugar de tener entradas binarias eran numéricas, cada entrada tiene un peso ponderado, se realiza una suma de estos pesos ponderados y se aplica una función para activar la neurona de salida. Para entrenar un perceptrón se utiliza la regla de Hebb, quién observó que en un cerebro biológico las neuronas que activan frecuentemente a otras refuerzan sus conexiones, por lo que se aplicó este principio a las redes neuronales, se alimenta un ejemplo de entrenamiento al perceptrón, se realiza la predicción y si esta fue incorrecta se ajustan los pesos de las conexiones para acercarnos a la predicción correcta.

Actualmente existen distintos tipos de redes neuronales, como indican Thakur y Konde (2021), las redes neuronales convolucionales (CNN), redes neuronales de

propagación hacia adelante (FNN), redes neuronales recurrentes (RNN), redes neuronales de base radial (RBF) y redes neuronales modulares (MNN).

2.5.1 Redes Neuronales Convolucionales

Wu (2017) explica el funcionamiento y la arquitectura de una Red Neuronal Convolucional, las CNN funcionan por capas, una capa de entrada, n cantidad de capas ocultas y una capa de salida. En la capa de entrada se recibe los datos que se van a manejar, generalmente en forma de matrices o tensor, como son llamados en el contexto de las redes neuronales. Esta entrada pasa a ser procesada en las capas ocultas, que pueden ser de distinto tipo, como una capa de convolución, una capa de normalización, una capa de pérdida, entre otras. Una vez que se los datos han sido procesados por todas las capas ocultas, llegan a la capa de salida donde nos dará el resultado. Se puede observar una representación gráfica de una red neuronal convolucional en la figura 2:

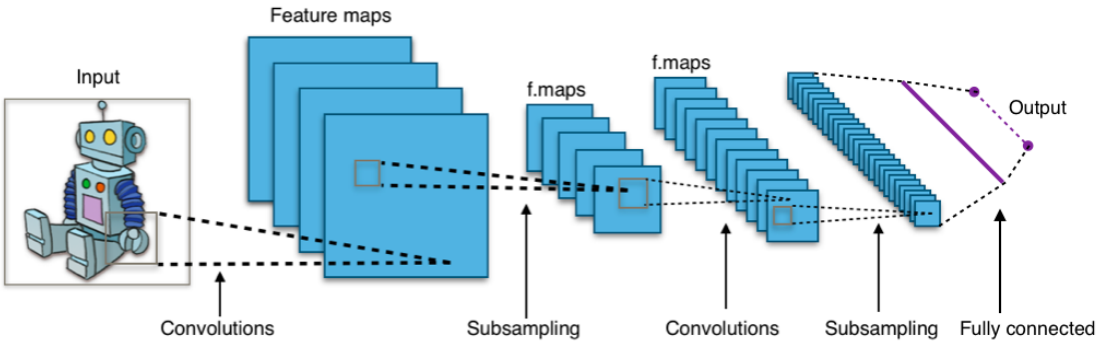

Ilustración 2. Ejemplo de red neuronal convolucional (Towards AI (2023))

2.5.2 Redes Neuronales de Propagación Hacia Delante

Otro tipo de red neuronal bastante utilizado es una red neuronal de propagación hacia delante (FNN), Pan (2024) nos cuenta que este tipo de redes neuronales es uno de los modelos más básicos y también de lo conoce como un perceptrón multicapa, se caracteriza por tener varias capas de neuronas, las capas de entrada procesan los datos con los pesos asignados para pasar sus datos a la siguiente capa hasta llegar a la última que es la capa de salida, este procesamiento es unidireccional y por eso son llamadas redes de propagación hacia delante. Además, nos menciona que este tipo de redes son especialmente buenas para resolver problemas de clasificación y regresión, por lo que algunos

ejemplos de uso son la clasificación de imágenes y calificación crediticia. Se puede ver un ejemplo de una red neuronal de propagación hacia delante en la figura 3:

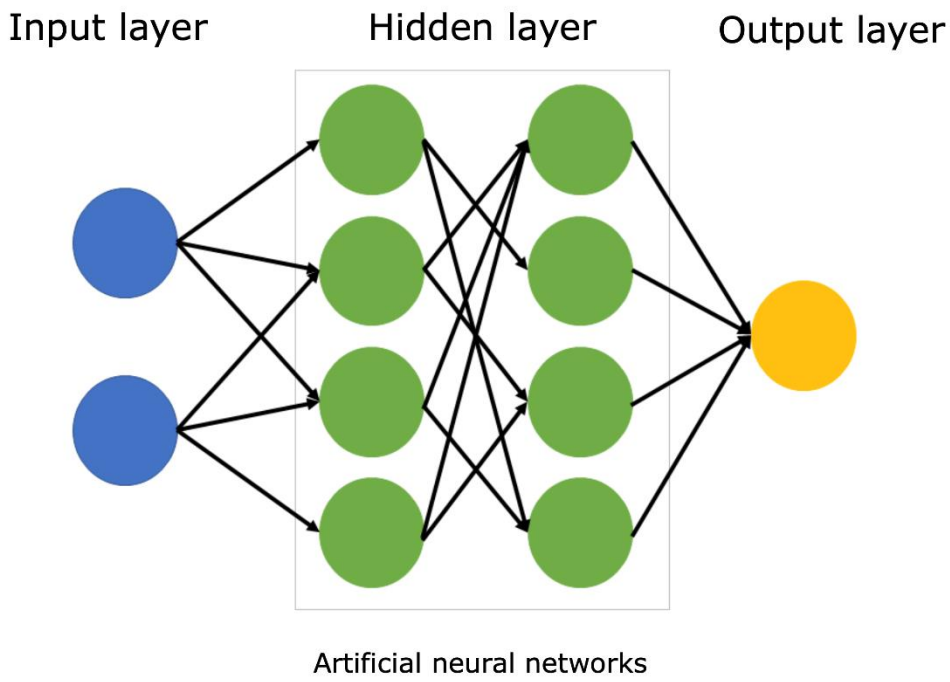

Ilustración 3. Ejemplo de red neuronal de propagación hacia delante (Science Learning Hub (2024))

2.5.3 Redes Neuronales Recurrentes

Un tipo de redes neuronales más complejo es la red neuronal recurrente (RNN), que como Pan (2024) nos indica, estas tienen conexiones bidireccionales a diferencia de las FNN que eran unidireccionales, esto les permite tener conocimiento de los pasos realizados en capas anteriores, lo que las vuelve muy buenas para modelos donde el tiempo es un factor importante. Estas se componen por células de entrada que reciben los datos en forma de vectores o embeddings, las células ocultas que procesan el paso temporal actual de los datos así como el paso de la célula anterior, lo que le permite tener este conocimiento de los pasos previos, las células recurrentes que se encargan de retener información histórica, procesar esos datos y aprender las características temporales de estos, finalmente la célula de salida que nos da el resultado del modelo. Algunas aplicaciones para este tipo de redes neuronales se dan en el

campo del procesamiento de lenguaje natural, específicamente en la traducción de textos donde es muy importante el orden de las palabras y el reconocimiento de voz, donde igualmente es fundamental tomar en cuenta la señal de audio como un conjunto. Se puede visualizar un ejemplo de una red neuronal recurrente en la figura 4:

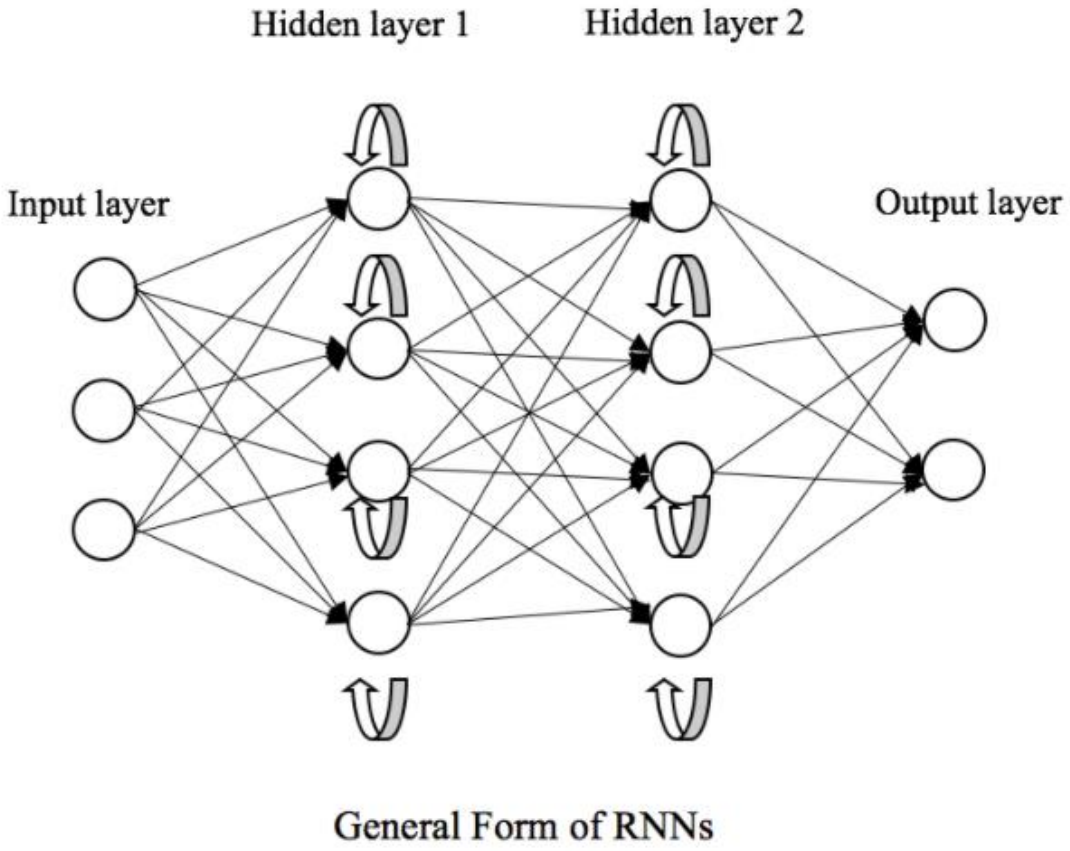

Ilustración 4. Ejemplo de red neuronal recurrente (Open Data Science (2020))

2.5.4 Redes Neuronales de Base Radial

Thakur y Konde (2021) nos explican cómo funcionan las redes neuronales de base radial, esta se caracteriza por utilizar funciones de base radial como lo indica su nombre, generalmente son funciones gaussianas, toma como entradas vectores y calcula la distancia de estos a lo que denomina un centro y las neuronas de la capa oculta se activan dependiendo de que tan cerca esté el vector del centro de esta. Se suelen aplicar este tipo de redes neuronales cuando se trata con funciones no lineales y común mente en sistemas de energía eléctrica para contrarrestar apagones. En la figura 5 se puede ver un ejemplo de una red neuronal de base radial:

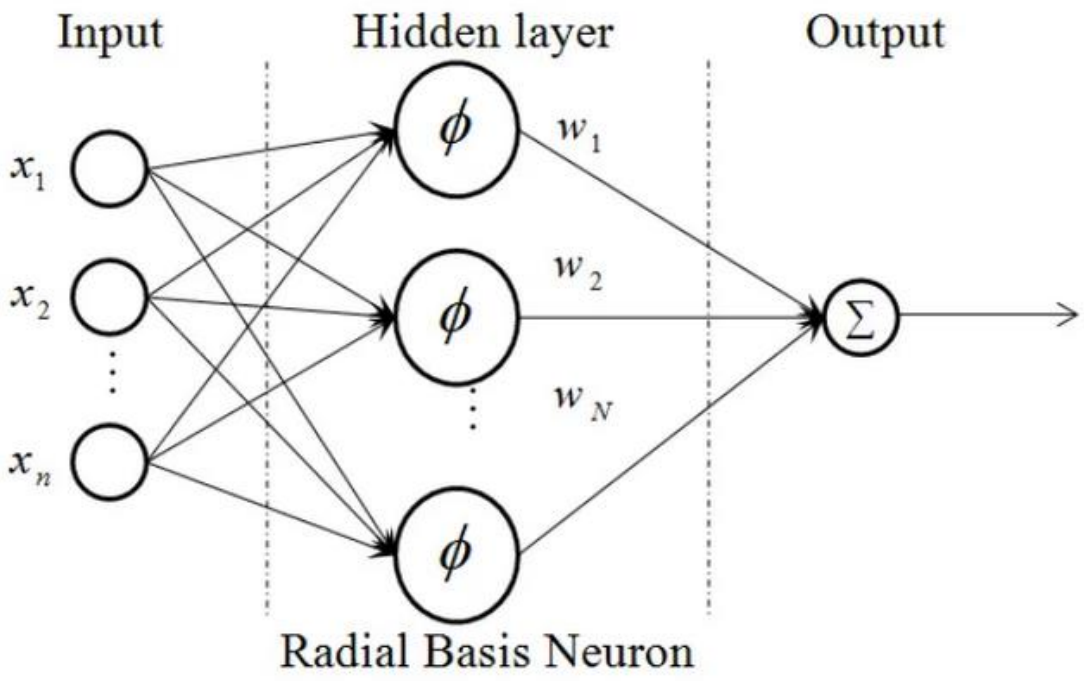

Ilustración 5. Ejemplo de red neuronal de base radial (Medium (2023))

2.5.5 Redes Neuronales Modulares

También existen las redes neuronales modulares, Thakur y Konde (2021) mencionan que este tipo de redes neuronales funcionan como varias redes independientes, es decir, no interactúan entre sí y cuentan con un mecanismo de integración para obtener la salida final de la red, se suelen dividir problemas complejos en otros más pequeños y se asignan estos a cada red, por lo que los tiempos de procesamiento suelen ser mucho menores. Este tipo de redes suelen ser aplicadas en el sector financiero para detectar fraudes y realizar predicciones de mercado. Se puede ver la arquitectura de este tipo de red en la figura 6:

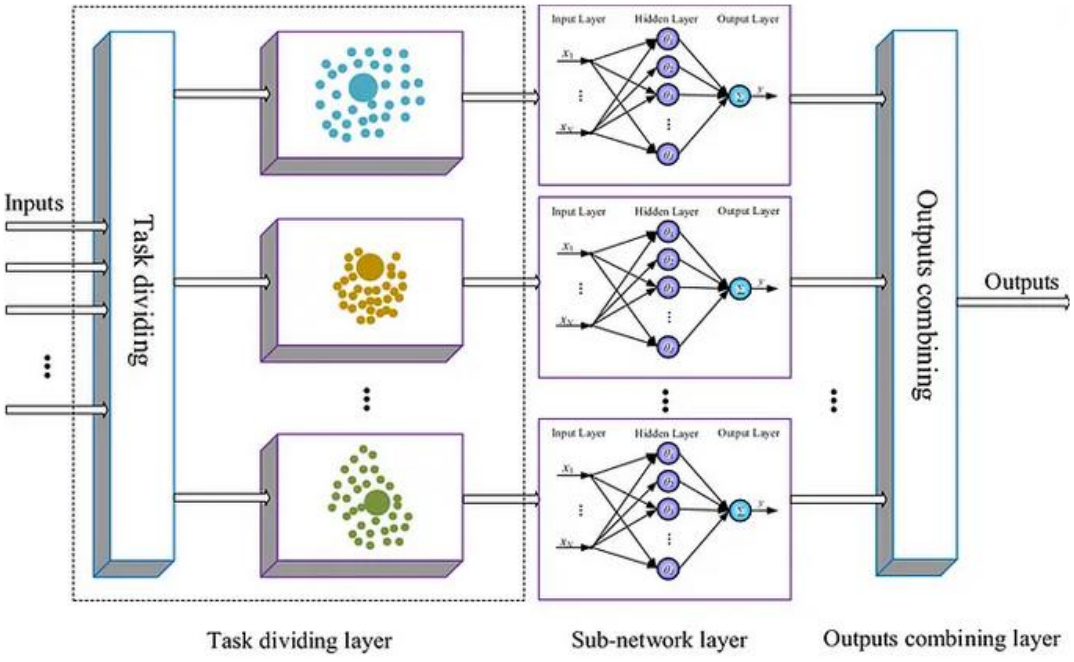

Ilustración 6. Ejemplo de red neuronal modular (Medium (2023))

2.6 Métodos de Vectorización

Como lo mencionan Abubakar, Umar y Bakale (2022) "*La representación de contenidos textuales en formatos entendibles por programas de computadora y algoritmos de machine learning es un paso vital en la identificación de sentimientos en textos y es conocido formalmente como vectorización".* Existen varios métodos para llevar a cabo este proceso, algunos nombrados por estos mismos autores son, Bag of Words (BoW), frecuencia de términos, frecuencia inversa del documento (Tf-Idf), word2vec y doc2vec.

2.6.1 Bag of Words

Qader, Ameen y Ahmed (2019) nos explican que el método Bag of Words puede ser utilizado dentro de los campos de visión por computadora y clasificación de textos. En el contexto de procesamiento de lenguaje natural este método obtiene los valores de los vectores con base en la frecuencia de aparición de cada palabra en una oración, sin tomar en cuenta la gramática o el orden de las palabras. Para entenderlo de mejor manera nos proveen el siguiente ejemplo:

Tenemos las siguientes oraciones:

Dara likes to go to cinema. Going to cinema is one of the hobbies of Azad.

Dara also wants to go to play football and to go to swim.

Entonces el método BoW genera diccionarios a partir de la cantidad de palabras repetidas en cada oración:

BoW 1 = {“Dara”:1, “likes”:1, “to”:3, “go”:1, “cinema”:2, “Going”:1, “is”:1, “one”:1, “of”:2, “the”:1, “hobbies”:1, “Azad:1”};

BoW 2 = {“Dara”:1, “also”:1, “wants”:1, “to”:4, “go”2, “play”:1, “football”:1, “and”:1, “and”:1, “swim:1”};

El siguiente paso es realizar una unión de todos los BoW calculados para formar uno final, que en este caso es BoW 3:

BoW3 = {“Dara”:2, “likes”:1, “to”:7, “go”:3, “cinema”:2, “going”:1, “is”:1, “one”:1, “of”:2, “the”:1, “hobbies”:1, “Azad”:1, “also”:1, “wants”:1, “play”:1, “football”:1, “and”:1, “swim”:1}

Con este BoW final, podemos conseguir los vectores de cada oración, estos son la cantidad de veces que se repite cada palabra del BoW 3 en cada oración, por lo que resultan los siguientes vectores:

1) [1, 1, 3, 1, 2, 1, 1, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0]

2) [1, 0, 4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]

2.6.2 Tf-Idf

Este es otro método que se utiliza para vectorizar textos, aunque este es más robusto que BoW ya que penaliza palabras que se repiten demasiado, como las palabras vacías “él”, “la”, entre otras que no aportan significado a la oración, como lo explican Havrlant y Kreinovich (2017) este método se basa en dos componentes, calcular la cantidad de veces que se repite un término en un documento, como lo hace BoW, este término es el TF, se explica con la siguiente ecuación donde *t* es el término que buscamos y *d* el documento

$$TF = tf(t, d)$$

Luego calcula la frecuencia inversa del documento (IDF), es decir, cuenta la cantidad total de documentos (N) y lo divide para la cantidad de documentos que

contienen la palabra específica (df(t)), de esta división se calcula el logaritmo natural.

$$IDF = ln\left(\frac{N}{df(t)}\right)$$

Finalmente se obtiene el valor de TF-IDF al multiplicar el resultado de ambos, con este valor resultante se arma el vector de cada oración, este método penaliza las palabras vacías que se repiten en todo el documento ya que el logaritmo natural de 1 es igual a 0.

2.6.3 Word2Vec

Lilleberg, Zhu y Zhang (2015) presentan Word2Vec como un método de procesamiento de lenguaje natural que ofrece una perspectiva única para la clasificación de textos al convertir palabras y frases en representaciones vectoriales. A diferencia de los métodos tradicionales como BoW y TF-IDF, Word2Vec busca capturar características semánticas adicionales que ayudan en la clasificación de textos.

Di Gennaro, Buonanno y Palmieri (2021) explican que el proceso de entrenamiento de Word2Vec utiliza dos arquitecturas principales: Continuous Bag-of-Words (CBOW) y Skip-gram. Para entender su funcionamiento, tomemos como ejemplo la oración:

El perro corre rápido en el parque

En el modelo Skip-gram, que según Lilleberg et al. (2015) es el más utilizado en la práctica, si seleccionamos "perro" como palabra central, el modelo intentará predecir las palabras contextuales ("el", "corre", "rápido") dentro de una ventana predefinida. El proceso de entrenamiento implica una red neuronal de dos capas con activación lineal en la capa oculta y sin sesgo.

El modelo genera dos matrices de pesos: una matriz de entrada:

$$H \in RV \times M$$

Y una matriz de salida:

$$Z \in RM \times V$$

Donde V es el tamaño del vocabulario y M la dimensión del espacio vectorial. Di Gennaro et al. (2021) señalan que el tamaño típico de la ventana de contexto y la dimensión del espacio vectorial son parámetros críticos que afectan el rendimiento del modelo.

Para mejorar la eficiencia computacional, Lilleberg et al. (2015) mencionan que Word2Vec utiliza una técnica llamada "muestreo negativo", que permite entrenar el modelo seleccionando aleatoriamente palabras que no aparecen en el contexto, reduciendo así la complejidad computacional del entrenamiento.

A diferencia de BoW que genera vectores basados en frecuencias simples, o TF-IDF que penaliza palabras comunes, Word2Vec genera vectores que codifican información semántica, permitiendo capturar relaciones más complejas del lenguaje.

2.6.4 Doc2Vec

Lau y Baldwin (2016) presentan Doc2Vec como una extensión del modelo Word2Vec para generar representaciones vectoriales de documentos completos. Este modelo se desarrolló con la capacidad de capturar la semántica de fragmentos de texto de diferente longitud, desde frases hasta documentos enteros, sin importar su extensión.

Doc2Vec opera bajo dos arquitecturas principales, como explican Dogru et al. (2021):

1. Distributed Memory Model (PV-DM):

Cada documento recibe un vector único que funciona como una memoria que captura el contexto global del texto. Durante el proceso de predicción de una palabra objetivo, el modelo combina tanto el vector del documento como los vectores de las palabras contextuales. Por ejemplo, para la oración:

el perro corre en el parque

El modelo sumaría los vectores para predecir la siguiente palabra como se muestra a continuación:

Vector_documento + Vector("el") + Vector("perro") → Predice "corre"

2. Distributed Bag of Words (PV-DBOW):

Presenta una estructura más simple, donde únicamente se utiliza el vector del documento para predecir las palabras que aparecen en él. Esta arquitectura ignora el orden de las palabras y no necesita almacenar vectores de palabras individuales, lo que resulta en un menor consumo de recursos computacionales.

El proceso de entrenamiento implica inicializar aleatoriamente los vectores de documentos y palabras. Durante el entrenamiento, el modelo ajusta estos vectores para maximizar la probabilidad de predecir correctamente las palabras del contexto (en PV-DM) o las palabras del documento (en PV-DBOW). Como demuestran Lau y Baldwin (2016), aunque PV-DM es más complejo, PV-DBOW suele obtener mejores resultados y requiere menos recursos computacionales.

2.7 Metodología y Técnicas

Para el presente trabajo se propone utilizar Redes Neuronales Recurrentes para entrenar el modelo de análisis de sentimientos, Kurniasari y Setyanto (2020) realizaron un estudio comparativo de diferentes algoritmos para análisis de sentimientos y determinar cuál obtenía los mejores resultados, en su comparación incluyeron el algoritmo de Naive Bayes, redes neuronales convolucionales y redes neuronales recurrentes, concluyeron que las redes neuronales recurrentes eran la mejor opción en conjunto con la técnica word2vec, para esto se propone utilizar el siguiente flujo:

- Recolección las respuestas de la encuesta de evaluación docente
- Limpieza de los datos para eliminar signos de puntuación y stopwords.
- Extracción de vectores de las respuestas mediante word2vec.
- Entrenamiento de la red neuronal.
- Evaluación y análisis de resultados.
- Iteración del flujo en caso de que los resultados no sean satisfactorios.

Para realizar esto se plantea utilizar el lenguaje de programación Python, debido a sus varias librerías orientadas al Deep learning y NLP, Sarkar (2016) menciona algunas como NLTK, spacy, gensim y textblob para el análisis de texto. También indica que existen librerías enfocadas en el machine learning y Deep learning,

como scikit-learn, numpy, tensorflow y keras. Por lo que Python es un lenguaje bastante robusto para este tipo de proyectos.

2.8 Python

Amazon Web Services (2024) nos define Python de la siguiente manera “es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML)”. También nos indica algunos de los beneficios de utilizar este lenguaje, como:

- Es fácil de leer y comprender debido a su sintaxis sencilla.
- El tiempo de desarrollo es menor que en otros lenguajes como C y Java.
- Cuenta con una gran cantidad de librerías disponibles para una variedad de tareas.
- Es multiplataforma, por lo que se puede ejecutar en sistemas operativos Windows, Linux y Mac.

Además, Amazon Web Services (2024) señala los distintos usos que podemos dar a este lenguaje de programación, tales como el desarrollo de sistemas back end, cuenta con librerías para interactuar con la base de datos, recibir y enviar datos al usuario, gestionar la seguridad del sistema, etc. Otro uso común es la automatización de tareas, como convertir archivos de un tipo a otro, cambiar sus nombres, enviar correos electrónicos, entre otros. Pero el uso que más nos interesa para el presente trabajo de titulación es la ciencia de datos y el machine learning, permite interactuar con distintos formatos de archivos de almacenamiento de datos, realizar cálculos estadísticos, generar gráficos estadísticos, entrenar distintos tipos de modelos de machine learning y Deep learning.

2.8.1 Pandas

La documentación oficial de Pandas (2024) define la librería como “un paquete de Python que ofrece estructuras de datos rápidas, flexibles y expresivas diseñadas para volver el trabajo con datos relacionados o etiquetados algo fácil e intuitivo”. Esta librería trabaja con dos tipos de estructuras de datos, las “series” de 1 dimensión y los “dataframe” de 2 dimensiones, cuenta con una variedad de funciones de mucha utilidad para el manejo de datasets, como manejar datos

faltantes, insertar y eliminar columnas de datos, manipular los tipos de datos de cada variable, entre muchas otras.

2.8.2 Numpy

Numpy (2024) indica en su documentación que es un paquete de Python enfocado a la computación científica, se basa en la manipulación de arreglos multidimensionales de datos para realizar cálculos algebraicos, estadísticos, entre otros. Además, indica que una gran ventaja de utilizar esta librería es su velocidad, ya que por detrás se basa en código optimizado y precompilado en C.

2.8.3 Gensim

Rehurek (2024) nos define Gensim de la siguiente manera “es una librería Python de código abierto y gratuita para representar documentos como vectores semánticos, de la manera más eficiente (desde el punto de vista computacional) y sencilla (desde el punto de vista humano) posible.” Fue diseñada con la idea de procesar datos en bruto mediante algoritmos de aprendizaje no supervisados para descubrir la estructura semántica de los textos, algunos de los algortimos que usa son Word2Vec, FastText, Latent Semantic Indexing, entre otros.

2.8.4 Spacy

La documentación provista por Explosion (2025) nos define a Spacy como “una biblioteca de código abierto gratuita para Procesamiento del Lenguaje Natural (NLP) avanzado en Python”. Sus funciones principales incluyen la tokenización de texto, etiquetado gramatical, análisis de dependencias sintácticas y lematización. La biblioteca permite reconocer entidades nombradas, detectar límites de oraciones y comparar la similitud entre palabras y documentos. Además, spaCy facilita la clasificación de texto, la coincidencia basada en reglas y el entrenamiento de modelos estadísticos.

2.8.5 Tensorflow

Abadi, et. Al. (2016) nos explican que Tensforflow es un sistema de machine learning de código abierto que opera a gran escala y en entornos heterogéneos. Se caracteriza por usar grafos de flujo de datos para representar la computación, el estado compartido y las operaciones que modifican ese estado. Se ha convertido en una de las bibliotecas más populares y ampliamente utilizadas para

la investigación y el desarrollo de machine learning desde su lanzamiento como proyecto de código abierto. Tiene una variedad de aplicaciones como el entrenamiento de redes neuronales profundas, visión por computadora, procesamiento de lenguaje natural, entre otras.

2.8.6 Scikit-learn

La documentación oficial de Scikit-learn (2025) da la siguiente definición “es una librería de aprendizaje automático de código abierto que soporta aprendizaje supervisado y no supervisado. También proporciona diversas herramientas para el ajuste de modelos, preprocesamiento de datos, selección de modelos, evaluación de modelos y muchas otras utilidades”. Es una librería muy potente y versátil dentro del campo del machine learning, las funciones y utilidades que brindan nos facilitan mucho el trabajo en todas las etapas de la elaboración de un modelo.

CAPITULO III: METODOLOGÍA

3.1 Metodología científica

El enfoque adoptado para esta investigación es de tipo cuantitativo con un diseño experimental, ya que se busca desarrollar, entrenar y evaluar distintos modelos de análisis de sentimientos que clasifiquen automáticamente las respuestas cualitativas de la encuesta de evaluación docente en las categorías positivo, neutro y negativo. La investigación sigue un diseño cuasi experimental, donde se desarrollaron modelos basados en algoritmos de machine learning y en redes neuronales profundas con vectores y embeddings obtenidos mediante diferentes técnicas de vectorización, permitiendo así evaluar la eficacia de cada uno en la clasificación de sentimientos.

La metodología se fundamenta en el NLP aplicado al español, utilizando técnicas avanzadas de representación de texto como Bag of Words, Tf-Idf y Word2Vec, que permite capturar las relaciones semánticas específicas del contexto educativo. El diseño experimental incluye la validación cruzada y la evaluación mediante métricas estándar de clasificación (precisión, recall, F1-score), asegurando la robustez y confiabilidad de los resultados obtenidos.

3.2 Metodología técnica

Para el desarrollo técnico del modelo de análisis de sentimientos, se utilizará la metodología CRISP-DM (Cross Industry Standard Process for Data Mining). Como lo indica Singgalen (2024) Este estándar proporciona un enfoque iterativo y estructurado para proyectos de minería de datos. Las fases son las siguientes: comprensión del negocio, comprensión de los datos, preparación de datos, modelado, evaluación, despliegue. Aplicar estos pasos asegura un acercamiento estructurado a la validación de los resultados del análisis de sentimientos.

En el presente trabajo de titulación se aplicarán todos los pasos de CRISP-DM exceptuando el de implementación, ya que el paso a producción del modelo de análisis de sentimientos queda fuera del alcance del proyecto.

CAPITULO IV: DESARROLLO DEL PRODUCTO

4.1 Compresión de negocio

En el contexto del presente proyecto de titulación, se plantea desarrollar un modelo de análisis de sentimientos aplicado a las respuestas abiertas de la encuesta de evaluación docente de la una institución educativa superior en Quito, Ecuador. Esta encuesta es una herramienta clave para medir la satisfacción de los estudiantes y detectar áreas de mejora en el desempeño docente. Sin embargo, debido a la naturaleza cualitativa de las preguntas abiertas y al alto volumen de respuestas generadas por más de 18000 estudiantes, el proceso de interpretación manual resulta laborioso y propenso a sesgos por parte del evaluador. Mediante la implementación de un modelo de análisis de sentimientos, se busca automatizar este análisis, clasificando las respuestas en categorías de sentimiento (positivo, neutro y negativo) y proporcionando datos procesables que optimicen la toma de decisiones y mejoren la eficiencia en la evaluación docente.

En este capítulo, se presenta el desarrollo de distintos modelos de análisis de sentimientos utilizando diferentes algoritmos, serán diseñados específicamente para procesar e interpretar las respuestas abiertas proporcionadas por estudiantes en la encuesta de evaluación docente de una institución de educación superior. El objetivo principal es convertir estas valiosas opiniones, expresadas en lenguaje natural, en información cuantitativa y accionable que facilite la toma de decisiones y mejore los procesos de evaluación y calidad educativa.

Los modelos propuestos aprovechan técnicas avanzadas de NLP, aprendizaje automático y aprendizaje profundo para capturar los matices semánticos y detectar automáticamente el sentimiento en cada respuesta, clasificándolas en categorías de positivo, negativo o neutro. Este enfoque permite superar las limitaciones del análisis manual, reduciendo el tiempo y esfuerzo requeridos, al tiempo que se mantiene un alto nivel de precisión y objetividad.

A continuación, se detallan las etapas clave del desarrollo de los modelos, la recolección y preprocesamiento de los datos, el desarrollo y evaluación del modelo de análisis de sentimientos.

4.2 Recolección y comprensión de datos

Los datos utilizados para desarrollar y evaluar el modelo de análisis de sentimientos provienen de las respuestas abiertas recopiladas en la encuesta de evaluación docente aplicada en una institución de educación superior durante el año 2024. Esta encuesta, que se realiza al finalizar cada semestre, busca obtener la retroalimentación de los estudiantes sobre el desempeño de sus profesores y diversos aspectos de la experiencia de aprendizaje.

Las respuestas fueron extraídas directamente de la base de datos institucional, donde se almacenan los resultados de las encuestas. Se obtuvieron un total de 36808 registros, cada uno correspondiente a la opinión de un estudiante sobre un docente específico. Los datos se recibieron en formato de hoja de cálculos en 2 archivos diferentes, uno correspondiendo a cada semestre del año 2024, conteniendo tres campos principales: un identificador único, la pregunta de la encuesta y el texto de la respuesta del estudiante. En este caso la evaluación docente cuenta con varias preguntas de opción múltiple y una pregunta abierta al final, para nuestro trabajo de titulación utilizaremos únicamente la pregunta abierta.

Es importante destacar que, durante el proceso de recolección, se aplicaron consideraciones éticas y de privacidad, asegurando que los datos estuvieran completamente anonimizados. Ninguna información personal identificable de estudiantes o docentes fue incluida o utilizada en cualquier etapa del desarrollo del modelo.

La técnica utilizada para anonimizar las respuestas de los estudiantes es la siguiente: primero se concatenan distintos códigos para identificar al docente, la materia que dicta y el periodo académico, luego se aplica salting con un código secreto y se aplica un hash sha256, finalmente se aplica codificación de tipo base64 y se toman los primeros 16 caracteres para generar los códigos asociados a las respuestas, de este manera se pueden asociar los resultados del análisis de sentimientos llevando a cabo este proceso nuevamente sobre los datos en bruto.

En la etapa inicial de análisis, se utilizó el lenguaje de programación Python y la librería Pandas para procesar los datos. Primero, se cargaron los datos de cada archivo en un dataframe.

```
df1 = pd.read_excel("datos tesis/Resultados_IDEA202410_completo.xlsx")
df1
```

✓ 1.0s

	CodUnion2	Pregunta	Respuesta
0	JmlNpqV5aBe-kiNX	46. Comentarios adicionales que usted pueda pr...	Buena clase
1	9h_4mbQRvKK1qw6N	46. Comentarios adicionales que usted pueda pr...	Ninguno.
2	mQxALHD3wYn6BnUC	46. Comentarios adicionales que usted pueda pr...	Excelente docente
3	dIt4w57MdiNuaW8g	46. Comentarios adicionales que usted pueda pr...	Excelente Docente
4	3kahMRgdFQOwNvkH	46. Comentarios adicionales que usted pueda pr...	N/D
...	...	...	...
18399	2PMisQ8rmS88INvV	46. Comentarios adicionales que usted pueda pr...	Es muy buena doctora pero no tenemos clase int...
18400	jlEFOiVJ5wmt2Tdq	46. Comentarios adicionales que usted pueda pr...	Muchas gracias por todo Dra. Con usted aprendí...
18401	BbYA1ya0oMviy6wh	46. Comentarios adicionales que usted pueda pr...	Es una excelente docente, enseña muy bien y mo...
18402	m5O4-B_xWW9q9KVU	46. Comentarios adicionales que usted pueda pr...	Excelente docente
18403	tInz5Tdh6HnMJnci	46. Comentarios adicionales que usted pueda pr...	Excelente docente, espero que la Dra no baje e...

18404 rows × 3 columns

Ilustración 7. Carga de datos del primer archivo

```
df2 = pd.read_excel("datos tesis/resultados_IDEA202420 Completo.xlsx")
df2
```

✓ 1.4s

	CodUnion2	Pregunta	Respuesta
0	cBEr0wUYVAfc-t3U	46. Comentarios adicionales que usted pueda pr...	Es un muy buen profesor que me a ayudado a ent...
1	wWvUzDlS2ye8Ap3y	46. Comentarios adicionales que usted pueda pr...	El profe si sabe, pero hace falta mas orden y ...
2	suE_k3a8ZrAy0Xh-	46. Comentarios adicionales que usted pueda pr...	Belen Espinel es la mejor profesora de la carr...
3	vsuIa9gWRSeH5-rX	46. Comentarios adicionales que usted pueda pr...	Es muy buena , no tiene la culpa de haber alum...
4	52nu2L80dz3-RQQb	46. Comentarios adicionales que usted pueda pr...	Excelente docente, las clases son muy dinámica...
...	...	...	...
18399	6aZt85NoQKflqfve	46. Comentarios adicionales que usted pueda pr...	ser más paciente
18400	3QbsyEoVqRUpmO4c	46. Comentarios adicionales que usted pueda pr...	Nd
18401	ZgOIbzEfsyO1weAs	46. Comentarios adicionales que usted pueda pr...	Nada hasta el momento
18402	WPmQghvxeTdFG5Fw	46. Comentarios adicionales que usted pueda pr...	Excelente docente.
18403	7hh-Z9LiSlIVynFN	46. Comentarios adicionales que usted pueda pr...	No mandar tantos trabajos

18404 rows × 3 columns

Ilustración 8. Carga de datos del segundo archivo

Luego, se integraron los archivos en un único dataframe, obteniendo un total de 36808 registros.

```
df = pd.concat([df1, df2])
df
```

✓ 0.0s

	CodUnion2	Pregunta	Respuesta
0	JmlNpqV5aBe-kiNX	46. Comentarios adicionales que usted pueda pr...	Buena clase
1	9h_4mbQRvKK1qw6N	46. Comentarios adicionales que usted pueda pr...	Ninguno.
2	mQxALHD3wYn6BnUC	46. Comentarios adicionales que usted pueda pr...	Excelente docente
3	dIt4w57MdiNuaW8g	46. Comentarios adicionales que usted pueda pr...	Excelente Docente
4	3kahMRgdFQOwNvkH	46. Comentarios adicionales que usted pueda pr...	N/D
...	...	...	...
18399	6aZt85NoQKflqfve	46. Comentarios adicionales que usted pueda pr...	ser más paciente
18400	3QbsyEoVqRUpmO4c	46. Comentarios adicionales que usted pueda pr...	Nd
18401	ZgOIbzEfsyO1weAs	46. Comentarios adicionales que usted pueda pr...	Nada hasta el momento
18402	WPmQghvxeTdFG5Fw	46. Comentarios adicionales que usted pueda pr...	Excelente docente.
18403	7hh-Z9LiSlIVynFN	46. Comentarios adicionales que usted pueda pr...	No mandar tantos trabajos

36808 rows × 3 columns

Ilustración 9. Concatenación de los dataframe

Posteriormente, se verificó la existencia de valores nulos en el campo de las respuestas y de registros duplicados. Tras eliminar estos casos, el dataframe final quedó compuesto por 22088 registros.

Ilustración 10. Eliminación de nulos y duplicados

A continuación, los datos procesados fueron exportados en formato CSV para facilitar su exploración manual.

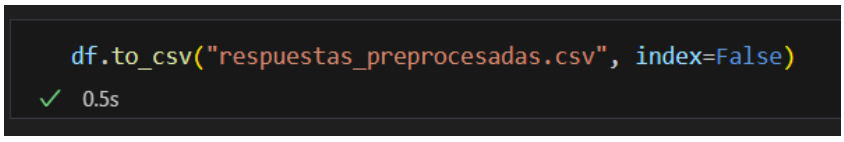

Ilustración 11. Almacenamiento de los datos

Durante esta revisión, se identificaron respuestas que no aportan información significativa, como aquellas compuestas únicamente por caracteres especiales (puntos, comas, entre otros), posiblemente debido a que algunos estudiantes completaron la encuesta por obligación. Asimismo, se observó la presencia de jerga estudiantil, expresiones informales y el uso de emojis, elementos que deberán considerarse durante las etapas de limpieza y preprocesamiento de datos.

A continuación, se puede ver algunos ejemplos de respuestas en bruto:

Excelente profe :), trabajos dinámicos y nos refuerza con ejemplos si un tema no fue comprendido del todo. Así mismo la Importancia de la asignatura en nuestra carrera y vida.
La mejor profe que existe, super buena, exigente (positivo), he aprendido mucho en su materia, me cayó muy bien y me alegra sentir que en cualquier situación puedo contar con ella, sea estudiantil, profesional o hasta personal.
Utilizar otras herramientas a parte de WSO2
sus clases son interesantes pero siento que debería ser unas dos horas al día porque 3 horas es muy cansado y se pierde el interés en la clase de ahí todo esta bien.
¡Es una docente increíble!

Tabla 1. Muestra de datos en bruto

4.3 Limpieza y preprocesamiento de datos

Antes de poder utilizar las respuestas de los estudiantes para entrenar el modelo de análisis de sentimientos, fue necesario someterlas a un proceso exhaustivo de limpieza y preprocesamiento. Este paso es crucial para asegurar la calidad de los datos y optimizar el rendimiento del modelo, previamente a procesar cada respuesta se eliminaron los registros con respuestas duplicadas y quedaron un total de 22088 registros únicos.

Otro problema que enfrentamos previo al entrenamiento del modelo es la ausencia de etiquetas en el dataset, para solucionar este inconveniente se utilizó un modelo pre-entrenado para realizar un etiquetado inicial de los datos. Para esto utilizamos el toolkit de NLP pysentimiento.

```
from pysentimiento import create_analyzer
✓ 2.3s
WARNING:tensorflow:From c:\Users\Jaime\anaconda3\envs\tesis\

analyzer = create_analyzer(task="sentiment", lang="es")
✓ 51.6s
```

Ilustración 12. Inicialización del modelo

Después, se cargaron los datos del archivo CSV que se generó anteriormente tras concatenar los datasets y eliminar registros duplicados, se iteró sobre las respuestas de los estudiantes y se etiquetaron.

```
import pandas as pd
✓ 0.0s

df = pd.read_csv("respuestas_preprocesadas.csv")
✓ 1.2s

sentiments = []
for text in df["Respuesta"].tolist():
    try:
        sentiments.append(analyzer.predict(text).output)
    except TypeError:
        sentiments.append("NEU")
df["Sentimiento"] = sentiments
✓ 13m 30.5s
```

Ilustración 13. Etiquetación de datos

Luego, se almacenaron los datos etiquetados en un nuevo archivo CSV, en el cuál nos basaremos para realizar la limpieza de las respuestas y posteriormente entrenar nuestra red neuronal.

```
df.to_csv("respuestas_etiquetadas.csv", index=False)
✓ 0.3s
```

Ilustración 14. Almacenamiento de datos etiquetados

El preprocesamiento de texto se realizó utilizando técnicas de NLP y las librerías especializadas de Python, como NLTK y spaCy. El proceso incluyó las siguientes etapas:

Se utilizaron las siguientes funciones en Python para realizar este proceso de limpieza:

```python
def preprocess_text(text):
    #Eliminar caracteres especiales y espacios en blanco
    text = str(text).lower()
    text = text.replace(".", "")
    text = text.replace(",", "")
    text = re.sub(r'[^a-záéíóúñü\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()

    return text

def texts_to_sequences(texts):
    #Convertir textos a secuencias
    sequences = tokenizer.texts_to_sequences(texts)
    return pad_sequences(sequences, maxlen=max_len)
```

Ilustración 15. Función de preprocesamiento de texto

```python
tokenized_texts = [
    [token.lemma_.lower() for token in nlp(text)
     if not token.is_punct and not token.is_space and not token.is_stop]
    for text in texts
]
```

Ilustración 16. Función de lematización y limpieza de stopwords

1. Limpieza de texto:

- Eliminación de caracteres especiales y signos de puntuación
- Normalización de espacios en blanco
- Conversión de todo el texto a minúsculas para un tratamiento uniforme

Se puede observar los resultados de esta fase en la siguiente tabla:

excelente profe trabajos dinámicos y nos refuerza con ejemplos si un tema no fue comprendido del todo así mismo la importancia de la asignatura en nuestra carrera y vida
la mejor profe que existe super buena exigente positivo he aprendido mucho

en su materia me cayó muy bien y me alegra sentir que en cualquier situación puedo contar con ella sea estudiantil profesional o hasta personal
utilizar otras herramientas a parte de wso
sus clases son interesantes pero siento que debería ser unas dos horas al día porque horas es muy cansado y se pierde el interés en la clase de ahí todo esta bien
es una docente increíble

Tabla 2. Muestra de datos preprocesados

2. Tokenización:

- División del texto en unidades significativas llamadas tokens (palabras, números, etc.)
- Uso de expresiones regulares para manejar casos especiales

Los resultados de esta etapa de procesamiento se pueden ver a continuación:

['excelente', 'profe', 'trabajos', 'dinámicos', 'y', 'nos', 'refuerza', 'con', 'ejemplos', 'si', 'un', 'tema', 'no', 'fue', 'comprendido', 'del', 'todo', 'así', 'mismo', 'la', 'importancia', 'de', 'la', 'asignatura', 'en', 'nuestra', 'carrera', 'y', 'vida']
['la', 'mejor', 'profe', 'que', 'existe', 'super', 'buena', 'exigente', 'positivo', 'he', 'aprendido', 'mucho', 'en', 'su', 'materia', 'me', 'cayó', 'muy', 'bien', 'y', 'me', 'alegra', 'sentir', 'que', 'en', 'cualquier', 'situación', 'puedo', 'contar', 'con', 'ella', 'sea', 'estudiantil', 'profesional', 'o', 'hasta', 'personal']
['utilizar', 'otras', 'herramientas', 'a', 'parte', 'de', 'wso']
['sus', 'clases', 'son', 'interesantes', 'pero', 'siento', 'que', 'debería', 'ser', 'unas', 'dos', 'horas', 'al', 'día', 'porque', 'horas', 'es', 'muy', 'cansado', 'y', 'se', 'pierde', 'el', 'interés', 'en', 'la', 'clase', 'de', 'ahí', 'todo', 'esta', 'bien']
['es', 'una', 'docente', 'increíble']

Tabla 3. Tokenización de datos

3. Eliminación de palabras vacías (stop words):

- Filtrado de palabras comunes que no aportan significado (preposiciones, artículos, etc.)
- Uso de listas predefinidas de stop words en español

Las respuestas sin stop words se pueden ver en la siguiente tabla:

['excelente', 'profe', 'trabajos', 'dinámicos', 'refuerza', 'ejemplos', 'tema', 'comprendido', 'importancia', 'asignatura', 'carrera', 'vida']
['profe', 'super', 'exigente', 'positivo', 'aprendido', 'materia', 'cayó', 'alegra', 'sentir', 'situación', 'contar', 'estudiantil', 'profesional', 'personal']
['utilizar', 'herramientas', 'wso']
['clases', 'interesantes', 'siento', 'debería', 'horas', 'horas', 'cansado', 'pierde', 'interés', 'clase']
['docente', 'increíble']

Tabla 4. Tokens sin stopwords

4. Lematización:

- Reducción de las palabras a su forma base o raíz (lema)
- Uso del modelo 'es_core_news_lg' de spaCy, entrenado en español

Se puede visualizar las palabras en su forma base a continuación:

['excelente', 'profe', 'trabajo', 'dinámico', 'reforzar', 'ejemplo', 'tema', 'comprender', 'importancia', 'asignatura', 'carrera', 'vida']
['profe', 'super', 'exigente', 'positivo', 'aprender', 'materia', 'caer', 'alegrar', 'sentir', 'situación', 'contar', 'estudiantil', 'profesional', 'personal']
['utilizar', 'herramienta', 'wso']
['clase', 'interesante', 'sentir', 'deber', 'hora', 'hora', 'cansado', 'perder', 'interés', 'clase']
['docente', 'increíble']

Tabla 5. Lematización de datos

Después del preprocesamiento, las respuestas quedaron limpias y estructuradas, listas para ser convertidas en representaciones numéricas aptas para el modelo de aprendizaje profundo. Se generaron secuencias de tokens de longitud fija (padding), y se crearon embeddings de palabras utilizando Word2Vec con un tamaño de vector de 200 y entrenado específicamente en el corpus de respuestas estudiantiles.

Se puede evidenciar la diferencia entre las respuestas en bruto y el texto completamente procesado en la siguiente tabla:

Texto en bruto	Texto procesado
Excelente profe :), trabajos dinámicos y nos refuerza con ejemplos si un tema no fue comprendido del todo. Así mismo la Importancia de la asignatura en nuestra carrera y vida.	['excelente', 'profe', 'trabajo', 'dinámico', 'reforzar', 'ejemplo', 'tema', 'comprender', 'importancia', 'asignatura', 'carrera', 'vida']

Tabla 6. Comparación de datos en bruto y procesados

4.4 Desarrollo y evaluación del modelo.

Para la selección de los modelos se tomó como base los trabajos previos de Kurniasari y Setyanto (2020), quiénes llevaron a cabo un estudio comparativo de distintos tipos de algoritmos para determinar cuál de estos es mejor para análisis de sentimientos, dentro de la comparativa se incluyó el algoritmo Naive Baiyes, el uso de CNNs en conjunto con word2vec, RNNs y RNNs en conjunto a word2vec, el resultado de su estudio determinó que el uso de RNNs en conjunto con word2vec alcanzaba la mayor exactitud de entre todos los algoritmos utilizados.

Por otro lado, Li y Qian (2016) realizaron un estudio comparativo del uso de RNNs convencionales y RNNs del tipo Long Short-Term Memory (LSTM) en el área de análisis de sentimientos con varios datasets distintos, los resultados de su artículo demuestran que las RNNs del tipo LSTM tienen una mejor exactitud en este tipo de tareas en comparativa a una RNN convencional.

Tomando en cuenta estos antecedentes se decidió entrenar, evaluar y comparar 3 modelos con distintos niveles de complejidad, que son: Naive Bayes con la

técnica de vectorización Bag of Words, Random Forest con el método de vectorización Tf-Idf y una red neuronal recurrente del tipo Long Short-Term Memory con la técnica de vectorización Word2Vec.

4.4.1 Naive Bayes

Para entrenar el modelo con el algoritmo Naive Bayes empezamos por transformar la variable de respuesta en numérica mediante el uso de un Label Encoder.

```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df["Sentimiento"])
✓ 0.0s
```

Ilustración 17. Transformación de la variable de respuesta

Luego, realizamos una división del dataset en datos de entrenamiento y de prueba para evaluar el rendimiento del modelo con valores del 80% y 20% respectivamente para cada conjunto.

```
X = list(texts)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
✓ 0.0s
```

Ilustración 18. División del dataset

Posteriormente, calculamos los vectores de las respuestas con la técnica Bag of Words.

```
count_vectorizer = CountVectorizer(max_features=10000)
X_train_bow = count_vectorizer.fit_transform(X_train)
X_test_bow = count_vectorizer.transform(X_test)
✓ 0.3s
```

Ilustración 19. Vectorización mediante Bag of Words

Después, inicializamos el modelo, lo entrenamos con los datos de prueba y medimos el tiempo que tardó en el entrenamiento y posteriormente en la predicción de los valores de prueba.

```
nb_model = MultinomialNB()
time_start = time.time()
nb_model.fit(X_train_bow, y_train)
nb_training_time = time.time() - time_start
✓ 0.0s
```

Ilustración 20. Entrenamiento del modelo Naive Bayes

```
time_start = time.time()
y_pred_nb = nb_model.predict(X_test_bow)
nb_predict_time = time.time() - time_start
0.0s
```

Ilustración 21. Predicción del modelo Naive Bayes

Para evaluar la eficiencia del modelo se tomaron en cuenta las métricas de precisión, recall, f1-score y la exactitud global sobre cada una de las clases, así como la matriz de confusión del dataset de prueba, los resultados son los siguientes:

	PRECISION	RECALL	F1-SCORE	SUPPORT
NEG	0.76	0.78	0.77	1007
NEU	0.72	0.56	0.63	1015
POS	0.86	0.93	0.89	2396
ACCURACY			0.81	4418
MACRO AVG	0.78	0.76	0.77	4418
WEIGHTED AVG	0.80	0.81	0.80	4418

Tabla 7. Resultados de Naive Bayes

Podemos ver que el modelo entrenado mediante Naive Bayes tuvo un buen rendimiento en cuanto a las respuestas con sentimientos negativos y positivos y presenta una exactitud global del 81% que es bastante buena, sin embargo, presenta un recall y f1-score bajos en comparación a los otros dos sentimientos, lo que nos demuestra que tiene problemas en identificar respuestas ambiguas.

	NEG	*NEU*	*POS*
NEG	790	112	105
NEU	177	572	266
POS	69	107	2220

Tabla 8. Matriz de confusión de Naive Bayes

Los resultados de la matriz de confusión muestran concordancia con las métricas anteriores, un buen resultado en el sentimiento positivo debido a la mayor

cantidad de datos en esta clase y la mayor cantidad de errores se muestra al clasificar sentimientos neutros como positivos.

4.4.2 Random Forest

Con el algoritmo de Random Forest se siguieron los mismos pasos iniciales que con Naive Bayes, es decir, se transformó la variable de respuesta mediante un Label Encoder y se realizó la partición del dataset. Tras estos pasos, se vectorizaron las respuestas utilizando Tf-Idf.

```
vectorizer = TfidfVectorizer(max_features=10000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

✓ 0.3s
```

Ilustración 22. Vectorización mediante Tf-Idf

Después, se instanció el modelo, se midió el tiempo de entrenamiento y de predicción con el algoritmo RandomForest.

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
time_start = time.time()
rf_model.fit(X_train_tfidf, y_train)
rf_training_time = time.time() - time_start
✓ 5.8s
```

Ilustración 23. Entrenamiento del modelo Random Forest

```
time_start = time.time()
y_pred_rf = rf_model.predict(X_test_tfidf)
rf_predict_time = time.time() - time_start
✓ 0.0s
```

Ilustración 24. Predicción del modelo Random Forest

Se utilizaron las mismas métricas de evaluación sobre este modelo para así poder comparar el rendimiento de cada uno con mayor facilidad, los resultados del modelo con el algoritmo Random Forest son los siguientes:

	PRECISION	RECALL	F1-SCORE	SUPPORT
NEG	0.76	0.71	0.73	1007
NEU	0.78	0.63	0.70	1015
POS	0.84	0.93	0.88	2396
ACCURACY			0.81	4418

MACRO AVG	0.79	0.76	0.77	4418
WEIGHTED AVG	0.81	0.81	0.80	4418

Tabla 9. Resultados de Random Forest

Podemos ver que Random Forest tuvo un buen rendimiento general, al igual que el modelo con Naive Bayes presenta una exactitud global del 81%, pero muestra un mejor rendimiento con los sentimientos neutros y baja ligeramente para los casos negativos.

	NEG	*NEU*	*POS*
NEG	720	85	202
NEU	151	639	225
POS	82	95	2219

Tabla 10. Matriz de confusión de Random Forest

La matriz de confusión nos muestra un buen rendimiento con los sentimientos positivos, pero también muestra que el modelo tiende a clasificar erróneamente los sentimientos negativos y neutros como positivos, lo cuál puede ser problemático en casos reales.

4.4.3 Red Neuronal Recurrente LSTM

Para entrenar la red neuronal necesitamos calcular los embeddings de las respuestas de los estudiantes, para esto vamos a aplicar la técnica Word2Vec y empezamos por inicializar el modelo.

```
w2v_model = Word2Vec(
    sentences=tokenized_texts,
    vector_size=embedding_dim,
    window=5,
    min_count=1,
    workers=4
)
```

Ilustración 25. Inicialización del modelo Word2Vec

El modelo de red neuronal toma como entrada secuencias de tokens de longitud fija, representadas por sus vectores de embedding, generamos esta matriz con el siguiente fragmento de código.

```
embedding_matrix = np.zeros((max_words, embedding_dim))
for word, i in tokenizer.word_index.items():
    if i >= max_words:
        break
    try:
        embedding_matrix[i] = w2v_model.wv[word]
    except KeyError:
        continue
```

✓ 0.0s

Ilustración 26. Matriz de embeddings

Luego, estas secuencias pasan a través de las siguientes capas:

1. Capa de Embedding: Inicializada con los pesos pre-entrenados de Word2Vec, permite representaciones densas y semánticamente significativas de las palabras en el contexto específico de las respuestas estudiantiles.
2. Capa de Dropout: Ayuda a prevenir el sobreajuste y mejora la generalización del modelo.
3. Capa LSTM Bidireccional: Captura las dependencias y el contexto tanto hacia adelante como hacia atrás en las secuencias de texto. Esto es especialmente útil para capturar el sentimiento general expresado en frases u oraciones completas.
4. Capa totalmente conectada con activación ReLU: Introduce no linealidad y aprende características de alto nivel a partir de las representaciones aprendidas por las capas anteriores.
5. Capa de salida con activación Softmax: Produce las probabilidades de pertenencia a cada clase de sentimiento (positivo, negativo, neutro).

El código utilizado para desarrollar el modelo descrito es el siguiente:

```python
def build_model(embedding_matrix, num_classes):
    #Construir la red neuronal recurrente
    model = Sequential([
        Embedding(max_words, embedding_dim,
                  weights=[embedding_matrix],
                  input_length=max_len),
        SpatialDropout1D(0.2),
        Bidirectional(LSTM(128, return_sequences=True)),
        Bidirectional(LSTM(64, return_sequences=False)),
        Dense(64, activation='relu'),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

Ilustración 27. Código de la red neuronal recurrente

A continuación, se puede observar una representación gráfica de la red neuronal utilizada.

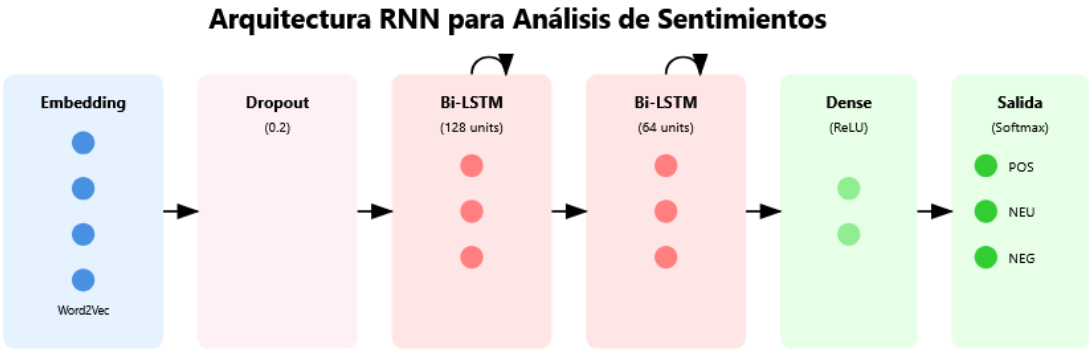

Ilustración 28. Representación de la red neuronal recurrente

El conjunto de datos de respuestas etiquetadas se dividió en subconjuntos de entrenamiento (80%) y prueba (20%), estratificados según las etiquetas de sentimiento para garantizar una representación balanceada de las clases.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
✓ 0.1s
```

Ilustración 29. División del dataset

El modelo se entrenó utilizando la función de pérdida de entropía cruzada categórica y el optimizador Adam con una tasa de aprendizaje de 0.001. Se aplicó un early stopping basado en la pérdida de validación para evitar el sobreajuste. El entrenamiento se realizó durante 10 épocas con un tamaño de batch de 32.

```
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    min_delta=0.001,
    restore_best_weights=True
)
✓ 0.0s

model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=10,
    batch_size=32,
    callbacks=[early_stopping],
    verbose=1
)
✓ 5m 46.3s
```

Ilustración 30. Entrenamiento de la red neuronal

Cabe mencionar que debido al uso de early stopping, el modelo no llegó a entrenarse durante las 10 épocas establecidas en ninguna ejecución del código, el entrenamiento se completó entre 4 y 6 épocas durante las distintas pruebas..

Los resultados de evaluación en el conjunto de prueba, presentados en el reporte de clasificación, muestran un excelente desempeño del modelo:

	PRECISION	RECALL	F1-SCORE	SUPPORT
NEG	0.84	0.79	0.82	1007
NEU	0.76	0.68	0.72	1015
POS	0.89	0.95	0.92	2396

ACCURACY			0.85	4418
MACRO AVG	0.83	0.81	0.82	4418
WEIGHTED AVG	0.85	0.85	0.85	4418

Tabla 11. Resultados de la RNN

Se obtuvo una exactitud global del 85%, con una precisión y recall balanceados en todas las clases. La clase positiva alcanzó el mejor f1-score de 0.92, seguida por la negativa con 0.82. La clase neutra presentó el rendimiento más bajo, pero aun así alcanzó un f1-score respetable de 0.72.

La matriz de confusión muestra una clara separación entre las clases, con pocos errores de clasificación entre sentimientos opuestos.

	NEG	*NEU*	*POS*
NEG	793	126	88
NEU	127	693	195
POS	19	92	2285

Tabla 12. Matriz de confusión de la RNN

4.5 Comparación de resultados.

Tras haber entrenado y validado la eficiencia de 3 modelos distintos, se puede comparar el rendimiento de cada uno con las métricas utilizadas anteriormente, en el siguiente gráfico podemos notar que en cuanto a rendimiento el modelo más complejo utilizando Word2Vec y redes neuronales recurrentes logró los mejores resultados de clasificación. Por otro lado, Naive Bayes y Random Forest presentaron rendimientos similares, ambos con una exactitud global casi igual y ligeras diferencias en cuanto a precisión y f1-score, con base en las matrices de confusión podemos notar que Naive Bayes presentó mayor cantidad de errores con sentimientos neutros mientras que Random Forest los presentó con sentimientos negativos.

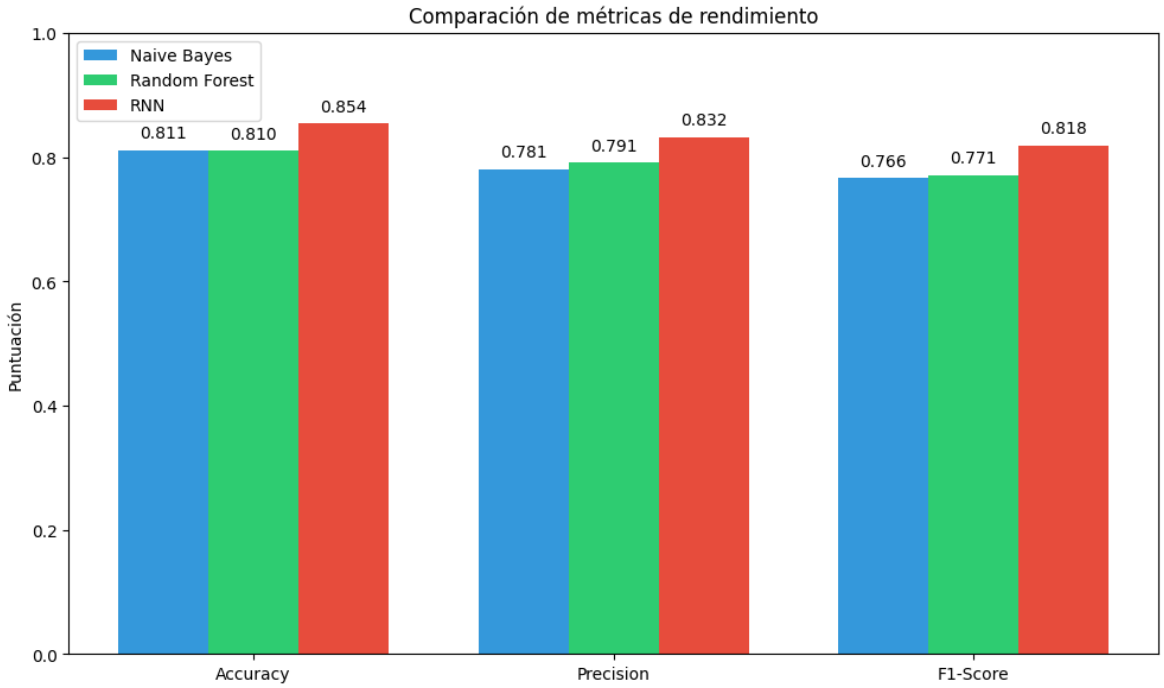

Ilustración 31. Comparación de los modelos

Además, se tomó en cuenta otra métrica importante que es el tiempo de entrenamiento y el de predicción que de forma indirecta nos dan una idea del costo de recursos computacionales necesarios para utilizar los modelos. Cabe mencionar que para el desarrollo del presente trabajo el computador utilizado cuenta con las siguientes características:

- Procesador Intel Core i7 con 4.2 GHz.
- 32 GB de memoria RAM DDR4.
- Tarjeta Gráfica NVIDIA GeForce GTX 1080 con 8 GB de memoria RAM.

El uso de la tarjeta gráfica marca una notable diferencia en el tiempo de entrenamiento y predicción, sobre todo de la red neuronal que fue desarrollada con el framework Tensorflow que soporta el uso de tarjetas gráficas dedicadas. Tomando en cuenta estas consideraciones, a continuación, se puede observar la diferencia de tiempo de uso de los 3 modelos:

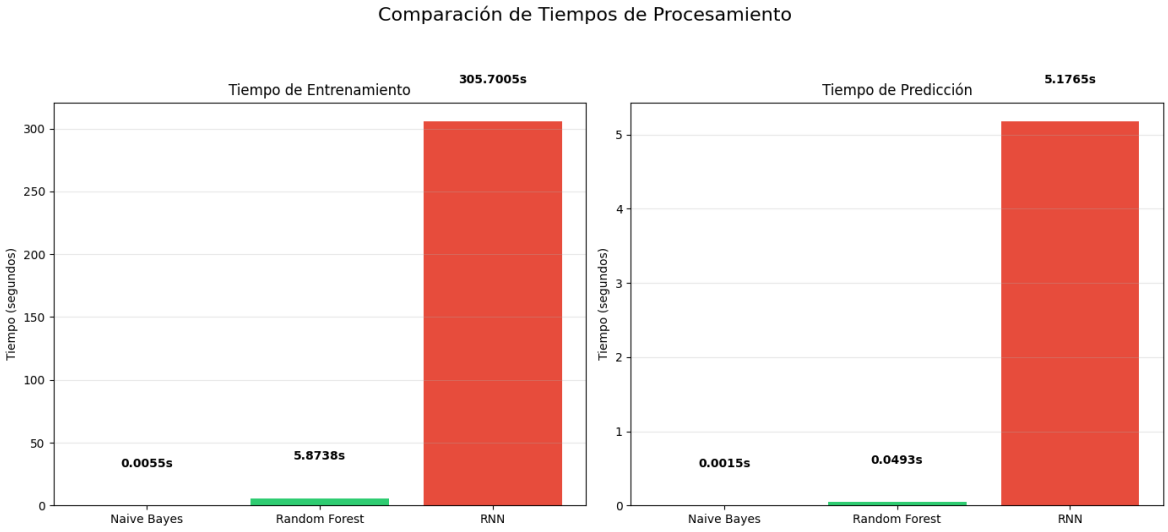

Ilustración 32. Tiempo de ejecución de los modelos

A pesar de la diferencia de magnitudes del tiempo de entrenamiento y del de predicción, notamos que tienen proporciones similares, siendo la red neuronal la que tarda mucho más tiempo y el algoritmo de Naive Bayes siendo el más rápido. Esto nos muestra una clara desventaja del uso de la red neuronal frente a los otros modelos, ya que es mucho más demandante de recursos computacionales.

CAPITULO V: CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- La arquitectura de una red neuronal recurrente de tipo Long Short-Term Memory en conjunto con la técnica de vectorización Word2Vec demostró ser la más eficiente al momento de capturar los matices semánticos y contextuales propios de un ambiente educativo y clasificarlos correctamente.
- A pesar de la diferencia de rendimiento, los 3 modelos lograron demostrar resultados competentes en cuanto a sus métricas globales, consiguiendo los 3 una exactitud global superior al 80%.
- La metodología CRISP-DM demostró ser el marco de trabajo indicado para estructurar el presente trabajo de titulación al permitirnos dar un enfoque sistemático al desarrollo del modelo, garantizando la obtención de buenos resultado.
- La clasificación de los sentimientos neutrales fue la más complicada para los 3 modelos al presentar métricas con los valores más bajos de las 3 clases, lo que nos indica la complejidad de identificar correctamente textos con una ausencia de polaridad.
- Los resultados del tiempo de ejecución de los modelos demuestran la capacidad de clasificar las respuestas de los estudiantes en tiempos mucho menores a lo que podría un ser humano, ya que incluso el modelo con redes neuronales que tardó más tiempo, logró clasificar el 20% del dataset en aproximadamente 5 segundos.

5.2 Recomendaciones

- Aplicar técnicas de balance de clases para mejorar el desempeño de los modelos, ya que se tuvo un mejor rendimiento en los sentimientos positivos que son los más numerosos en el dataset utilizado, esto podría mejorar las métricas de las clases neutro y negativa.
- Extender el alcance del presente trabajo de titulación más allá del análisis de sentimientos, llegando a obtener resultados de clasificación más granulares, por ejemplo, reseñas hablando de la metodología del docente, la puntualidad, el trato del docente hacia el estudiante, entre otros.

- Tomar en cuenta los recursos computacionales disponibles al momento de implementar un modelo de análisis de sentimientos, ya que, a pesar de mostrar resultados inferiores, los algoritmos Naive Bayes y Random Forest presentaron tiempos de ejecución mucho menores que la red neuronal recurrente.
- Planificar el proceso de implementación del modelo en un ambiente de producción para que la institución pueda darle uso y beneficiarse de los tiempos de análisis cortos, pertinentes y libres de sesgos que presenta el modelo.
- Continuar con el uso del marco de trabajo CRISP-DM para futuros trabajos de ciencia de datos y machine learning, debido a que su enfoque organizado y estructurado permite llevar a cabo este tipo de proyectos de la mejor manera.

BIBLIOGRAFÍA

1) Hajrizi, R. Pireva, Krenare. (2020). *Aspect-Based Sentiment Analysis in Education Domain*, University for Business and Technology, Prishine, Kosovo, https://doi.org/10.48550/arXiv.2010.01429
2) Bhalla, R. (2022). *A Review Paper on the Role of Sentiment Analysis in Quality Education*, SN Computer Science, https://doi.org/10.1007/s42979-022-01366-9
3) Gutierrez, G. De Luna, A. Ochoa, A. Hernandez, A. Ponce, Julio. Álvarez, M. Cossio, E. Nava, J. (2018), *A Sentiment Analysis Model to Analyze Students Reviews of Teacher Performance Using Support Vector Machines*, Distributed Computing and Artificial Intelligence, 14th International Conference, Advances in Intelligent Systems and Computing 620, https://doi.org/10.1007/978-3-319-62410-5_19
4) Altrabsheh, N., Gaber, M., & Cocea, M. (2013). *SA-E: Sentiment Analysis for Education*. Paper presented at 5th KES International Conference on Intelligent Decision Technologies, Sesimbra, Portugal. http://idt-13.kesinternational.org/
5) Chowdhary, K. (2020*), Fundamentals of Artificial Intelligence*, Springer
6) Sarkar, D. (2016), *Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Inisghts from Your Data*, Apress
7) Shaik, T. Tao, X. Dann, C. Xie, H. Li, Y. Galligan, L. (2023), *Sentiment análisis and opinión mining on educational data: a survey*, Natural Language Processing Journal, https://doi.org/10.1016/j.nlp.2022.100003
8) Liu, B. (2020), *Sentiment Analysis: Mining Opinions, Sentiments and Emotions,* Cambridge University Press
9) Mao, Y. Liu, Q. Zhang, Y. (2024), *Sentiment análisis methods, applications and challenges: a systematic literatura review*, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2024.102048
10) Webb, G. I., Keogh, E., & Miikkulainen, R. (2010). *Naïve Bayes*. Encyclopedia of machine learning, *15*(1), 713-714.

11) Noble, W. S. (2006). *What is a support vector machine?.* Nature biotechnology, *24*(12), 1565-1567.

12) Kingsford, C., y Salzberg, S. L. (2008). *What are decision trees?.* Nature biotechnology, *26*(9), 1011-1013.

13) Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). *How many trees in a random forest?*. In Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings 8 (pp. 154-168). Springer Berlin Heidelberg.

14) Gerón, A. (2019), *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems,* O'Reilly Media

15) Thakur, A. Konde, A. (2021), *Fundamentals of Neural Networks,* International Journal for Research in Applied Science & Engineering Technology, 9. 407-426. 10.22214/ijraset.2021.37362.

16) Wu, J. (2017), *Introduction to Convolutional Neural Networks*, Nanjing University.

17) Towards AI, (2023), *Beginners Guide to Convolutional Neural Networks from Scratch,* recuperado de: https://towardsai.net/p/machine-learning/beginner-guides-to-convolutional-neural-network-from-scratch-kuzushiji-mnist-75f42c175b21

18) Pan, Y. (2024). *Different Types of Neural Networks and Applications: Evidence from Feedforward, Convolutional and Recurrent Neural Networks. Highlights in Scienc*e, Engineering and Technology. 85. 247-255. 10.54097/6rn1wd81.

19) Science Learning Hub, (2024), *Neural network diagam*, recuperado de: https://www.sciencelearn.org.nz/images/5156-neural-network-diagram

20) Open Data Science, (2020), *Understanding the Mechanism and Types of Recurrent Neural Networks,* recuperado de: https://opendatascience.com/understanding-the-mechanism-and-types-of-recurring-neural-networks/

21) Parakatta, A. (2023), *Radial Basis Function,* recuperado de: https://medium.com/@parakatta/radial-basis-function-e3121f85a29a

22) Gomede, E. (2023), *Exploring the Efficacy and Applications of Modular Neural Networks in Modern AI,* recuperado de: https://medium.com/the-modern-scientist/exploring-the-efficacy-and-applications-of-modular-neural-networks-in-modern-ai-3ea3d56950a9

23) Abubakar, H.D., Umar, M. Bakale, A. (2022). *Sentiment Classification: Review of Text Vectorization Methods: Bag of Words, Tf-Idf, Word2vec and Doc2vec*. SLU Journal of Science and Technology. Doi: https://doi.org/10.56471/slujst.v4i.266

24) Qader, W. M. Ameen, M. & Ahmed, B. (2019). *An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges*. 2019 International Engineering Conference (IEC). 200-204. 10.1109/IEC47844.2019.8950616.

25) Havrlant, L., & Kreinovich, V. (2017). A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation). *International Journal of General Systems*, *46*(1), 27–36. https://doi.org/10.1080/03081079.2017.1291635

26) Lilleberg, J., Zhu, Y., & Zhang, Y. (2015, July). Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)* (pp. 136-140). IEEE.

27) Di Gennaro, G., Buonanno, A., & Palmieri, F. A. (2021). Considerations about learning Word2Vec. *The Journal of Supercomputing*, 1-16.

28) Lau, J. H. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *arXiv preprint arXiv:1607.05368*.

29) Dogru, H. B., Tilki, S., Jamil, A., & Hameed, A. A. (2021, April). Deep learning-based classification of news texts using doc2vec model. In

2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA) (pp. 91-96). IEEE.

30) Kurniasari, L. Setyanto, A. (2020), *Sentiment Analysis using Recurrent Neural Networks,* Phys.: Conf. Ser. **1471** 012018, doi: 10.1088/1742-6596/1471/1/012018

31) Amazon Web Services, (2024), *¿Qué es Python?,* recuperado de: https://aws.amazon.com/es/what-is/python/

32) Pandas, (2024), *Package Overview,* recuperado de: https://pandas.pydata.org/docs/getting_started/overview.html

33) Numpy, (2024), *What is Numpy?,* recuperado de: https://numpy.org/doc/stable/user/whatisnumpy.html#whatisnumpy

34) Rehurek, R. (2024), *What is Gensim?,* recuperado de: https://radimrehurek.com/gensim/intro.html

35) Explosion (2025), *spaCy 101: Everything you need to know,* recuperado de: https://spacy.io/usage/spacy-101

36) Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). TensorFlow: a system for Large-Scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265-283).

37) Singgalen, Y. (2024). *Travel Content Evaluation through Sentiment and Toxicity Analysis using CRISP-DM*. Building of Informatics Technology and Science (BITS). 6. 365-377. 10.47065/bits.v6i1.5397.

38) Li, D., & Qian, J. (2016, October). *Text sentiment analysis based on long short-term memory*. In 2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI) (pp. 471-475). IEEE.

ANEXO 1

Enlace del código utilizado: https://github.com/Jagsec/trabajo_titulacion_maestria