

**MÉTODO AUTOMATIZADO BASADO EN INTELIGENCIA ARTIFICIAL PARA
IDENTIFICAR MENSAJES DE VIOLENCIA EN CONTRA DE LA MUJER EN
LETRAS DE CANCIONES**

DIDIER ENRIQUE RAMÍREZ DÍAZ

INSTITUCIÓN UNIVERSITARIA COLEGIO MAYOR DEL CAUCA

FACULTAD DE INGENIERÍA

INGENIERÍA INFORMÁTICA

POPAYÁN

2020

**MÉTODO AUTOMATIZADO BASADO EN INTELIGENCIA ARTIFICIAL PARA
IDENTIFICAR MENSAJES DE VIOLENCIA EN CONTRA DE LA MUJER EN
LETRAS DE CANCIONES**

DIDIER ENRIQUE RAMÍREZ DÍAZ

DIRECTORA:

MSC. MARIA ISABEL VIDAL CAICEDO

INSTITUCIÓN UNIVERSITARIA COLEGIO MAYOR DEL CAUCA

FACULTAD DE INGENIERÍA

INGENIERÍA INFORMÁTICA

POPAYÁN

2020

TABLA DE CONTENIDO

CAPÍTULO 1.	
PRELIMINARES	4
1.1 PLANTEAMIENTO DEL PROBLEMA	4
1.2 JUSTIFICACIÓN	6
1.3. ALCANCE	7
1.4. MARCO TEÓRICO	7
1.4.1. Violencia De Género Contra La Mujer.	8
1.4.2. Inteligencia Artificial.	8
1.4.3. Machine Learning.	8
1.4.4. Redes Neuronales Convolucionales.	9
1.4.5. Clasificación de Textos.	9
1.5. ANTECEDENTES	9
1.6. OBJETIVOS	13
1.6.1 Objetivo General	13
1.6.2. Objetivos Específicos.	13
1.7. METODOLOGÍA.	14
CAPÍTULO 2. BASE CONCEPTUAL SOBRE EL ESTADO DEL DESARROLLO ACTUAL DE HERRAMIENTAS Y PROCESOS QUE PERMITAN LA IDENTIFICACIÓN DE PATRONES EN TEXTO.	16
CAPÍTULO 3. TÉCNICAS DE INTELIGENCIA ARTIFICIAL UTILIZADAS ACTUALMENTE PARA LA DETECCIÓN DE PATRONES TEXTUALES	28
3.1.ARQUITECTURA DE REDES NEURONALES CONVOLUCIONALES	33
CAPÍTULO 4. CONSTRUIR UN MÓDULO SOFTWARE BASADO EN LAS TÉCNICAS DE INTELIGENCIA ARTIFICIAL, QUE PERMITA LA IDENTIFICACIÓN Y EL ANÁLISIS AUTOMÁTICO DE PATRONES DE VIOLENCIA CONTRA LA MUJER EN EL TEXTO DE LAS LETRAS DE CANCIONES	38

4.1 FASE 1: COMPRENSIÓN DE LOS DATOS Y EL NEGOCIO	40
4.1.1 Alcance De La Aplicación ML	40
4.1.2 Criterios De Éxito.	41
4.1.3 Factibilidad.	41
4.1.4 Recolección de datos.	42
4.1.5 Verificación En La Calidad De Los Datos.	49
4.2 FASE 2: PREPARACIÓN DE DATOS.	54
4.2.1 Seleccionar Los Datos	54
4.2.2 Limpiar Los Datos.	54
4.2.3 Construir Los Datos.	56
4.2.4 Estandarización De Datos.	56
4.3 FASE 3: MODELADO.	57
4.3.1 Investigación De La Literatura Sobre Problemas Similares.	57
4.3.2 Definir Medidas De Calidad Del Modelo.	57
4.3.3 Seleccionar El Modelo.	57
4.3.4 Entrenamiento Del Modelo.	58
4.4 FASE 4: EVALUACIÓN.	60
4.4.1 Validar desempeño.	60
4.4.2 Comparar Resultados De Criterios De Éxito Con Los Definidos.	70
4.5 FASE 5: DESPLIEGUE.	73
4.5.1 Evaluación Del Modelo En Condiciones De Producción.	73
4.5.2 Estrategia De Implantación.	74
4.6. FASE 6. MONITOREO Y MANTENIMIENTO.	74
4.6.1 Monitorear datos.	74
4.6.2 Actualización de librerías y datos.	74

CAPÍTULO 1. PRELIMINARES

1.1 PLANTEAMIENTO DEL PROBLEMA

Desde hace mucho tiempo se ha identificado que existe un nivel de desigualdad entre las personas. Esto se ha visto reflejado en la violencia de género, específicamente en contra de la mujer. La cual se puede presenciar de muchas maneras, no importa la edad, nacionalidad, cultura, color de piel, etc. Son factores que además de afectar a la víctima, puede afectar al entorno que las rodea, como: hijos, amigos, familiares, compañeros, etc.

La violencia contra la mujer se presenta como un contexto de desigualdad estructural para las mujeres en cuanto a menos posibilidades de acceso a recursos, autonomía económica y participación en la vida pública, lo que lleva a prestar atención sobre los factores que lo causan y las consecuencias que eso conlleva hacia las víctimas.

Según el informe del Instituto Nacional de Medicina Legal en Colombia (INMLCF) en el 2014 fueron asesinadas 1.007 mujeres, se registraron 37.881 casos de violencia contra las mujeres en el ámbito de la pareja y 16.088 casos de violencia sexual fueron contra mujeres, el 86% del total de las víctimas de este delito, siendo además las niñas y las adolescentes las principales afectadas por esta forma de violencia. La persistencia de la violencia contra las mujeres se explica por hallazgos de estudios, como el de la Segunda Medición sobre Tolerancia Social e Institucional a la Violencia Basada en Género presentado por Consejería Presidencial para la Equidad de la Mujer, según el cual un 45% de las personas encuestadas opina que “las mujeres que siguen con sus parejas después de ser golpeadas son porque les gusta” (Mujeres, 2017).

La violencia contra la mujer es un acto que puede producir un daño físico, psicológico o emocional que se puede presentar verbal o físico en cualquier

contexto. Los siguientes son algunos tipos de violencia que se pueden presentar individualmente o en conjunto:

Violencia laboral

Violencia económica

Violencia institucional

Violencia psicológica

Violencia física

Violencia sexual

Violencia simbólica

A partir de lo expuesto anteriormente, se tomará como caso de estudio específico el análisis de letras de canciones que violentan contra la mujer, esta es una iniciativa novedosa que no tiene desarrollos en las plataformas digitales de música como Spotify, Deezer, Apple music, etc.

De esta manera se pretende hacer énfasis en una forma de violencia en específico, las canciones. De esta manera se puede percibir muchos tipos de violencia de las cuales se mencionarán algunas que siempre están presentes y quizá no se perciben de esa manera, como, por ejemplo: Violencia sexual, psicológica, emocional, entre otras. En el presente año 2020 se hizo una denuncia en Twitter por parte de una usuaria, esta mujer aseguraba que existía un cantante que incitaba a la violencia de formas muy explícitas en sus canciones, esto es lo que la mujer escribió: “Hey qué onda [@Spotify LATAM](#) oye y si en vez de quitarnos Safaera de [@sanbenito](#) quitas las canciones de Johnny Escutia en donde describe cómo violar, torturar, matar y desmembrar mujeres y niñas?” (Luz, 2020). Luego de esta declaración Spotify y otras plataformas reconocidas, decidieron quitar la “música” de este cantante de sus listas de reproducción.

Debido a lo anterior y a todo el problema que se vive con la violencia contra la mujer, se propone la investigación e implantación de un prototipo funcional que permita el

reconocimiento de palabras, frases o conductas presentadas dentro de las canciones que puedan promover la violencia contra la mujer.

1.2 JUSTIFICACIÓN

Considerando que la violencia de género contra la mujer se refleja y se aplica en muchos contextos, por ejemplo, en la postulación a cargos públicos que “solo pueden ser ocupados por hombres” o trabajos del hogar que “solo pueden ser desempeñados por mujeres” (Colombia, 2015). Como estos ejemplos hay muchos más. Sin embargo, este proyecto busca dar un enfoque de género, respecto a la violencia contra la mujer que puede presentarse de manera desapercibida en las letras de canciones de diferentes géneros musicales. (Díez, 2015)

En este proyecto se propone una alternativa basada en inteligencia artificial para el análisis de violencia de género contra la mujer en letras de canciones, haciendo uso de inteligencia artificial, redes convolucionales, aplicaciones móviles o de escritorios y demás tecnologías que se puedan implementar para un mejor desempeño del sistema a crear. Se busca que el sistema sea autónomo y objetivo en cuanto a la clasificación de las letras de las canciones, evitando juicios de valor de los implicados en la investigación.

Así pues, este análisis busca replantear la manera en la que escuchamos y percibimos las letras de las canciones, ya que la violencia contra la mujer en este tipo de espacios es un fenómeno que se encuentra escondido y pasa desapercibido, no solo por jóvenes sino por adultos también, teniendo en cuenta muchos aspectos como, la edad de las personas que la escuchan, la cultura en la que viven, la aceptación de la sociedad sobre ciertos géneros musicales, la percepción sobre cada contenido puede variar en función de las categorías mencionadas anteriormente.

1.3. ALCANCE

Para el alcance de este proyecto se plantea sentar las bases de una profunda investigación aplicada en el desarrollo de un sistema que pueda identificar la violencia de género contra la mujer en letras de canciones, para el semillero de investigación de I+D y los estudiantes de ingeniería informática de la Institución Universitaria Colegio Mayor del Cauca, puedan continuar con el proceso de implementación tanto en inteligencia artificial como en violencia de género contra la mujer.

1.4. MARCO TEÓRICO

En este apartado, se definen algunas de las características más importantes que se tienen en cuenta para el desarrollo e implementación de esta investigación.

1.4.1. Violencia De Género Contra La Mujer.

Las Naciones Unidas definen la violencia contra la mujer como «todo acto de violencia de género que resulte, o pueda tener como resultado un daño físico, sexual o psicológico para la mujer, inclusive las amenazas de tales actos, la coacción o la privación arbitraria de libertad, tanto si se producen en la vida pública como en la privada» (OMS, 2017).

1.4.2. Inteligencia Artificial.

Darle una definición a la inteligencia artificial es algo que se ha tratado desde que los primeros conceptos empezaron a surgir, por lo tanto, una de las tantas definiciones que puede tener es que la inteligencia artificial es la combinación de una serie de tecnologías y conceptos que permiten emular características o capacidades exclusivas del ser humano, agregando también que estas deben tener una de las características más importantes del ser humano, la cual es autoaprendizaje. (SAS, 2020)

1.4.3. Machine Learning.

Machine learning es una forma de inteligencia artificial que consiste en que un sistema aprenda de los datos o la información que recibe. El sistema se basa en construir modelos predictivos más precisos, los cuales al recibir una cantidad de datos y tratarlos como una entrada, estos arrojan una salida. (IBM, 2018)

Existen categorías de machine learning, las cuales se verán a continuación:

Aprendizaje supervisado: Este tiene la función de encontrar patrones que persisten en datos y aplicarlos a procesos de analítica. Los datos que se emplean para el uso de este método tienen características de etiquetas, las cuales sirven para hacer la clasificación de conjuntos de datos muy grandes previamente establecidos (IBM, 2018).

Aprendizaje no supervisado: Este tipo de categoría se usa cuando la cantidad de datos es demasiado grande y contiene datos sin etiquetar. El aprendizaje no supervisado realiza el análisis de datos sin intervención humana además utiliza algoritmos que clasifican datos con base en patrones o clústeres que encuentra (IBM, 2018).

1.4.4. Redes Neuronales Convolucionales.

Las redes neuronales convolucionales son básicamente una arquitectura del Deep learning (aprendizaje profundo) que permite a los equipos computacionales procesar y clasificar todo tipo de imágenes y textos. Aunque su principal objetivo es el procesamiento de imágenes, en los últimos años se ha popularizado enormemente al conseguir resultados sorprendentes en el procesamiento de texto, impactando así en el área de visión por computador (Alvares, 2017).

De tal manera se puede decir que el propósito principal es detectar características o rasgos sintácticos en las oraciones como la distancia entre palabras, nombre, etc. Siendo una de las propiedades más importantes porque una vez se aprende la

característica en el lugar concreto de la oración la puede reconocer después en cualquier parte de la misma (Alvares, 2017).

1.4.5. Clasificación de Textos.

La clasificación de textos representa una parte fundamental en la investigación porque permite innovar como técnica para la identificación de palabras que puedan simbolizar violencia hacia la mujer. “la clasificación de texto tiene entre sus particularidades una alta dimensionalidad y desbalance entre categorías, lo que la distingue y la hace ser más complicada que la de otro tipo de datos, como por ejemplo imágenes” (Alvares, 2017)

1.5. ANTECEDENTES

Teniendo en cuenta las numerosas dificultades sociales que se presentan ocasionados por la violencia de género y las constantes búsquedas de igualdad que se persiguen en la actualidad, es labor de la academia apoyar este tipo de causas.

Más aún bajo cifras tan significativas como las descritas por el informe del Instituto Nacional de Medicina Legal en Colombia (INMLCF) en el 2014 fueron asesinadas 1.007 mujeres, se registraron 37.881 casos de violencia contra las mujeres en el ámbito de la pareja y 16.088 casos de violencia sexual fueron contra mujeres, el 86% del total de las víctimas de este delito, siendo además las niñas y las adolescentes las principales afectadas por esta forma de violencia.

Ello implica que se debe empezar a reevaluar no solo las manifestaciones más evidentes y opulentas de violencia contra la mujer sino extenderse a los espacios más cotidianos de dichas manifestaciones, las cuales pueden estar disimuladas en numerosos espacios culturales, políticos y sociales.

Para lograr un entendimiento más claro de lo mencionado anteriormente, este proyecto toma como referencia algunos proyectos e investigaciones se enfocan en el análisis de aspectos psicológicos a través de la tecnología.

Organización mundial de la salud

Según la organización mundial de la salud (OMS) “todo acto de violencia de género que resulte, o pueda tener como resultado un daño físico, sexual o psicológico para la mujer, inclusive las amenazas de tales actos, la coacción o la privación arbitraria de libertad, tanto si se producen en la vida pública como en la privada”, (OMS, 2017).

El 35% de las mujeres han sido objeto de violencia física o violencia sexual dentro o fuera de la pareja, según un estudio realizado en más de 80 países por la (OMS, 2017). Esto es demasiado grave teniendo en cuenta todo el daño psicológico que dejan los actos de esta magnitud generan muchas secuelas psicológicas para la persona que sufre de estos vejámenes, algunos de estos son: trastorno de personalidad, exposición al maltrato infantil, uso nocivo del alcohol, un acceso reducido de la mujer a un empleo bien remunerado, depresión, baja autoestima, actitudes que toleran la violencia.

El trabajo titulado Violencia contra la mujer en la música, una aproximación metodológica, en un proyecto de investigación por parte de la universidad Rey Juan Carlos en España, se formuló una propuesta de una aproximación metodológica acerca de la violencia contra las mujeres en la música. Este artículo plantea la idea de que el entorno creativo en que se escriben las canciones se ve reflejado en su temática, forma de pensar, forma de percibir la canción y cómo esto puede debatir los problemas sociales que son propios o están involucrados, como la violencia contra la mujer. “A partir de esto se da comienzo a la identificación de diferentes variables que permiten dar una caracterización de la violencia, efectos generadores, catalizadores y/o potenciadores, consecuencias de la violencia, la influencia y el apoyo parte del entorno” (Gómez Escarda, 2016).

Para poder realizar un análisis más congruente y que soporte la caracterización de las variables mencionadas, se propone que, en las características estructurales, se estudia el año que sale la canción, la década, el sexo del autor o autores, puesta en escena, estilo, público masivo o específico, etc. Teniendo que estas variables que se mencionan buscan dar a conocer en qué época se marca la violencia, analizando también que cuando a la canción se le aísla el sonido, esta deja de tener la misma interpretación. Para la caracterización de la violencia, se analiza si la canción la

fomenta o denuncia la violencia contra la mujer, si es violencia física o psicológica la que se genera con el contenido de la misma, el tipo de víctima (adulto o niño/a) a la cual va dirigida, destinatario de la canción (víctima, victimario, sociedad) ya que este puede hacer que el mensaje se perciba de manera diferente. Así continúa con la caracterización de los efectos generadores, potenciadores o catalizadores, se menciona que algunos pueden ser alcohol, drogas, los problemas sociales del agresor o de la víctima y malos tratos en la niñez. “Los estudios previos muestran que los niños y niñas expuestos a la violencia en el hogar (física, psicológica, sexual, emocional o intelectual) presentan mayores probabilidades de sufrir problemas emocionales y de conducta (ansiedad, depresión, mal rendimiento escolar, baja autoestima, pesadillas, etc.)” (Gómez Escarda, 2016).

El trabajo titulado Análisis de sentimientos en Twitter, (Sande, 2018), se basa en el análisis de sentimientos en la red social Twitter, haciendo uso de inteligencia artificial, procesamiento del lenguaje natural, machine learning y aprendizaje supervisado. Con la llegada de la web 2.0 el auge de las redes sociales ha venido en aumento, no solo para la distracción, sino también, para una nueva era de comercialización de productos o servicios. Así bien, el proyecto se enfoca en dar una opinión positiva o negativa acerca de un texto dado, ya sea hacia un producto, servicio, usuario o una empresa. El proyecto implementó un clasificador de sentimientos para los mensajes de Twitter basado en algoritmos de aprendizaje profundo.

Análisis de sentimientos y emociones sobre el discurso de firma del acuerdo de paz en Colombia

En esta investigación se hace referencia de nuevo al análisis de sentimientos, eso hace notar la importancia que está tomando este nuevo rumbo de investigación que usa también la inteligencia artificial, el aprendizaje profundo y la computación afectiva como bases para su desarrollo (G & Elizabeth, 2019). El análisis de sentimientos y emociones se realiza sobre el discurso del ex presidente Juan Manuel Santos en la firma del acuerdo de paz en Colombia, el cual se evalúan dos partes, el audio y el texto del discurso. Siendo Aurosal y Valencia (Pedro Angel Ruiz,

2019) las propiedades que se utilizaron para medir la emoción en pistas de audio y el portal ParallelDots para realizar el análisis de sentimientos en el texto. Para explicar los resultados se puede definir que la medición de las emociones se daba en 3 estados (feliz, triste o enojado) y los sentimientos se representaban en positivo o negativo. Ahora bien, los resultados arrojaban que la emoción feliz se presentaba en un 53% y el sentimiento positivo en un 58%, lo cual refleja que los análisis tenían congruencia en los valores obtenidos.

Se puede deducir que el uso de las tecnologías como aprendizaje profundo, inteligencia artificial, computación efectiva, redes convolucionales y demás, se puede aplicar en muchos de los ámbitos de la sociedad y que su uso puede traer mejoras a los procesos hechos a mano o de manera rudimentaria.

Solución Informática para el Reconocimiento de Acciones Básicas de Violencia en Tiempo Real, a partir del uso de Redes Neuronales Convolucionales, Secuencias de Video y Computación de Alto Desempeño

Este es un proyecto que se desarrolló en Perú (Laureano, 2019) el cual trabaja haciendo uso de redes neuronales convolucionales y técnicas de aprendizaje supervisado, para identificar por medio de imágenes la presencia de violencia física. El proceso consiste en crear una base de datos con imágenes y/o videos reales en los que exista una agresión física, como patadas y puños, con el fin de hacer que el sistema aprenda y más adelante pueda identificar imágenes o acciones similares a las que ya están presentes en la base de datos, luego de realizar los análisis dar un resultado sobre la coincidencia que se presenta entre la imagen que ve y las que ya existen. Los resultados arrojados en su momento indican un 84% de precisión en la clasificación de patadas y puños.

El nivel de aceptación de esa propuesta ha sido muy grande, lo cual llevó a que los investigadores de esta iniciativa participaran en el evento internacional denominado XLV Latin American Computing Conference en Panamá (UNAP, 2019).

1.6. OBJETIVOS

1.6.1 Objetivo General

Construir un método automatizado basado en Inteligencia Artificial para identificar mensajes de violencia en contra de la mujer en letras de canciones.

1.6.2. Objetivos Específicos.

Elaborar una base conceptual sobre el estado del desarrollo actual de herramientas y procesos que permitan la identificación de patrones en texto.

Identificar las técnicas de Inteligencia Artificial utilizadas actualmente para la detección de patrones textuales

Construir un módulo software basado en las técnicas de Inteligencia Artificial, que permita la identificación y el análisis automático de patrones de violencia contra la mujer en el texto de las letras de canciones.

1.7. METODOLOGÍA.

FASE 1: Elaborar una base conceptual sobre el estado del desarrollo actual de herramientas y procesos que permitan la identificación de patrones en texto

Actividades:

A1: Búsqueda de artículos científicos y literatura sobre proyectos que utilicen herramientas para la identificación de patrones en texto

A2: Construir un documento que consigne las herramientas y procesos utilizados en la identificación de patrones de texto

FASE 2: Identificar las técnicas de Inteligencia Artificial utilizadas actualmente para la detección de patrones textuales

Actividades

A3: Revisar las técnicas de inteligencia artificial utilizadas en la detección de patrones de texto

A4: Construir un cuadro comparativo de las técnicas de Inteligencia Artificial utilizadas

A5: Seleccionar la técnica de Inteligencia Artificial a utilizar

FASE 3: Construir un módulo software basado en las técnicas de Inteligencia Artificial, que permita la identificación y el análisis automático de patrones de violencia contra la mujer en el texto de las letras de canciones.

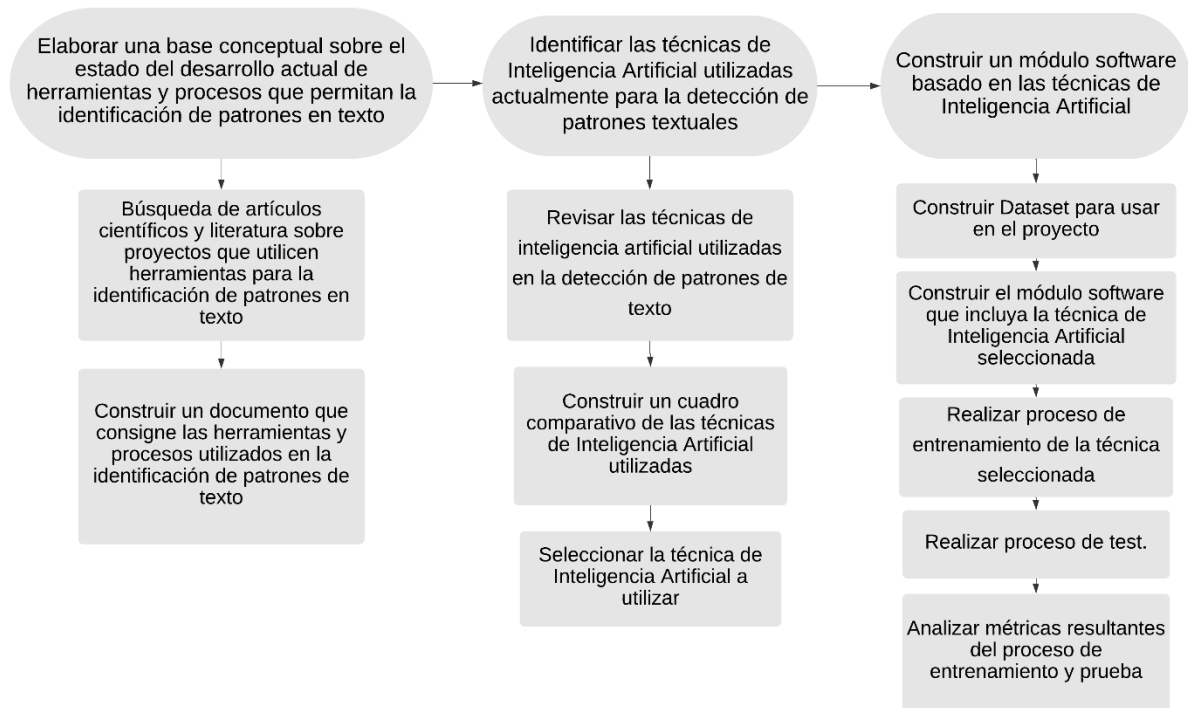
A6: Construir Dataset para usar en el proyecto

A7: Construir el módulo software que incluya la técnica de Inteligencia Artificial seleccionada

A8: Realizar proceso de entrenamiento de la técnica seleccionada

A9: Realizar proceso de test.

A10: Analizar métricas resultantes del proceso de entrenamiento y prueba



Gráfica 1 Metodología

CAPÍTULO 2

BASE CONCEPTUAL SOBRE EL ESTADO DEL DESARROLLO ACTUAL DE HERRAMIENTAS Y PROCESOS QUE PERMITAN LA IDENTIFICACIÓN DE PATRONES EN TEXTO.

En este capítulo se hablará sobre algunos trabajos, investigaciones y/o proyectos que se han desarrollado, los cuales cuentan con un amplio conocimiento acerca de la implementación e identificación de patrones de texto. A continuación, se presentan trabajos relacionados con la temática en donde se observa los conceptos claves para el desarrollo del proyecto

Reconocimiento De Agresión Verbal En Twitter Con El Uso De Patrones Lingüísticos

En este proyecto se hace un estudio basado en el análisis de sentimientos en tweets de una de las redes sociales más usadas en el mundo, la cual es Twitter. Para llevarla a cabo se requirió de una gran cantidad de tweets y clasificarlos según su agresividad. “La clasificación de textos tiene entre sus particularidades una alta dimensionalidad y desbalance entre categorías, lo que la distingue y la hace ser más complicada que la de otro tipo de datos, como por ejemplo imágenes. Por esta razón, es un problema abierto de gran desarrollo en las Ciencias de la Informática” (Alvares, 2017).

Según (Alvares, 2017), la definición formal de la clasificación de textos dice que, se hace una comparación entre el dominio de documentos y un conjunto de clases ya

predeterminadas, su objetivo es encontrar una función (regla, hipótesis, modelos, etc.) que busca la mejor opción de clasificación de los documentos. Para llevar a cabo esta tarea, el clasificador lee y utiliza técnicas de preprocesamiento de documentos para reducir su dimensión, técnicas de extracción y selección de las características. Sin embargo, la extracción de las características está compuesta por tres tareas, que son:

Técnicas De Procesamiento De Texto: Lo que hace es la eliminación de elementos redundantes sin que pierda significado, para obtener formas que ocupan menos recursos de cómputo.

Tokenización: “Consiste en separar el texto recibido en pequeñas partes denominadas token, ya sea hashtags, links, emoticones, palabras, palabras compuestas, etc. Esto permite procesar cada elemento, para luego aplicar detección de stopwords, enraizamiento, normalización del texto, etc. ya que son más fáciles de detectar, a su vez resulta más útil en la formación de N-gramas para reconocer significados que sólo la mezcla de conceptos puede generar.” (Alvares, 2017).

Eliminación De Stopwords Y Enlace: En esta tarea se eliminan palabras como verbos auxiliares, conjugaciones, entre otros. Además de eliminar enlaces que dentro de los tweets pueden estar insertados y que no aporten a la valoración del contenido (Alvares, 2017).

Lematizadores: La cual es una técnica que consiste en reducir una palabra a su raíz, para agrupar las que contengan la misma raíz y así evitar redundancias. Por ejemplo, “consentimiento, consentir, consentido” las tres se reducen a “consentir” (Alvares, 2017).

Las técnicas mencionadas anteriormente fueron abordadas en este proyecto. Así bien la clasificación de los textos busca que el clasificador no haga interpretaciones, sino que deduzca con la información que es suministrada.

Uso De Redes Neuronales Convolucionales Aplicado A Sentiment Analysis

Esta investigación realizada en la universidad de Chile pretende comprobar la aplicación de Deep learning en el análisis de textos. Sin embargo, en esta sección se explicará el proceso que se lleva a cabo con el procesamiento de texto. Según (López, 2019) plantea que la finalidad del procesamiento de texto es convertir los datos que entran al sistema en información con conocimiento y contexto. Aunque para lograr esto se deben seguir ciertas etapas, las cuales son: recogida, preparación, entrada de datos, procesamiento, interpretación y análisis. A continuación, se explica brevemente cada una de las etapas mencionadas.

Recogida: Es la primera y más relevante, se debe tener datos de calidad ya que estos determinan la base para determinar el sistema de métricas y la información acerca de los objetivos de los procesos de datos. (López, 2019)

Preparación: Aquí se manipulan los datos para darles un formato adecuado para un mejor análisis y procesamiento. (López, 2019)

Entrada de datos: Esta etapa requiere bastante tiempo, ya que es aquí donde se ingresan los datos, ya sea manual, digital o automatizada. Para así convertirla en información procesable.

Procesamiento: Aquí se evalúan, clasifican y organizan los datos para obtener información útil, a través de diversos métodos, cada uno con sus instrucciones.

Interpretación y análisis: “La información procesada se transmite al usuario de negocio, quien se ocupará de acceder a ella a través de informes, visualizaciones, montajes de vídeo o audio; para obtener el conocimiento que guiará las decisiones futuras de la empresa.” (López, 2019).

Además, dentro de las etapas mencionadas, específicamente en la etapa de procesamiento, se llevan a cabo algunos procesos o pasos que son indispensables para el procesamiento del texto, los cuales hacen que el texto recogido sea entendible y homogéneo. Ahora bien, los pasos que se deben llevar a cabo son:

Tokenización: “La tokenización es un paso que divide cadenas de texto más largas

en piezas más pequeñas o tokens. Los trozos de texto más grandes pueden ser convertidos en oraciones, las oraciones pueden ser tokenizadas en palabras, etc.” (Lopez, 2019).

Eliminación del ruido: Aquí se trata de eliminar elementos del texto que no aportan información relevante o importante. Por ejemplo, hashtag #, menciones @, direcciones web, URL, caracteres especiales. (Lopez, 2019).

Normalización: “La normalización generalmente se refiere a una serie de tareas relacionadas destinadas a poner todo el texto en igualdad de condiciones: convirtiendo todo el texto en el mismo caso (superior o inferior), eliminando la puntuación, convirtiendo los números a sus equivalentes de palabras, y así sucesivamente.” (Lopez, 2019).

Luego de realizar todos los procesos mencionados anteriormente se procede a trabajar con las redes neuronales que se mencionan en el próximo capítulo.

En el proyecto **Análisis De Emociones Y Sentimientos Sobre El Discurso De Firma Del Acuerdo De Paz En Colombia** (G & Elizabeth, 2019), se plantea el análisis de sentimientos de dos maneras, la primera, es el análisis de sentimientos en audios, la cual se hace a través de un software desarrollado en java que recibe audios como parámetros de entrada, para luego analizar y clasificar las emociones encontradas, según las propiedades de Arousal y Valencia, las cuales permiten la identificación de emociones como enojo, tristeza, felicidad, etc. (G & Elizabeth, 2019).

Sin embargo, en lo que concierne a este capítulo, la segunda manera en la que realizan el análisis de sentimientos, es a través de un portal web llamado ParallelDots, el cual permite analizar y clasificar textos en tres diferentes tipos de sentimientos: positivo, negativo y neutral. Este programa realiza la clasificación de texto de la siguiente manera:

Tareas y análisis de datos: esta etapa consiste en clasificar las emociones que se pueden encontrar a lo largo del texto, como: enojado, triste, feliz y otro. Esta tarea se realiza teniendo en cuenta la jerga, errores ortográficos y oraciones incompletas (Akansha Jain, 2019).

Pre procesamiento, en el cual se realizan tareas similares a los proyectos anteriores, aquí evitan eliminar los stopwords y Lematizadores, ya que al quitarlos podría conllevar a la pérdida de información, se convierten todas las palabras a minúscula y se eliminan los espacios en blanco junto a la puntuación repetida (Akansha Jain, 2019).

Palabras, sentencias y emojis incrustados, en este paso realizan la inserción de palabras, oraciones, sentencias y varios tipos de emojis en la inteligencia artificial, para poder hacer la comparación con los textos que se pretenden analizar, además de hacer uso de 1.3 billones de tweets de sentimiento, emoción y sarcasmo. (Akansha Jain, 2019).

Así de esta manera se logra el desarrollo del proyecto de análisis de sentimientos, dando una similitud en los resultados del audio y el texto recibido en el discurso de paz del expresidente Juan Manuel Santos.

Análisis De Sentimientos En Twitter

En este proyecto se pretende abarcar el área de investigación en el campo del procesamiento del lenguaje natural, cuyo objetivo es el tratamiento computacional de opiniones, sentimientos y subjetividad de los textos (Sande, 2018).

Para dar inicio a la explicación sobre cómo se realizó el análisis de sentimientos en este proyecto, se empezará mencionando que el corpus (conjunto cerrado de textos o de datos destinado a la investigación científica) se creó con base en mensajes de Twitter que ya estaban publicados (Sande, 2018). Ya teniendo el corpus de entrenamiento se deben seguir ciertos pasos para el correcto análisis de los textos:

Preprocesamiento, cuyo objetivo es limpiar y normalizar la información, ya que los mensajes extraídos de las redes sociales pueden tener errores de ortografía, repeticiones de caracteres, letras en mayúsculas o minúsculas, eliminación de hashtags, etiquetas, enlaces, eliminación de retweets, eliminación de números, eliminación de tildes, la normalización del uso de jergas y abreviaturas que pueden generar confusión y la normalización de risas (Sande, 2018).

Tokenización, que consiste en dividir los textos en unidades más pequeñas llamados tokens, de esta manera el análisis y la clasificación se hace de manera más sencilla (Sande, 2018).

Extracción de las características, para este paso se toman los tokens del paso anterior y se crean características con base en ellos, dándoles un significado.

Reducción de las características, en esta fase se pretende eliminar parte de las características y para ello, se utilizan las siguientes técnicas: En primer lugar, eliminación de stopwords, que básicamente elimina palabras que carecen de sentido y no aportan nada a la oración, por ejemplo: conectores, pronombres, etc. En segundo lugar, la lematización, que transforma cada palabra en su lema mediante el uso de diccionarios. Y, en tercer lugar, el stemming, que convierte a su raíz cada palabra por medio de la supresión de sufijos e inflexiones (Sande, 2018).

A partir de todo el proceso mencionado anteriormente, se procede con la clasificación y análisis de sentimientos, haciendo uso de programas o herramientas tecnológicas.

En el **Proyecto De Investigación De Análisis De La Dinámica Del Contenido Semántico De Textos** (Edgar Altszyler, 2015), se plantea el análisis de textos desde un enfoque semántico buscando identificar patrones similares en cualquier tipo de corpus (conjunto de datos). El corpus que se plantea analizar es proveniente de libros, en este caso de la saga de Harry Potter.

Para realizar el análisis se utilizó un algoritmo de procesamiento de lenguaje natural, llamado Análisis Semántico Latente (LSA) el cual sirve para reducir y cuantificar las

palabras semánticamente similares. Además de hacer el proceso semántico, puede identificar la cercanía de una palabra en un texto el cual no contenga esa palabra.

Para realizar el análisis de texto se tomaron corpus de novelas, noticias y TASA (Touchstone Applied Science Associates, Inc) que contiene una colección de material educativo de EEUU. Luego se analizó el idioma en el cual todos los textos debían estar en inglés. También se usan técnicas de preprocesamiento de texto como la eliminación de Stopwords, además del uso de Lematizadores a través de algoritmos que lleven cada palabra a su raíz para un mejor análisis (Edgar Altszyler, 2015).

En la parte final para hacer el análisis de la palabra “oscuridad” se hace uso de un software llamado *wortsurfer* el cual permite hacer una lista de palabras semánticamente cercanas, todo eso con el fin de evitar la polisemia que se puede presentar con la palabra mencionada, ya que “oscuridad” puede significar también “maldad”, “ignorancia” o “falta de luz”, así con la ayuda del programa mencionado solo busca similitudes con la primer interpretación que es “maldad (evil)” (Edgar Altszyler, 2015).

¿Qué Se Escribe Respecto Al Marxismo En Redes Sociales? Análisis De Patrones De Texto A Través De Twitter Por Medio De Data Mining.

“Este estudio trata de analizar por medio de patrones de texto las opiniones públicas en Twitter sobre el pensamiento crítico. El objetivo es obtener algunas nociones que permitan entender las expresiones e ideas que se gestan en esta red social.” (Olguín, 2015).

Para el desarrollo de este tema se tomaron más de 2.000 mensajes de Twitter o tweets que contengan la palabra “marxismo”, las cuales se usan para realizar el análisis que plantea el estudio. Las condiciones para la recolección de esos datos son dos: primero que los textos encontrados se encuentren en idioma español y segundo que la búsqueda se haga de manera aleatoria (Olguín, 2015).

Luego de obtener los datos se procede a realizar una limpieza dentro de los tweets, la cual consiste en eliminar números, signos de puntuación, enlaces de sitios web,

referencias a otros usuarios, texto sin sentido y cualquier otro tipo de datos que no sea analizable. A partir del paso anterior limpieza se genera un nuevo corpus o lista de información, la cual se procede con una nueva limpieza que convierte todo el texto a minúsculas y se remueven particularidades del español, como, pronombres, artículos, preposiciones, etc. Además, se hacen procesos que ya se han mencionado antes como los stopwords y Lematizadores (Olguín, 2015).

Como paso final del estudio se procede a la eliminación de las palabras “marxismo”, con el fin es evitar la sobreestimación de la palabra en el análisis del contenido, ya que es el patrón original de texto buscado inicialmente.

Uso Del Software Sisam Para La Detección De Contenidos Denigrantes Hacia La Mujer En Productos Audio Musicales Del Género Reggaetón 2019

El proyecto (Magino, 2020) plantea el análisis de letras de canciones del género de reggaetón, haciendo uso del análisis de sentimientos, con el fin de identificar contenido denigrante contra la mujer. Para el desarrollo de este proyecto se recolectaron canciones de reggaetón que comprenden desde el año 2009 hasta 2019, en las cuales se consiguieron las letras de más de 400 canciones tomadas de Spotify y algunas plataformas que muestran las letras de las canciones. Además, se clasificaron los mensajes en 4 diferentes tipos: violencia física, violencia psicológica, violencia sexual y violencia simbólica. Sin embargo, para la clasificación de lo mencionado anteriormente hacen uso de SISAM, que es un software para el análisis musical basado en análisis de sentimiento y diccionario léxico (Magino, 2020).

Ahora bien, en lo que concierne a este capítulo se va a analizar las técnicas de análisis de patrones de texto que se puedan extraer del proyecto (Magino, 2020). Por lo tanto, la primer fase llamada **flujo del análisis de sentimiento**, la cual se divide en **Pre procesamiento**, que consiste en transformar los textos de entrada a través de una serie de pasos, lo cuales son:

- Corrección de signos de puntuación.
- Corrección de abreviaturas, por ejemplo, (“xp” □ “porque”).

- Corrección de palabras.
- Normalización y acentuación de términos, que consiste en eliminar los caracteres repetidos en palabras, por ejemplo, (“largoooo” se reduce a “largo”).
- Eliminación del texto no útil.
- Algoritmo de Levenshtein, el cual consiste en realizar operaciones de inserción, sustitución y extracción de caracteres hasta dar con el mínimo número de cambios o distancias entre palabras (Jacobo, 2016).

La segunda fase son las **técnicas de clasificación de sentimientos u opinión**, en la que básicamente se trabajan con dos opciones, el enfoque basado en diccionario o el enfoque basado en el corpus, que ambas buscan dar un enfoque semántico a las palabras o textos, luego que han pasado por un proceso de lematización o stemming (Magino, 2020).

Luego está la fase **niveles del análisis de sentimiento**, que busca clasificar el sentimiento en 3 diferentes tipos de niveles: Nivel de documento, que mide cuál es la intención del documento, nivel de oración que determina si una opinión es neutral, positiva o negativa, por último, nivel aspecto entidad, la cual se centra en si la opinión está compuesta por un sentimiento y un objetivo (Magino, 2020).

Y por último los **intensificadores y negadores en el análisis de sentimientos**, los cuales se encarga de denotar si las expresiones o frases pueden incrementar o reducir su potencia de sentimiento, en el caso de los intensificadores. Y para el caso de los negadores se analizan ya que estos pueden generar polaridad en los textos (Magino, 2020).

Hugo Librado Jacobo en la tesis doctoral “**Análisis Automático De Opiniones De Productos En Redes Sociales**” (Jacobo, 2016), propone una metodología y el desarrollo de una aplicación que permite el análisis de textos tomados de la plataforma de Twitter, clasificándolos en 6 posibles estados, positivo, muy positivo, neutral, negativo, muy negativos, sin opinión (Jacobo, 2016).

El propósito de la aplicación consistió en analizar textos cortos sin importar el tema del que se hablará en cada uno, por ejemplo, política, fútbol, economía, etc. Ya que en cada tema la presentación de los productos no se ve de la misma manera, por ejemplo, la imagen de una persona, un bien o servicio de consumo (Jacobo, 2016).

Como primera fase, siendo común en la mayoría de las propuestas anteriores, es el **preprocesamiento**, que básicamente se encarga de arreglar el texto para poder realizar procesos de análisis adecuados, los cuales hacen que mejore en gran porcentaje de los resultados obtenidos. En esta fase se realizan tareas como: **Eliminación de texto no útil**, que se encarga de limpiar nombres, hashtags, direcciones web y demás características que no aportan información. **Normalización y acentuación de términos**, que se aplica a aquellos que han sido modificados por la repetición de caracteres (Jacobo, 2016). Así como en la propuesta anterior (Magino, 2020), se hace uso del algoritmo de Levenshtein, el cual consiste en realizar operaciones de inserción, sustitución y extracción de caracteres hasta dar con el mínimo número de cambios o hasta convertir una palabra en otra (Waki, 2017). Y como última parte del preprocesamiento, realizan un diagrama de secuencia del preprocesamiento, el cual lo usan para el corpus de entrenamiento y el corpus de prueba. (Jacobo, 2016).

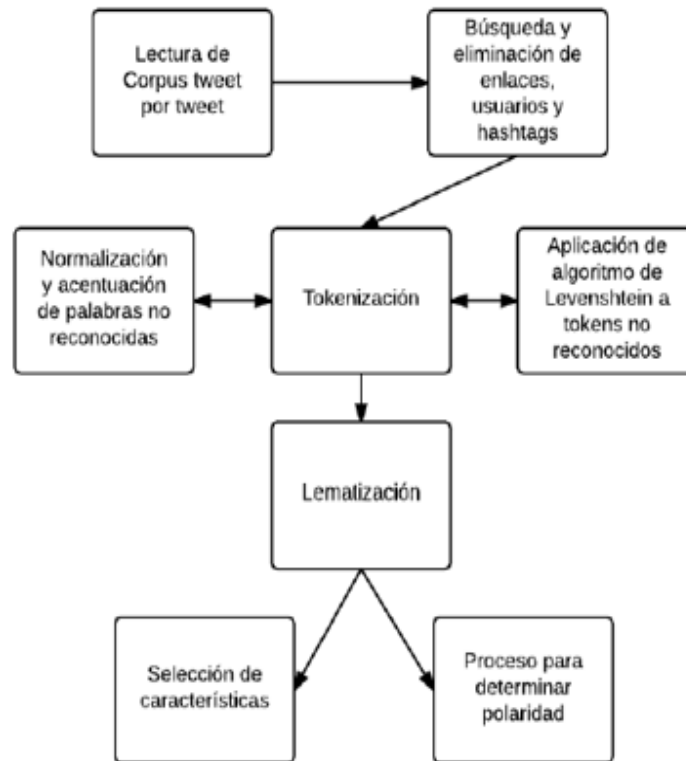


Figura 1 Ilustración de diagrama de secuencia del preprocesamiento

Fuente (Jacobo, 2016).

La segunda fase es **Selección de características**, en la cual se procede a hacer la eliminación de stopwords, que son el caso de algunos conectores que no aportan información al texto. Sin embargo, este proceso se realiza con una serie de procedimientos matemáticos en los cuales mediante fórmulas asignan valores a los términos y si estos no están en la frecuencia del umbral permitido, se procede con la eliminación del término (Jacobo, 2016).

Por último, está la fase de “**procesamiento y determinación de clase**”, la cual consiste en determinar la clase o polaridad del texto del tweet, y esto se hace a través de las siguientes tareas:

- Fase 1 presencia o ausencia de polaridad. En la cual se forman 2 categorías (con polaridad o sin polaridad) con la sumatoria de los valores de las características de los documentos (Jacobo, 2016).

- Fase 2 orientación positiva, negativa o neutral. Aquí se toma la categoría con polaridad y se crean dos categorías más para los términos positivo y negativo, quedando así (P+, P) y (N+, N) (Jacobo, 2016).
- Fase 3 polaridad final. “Si el documento en evaluación ha superado las fases anteriores, se procede a determinar la clase que le corresponde (P+, P, N+, N). Si la categoría es positiva se procede a elegir entre P+ y P tomando como base el peso de cada categoría, y lo mismo aplica en caso de que la categoría sea negativa. “(Jacobo, 2016).

De esta manera concluye el procedimiento de análisis de texto en la propuesta anterior.

Aplicación De Redes Neuronales Artificiales En La Clasificación De Textos Académicos Según Disciplina: Biometría, Filosofía Y Lingüística Informática.

Este proyecto propone el uso de redes neuronales artificiales con aprendizaje supervisado con el objetivo de ayudar a la clasificación automática de textos para categorizar documentos dentro de un número fijo de categorías según su contenido (Beltrán, 2012).

Para esta propuesta el análisis y clasificación de texto inicia con la fase de diseño de la muestra, que consiste en la selección de textos académicos resúmenes de trabajos presentados a congresos y revistas, pertenecientes a las disciplinas de biometría, lingüística informática y filosofía, de los cuales se seleccionaron 60 textos de cada una (Beltrán, 2012).

En la segunda fase de etiquetado, se apoyan de un software llamado smorph, el cual es un analizador y generador morfosintáctico que realiza la Tokenización y análisis morfológico en una sola etapa, que con un texto de entrada arroja un texto lematizado con los valores correspondientes (Beltrán, 2012). Cabe resaltar que con este software el proceso se simplifica en un gran porcentaje en clasificaciones no tan robustas como esta.

En la tercera fase de diseño y desarrollo de base de datos, los resultados obtenidos en la fase de etiquetado se proceden a guardar en un base de datos por palabra, la

cual se usa para generar una matriz que contiene la cantidad de lemas, etiquetas, ocurrencias y textos (Beltrán, 2012).

En la fase final, se hace uso de las redes neuronales artificiales, fase en la que se debe realizar un entrenamiento con los datos que se obtienen en las fases anteriores, después de entrenar la red neuronal se realizan las pruebas suficientes, para determinar cuál es el rango de error. Por último, se calculan los porcentajes de clasificación correcta en cada categoría de texto.

Con estos proyectos y propuestas presentadas a lo largo de este capítulo, se puede evidenciar que existen varios métodos para el análisis de patrones de texto, y que a raíz de eso se pueden adaptar algunas de ellas en el desarrollo de este método de automatización de inteligencia artificial.

Proyecto	Técnica
Análisis De La Dinámica Del Contenido Semántico De Textos	Análisis de textos en inglés, lematización y stemming
Análisis De Emociones Y Sentimientos Sobre El Discurso De Firma Del Acuerdo De Paz En Colombia	Audios y textos
Reconocimiento De Agresión Verbal En Twitter Con El Uso De Patrones Lingüísticos	Tokenizador, stopwords, Lematizadores.
Análisis De Sentimientos En Twitter	Preprocesamiento de texto, extracción y reducción de características.

De Redes Neuronales Convolucionales Aplicado A Sentiment Analysis	Preparación de datos, limpieza, normalización, eliminación de ruido.
Uso Del Software Sisam Para La Detección De Contenidos Denigrantes Hacia La Mujer En Productos Audio Musicales Del Género Reggaetón 2019	Tokenización, preprocesado de texto, eliminación de stopwords.

CAPÍTULO 3.

TÉCNICAS DE INTELIGENCIA ARTIFICIAL UTILIZADAS ACTUALMENTE PARA LA DETECCIÓN DE PATRONES TEXTUALES

En este capítulo se pretende evidenciar algunas de las técnicas que se han usado a lo largo de las investigaciones y proyectos mencionados anteriormente, las cuales sirven de base conceptual para identificar técnicas de inteligencia artificial (se explica en el marco teórico de esta investigación) para la detección de patrones de texto.

En el proyecto de (Alvares, 2017), se hace el uso de **machine learning** (aprendizaje de máquina), un área de la inteligencia artificial que se encarga de estudiar algoritmos computacionales que permiten a los computadores aprender hacer tareas o predicciones con cierto grado de certidumbre y su aprendizaje se basa en observaciones y ejemplos del pasado (Alvares, 2017).

Para realizar entrenamientos o enseñar a las máquinas, existen 3 tipos de métodos los cuales son:

Aprendizaje supervisado: Consiste en entregar colecciones de datos previamente clasificadas para poder realizar un entrenamiento adecuado, así cuando se reciban otro tipo de datos ya sabrá cómo comparar la información, con el aprendizaje que ya obtuvo (Alvares, 2017).

Aprendizaje no supervisado: Este a diferencia del anterior, recibe datos sin una previa clasificación o etiquetado, ya que este tipo de aprendizaje tiene como fin hacer que el algoritmo busque similitudes entre cada conjunto de datos que recibe y se encargue de la clasificación, así los datos se agrupan según su contenido. Además, que este proceso se hace de manera automática o sin intervención manual (Alvares, 2017).

Aprendizaje por refuerzo: Este básicamente se refiere a que una máquina aprende por sí misma a través de realizar las tareas o funciones repetidamente, así la máquina descarta o privilegia acciones hasta perfeccionar su modelo de aprendizaje. Todo esto se logra haciendo uso de la psicología conductista (Alvares, 2017).

Ahora otra técnica de la inteligencia artificial utilizada en el proyecto de (Lopez, 2019), es el uso de **redes neuronales convolucionales**, que son un tipo concreto de redes neuronales de Deep learning. Ahora bien, una red neuronal es un conjunto de nodos interconectados que imitan las neuronas del cerebro humano, las cuales usan algoritmos que pueden reconocer patrones ocultos y correlacionarlos en datos, agruparlos y clasificarlos para con el tiempo aprender y mejorar continuamente (SAS, 2020). Sin embargo, un tipo de redes neuronales son las redes neuronales convolucionales, las cuales se crearon principalmente para la detección de imágenes y objetos, aunque en los últimos tiempos se han usado de manera muy efectiva para el procesamiento de texto (Lopez, 2019).

Otro ejemplo es (G & Elizabeth, 2019), en el cual para el análisis de sentimientos hacen uso de un programa llamado ParallelDots que se basa en inteligencia artificial, específicamente en **Deep learning**, que es una rama del machine learning pero que ha evolucionado y su función es un poco más precisa que su antecesora. La finalidad del Deep learning es que, en lugar de darle instrucciones o reglas para la evaluación de un problema, a este se le da un modelo de datos y unas pocas instrucciones de cómo evaluar los datos, para que así aprenda cómo solucionar un problema y en cada repetición mejorará la técnica que creó hasta llegar a perfeccionarla y usarla con cualquier tipo de problema (SAS, 2020).

Otro uso de la inteligencia artificial en clasificación de patrones de texto se ve en el proyecto de (Sande, 2018), en el que se hace uso de algunas técnicas mencionadas antes como, machine learning y aprendizaje supervisado. Sin embargo, aquí se aplica otro concepto llamado **Procesamiento del lenguaje natural (NLP)** (abreviado en inglés), esta rama de la inteligencia artificial se basa en facilitar la comunicación entre las personas y las computadoras, mediante el uso de protocolos del lenguaje natural, los cuales son aquellos que usan las personas para comunicarse entre ellas de forma oral o escrita. Dentro de esta área existen varios niveles que se deben tener en cuenta para un óptimo entendimiento entre el usuario y el sistema, los niveles son: Análisis morfológico, análisis sintáctico, análisis semántico y análisis pragmático. Sin embargo, estos no deben ser implementados a la vez, ya que debe ser el sistema el que determine cuál usar (Sande, 2018).

De la misma manera que el anterior proyecto, la propuesta de (Edgar Altszyler, 2015), hace uso del **procesamiento del lenguaje natural**, en este caso particular se utiliza para la cuantificación semántica de las palabras, a través de un algoritmo llamado **análisis semántico latente (LSA)**, el cual consiste en realizar una descomposición de valores y reducción de dimensionalidad de las palabras para hallar similitud semántica entre palabras y textos, ya que el significado de la palabra puede estar contenida en un texto sin que la palabra esté presente. Aunque este algoritmo fue desarrollado para la búsqueda y recuperación de información, también se emplea para cuantificar la cercanía de conceptos y documentos (Edgar Altszyler, 2015).

En el proyecto de (Magino, 2020) también se hace uso del procesamiento del lenguaje natural, ya que este es uno de los procesos más acertados en el análisis de textos. Sin embargo, también se hace uso de un software desarrollado en java por (Magino, 2020), el cual utiliza librerías que permite hacer todo el proceso de procesamiento de texto mencionado en el capítulo anterior.

Ahora en el proyecto (Olguín, 2015), se aplican técnicas de **minería de datos (Data Mining)**, la cual es un conjunto de técnicas y tecnologías que intentan explorar y descubrir patrones en una cantidad enorme de datos, haciendo uso de métodos de

inteligencia artificial. Además, es un campo de las ciencias de la computación y la estadística (Olguín, 2015). En pocas palabras el uso de la minería de datos sirve para la extracción y análisis de conjuntos de datos y hacerla convertirla en información comprensible.

Según (Rosa Montañes, 2018) una de las técnicas de inteligencia artificial que se están usando hoy en día para el análisis de patrones de texto son las redes neuronales convolucionales, que se encuentran dentro de la rama del aprendizaje profundo (Deep learning), ya que, a través de los resultados obtenidos por su alta precisión en la clasificación de imágenes, se han llevado a cabo modelos de prueba para el análisis de texto. Sin embargo, (Rosa Montañes, 2018) hace uso de otro tipo de modelo el cual es **Long Short Term Memory (LSTM)**, los cuales combina para crear un modelo híbrido que permita extraer lo mejor de ambos modelos y ponerlos en uno solo. **LSTM** es una arquitectura de una **red neuronal recurrente**, que básicamente surge de la necesidad de mantener información a largo plazo y hacer que la información persista. Aunque la red neuronal recurrente ya hace el proceso de persistir o mantener la información, esta red por sí sola, no puede mantener la información por mucho tiempo (Colah, 2015).

Ahora en el caso de (Luis Jiménez, 2017), hace uso de Deep learning en una de sus ramas como **redes neuronales convolucionales (CNN)**, en el cual construyen y comparan el rendimiento de dos diseños CNN para la clasificación de dígitos escritos a mano. Los resultados fueron muy satisfactorios ya que a partir de su primer entrenamiento ya alcanzan un nivel de precisión del 95% y luego de 7 entrenamientos alcanzan un 99% de precisión (Luis Jiménez, 2017).

El trabajo de (Beltrán, 2012) plantea el uso de **redes neuronales artificiales con aprendizaje supervisado** para la clasificación de textos académicos de distintas disciplinas. Este tiene una característica importante la cual consiste en realizar el entrenamiento de manera supervisada por un agente externo, ya sea un maestro o supervisor que determina la respuesta que debe generar la red (David Llanos, 2013). El análisis de los textos con el uso de esta técnica arroja resultados

satisfactorios con porcentajes de 100% para biometría, 100% para filosofía, 93% para lingüística computacional y un error global de 2.2% (Beltrán, 2012).

A continuación, en la Tabla 1 se presenta una comparativa de las diferentes técnicas de inteligencia artificial empleadas para el análisis de patrones de texto.

Método	Características
Machine learning (aprendizaje supervisado, no supervisado)	Esta técnica se ha llevado desde hace mucho tiempo, la cual permite que su implementación sea más sencilla que otras técnicas, es muy versátil en cuanto a evolución se refiere.
Deep learning	Es una de las técnicas más recientes, dentro de esta se encuentran las redes neuronales. En esta rama se han desarrollado muchas soluciones como, métodos de aprendizaje basado en máquina, redes neuronales para traducción de texto y clasificación de imágenes, computación en la nube distribuida, etc. (SAS, 2020).
Redes neuronales convolucionales (CNN)	Las CNN tienen un alto impacto en el análisis de imágenes, ya que proyectos como (Laureano, 2019) evidencia el alto grado de aceptabilidad y precisión que tiene este modelo. Aunque en (Luis Jiménez, 2017), se emplea como reconocimiento de texto arroja excelentes resultados.
Redes neuronales recurrentes (RNN)	Este tipo de redes son muy buenas, ya que persisten la información punto, eso hace que sea mucho más fácil de realizar

	tareas o procesos. Hace más corto el proceso de entrenamiento, aunque es difícil de implementar y la persistencia de información es limitada.
Data Mining	Esta rama del machine learning, hace que la clasificación de datos sea muy robusta, aunque esta es más utilizada en análisis de grandes cantidades de datos para obtener como resultado información entendible, no es muy común usarla para el análisis de textos como tal.
Procesamiento de lenguaje natural (NLP)	Se enfoca en mejorar el proceso de comunicación entre el humano y la computadora, además tiene amplios usos en la mejora de experiencia de usuario.
Análisis Semántico Latente (LSA)	Según (Gutiérrez, 2005) es un tipo de análisis computacional que se basa en un algoritmo matemático, para poder determinar la similitud del significado entre piezas textuales.
Long Short-Term Memory RNN (LSTM)	Esta es una mejora de RNN, que permite ampliar el tiempo de persistencia de la información, genera modelos más precisos, pero consume demasiados recursos de hardware y software, su implementación es más compleja que las demás.

Tabla 1 Comparativa de diferentes técnicas de inteligencia artificial para análisis de texto

Fuente: Elaboración Propia

Ahora como punto final de este capítulo, se planea escoger la técnica de inteligencia artificial que se usará para el desarrollo del software en el capítulo siguiente. Por todo lo mencionado, se escoge **Redes Neuronales Convolucionales**, la cual sus implementaciones en el área de reconocimiento de texto han sido muy pocas y se planea hacer un aporte demostrando su eficacia en esta rama de la inteligencia artificial.

3.1 ARQUITECTURA DE REDES NEURONALES CONVOLUCIONALES

Una red neuronal convolucional CNN (Convolutional Neural Networks) es una clase de Redes Neuronales Artificiales (RNA), al igual que estas, las CNN están conformadas por neuronas que tienen entradas y salidas. Las CNN a diferencia de las RNA permiten que las entradas a sus neuronas sean datos no numéricos como imágenes, sonido, texto, sin necesidad de que estas entradas tengan un preprocesamiento previo (Torres, 2020).

El surgimiento de las redes neuronales convolucionales data de los años noventa en un experimento en donde se propone un nuevo modelo de redes neuronales que permitiera clasificar dígitos escritos a mano a partir de sus imágenes (Raschka & Mirjalili, 2019). Este experimento fue publicado en el trabajo titulado Handwritten Digit Recognition with a Back-Propagation Network (Cun, Boser, Denker, & Henderson, 1990).

Aunque las CNN han sido utilizadas principalmente para procesamiento de imágenes, en los últimos años se ha visto un avance muy significativo en el análisis y clasificación de texto. Por consiguiente, se pretende describir a brevedad la arquitectura que emplea este tipo de redes.

Un modelo CNN se compone básicamente de tres capas, una capa de entrada, capas ocultas y capa de salida, donde las capas ocultas se componen de capas de convolución, la capa de entrada recibe los parámetros que se van a evaluar y las capas de salida dan el resultado de todo el análisis y proceso de entrenamiento y extracción de características ver. En la Figura 2 se muestra la arquitectura base de un modelo convolucional, a continuación, se explicará el funcionamiento del modelo CNN junto con sus capas.

La capa de entrada es la primera capa en inicializar dentro de un modelo CNN, esta capa cumple la misma finalidad tanto para imágenes como para textos. Su función consiste en tomar los datos y transformarlos en un vector de números con todas las características del dato ingresado, por ejemplo, una imagen de tamaño 28*28 píxeles de alto y de ancho y en escala de grises, crearía un vector de 784 posiciones lo cual es equivalente a 784 neuronas. Sin embargo, usan capas diferentes para realizar esa transformación de datos.

Para el procesado de imágenes se usa la capa Input, la cual requiere de un parámetro específico (shape) para su correcto funcionamiento y desempeño, donde se definen los valores que determinan el tamaño del vector, como en el ejemplo anterior.

En el procesado de texto, se requiere una capa llamada Embedding y debe recibir tres parámetros, que al igual que la anterior ayudan en su desempeño y correcto funcionamiento: el número máximo de palabras (dimensión de entrada), tamaño del vector de salida (dimensión de salida) y la secuencia máxima de caracteres.

Una vez definidas las capas de entrada, se procede con la creación de las capas ocultas, que pueden ser capas convolucionales (conv1D, conv2D), capas agrupadas (pooling), capas completamente conectadas y capas de normalización. La capa convolucional opera tomando una señal de entrada, aplica un filtro sobre ella y multiplica la señal de entrada para obtener la señal modificada. La capa de pooling reduce la dimensionalidad de un vector agrupando las características de un valor o dato ingresado y devuelve las mejores características en otro vector, así lo entrena la red con los mejores datos posibles.

La capa final, es la última capa del modelo creado, es la que me retorna los resultados de todo el proceso anterior. Para dar la salida de las demás capas, se usa una capa completamente conectada (Dense), lo que significa que todas las neuronas entrenadas en los pasos anteriores se condensan en una sola capa.

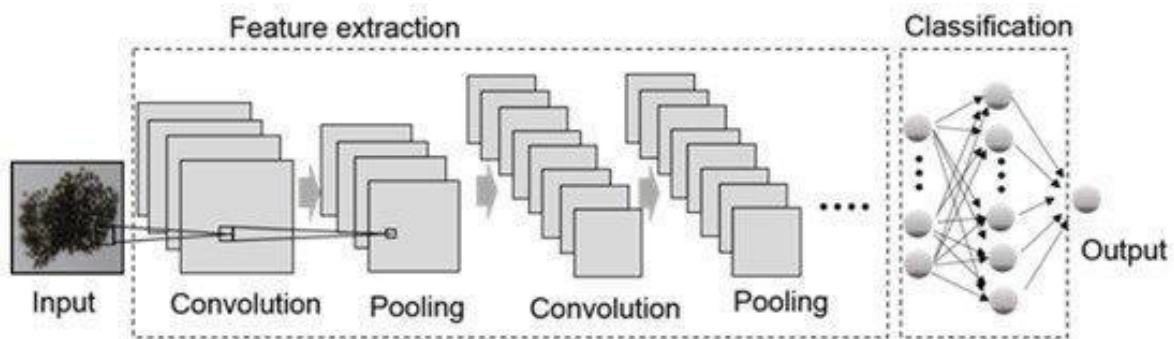


Figura 2 modelo de arquitectura CNN. (Jungsu Park, 2019)

Teniendo en cuenta lo mencionado anteriormente, se presenta el modelo de arquitectura que se emplea en el desarrollo del proyecto, ver Figura 3. Se evidencia la implementación de las capas que se requieren para conformar una red neuronal convolucional.

La capa de Embedding que tiene un tamaño máximo de entrada de 900 palabras y de salida un vector de tamaño 100 y funciona como la capa de entrada. Posteriormente, en las capas ocultas se encuentran: la capa convolucional, que recibe como entrada los valores que se ocupan en la capa anterior, además utilizan una cantidad de 128 filtros y un kernel de 5 para aumentar la cantidad de parámetros. Una capa Dropout que reduce la capacidad de aprendizaje de la red para evitar sobreentrenamiento, una capa Densa, donde todas las neuronas y datos entrenados se reúnen en esta capa, una capa Flatten que convierte los vectores creados en un vector de una sola dimensión, luego ocupa otra capa Dropout y por últimos una capa densa con un método de activación softmax, esta capa retorna la clasificación de los datos según el número de etiquetas o clases definidas en el dataset.

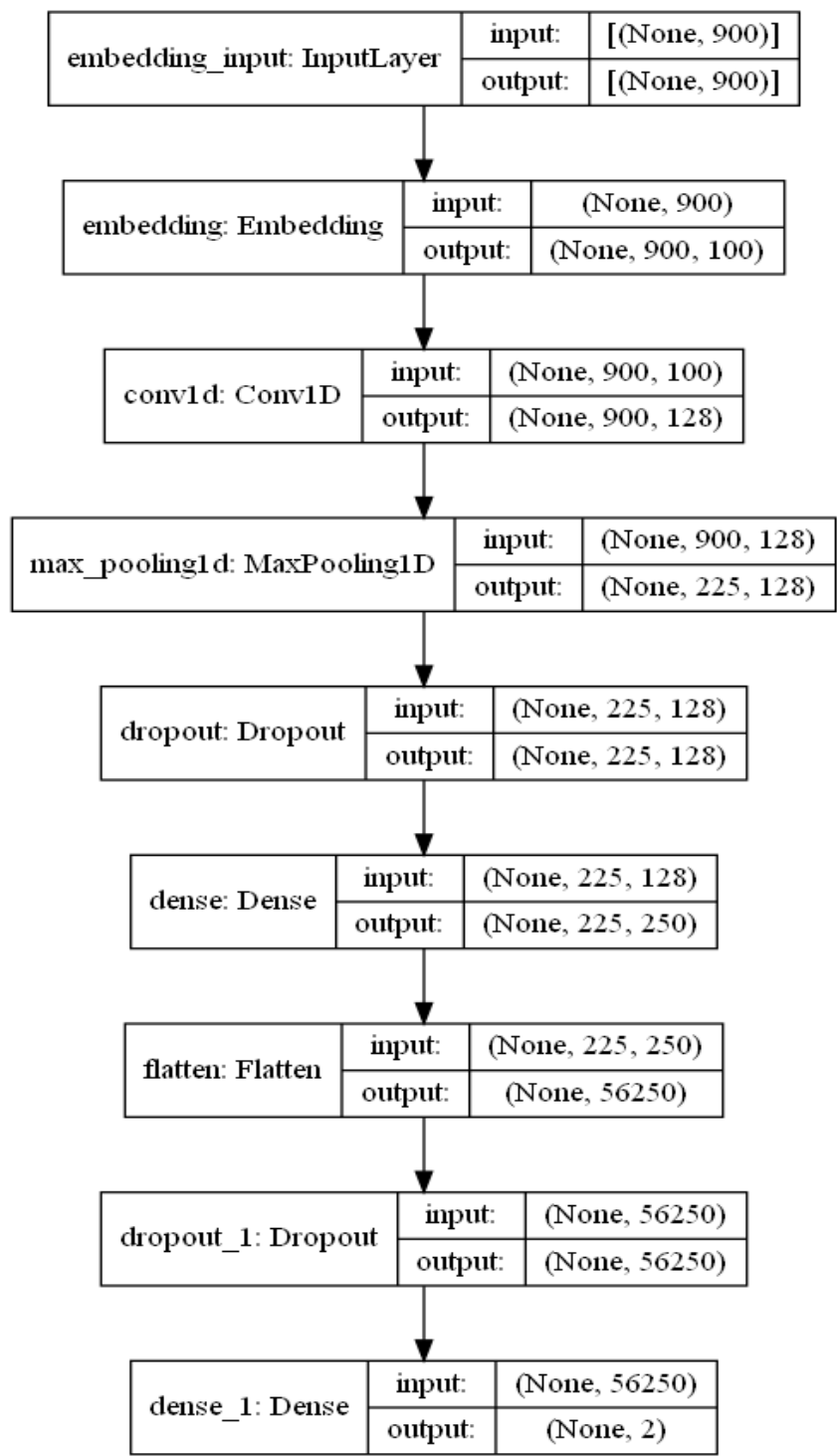


Figura 3 modelo de arquitectura (Jungsu Park, 2019)

CAPÍTULO 4

CONSTRUIR UN MÓDULO SOFTWARE BASADO EN LAS TÉCNICAS DE INTELIGENCIA ARTIFICIAL, QUE PERMITA LA IDENTIFICACIÓN Y EL ANÁLISIS AUTOMÁTICO DE PATRONES DE VIOLENCIA CONTRA LA MUJER EN EL TEXTO DE LAS LETRAS DE CANCIONES

En este capítulo se presenta la construcción de un modelo de inteligencia artificial, el cual tiene como finalidad identificar y clasificar los patrones de violencia contra la mujer que se presentan en letras de canciones. Para el correcto desarrollo del módulo de software, se ha determinado que se debe hacer uso de una metodología de trabajo que apoye en todo el proceso de planeación, desarrollo y ejecución del proyecto.

Se selecciona la metodología CRISP-ML (machine learning) para el desarrollo del modelo, la cual es una variante de la metodología Cross-Industry Standard Process for Data Mining (CRISP-DM) que es modelo probado para orientar el trabajo de minería de datos, en este modelo se deben seguir seis fases para el buen funcionamiento y desarrollo del proyecto que se desea realizar. CRISP-DM fue usada para el desarrollo de proyectos con machine learning, sin embargo, no cubría algunas de las necesidades de este tipo de desarrollo y carecía de tareas específicas para esta área (Stefan Studer, 2021).

Además, de CRISP-ML existen algunas otras metodologías que se pueden usar para el desarrollo de aplicaciones con machine learning, a continuación, una breve comparativa en la *Figura 2*.

CRISP-ML (Q)	CRISP-DM	(Saleema Amershi, 2019)	(Sculley, 2017)
Business & Data Understanding	Business Understanding	Requirements	-
	Data Understanding	Collection	

Data Preparation	Data Preparation	Cleaning	Data	Infrastructure
		Labeling		
		Featuring Engineering		
Modeling	Modeling	Training	Model	
Evaluation	Evaluation	Evaluation	-	
Deployment	Deployment	Deployment	-	
Monitoring & Maintenance	-	Monitoring	Monitoring	

Tabla 2 Comparativa de metodologías. (Stefan Studer, 2021).

CRISP-ML al basarse en la CRISP DM, modifica y agrega algunas fases, aunque la cantidad de fases es igual a su antecesora, abarca seis fases que van desde definición del alcance hasta el mantenimiento de la aplicación de machine learning. Además, cada tarea del proceso de implementación propone métodos que se encargan de asegurar la calidad, tanto de los datos que se usan, hasta el modelo que se crea para tratar los mismos datos.

En la *Figura 4* se presenta un diagrama de flujo en el cual se evidencian las 6 fases que se deben cumplir con esta metodología, además cada fase cuenta con una serie de tareas que son primordiales para continuar con la siguiente tarea y posteriormente con otra fase.

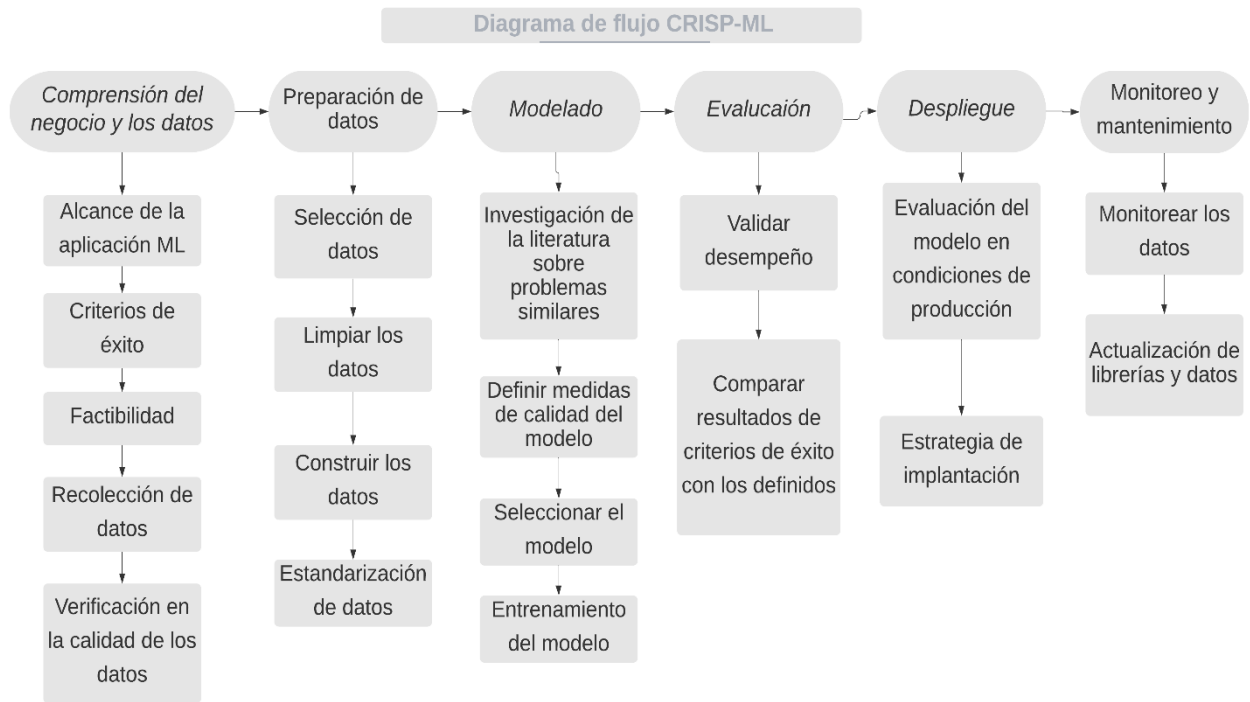


Figura 4 Diagrama de flujo CRISP-ML (Stefan Studer, 2021)

Teniendo en cuenta las fases de la metodología, se inicia la implementación de la misma dentro del proyecto, como primera medida, se abordará la primera fase aplicando cada una de sus tareas, teniendo en cuenta que también se explicará la finalidad de cada etapa.

4.1 FASE 1: COMPRESIÓN DE LOS DATOS Y EL NEGOCIO

Esta fase se encarga de las tareas definir objetivos comerciales y transformarlos en objetivos de ML, así como recopilar y verificar la calidad de los datos que se usarán.

4.1.1 Alcance De La Aplicación ML

El alcance de este proyecto, se tiene en cuenta el objetivo principal, el cual es crear un modelo de inteligencia artificial que permita la identificación de patrones de texto que denotan violencia contra la mujer en letras de canciones y posteriormente clasificar más canciones como violentas o no violentas.

4.1.2 Criterios De Éxito.

De acuerdo a los estándares IEEE (Stefan Studer, 2021), se propone medir los criterios de éxito establecidos en los siguientes puntos:

- La métrica de precisión en validación (`val_accuracy`), debe estar por encima del setenta por ciento (70%) en los datos de validación y la precisión en entrenamiento no debe superar por más de quince (15) puntos a la validación, para así garantizar la predicción del modelo en futuras clasificaciones de letras de canciones.
- Se debe emplear una red neuronal convolucional como modelo Deep learning en el desarrollo del proyecto.
- Desarrollar un entorno visual para realizar pruebas de predicción del modelo entrenado, ya sea con Django o Flask.

Para el criterio de éxito económico, no se aplica para este proyecto, ya que es un proyecto de investigación para adquirir el título de pregrado y no se cuenta con una ayuda o financiación monetaria, teniendo en cuenta también que se puede clasificar como un proyecto social en algún futuro.

4.1.3 Factibilidad.

Se debe verificar la viabilidad del proyecto teniendo en cuenta factores como: la disponibilidad, tamaño y calidad de la información que se va obtener, revisar si hay proyectos similares ya desarrollados, determinar con base en los proyectos hechos si la cantidad de datos que se pueden obtener son suficientes o no, y de no ser suficientes se de detener el desarrollo en esta fase.

En esta etapa, se pueden evidenciar varios proyectos que se acercan a lo que se está desarrollando, como es el caso de (Hecmaramco, 2020) que plantea un proyecto que permite clasificar si hay violencia contra las mujeres en las canciones, haciendo uso de machine learning y con un dataset de 50 canciones en español clasificadas en diferentes tipos de violencia.

También están diferentes personas que han realizado proyectos con Deep learning con la clasificación de lenguaje ofensivo en Twitter como el caso de (machado, 2021), que usa Redes convolucionales para realizar la clasificación de tweets en inglés y se apoya de un dataset de veinte mil tweets en inglés también.

Otro ejemplo es el de (Md Abul Bashar, 2018) quienes realizan una investigación de detección de misoginia en tweets con redes neuronales convolucionales con dataset pequeño.

4.1.4 Recolección de datos.

En esta tarea de recopilación de los datos son fundamentales factores como el tiempo y costos de recolección, la cantidad de datos que se obtienen y el control de versiones, ya que esta no es una tarea que se haga una sola vez, debe ser repetitiva hasta que los datos sean fiables para llevarlos a la siguiente etapa.

El control de versiones se puede evidenciar por medio de la recolección de los datos, la cual se llevó a cabo en 3 iteraciones, cada una con resultados diferentes y con sus propios procesos de recolección. Así cada una de las iteraciones tiene como resultado un cambio y mejoramiento de los datos.

La primera iteración tomó alrededor de 4 días ya que se buscó en varios repositorios de datasets que pudieran contener letras de canciones en español. Los sitios a los que se realiza la consulta, en su mayoría fueron propuestos por la docente encargada de este proyecto, los demás fueron encontrados por medio de

investigación, estos son los sitios en los que se realizó la búsqueda: kaggle.com/, archive.ics.uci.edu/ml/index.php, pub.towardsai.net/, paperswithcode.com, catalogodatos.gub.uy/dataset, de los cuales se obtuvo cuatro tipos de dataset que se mencionan a continuación:

- Un dataset que contiene una carpeta con 49 archivos, cada uno de un artista diferente con alrededor de 60 canciones cada artista, dando un total de trescientas canciones, sin embargo, las canciones que se obtuvieron eran en inglés, así que no servía para el proyecto ver *Figura 5*.

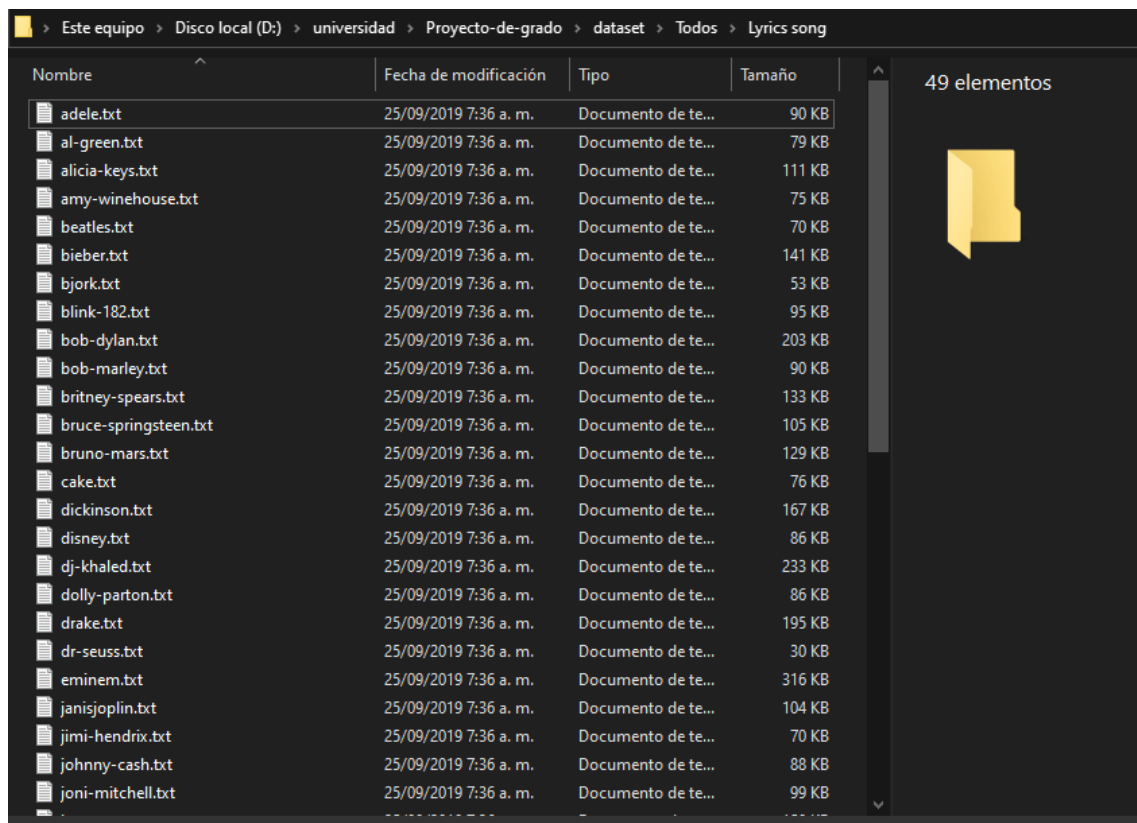


Figura 5 Canciones en inglés primer iteración

Dataset tomado de <https://www.kaggle.com/paultimothymooney/poetry>

- Un archivo tipo csv que contiene dos mil canciones aproximadamente, pero como en el caso anterior las canciones estaban en inglés ver *Figura 6*.

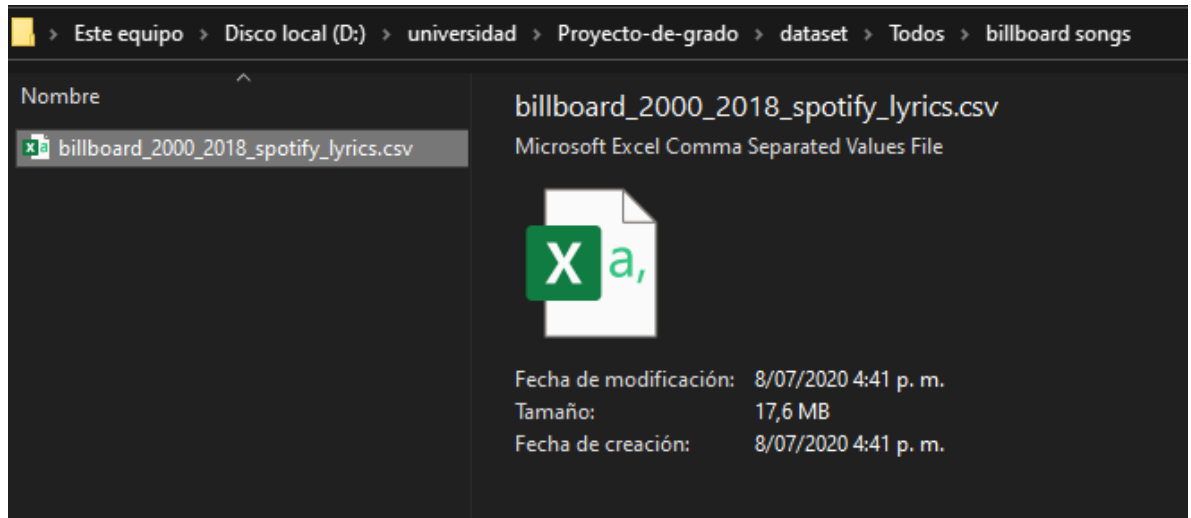


Figura 6 Archivo csv con más de 2 mil canciones en ingles

Dataset tomado de <https://www.kaggle.com/danield2255/data-on-songs-from-billboard-19992019>

- Un dataset conformado por canciones clasificadas por sentimiento, en feliz, triste, enojado y relajado. Sin embargo, todas estaban en el idioma inglés ver *Figura 7*.

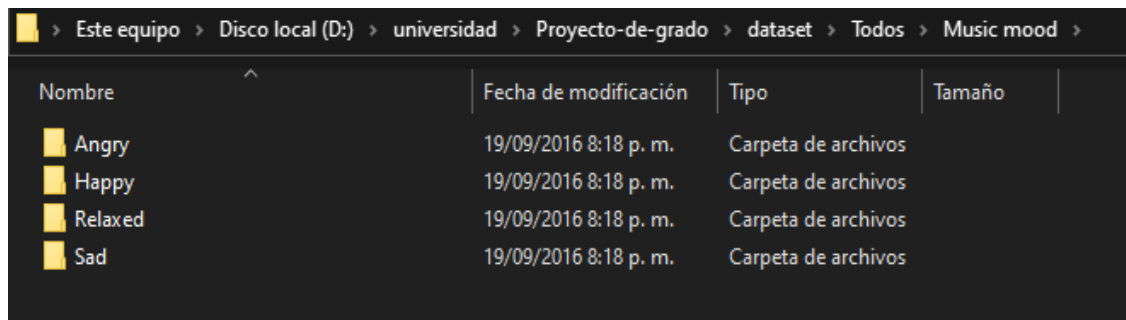


Figura 7 Canciones clasificadas por sentimiento

Dataset tomado de <https://cs.nju.edu.cn/sufeng/data/musicmood.html>

- Por último, un archivo csv con canciones de rap en español, este contiene un aproximado de nueve mil canciones, por un momento se pensó en trabajar con ese dataset, pero se limitaba el aprendizaje a un solo género musical ver *Figura 8*. Por lo tanto, se procede a realizar la recolección de datos de otra

manera, descartando así la idea que conseguir un dataset construido y etiquetado.

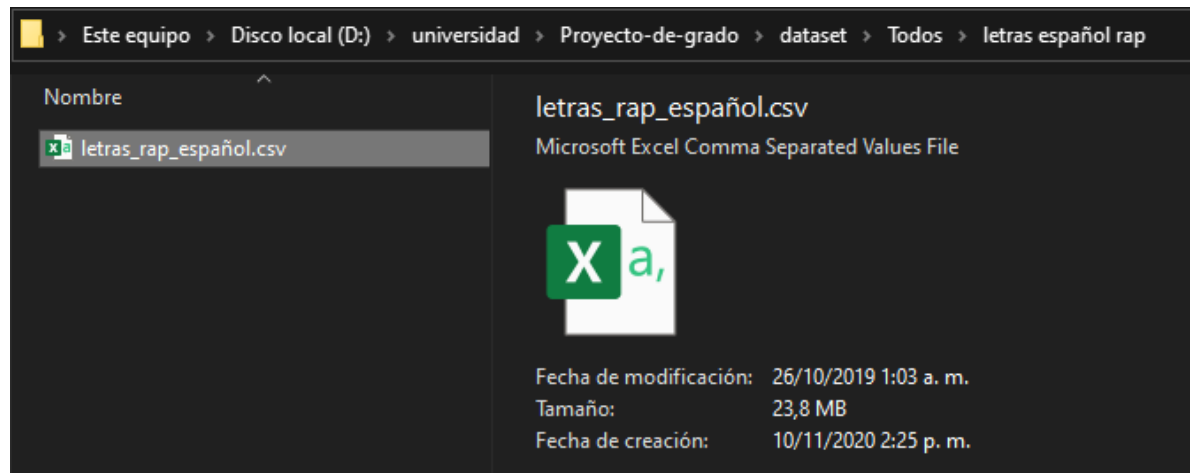


Figura 8 Archivo de canciones rap en español

Dataset tomado de <https://www.kaggle.com/smunoz3801/9325-letras-de-rap-en-espaol>

En la segunda iteración se plantea la solución de crear un propio dataset, ya que no se encontraron datasets con canciones en español o la cantidad de canciones no era suficiente. Por lo tanto, la primera opción de recolección de letras de canciones era hacerlo manualmente, buscar las letras en sitios como lyricfind, musixmatch, Spotify, YouTube, letras.com, música.com, pero el tiempo que esto toma era demasiado y se salía del cronograma de trabajo establecido, estimando aproximadamente veinte días de trabajo con dos personas y cada una recolectando 100 canciones diarias, para un total de cuatro mil canciones.

Sin embargo, esta opción manual no era viable, por lo que se optó por buscar algún tipo de servicio de API que proveyera las letras de canciones. Al iniciar se tomaron algunas Apis y librerías como: musixmatch, lyrics.com, Spotify, Mourits Lyrics, LastFm, Chart Lyrics, billboard de las cuales solo musixmatch obtuvo resultados positivos, aunque lastimosamente luego de realizar la conexión con la API a través de código propio desarrollado en lenguaje Python, el servicio de musixmatch solo arrojaba el treinta por ciento (30%) de la letra de la canción, lo cual no era suficiente y quedaba incompleta y no es posible realizar la clasificación con fragmentos de

canciones. . Además, para acceder al cien por ciento (100%) de la letra, se debía realizar una solicitud de compra de los servicios de musixmatch, al que solicite información de los precios por medio de un correo electrónico, el cual nunca tuvo ningún tipo de respuesta.

A partir de ahí, se decide realizar la extracción de las letras con una técnica llamada web scraping, según (ionos, s.f.) se define como un proceso de extracción de datos o información de páginas web y se puede realizar de manera manual o a través de software o códigos desarrollados en Python o Perl. Luego de crear y probar varias funciones desarrolladas en Python ver *Figura 9* y *Figura 10*, se procede a realizar la extracción de las letras para más adelante revisar la legibilidad del contenido, el cual no arroja los resultados esperados, ya que no alcanzaba ni las dos mil quinientas canciones correctas, por lo tanto, también se descartó esta opción de recolección, teniendo en cuenta también que el tiempo era demasiado alto sumando todos los pasos que se debían realizar para obtener letras óptimas y de calidad, dando un total de quince días.

```

def get_lyrics(song, artist):
    api_key = 'e6ada0d24fedeb1162187cb9af21d4f5'
    q_track = song
    q_artist = artist
    base = 'http://api.musixmatch.com/ws/1.1/track.search?'
    url = f"{base}apikey={api_key}&q_track={q_track}&q_artist={q_artist}"
    response = requests.get(url)
    json_response = response.json()
    get_genre(json_response)
    try:
        url_lyrics = json_response['message']['body']['track_list'][0]['track']['track_share_url']
        response = requests.request(
            "GET", url_lyrics, headers=headers, data=payload)
        soup = BeautifulSoup(response.text)
        data = soup.find_all('span', class_='lyrics_content_ok')
        preformat_data = str(data)
        text = preformat_data.replace(
            '<span class="lyrics_content_ok">', '')
        text_1 = text.replace('</span>', '')
        text_2 = text_1.replace('[', '')
        text_3 = text_2.replace(']', '')
        if not str(text_3):
            lyrics.append('Sin Letra')
        else:
            lyrics.append(str(text_3))
    except:
        lyrics.append('Sin Letra')

```

Figura 9 Función Web Scraping en Musixmatch

```

# función para obtener los nombres de canciones
def get_songs():
    chart = billboard.ChartData('latin-songs')
    i = 0
    while chart.previousDate:
        for x in chart:
            title.append(x.title)
            artist.append(x.artist)
            get_lyrics(x.title, x.artist)
        chart = billboard.ChartData('latin-songs', chart.previousDate)
        i += 1
    # dictionary of lists
    dict = {'Titulo': title, 'Artista': artist,
            'Genero': genres, 'Letra': lyrics}
    df = pd.DataFrame(dict)
    df.to_csv(r'C:\Users\didie\Desktop\data.csv')

```

Figura 10 Función para obtener nombres de canciones

En la tercera y última iteración, se usan dos Apis y una librería de Python: billboard, que se usa para obtener los nombres de las canciones top latinas de los últimos años, genius como un api que trae las letras de las canciones top obtenidas con billboard. A partir de ahí, se inicia con la creación de varias funciones de código que permitían obtener los nombres de las canciones ver *Figura 11*, obtener la letra de las canciones obtenidas en la función anterior ver *Figura 12* y una función que permite eliminar las canciones que no tenían letra ver *Figura 13*. Satisfactoriamente esta tarea de recolección da como resultado más de cuatro mil canciones latinas de todos los géneros musicales y en un tiempo de tres días, contando el tiempo de desarrollo e implementación del código.

```
# FUNCION PARA OBTENER NOMBRES DE LAS CANCIONES
def get_songs():
    chart = billboard.ChartData('latin-songs')
    i = 0
    while i < 2000:
        for x in chart:
            if x.title in titles:
                print('rep')
            else:
                print('Libre')
                title = re.sub('[^a-zA-Z0-9 \n\.]', '', x.title)
                path = 'C://datos/' + str(title) + '.txt'
                file = open(path, 'w', encoding='utf8')
                file.write(get_lyrics(x.title, x.artist))
                file.close()
                titles.append(x.title)
                artists.append(x.artist)

        chart = billboard.ChartData('latin-songs', chart.previousDate)
        i += 1
        print('iteracion # : ', i)
```

Figura 11 Función para obtener títulos de las canciones

```

# FUNCION PARA OBTENER LA LETRA DE LAS CANCIONES
def get_lyrics(title, artist):
    try:
        data = genius.search_song(title, artist).lyrics
        return data
    except:
        return 'Sin letra'

```

Figura 12 Función para obtener letra de las canciones

```

# FUNCION PARA ELIMINAR LAS CANCIONES QUE VIENEN SIN LETRA
def del_song():
    path = "D://universidad/Proyecto-de-grado/dataset/datos/"
    songs_remove = []
    for root, dirs, files in os.walk(path):
        for file in files:
            with open(os.path.join(root, file), 'r', encoding="utf8") as f:
                letra = f.read()
                res = len(letra.split())
                full = root + file
                if letra == 'Sin letra':
                    songs_remove.append(full)
                    print('Sin letra')

                elif res >= 900:
                    songs_remove.append(full)

                else:
                    print('No cambiada')

    for song in songs_remove:
        os.remove(song)
        print('removidas')

```

Figura 13 Función para eliminar canciones sin letra

4.1.5 Verificación En La Calidad De Los Datos.

En esta fase se consideran tres tareas que se deben realizar:

- Descripción de los datos
- Requisitos de datos
- Verificación de datos

Esto con el fin de que los datos recolectados cumplan los objetivos comerciales y de Machine Learning (ML).

Descripción de los datos: En esta tarea se describe brevemente las características de los datos, propiedades, formatos y demás. Los datos recolectados son letras de canciones en español, la cantidad exacta de canciones verificadas es de 4,366, todas en formato de texto, el tamaño de la letra de la canción es indiferente, pueden ser letras que contengan desde cincuenta palabras hasta novecientas palabras, sin embargo, es muy difícil encontrar canciones de más de mil palabras y en caso de haberlas igual se tienen en cuenta.

Requisitos de los datos: Esta tarea define las condiciones esperadas de los datos para que sean aceptados como válidos. Aquí se definen algunos criterios o requisitos que deben cumplir los datos:

- Las letras de las canciones deben estar en el idioma español.
- Las canciones que se utilicen no deben pertenecer a un solo género musical, debe ser imparcial con respecto a los objetivos del proyecto.
- Debe haber un total mínimo de tres mil canciones.
- Eliminar los signos de puntuación para poder guardarlos en archivos csv.
- Deben clasificarse con violencia o sin violencia las letras de las canciones.

Verificación de los datos: Esta es una tarea que se realiza de manera transversal a todas, ya que de esta depende que la calidad de los datos sea la mejor posible. Sin embargo, se programaron funciones ver *Figura 14*, *Figura 15*, *Figura 16*, que permiten la limpieza, clasificación y creación de un archivo csv con las letras de canciones ya etiquetadas como con violencia o sin violencia. Para realizar esa clasificación se usa librerías de procesamiento de lenguaje natural, para la identificación de palabras o frases que pueden denotar los tipos de violencia contra la mujer mencionados en los preliminares de este proyecto ver apartado 1.1

```

def clean_text(txt):
    # Converting all string to lower case
    txt = str(txt).lower()
    # Removing Punctuations
    txt = re.sub(r'^[\s\w]', '', txt)
    # Removing Numbers
    txt = re.sub(r'[0-9]', '', txt)
    # Tokenizing
    tokens = word_tokenize(txt)

    # Removing the stop words and Rejoining
    to_remove = ['me', 'eres']
    stops = set(stopwords.words("spanish")).difference(to_remove)
    stops.update([...])
    filtered_sentence = [word for word in tokens if word not in stops]
    filtered_sentence = " ".join(filtered_sentence)

    ...
    bad_words = "... "
    words = re.findall(bad_words, filtered_sentence)
    num_bad_words = len(words)

    return num_bad_words

```

Figura 14 Función para limpiar texto

```

# FUNCION PARA MOVER LAS CANCIONES DE CARPETAS
def move_song():
    path = "D://universidad/Proyecto-de-grado/dataset/data2/"
    songs = []
    buenas = []
    malas = []
    songs_new = []
    i = 0
    for root, dirs, files in os.walk(path):
        for file in files:
            with open(os.path.join(root, file), 'r', encoding="utf8") as f:
                i = i+1
                full = root + file
                songs.append(full)
                letra = f.read()
                resp = clean_text(letra)
                if resp >= 1:
                    target = root + 'con violencia/' + file
                    songs_new.append(target)
                    malas.append(resp)
                else:
                    target = root + 'sin violencia/' + file
                    songs_new.append(target)
                    buenas.append(resp)
                print('vuelta ' + str(i))

    for origin, target in zip(songs, songs_new):
        shutil.move(origin, target)

```

Figura 15 Función para clasificar canciones con violencia o sin violencia

```

def create_csv():
    path1 = "D://universidad/Proyecto-de-grado/dataset/data2/con violencia"
    path2 = "D://universidad/Proyecto-de-grado/dataset/data2/sin violencia"
    for root, dirs, files in os.walk(path1):
        for file in files:
            with open(os.path.join(root, file), 'r', encoding="utf8") as f:
                title = clean_song(file)
                letra = clean_song(f.read())
                titles1.append(title)
                lyrics1.append(letra)
                labels1.append(1)
    for root, dirs, files in os.walk(path2):
        for file in files:
            with open(os.path.join(root, file), 'r', encoding="utf8") as f:
                title = re.sub(r'txt', '', file)
                letra = clean_song(f.read())
                titles1.append(title)
                lyrics1.append(letra)
                labels1.append(0)

    # dictionary of lists
    dict = {'titulo': titles1, 'letra': lyrics1, 'clase': labels1}
    df = pd.DataFrame(dict)
    df.to_csv(r'D:\universidad\Proyecto-de-grado\dataset\canciones\data2.csv')

```

Figura 16 Función para crear archivo csv con letras y clases

Teniendo en cuenta lo anterior, se puede afirmar que la calidad de los datos cumple con el requisito mínimo de calidad para este proyecto, el cual es que la canción debe estar en español, además las letras se guardan en un archivo csv que exige que los signos de puntuación como la coma, no deban estar para preservar la legibilidad del documento, así mismo con las etiquetas de html o saltos de línea que contenga la letra de la canción.

De esta manera quedan realizadas las tareas de este punto, sin embargo, de ser necesaria una nueva iteración que requiera esta fase, para pulir o mejorar la calidad de los datos, se debe llevar a cabo.

4.2 FASE 2: PREPARACIÓN DE DATOS.

Esta fase sirve para producir un conjunto de datos para la fase de modelado posterior a esta, por lo tanto, se deben realizar cuatro tareas para su correcta ejecución: Seleccionar los datos, limpiar los datos, construir los datos y la estandarización de los datos. Sin embargo, esta fase no es estática, ya que si en las fases posteriores resultan datos erróneos se debe recurrir a esta. A continuación, las tareas de esta fase.

4.2.1 Seleccionar Los Datos.

En la selección de los datos se debe realizar un descarte de muestras y basarse únicamente en los criterios de calidad del objetivo. La solución para esta tarea es sencilla, teniendo en cuenta que en la fase anterior se realizaron filtros de verificación de calidad. Sin embargo, el archivo contiene varios campos en su interior, los cuales son: título, letra, clase. De los cuales, los datos que se van a utilizar son la letra y la clase de cada canción, ya que con estos se realiza la creación del dataset de entrenamiento y los demás campos son descartados para su uso porque no son necesarios.

En caso de que las clases no estén balanceadas, es decir que el número de muestras por clase esté sesgado se pueden aplicar diferentes técnicas (Guillaume Lemaitre, 2017) que evita el sobre muestreo y submuestreo. En este caso las muestras por clase se encuentran en el rango de los miles cada uno, así que se puede usar sin problemas.

4.2.2 Limpiar Los Datos.

Para limpiar los datos se establecen dos tareas, la reducción de ruido e imputación de datos.

Reducción de ruido: En esta tarea se deben aplicar filtros de procesamiento para reducir datos que no aportan al modelo de aprendizaje. Por lo tanto, se decide aplicar técnicas de preprocesamiento de texto como:

- Eliminación de signos de puntuación y caracteres especiales.
- Eliminación de números y vocales con tilde.
- Tokenización de palabras.
- Eliminación de stopwords (artículos, preposiciones, conectores) que son palabras que no aportan al contenido lírico de una canción.

Imputación de datos: Esta tarea busca reemplazar datos por un símbolo o valor numérico, para generar un patrón informativo. Por consiguiente, se usa una técnica muy conocida en el preprocesamiento de texto, llamada tokenize que convierte las palabras en un token numérico de único valor, para así clasificar y contar las repeticiones y posibles patrones en diferentes textos ver *Figura 17*.

```
df1 = pd.read_csv(r'D:\universidad\Proyecto-de-grado\dataset\canciones\data2.csv')

# Converting all string to lower case
df1 = df1.apply(Lambda x: x.astype(str).str.lower())

# Removing Punctuations
df1.letra = df1.letra.str.replace('[^\s\w]', '')

# Removing Numbers
df1.letra = df1.letra.str.replace('[0-9]', '')

# Removing Accent
df1.letra = df1.letra.str.replace('áááááá', 'a')
df1.letra = df1.letra.str.replace('ééééé', 'e')
df1.letra = df1.letra.str.replace('ííííí', 'i')
df1.letra = df1.letra.str.replace('óóóóó', 'o')
df1.letra = df1.letra.str.replace('úúúúú', 'u')

# Tokenizing
nltk.download('punkt')
df1['letra_token'] = df1['letra'].apply(lambda x: word_tokenize(x))
# print(df1['letra_token'])

# Removing the stop words and Rejoining
nltk.download('stopwords')
stops = set(stopwords.words("spanish"))
stops.update(
    ["el", "sí", "yo", "y", "ya", "es", "aca", "la", "tu", "length", "object", "text", "dtype", "name", "eh", "ay",
     "n", "acá", "ser0", "Es", "túLa", "IEX", "IEXEL", "heyEs", "boo", "oi'te", "e'", "Rc", "DY", "yoy", "pa", "c",
     "uzi", "uh", "ah", "dj", "kushy", "mas", "ey", "na", "juju", "yeah", "mmmmmm", "yeh", "yehyeahyeah", "yeahyeahyeahyeah",
     "yehyeh", "yehyehyehyeh", "ma", "ohohohoh", "huh", "to", "ta", "vip", "migo", "toa", "tigo", "lao", "so", "eh", "jah", "hey", "ey",
     "ooh", "oh", "oooh", "aunque", "mientras"])

df1.letra = df1.letra_token.apply(lambda x: ' '.join(list(i for i in x if i not in stops)))
```

Figura 17 Limpiar datos y tokenizado

4.2.3 Construir Los Datos.

En esta tarea de construcción de datos solo cabe resaltar la transformación de las letras de canciones en valores numéricos, además de la creación de los dataset de entrenamiento y prueba, el cual se crea con las canciones ya tokenizadas y con las etiquetas o clase de la canción, dando un resultado de un setenta por ciento (70%) de datos de entrenamiento y un treinta por ciento (30%) de datos de prueba ver *Figura 18*.

```
MAX_NB_WORDS = 50000
MAX_SEQUENCE_LENGTH = 900
EMBEDDING_DIM = 100

tokenizer = Tokenizer(num_words=MAX_NB_WORDS, filters='!"#$%&()*+,-./:;<=>@[\\]^_`{|}~\t\n', lower=True, split=' ')
tokenizer.fit_on_texts(texts=df1.letra)
word_index = tokenizer.word_index
# print(len(word_index))

X = tokenizer.texts_to_sequences(texts=df1['letra'].values)
X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH, padding='post')

Y = pd.get_dummies(df1['clase']).values

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

Figura 18 Creación de datos de prueba y entrenamiento

4.2.4 Estandarización De Datos.

Para la estandarización de los datos se deben desarrollar dos tareas:

- Formato de archivo: Algunas herramientas de ML requieren tipos de entrada específicos y el archivo separado por comas (CSV) es el estándar más genérico RFC 4180. En la *Figura 16* se evidencia la creación del archivo en la que están las letras de canciones y la clase de cada una.
- Normalización: Esta tarea se ve realizada a través del paso anterior construcción de los datos, en la reducción y estandarización de características, eliminando datos que no aportan a la clasificación del texto ver *Figura 17*.

4.3 FASE 3: MODELADO.

El objetivo de esta fase es crear uno o varios modelos que satisfagan las condiciones o requisitos dados. Para realizar esta fase se recomiendan algunas características o condiciones que aseguran la calidad y viabilidad del modelo a crear, teniendo en cuenta también los objetivos comerciales y de ML.

4.3.1 Investigación De La Literatura Sobre Problemas Similares.

Se considera una buena práctica examinar la literatura existente para obtener una descripción general de tareas similares. Por consiguiente, se han revisado más de 15 proyectos o propuestas que cumplen algunas de las tareas que se plantean aquí. Algunos de los proyectos que se relacionan o que se tomaron como base se encuentran en la sección de **ANTECEDENTES** o en la sección 4.1.3 **Factibilidad**.

4.3.2 Definir Medidas De Calidad Del Modelo.

Existen algunos modelos que precisan de medidas de calidad muy específicas, todas y cada una de esas medidas depende de los objetivos establecidos. Así que para este proyecto solo se piensa definir como medida principal la métrica de precisión del modelo en predicción, también tratar de obtener la pérdida de entrenamiento y validación más baja en los posibles modelos que se plantean.

4.3.3 Seleccionar El Modelo.

Esta tarea es una de las más sencilla, desde un inicio se ha planteado el uso de redes neuronales convolucionales como modelo de Deep learning. También se han planteado modelos de redes neuronales recurrentes que tienen un amplio uso en el análisis de texto, sin embargo, ese tipo de redes consume más recursos software y hardware y por ende la cantidad de datos debe ser mucho mayor. Las especificaciones y demás características del modelo se mencionan más adelante.

4.3.4 Entrenamiento Del Modelo.

En este apartado es importante considerar varios factores para realizar el entrenamiento del modelo según (Verma, 2019), (Vimal Shrivastava, 2020) y (Ameya D. Jagtap, 2020), tales como, el optimizador, regularización, validación cruzada, función de pérdida, función de activación, entre otras características propias que se mencionan más adelante.

En la generación de este modelo, se usa ADAM como optimizador del modelo, producto de la investigación varias fuentes afirman la superioridad de este optimizador gracias a sus características como:

- Reduce la exigencia en recursos hardware y software.
- Maneja una tasa de aprendizaje adaptativa, lo que permite que esparza los datos de manera eficiente para evitar sobre entrenamiento.
- Reduce el tiempo de entrenamiento.
- Aumenta la eficiencia del modelo.

La regularización es una técnica para ajustar la carga de datos y prevenir sobre entrenamiento o sub-entrenamiento y se puede aplicar a través de diferentes funciones. En este modelo se usa la función Dropout, la aleatoriamente elimina algunas neuronas para regular los pesos de los datos. También se usa la función MaxPooling la cual reduce el tamaño de los parámetros para un mejor entendimiento y procesamiento de las demás neuronas.

La validación cruzada es una técnica que divide los datos para entrenamiento y validación, de esa manera los datos que usa para validar no serán los mismo que se usan para entrenar y así genera predicciones más acertadas ver *Figura 19*.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

Figura 19 Creación del dataset de entrenamiento y validación (validación cruzada)

La función de pérdida no es más que el error de predicción de la red neuronal y existen varios tipos de funciones, cada una para un problema específico (Verma, 2019). En este modelo se usa categorical-crossentropy por lo que se usa para la clasificación de varias clases (con violencia o sin violencia en este proyecto).

La función de activación es un nodo que se ubica al final o entre las redes neuronales, este determina el valor de la salida. Para este se usan dos tipos de activaciones RELU, que es la más usada en Deep learning y arroja valores entre cero e infinito, activación softmax que es necesaria para usar la función de pérdida mencionada anteriormente, ya que asigna un valor entre uno y cero, además esta recibe la salida de la activación anterior para luego clasificarlos datos.

Finalmente, en el proceso de entrenamiento se usa el modelo predefinido que se muestra en la *Figura 19*, sin embargo existen algunos valores que se modifican en varias pruebas para determinar su precisión, como: número de épocas en que se realiza en entrenamiento, batch-size que es el número que se divide por la cantidad total de datos y el resultado son los datos que se entrenan por época, la tasa de aprendizaje que se define dentro del optimizador seleccionado, el número de filtros y el tamaño de kernel de la capa convolucional y el número de Dropout que se desea establecer la función.

Antes de finalizar, se menciona que existe una capa que debe ir por obligación el cualquier modelo que pretende trabajar con análisis o clasificación de texto, se llama Embedding donde esta capa convierte cada palabra en un vector de valores y longitud fija, el cual tiene valores de ceros o unos. Al final esta capa trabaja como una tabla de búsqueda para las palabras, donde las palabras son las claves y los vectores de palabras son los valores. Además, esta capa recibe unos valores fijos para realizar un entrenamiento acertado, como tamaño del vocabulario, longitud del vector de cada palabra y longitud máxima de una secuencia, en la segunda línea de la *Figura 20* se puede observar la capa de Embedding.

```

model = Sequential() # inicializing the Sequential nature for CNN model
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=MAX_SEQUENCE_LENGTH ))
model.add(Conv1D(128, 5, padding='same', activation='relu'))
model.add(MaxPooling1D(4))
model.add(Dropout(0.5))
model.add(Dense(250, activation='relu'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
...
opt = tf.keras.optimizers.Adam()

model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy', 'Precision', 'Recall', 'AUC'])
model.summary()
automodelo = keras.callbacks.ModelCheckpoint('cnnmodel.h5', monitor='val_loss', verbose=1, save_best_only=True,
                                           save_weights_only=False, mode='auto', period=1)

epochs = 10
# Fitting the data onto model
history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test),
                  callbacks=[automodelo],
                  epochs=epochs, batch_size=32, verbose=1)

```

Figura 20 Modelo final de entrenamiento

4.4 FASE 4: EVALUACIÓN.

En esta fase se pretende evaluar los resultados que arrojan los diferentes modelos que se emplearon. La evaluación se realiza acorde a los criterios de éxitos establecidos a través de dos tareas: validación de desempeño y comparación de resultados.

4.4.1 Validar desempeño.

Esta tarea buscar mostrar el proceso y los resultados de los diferentes modelos que se entrenaron, por lo tanto, una buena manera de evaluar la efectividad de los modelos es utilizando los indicadores que se establecieron en criterios de éxito de Machine Learning, los cuales son: precisión en validación (`val_accuracy`) mayor de setenta por ciento (70%) mientras la precisión en entrenamiento no debe superar por más de quince (15) puntos a la validación y el uso de una red convolucional Deep learning. Aun así, se deben tener en cuenta la validación en entrenamiento (`accuracy`), la pérdida del modelo (`loss`).

Teniendo en cuenta lo anterior se crea un modelo base ver *Figura 21* con el que se pretende realizar los entrenamientos, teniendo en cuenta que se cambian sus parámetros de entrenamiento como: tasa de aprendizaje (learning rate), épocas de entrenamiento (epochs), tamaño de lotes de entrenamiento (batch-size), estos parámetros según (Aitor Lewkowycz, 2020), (Aboozar Taherkhani, 2020) son los que definen la precisión del modelo y como se va entrenar.

```
model = Sequential() # initilaizing the Sequential nature for CNN model
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=MAX_SEQUENCE_LENGTH ))
model.add(Conv1D(128, 5, padding='same', activation='relu'))
model.add(MaxPooling1D(4))
model.add(Dropout(0.5))
model.add(Dense(250, activation='relu'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
```

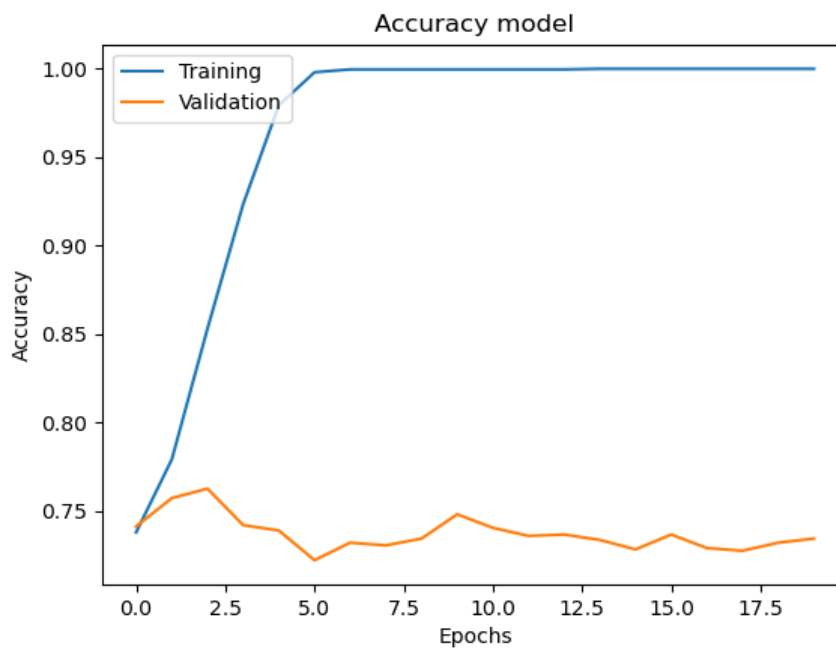
Figura 21 Modelo base de entrenamiento

El primer entrenamiento realizado por este modelo, se realiza con los siguientes parámetros: Batch-size de 128, epochs de 20, learning rate de 0.0001, ver *Figura 22*. Los resultados obtenidos por este modelo son de: 73,44% de precisión en validación y 100% en precisión de entrenamiento ver *Gráfica 1*, una pérdida de 1.6450 en validación y de 0,0 en entrenamiento, lo cual es muy alto aún ver *Gráfica 2*.

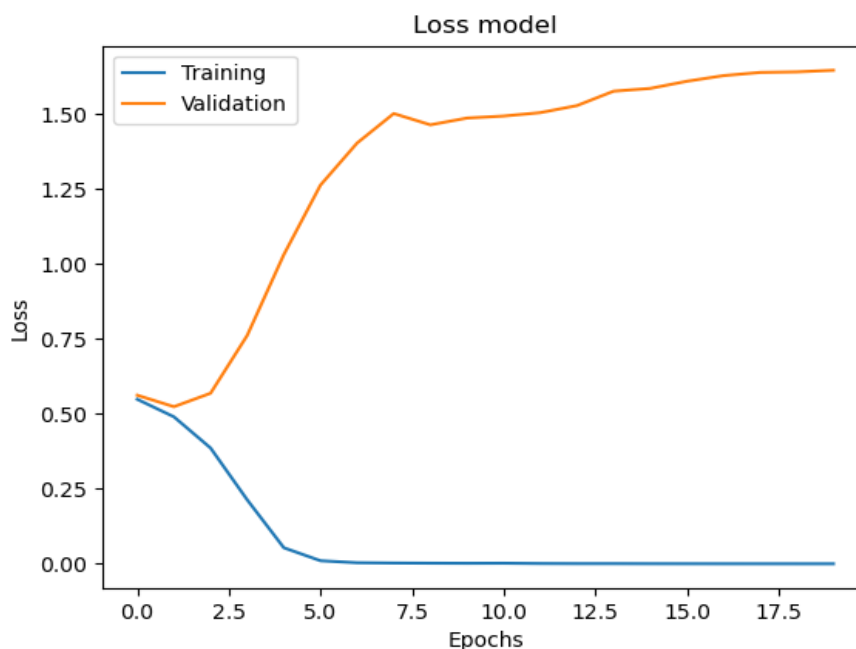
```
lr = 0.0001
opt = tf.keras.optimizers.Adam()
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy', 'Precision', 'Recall', 'AUC'])
model.summary()
automodelo = keras.callbacks.ModelCheckpoint('finalmodel1.h5', monitor='val_loss', verbose=1, save_best_only=True,
                                             save_weights_only=False, mode='auto', period=1)

epochs = 20
# Fitting the data onto model
history = model.fit(X_train, Y_train,
                   validation_data=(X_test, Y_test),
                   callbacks=[automodelo],
                   epochs=epochs, batch_size=128, verbose=1)
```

Figura 22 Parámetros primer entrenamiento



Gráfica 1 Precisión (Accuracy) primer entrenamiento



Gráfica 2 Pérdida (Loss) primer entrenamiento

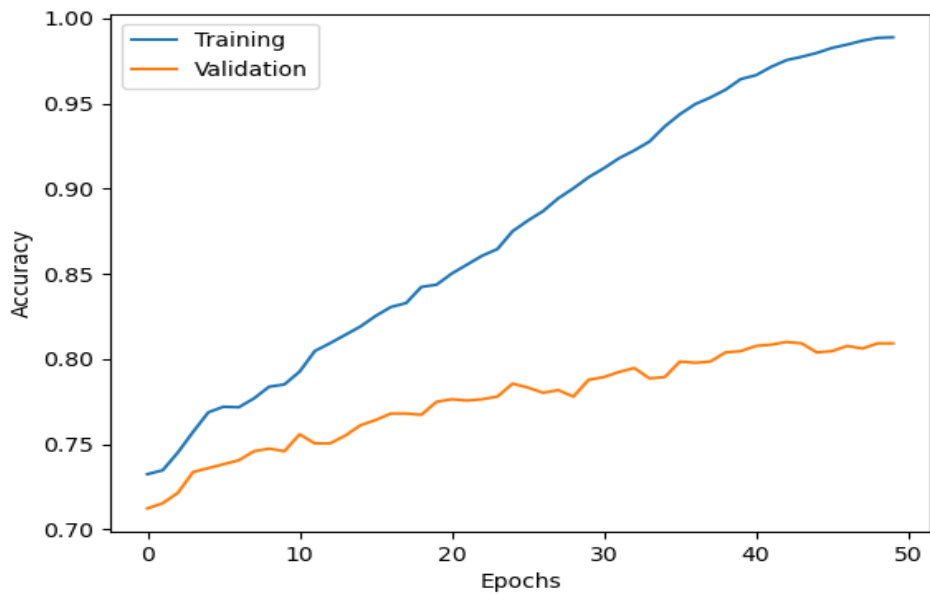
En un segundo entrenamiento de este modelo, se cambian los parámetros: Epochs de 50, learning rate de 0.001, ver *Figura 23*. Los resultados obtenidos por este modelo son de: 80,02% de precisión en validación y 99% de precisión en entrenamiento ver *Gráfica 3* y una pérdida en validación de 0.51 y pérdida en entrenamiento de 0,05, ver *Gráfica 4*.

```

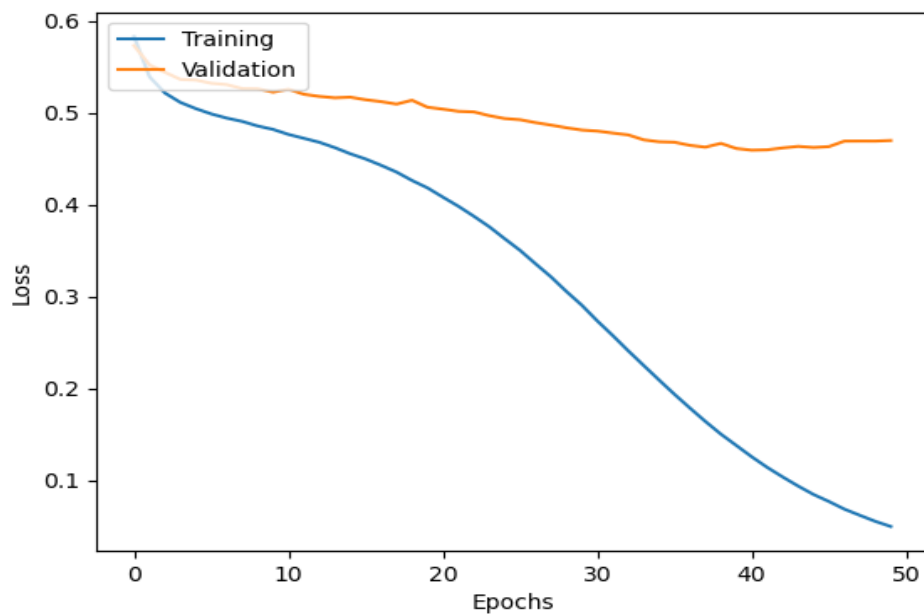
lr = 0.0001
opt = tf.keras.optimizers.Adam()
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy', 'Precision', 'Recall', 'AUC'])
model.summary()
automodelo = keras.callbacks.ModelCheckpoint('finalmodel2.h5', monitor='val_loss', verbose=1, save_best_only=True,
                                             save_weights_only=False, mode='auto', period=1)
epochs = 20
# Fitting the data onto model
history = model.fit(X_train, Y_train,
                    validation_data=(X_test, Y_test),
                    callbacks=[automodelo],
                    epochs=epochs, batch_size=128, verbose=1)

```

Figura 23 Parámetros segundo entrenamiento



Gráfica 3 Precisión (Accuracy) segundo entrenamiento



Gráfica 4 Pérdida (Loss) segundo entrenamiento

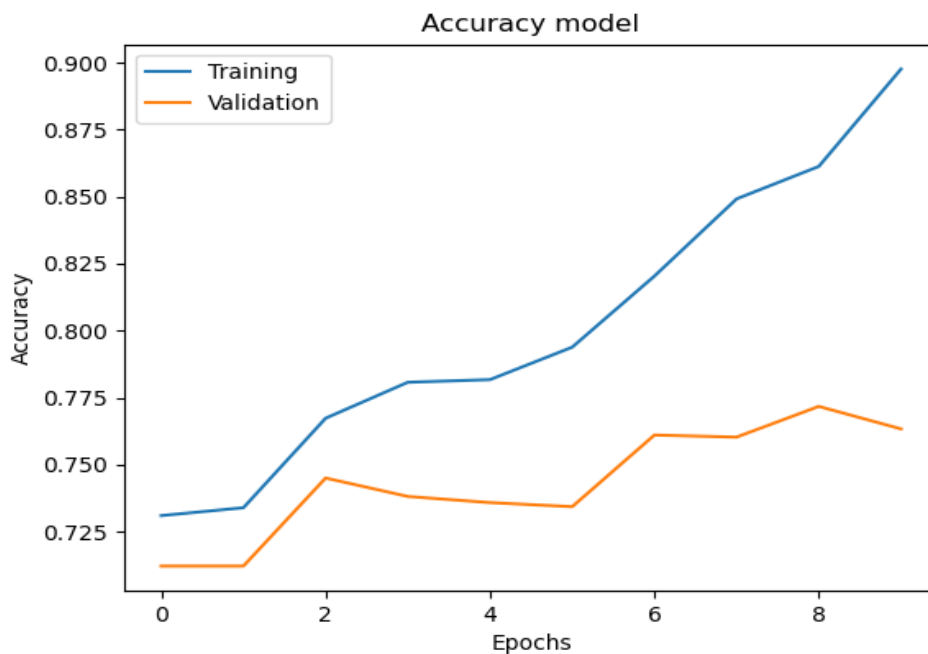
En un tercer entrenamiento usando el modelo base, se reduce la cantidad de épocas de entrenamiento, así de esa manera se evita el sobre entrenamiento que se ve reflejado en las gráficas de los modelos anteriores. El parámetro de batch-size se

igual a 10 y un learning rate de 0,0001, ver *Figura 24*. Los resultados arrojados en este entrenamiento son: 76,2% de precisión en validación y 89,77% de precisión en entrenamiento, ver *Gráfica 5* y una pérdida en validación de 0,53 y 0,15 en pérdida de entrenamiento, ver *Gráfica 6*.

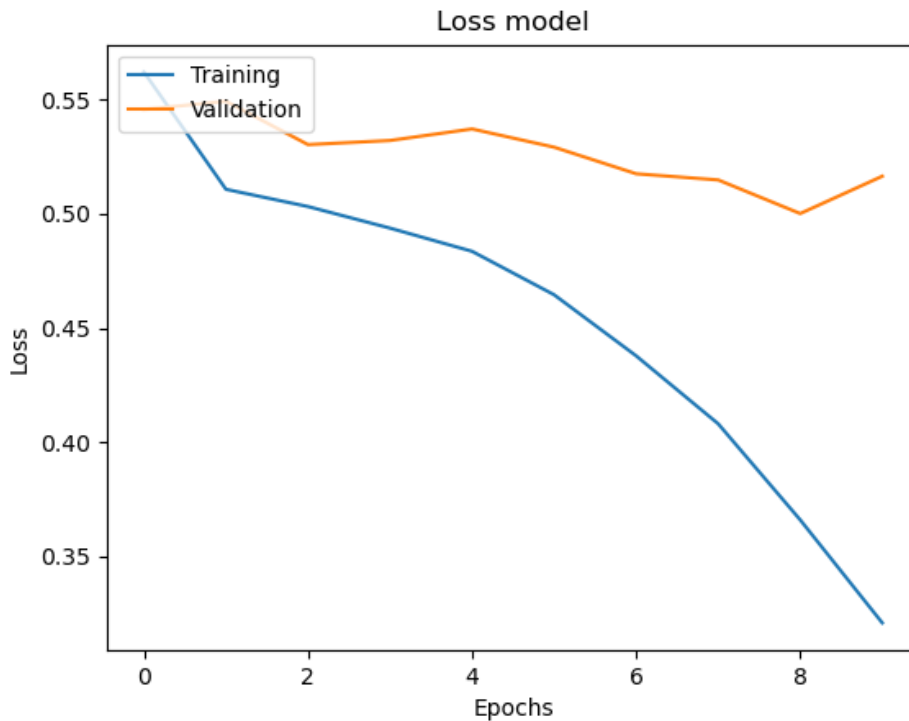
```
lr = 0.0001
opt = tf.keras.optimizers.Adam()
model.compile(loss='categorical_crossentropy', optimizer=opt,
              metrics=['accuracy', 'Precision', 'Recall', 'AUC'])
model.summary()
automodelo = keras.callbacks.ModelCheckpoint('modeloglobalmax.h5',
                                             monitor='val_loss',
                                             verbose=1, save_best_only=True,
                                             save_weights_only=False, mode='auto',
                                             period=1)

epochs = 10
# Fitting the data onto model
history = model.fit(X_train, Y_train,
                   validation_data=(X_test, Y_test),
                   callbacks=[automodelo],
                   epochs=epochs, batch_size=128, verbose=1)
```

Figura 24 Parámetros tercer entrenamiento



Gráfica 5 Precisión (Accuracy) tercer entrenamiento

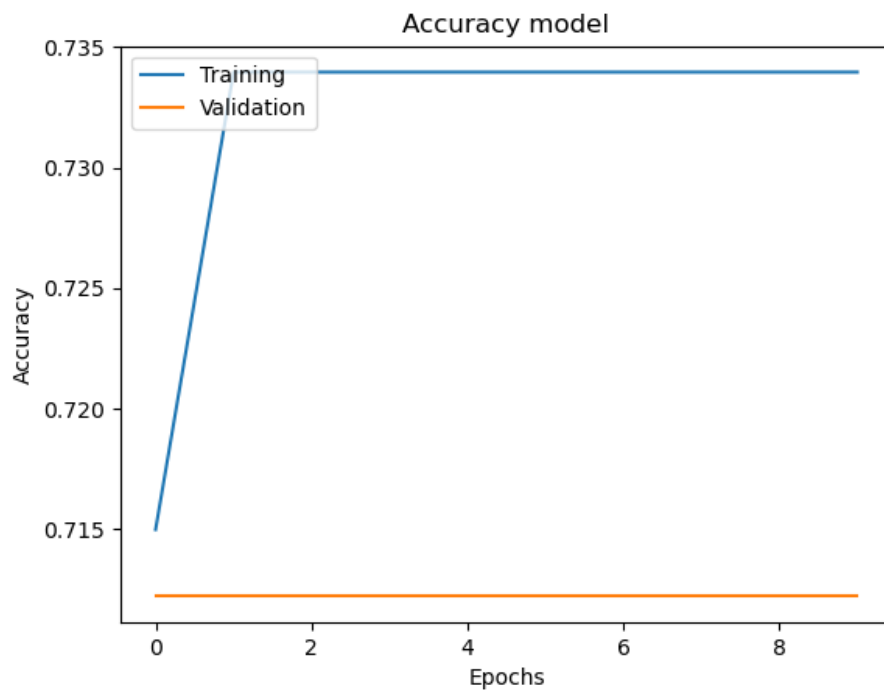


Gráfica 6 Pérdida (Loss) tercer entrenamiento

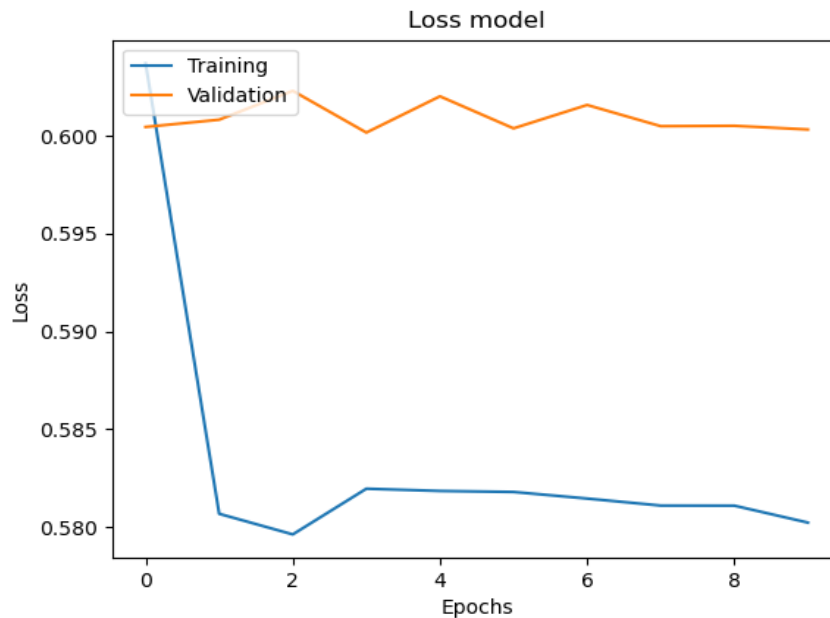
El cuarto modelo de entrenamiento ver *Figura 25*, se toma de programgeek.com, una página dedicada a desarrolladores que tienen modelos de Deep Learning y Machine Learning para uso y replicación de la comunidad del software. Para este modelo se cambian las capas ocultas de Dropout, Dense, para agregar una capa LSTM que son usadas para que la información pueda persistir a través del tiempo en cada iteración realizada en el entrenamiento. Los parámetros del modelo son los mismos del último modelo ver *Figura 24*. Los resultados obtenidos con este modelo son: 71,22% de precisión en validación y un 73,5% de precisión en entrenamiento, ver *Gráfica 7*, una pérdida en validación de 0.6003 y perdida en entrenamiento de 0,580, ver *Gráfica 8*.

```
model = Sequential() # initilaizing the Sequential nature for CNN model
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=MAX_SEQUENCE_LENGTH ))
model.add(Conv1D(128, 5, padding='same', activation='relu'))
model.add(MaxPooling1D(4))
model.add(LSTM(64))
model.add(Dense(2, activation='softmax'))
```

Figura 25 modelo 2 con capa LSTM



Gráfica 7 Precisión (Accuracy) cuarto entrenamiento LSTM



Gráfica 8 Pérdida (Loss) cuarto entrenamiento LSTM

El quinto y último entrenamiento usa el modelo base, pero se le agrega una capa LSTM como en el anterior entrenamiento ver *Figura 26*. Se hace un cambio de parámetros como: Batch-size de 128, epochs de 50, learning rate de 0,0001, ver *Figura 27*. Los resultados obtenidos de este entrenamiento son: 71,0% de precisión en validación y 100% precisión en entrenamiento, ver *Gráfica 9*, una pérdida en validación de 2,25 y 0,02 de pérdida en entrenamiento, ver *Gráfica 10*.

```

model = Sequential() # initalaizing the Sequential nature for CNN model
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=MAX_SEQUENCE_LENGTH ))
model.add(Conv1D(128, 5, padding='same', activation='relu'))
model.add(MaxPooling1D(4))
model.add(LSTM(64))
model.add(Dropout(0.5))
model.add(Dense(250, activation='relu'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

```

Figura 26 Quinto entrenamiento modelo base con LSTM

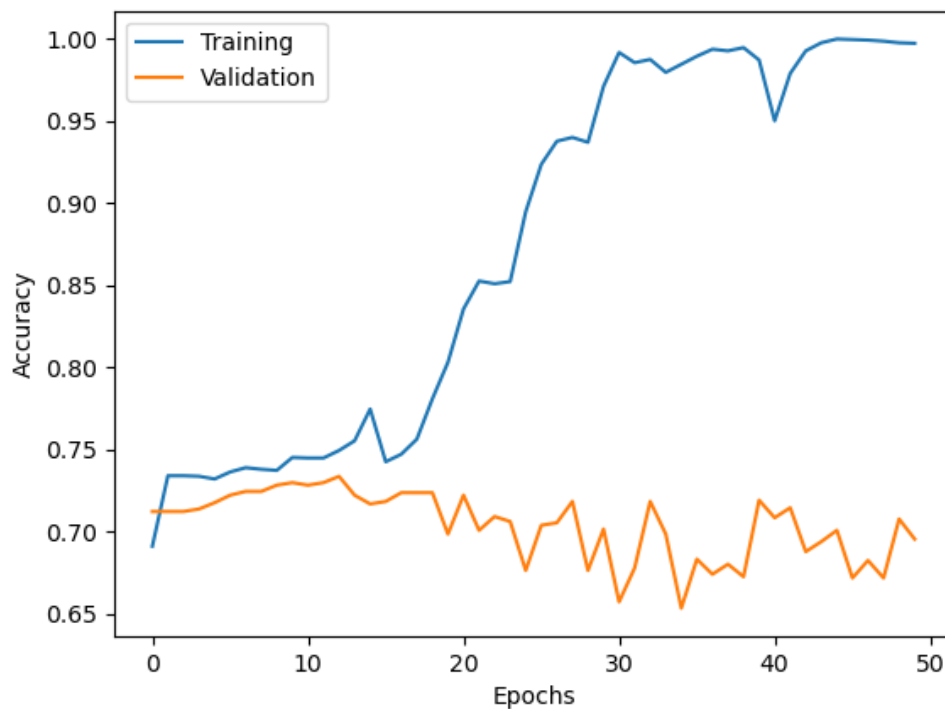
```

lr = 0.0001
opt = tf.keras.optimizers.Adam()
model.compile(loss='categorical_crossentropy', optimizer=opt,
              metrics=['accuracy', 'Precision', 'Recall', 'AUC'])
model.summary()
automodelo = keras.callbacks.ModelCheckpoint('finalmodel4.h5',
                                             monitor='val_loss',
                                             verbose=1, save_best_only=True,
                                             save_weights_only=False, mode='auto',
                                             period=1)

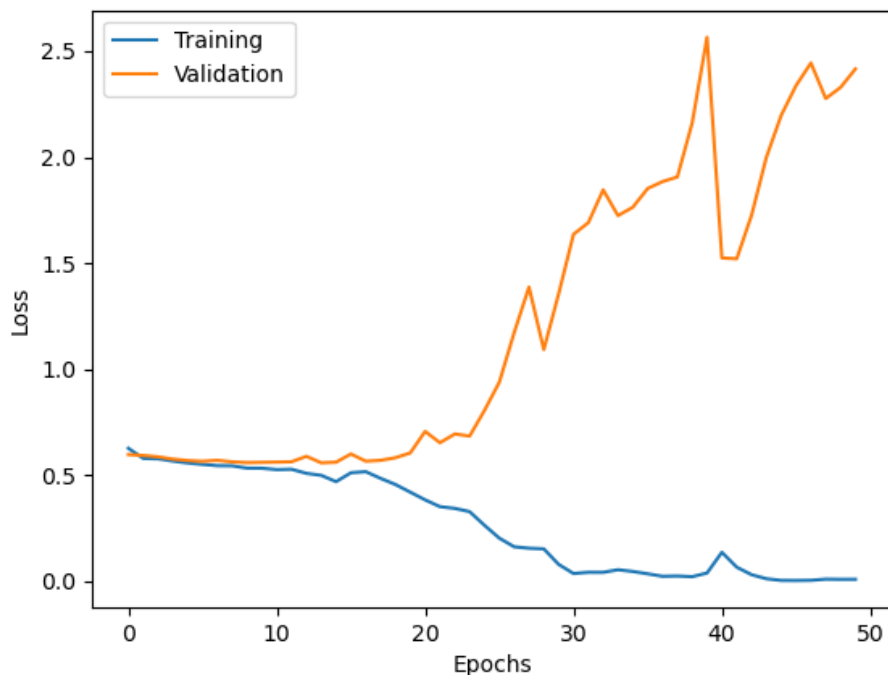
epochs = 50
# Fitting the data onto model
history = model.fit(X_train, Y_train,
                   validation_data=(X_test, Y_test),
                   callbacks=[automodelo],
                   epochs=epochs, batch_size=128, verbose=1)

```

Figura 27 Parámetros quinto entrenamiento modelo base LSTM



Gráfica 9 Precisión (Accuracy) quinto entrenamiento modelo base LSTM



Gráfica 10 Pérdida (Loss) quinto entrenamiento modelo base LSTM

Los resultados que se presentan en esta etapa, son una pequeña parte de todos los intentos posibles, pese a la cantidad realizada solo se posee registros de gráficas y porcentajes de los mostrados anteriormente, debido al tiempo que toma en ejecutarse un entrenamiento de manera exitosa y los recursos de hardware que consume. Ahora bien, en la siguiente etapa se realizará un análisis de los resultados previamente mostrados.

4.4.2 Comparar Resultados De Criterios De Éxito Con Los Definidos.

En esta etapa se realiza una comparativa de los resultados de los modelos entrenados en la etapa anterior, con objetivo de determinar el mejor modelo y realizar su implementación. Los valores que se pretenden evaluar son: precisión en entrenamiento (accuracy) mayor al 70%, precisión en validación (val_accuracy) mayor al 70%, pérdida en entrenamiento (loss) y pérdida en validación (val_loss), aunque uno de los criterios de evaluación sea solo la precisión en validación, es

necesario tener en cuenta las demás métricas para realizar una elección más acertada. A continuación, en la *Tabla 3* se muestran los resultados.

	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5
Accuracy	100%	99%	89,77%	73,5%	100%
Val_accuracy	73,44%	80,02%	76,2%	71,22%	71,0%
Loss	0,0	0,05	0,15	0,580,	0,02
Val_loss	1.6450	0.51	0,53	0.6003	2,25
Diferencia de precisión	26,56	18,98	13,57	2,28	29,0
Diferencia de pérdida	1,6450	0,46	0,38	0,0203	2,23

Tabla 3 Tabla de comparativa de resultados

Fuente: Elaboración Propia

Primer criterio de éxito, precisión en validación (accuracy) debe superar el setenta por ciento (70%) mientras la precisión en entrenamiento no debe superar por más de quince (15) puntos a la validación.

Este criterio se cumple analizando la *Tabla 3*, en la que se observa que, los cinco modelos cumplen con la condición de superar el 70% de precisión en validación, no obstante, solo dos modelos cumplen con la condición de que la diferencia entre la precisión en validación y en entrenamiento no debe ser mayor a 15, además de tener la menor diferencia en pérdida. A raíz de esto, el modelo 3 y el modelo 4 son los seleccionados.

La razón de este criterio es que cuando la métrica accuracy se aleja demasiado de val-accuracy causa sobre-entrenamiento, es por eso que un rango aceptable es de 15 unidades entre ellas.

Ahora bien, para elegir el modelo más óptimo entre los dos seleccionados, se debe analizar las gráficas de resultados de cada uno que están en la etapa anterior.

La *Gráfica 5* perteneciente al modelo 3, muestra un crecimiento constante de la métrica de precisión, donde solo se usan 10 épocas para evitar el sobreentrenamiento y se evidencia que existe un 76% de precisión.

Por otro lado, el modelo 4 cumple con los criterios de aceptación, sin embargo, la *Gráfica 7* muestra que el entrenamiento y validación nunca mejoran en ninguna época y quedan estáticos en un valor de aprendizaje, por lo tanto, se toma la decisión de descartar este modelo, dejando así el modelo 3 para despliegue e implementación.

Para el cumplimiento del segundo criterio de éxito se desarrolló una red neuronal convolucional que cumple las características mencionadas en la sección de la arquitectura del modelo. La arquitectura de la red neuronal que atañe a esta investigación puede ser verificada de manera detallada en la *Figura 3*. Dentro de esta se puede probar el cumplimiento de las características que debe contener una red convolucional.

A fin de cumplir el criterio de éxito establecido como la creación de un entorno visual para realizar las pruebas de predicción del modelo, se desarrolló una página web básica, en la que se ingresa la letra de una canción y este usa el modelo entrenado para realizar la predicción y la clasificación. En la *Figura 28*, se evidencia un fragmento del código con el cual se carga el modelo entrenado anteriormente para su posterior uso en predicción y clasificación de las canciones.

Para cargar y usar el modelo de manera correcta, se debe tener las mismas condiciones y características del tokenizador usado para el entrenamiento, así de esa manera, las palabras nuevas que ingresen para ser tokenizadas, van a tener similitud con las palabras del modelo entrenado. De no cumplirse los mismos criterios, las predicciones no serán fiables.

En la siguiente fase se evidenciará que la página creada satisface el tercer criterio de éxito. Así de esta manera, se concluye que esta fase da como resultado satisfactorio el cumplimiento de los criterios de éxito establecidos.

```
def get_prediction(texto):
    texto = pro_text(texto)
    # CARGANDO MODELO ENTRENADO
    model = tf.keras.models.load_model('modeloglobalmax.h5')

    # TOKENIZADO DE LAS LETRAS DE CANCIONES
    tokenizer = Tokenizer(num_words=MAX_NB_WORDS,
                          filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
                          lower=True, split=' ')
    tokenizer.fit_on_texts(texts=texto)
    X = tokenizer.texts_to_sequences(texts=texto)
    X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH, padding='post')
    # pred = np.argmax(model.predict(X), axis=-1)
    pred = model.predict(X)
    labels = ['no contiene violencia', 'contiene violencia']
    print("Valores de predicción : \n ", pred)
    print("Resultado aproximado: \n ", labels[np.argmax(pred)])

    return labels[np.argmax(pred)]
```

Figura 28 Cargar modelo entrenado

4.5 FASE 5: DESPLIEGUE.

El objetivo de esta fase es explicar la implementación y ejecución del proyecto realizado, de manera que el cliente o interesado puedan entender fácilmente su uso.

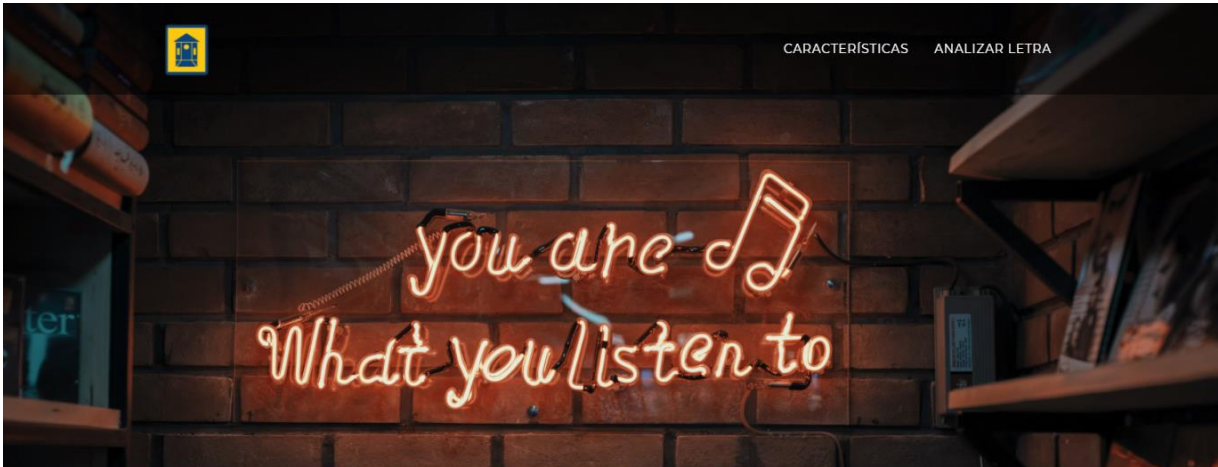
4.5.1 Evaluación Del Modelo En Condiciones De Producción.

La evaluación del modelo se puede ver evidenciada en la *Figura 29*, la cual es desarrollo con base en el tercer criterio de aceptación, que permite la visualización

de las predicciones del modelo en tiempo real ingresando una canción y clasificándola, además de mostrar algunas de las palabras o frases que se determinan violentas contra la mujer en una canción.

La predicción del modelo al comprender un porcentaje de 76% de precisión puede fallar en la clasificación, además de que algunas frases que denotan violencia contra la mujer pueden no mostrarse en la tabla. Se destaca también que, el porcentaje más alto de las canciones que se clasificaron como violentas, contienen violencia de tipo sexual y violencia simbólica.

Esta etapa se puede dar como aceptada, teniendo en cuenta el funcionamiento del modelo a través del entorno visual desarrollado y como se evidencia en la *Figura 30*, se prueba con una canción llamada “tremenda sata” de género de reggaetón, en la que se evidencia el alto contenido sexual que maneja la canción, clasificándola, así como violenta.



CARACTERÍSTICAS

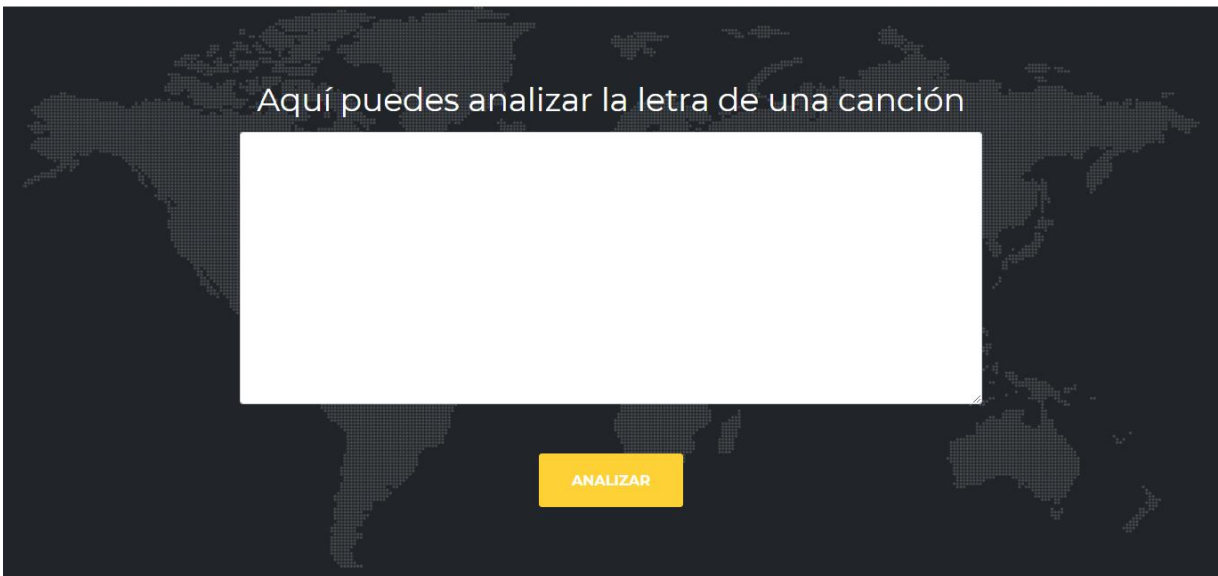
Este prototipo propone una alternativa basada en inteligencia artificial para el análisis de violencia de genero contra la mujer en letras de canciones. Se presenta como proyecto de investigación del semillero I+D en cabeza de la ingeniera María Isabel Vidal y el estudiante de decimo semestre de ingeniería informática Didier Enrique Ramirez Diaz, de la institución universitaria colegio mayor del cauca.

Deep Learning

Las redes neuronales convolucionales son básicamente una arquitectura del Deep learning (aprendizaje profundo) que permite a los equipos computacionales procesar y clasificar todo tipo de imágenes y textos. Aunque su principal objetivo es procesamiento de imágenes, en los últimos años se ha popularizado enormemente al conseguir resultados sorprendentes en el procesamiento de texto, impactando así en el área de vision computer (Alvares, 2017).

Violencia contra la mujer

La violencia contra la mujer es acto que puede producir un daño físico, psicológico o emocional que se puede presentar en cualquier contexto. Desafortunadamente hay cientos de mensajes que pasan desapercibidos en las canciones, debido a esto se propone la investigación e implantación de un prototipo funcional que permita el reconocimiento de palabras o frases presentadas dentro de las canciones que puedan promover la violencia contra la mujer.



© Didier Ramirez 2021

Figura 29 Página de implementación del modelo entrenado



Figura 30 Prueba del modelo entrenado

4.5.2 Estrategia De Implantación.

En esta etapa se realiza un plan de implementación del proyecto, la manera correcta para que el usuario o personas interesadas puedan ejecutar el proyecto de manera adecuada.

Para poder implantar este proyecto, es necesario en primer lugar, tener acceso al dataset donde se encuentran las canciones seleccionadas y clasificadas como violentas o no violentas. Si no se tiene el dataset ya etiquetado, se puede realizar una clasificación a mano o con los métodos de clasificación mencionados en 4.2.3 *Construir Los Datos*. Así, de esa manera se tiene más control de los parámetros que se desean etiquetar como violentos.

En segundo lugar, es necesario realizar las instalaciones necesarias de los frameworks, librerías, extensiones y demás recursos que se utilizaron en la ejecución de este proyecto, sin embargo las instalaciones requeridas o principales de este proyecto deben ser: Anaconda para crear un entorno virtual en el equipo para el desarrollo, Python ya que es el lenguaje principal de desarrollo de este proyecto, TensorFlow y Keras que son precisamente para el desarrollo de proyectos Deep Learning y Machine Learning, por ultimo un entorno de desarrollo IDE como PyCharm o Visual Studio Code.

Por último, luego de realizar los ajustes necesarios y probar que todo esté en orden, se ejecuta el archivo main.py ver *Figura 31*, el cual contiene todo el código de preprocesado de texto y entrenamiento del modelo de red neuronal convolucional, se define un nombre al modelo para guardarlo y usarlo posteriormente ver *Figura 32*.

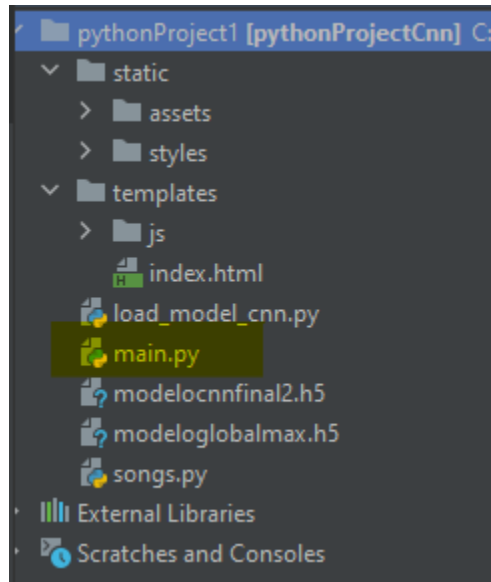


Figura 31 Main.py

```
automodelo = keras.callbacks.ModelCheckpoint('finalmodel4.h5',  
                                             monitor='val_loss',  
                                             verbose=1, save_best_only=True,  
                                             save_weights_only=False, mode='auto',  
                                             period=1)
```

Figura 32 Cambiar nombre de modelo a entrenar

Posteriormente, para usar el modelo en el entorno visual desarrollado, se debe ejecutar el archivo `load_model_cnn.py`, ver *Figura 33*, no obstante, si se desea cambiar y usar su propio modelo puede hacer cambiando el nombre el modelo en el siguiente campo ver *Figura 34*. Para ver la página en el navegador, se hace clic en la dirección que aparece en el log de ejecución del proyecto, ver *Figura 35*.

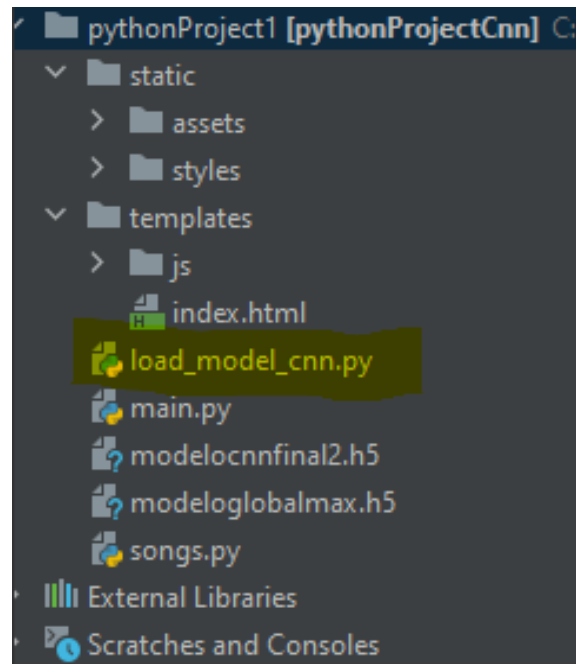


Figura 33 Archivo load_model_cnn.py

```
def get_prediction(texto):
    texto = pro_text(texto)
    # CARGANDO MODELO ENTRENADO
    model = tf.keras.models.load_model('finalmodel4.h5')

    # TOKENIZADO DE LAS LETRAS DE CANCIONES
    tokenizer = Tokenizer(num_words=MAX_NB_WORDS,
                          filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
                          lower=True, split=' ')
    tokenizer.fit_on_texts(texts=texto)
    X = tokenizer.texts_to_sequences(texts=texto)
    X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH, padding='post')
    # pred = np.argmax(model.predict(X), axis=-1)
    pred = model.predict(X)
    labels = ['no contiene violencia', 'contiene violencia']
    print("Valores de predicción : \n ", pred)
    print("Resultado aproximado: \n ", labels[np.argmax(pred)])

    return labels[np.argmax(pred)]
```

Figura 34 Cambiar nombre de modelo a cargar

```
Run: load_model_cnn x
* Debugger is active!
* Debugger PIN: 323-673-279
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Detected change in 'C:\\Users\\didie\\PycharmProjects\\pythonProject1\\load_model_cnn.py', reloading
* Restarting with stat
2021-06-07 17:55:31.629731: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cudart64_110.dll'; dLError:
cudart64_110.dll not found
2021-06-07 17:55:31.630051: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
* Debugger is active!
* Debugger PIN: 323-673-279
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figura 35 Url del entorno visual de despliegue

Por último, para usar la interfaz gráfica y realizar la predicción de alguna canción, se debe copiar cualquier letra de alguna canción en español, luego se hace clic en el botón “analizar” de color amarillo ver *Figura 29*, la página se recargará con los resultados de la predicción, donde se muestra en un texto si la canción contiene violencia o no y seguido de ello, se ven un par de tablas que contienen la frecuencia de palabras más usadas en esa canción y las palabras que generan violencia ver *Figura 30*.

4.6. FASE 6: MONITOREO Y MANTENIMIENTO.

Esta fase no se cumple, debido a que el proyecto finaliza en la fase anterior. Al ser este un proyecto de investigación para el semillero de I+D de la institución universitaria colegio mayor del cauca, finaliza con la entrega y sustentación ante el jurado propuesto por el consejo estudiantil. Se espera que en algún futuro el semillero pueda continuar con esta propuesta con la ayuda de otro estudiante.

4.6.1 Monitorear datos.

4.6.2 Actualización de librerías y datos.

Para la implementación de un proyecto Deep Learning se requieren herramientas de software y hardware sofisticadas o con una gran capacidad de hardware, se sugiere una GPU (Unidad de procesamiento gráfico), un procesador de cuarta generación y la memoria RAM de 8gb o mayor.

El procesamiento de texto es un área muy amplia que puede abarcar diferentes usos, como por ejemplo: control parental de padres e hijos, detección de correos spam, clasificación de textos, entre otros.

A pesar de las dificultades presentadas en el proyecto, se logra cumplir con los objetivos planteados en el proyecto y se propone continuar con este tipo de proyectos con características sociales haciendo uso de la tecnología.

BIBLIOGRAFÍA

- Aboozar Taherkhani, T. M. (09 de 2020). AdaBoost-CNN: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. Obtenido de <https://www.sciencedirect.com/science/article/abs/pii/S0925231220304379>
- Aitor Lewkowycz, Y. B.-D.-A. (04 de 03 de 2020). The large learning rate phase of deep learning: the catapult mechanism. Obtenido de <https://arxiv.org/abs/2003.02218>
- Akansha Jain, I. A. (2019). https://scholar.google.com.co/scholar?q=parallel+dots+text+analysis&hl=es&as_sdt=0&as_vis=1&oi=scholart. Obtenido de ParallelDots: <https://www.aclweb.org/anthology/S19-2029.pdf>
- Alvares, C. y. (12 de 2017). Uso de redes neuronales convolucionales aplicado a sentiment analysis. Obtenido de <http://repositorio.uchile.cl/handle/2250/174536>
- Ameya D. Jagtap, K. G. (01 de 03 de 2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. Obtenido de <https://www.sciencedirect.com/science/article/pii/S0021999119308411>
- Beltrán, C. (2012). Obtenido de <https://rehip.unr.edu.ar/handle/2133/14279>
- colah. (27 de 8 de 2015). colah. Obtenido de <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Colombia, O. M. (2015). ONU Mujeres Colombia. Obtenido de ONU Mujeres Colombia: <https://colombia.unwomen.org/es/onu-mujeres-en-colombia/las-mujeres-en-colombia>
- Cun, L., Boser, B., Denker, J. S., & Henderson, J. (1990). Handwritten Digit Recognition with a Back-Propagation Network. In Advances in neural information processing systems, 396- 404.
- David Llanos, M. R. (4 de 6 de 2013). Aprendizaje Supervisado y no Supervisado. Obtenido de

<http://redesneuronal.blogspot.com/#:~:text=Redes%20con%20aprendizaje%20supervisado,partir%20de%20una%20entrada%20determinada.>

Díez, C. (1 de 11 de 2015). amecopress. Obtenido de amecopress: <https://amecopress.net/Musica-y-Violencia-Machista-Aunque-sea-de-forma-sutil-aun-hay-muchas-letras-sexistas>

Edgar Altszyler, P. B. (2015). Laboratorio de Inteligencia Artificial Aplicada, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires. Obtenido de <http://sedici.unlp.edu.ar/handle/10915/52169>

G, G. E., & elizabeth, A. (Agosto de 2019). Revista Ibérica de Sistemas e Tecnologías de Información. Obtenido de <https://search.proquest.com/openview/be0e95c93d8a58159ee4fbaf913e1661/1?pq-origsite=gscholar&cbl=1006393>

Gómez Escarda, M. &. (2016). La violencia contra las mujeres en la música: Una aproximación metodológica. metadatos. Revista De Ciencias Sociales, 4(1). Obtenido de La violencia contra las mujeres en la música: Una aproximación metodológica.methaodos. Revista De Ciencias Sociales, 4(1).: <https://doi.org/10.17502/m.rcs.v4i1.115>

Guillaume Lemaître, F. N. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in. Obtenido de <https://jmlr.org/papers/volume18/16-365/16-365.pdf>

Gutiérrez, R. M. (2005). Obtenido de https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-09342005000300003&lng=es&nrm=iso#*

Hecmaramco. (25 de 11 de 2020). Medium. Obtenido de <https://medium.com/saturdays-ai/o%C3%ADmos-o-escuchamos-m%C3%BAsica-2859085f057f>

IBM. (2018). IBM. Obtenido de <https://www.ibm.com/co-es/analytics/machine-learning>

ionos. (s.f.). Obtenido de <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-el-web-scraping/#:~:text=Web%20Scraping%3A%20definici%C3%B3n,-Durante%20el%20web&text=Con%20el%20web%20scraping%20se,se%20denomina%20acertadamente%20image%20scraping.>

Jacobo, H. L. (2016). Obtenido de <https://www.gelbukh.com/thesis/Hugo%20Librado%20Jacobo%20-%20MSc.pdf>

- JOSÉ THOMAS VILLAVICENCIO ALVAREZ, M. E. (12 de 2017). Obtenido de http://opac.pucv.cl/pucv_txt/txt-8000/UCC8097_01.pdf
- Jungsu Park, H. L.-Y. (2019). Algal Morphological Identification in Watersheds for Drinking Water Supply Using Neural Architecture Search for Convolutional Neural Network. Obtenido de <https://www.mdpi.com/2073-4441/11/7/1338>
- Laureano, A. (8 de 2019). Obtenido de <http://clei2019.utp.ac.pa/storage/app/uploads/public/5d8/cf7/d74/5d8cf7d74d341949576565.pdf>
- Lopez, R. A. (2019). Obtenido de <http://repositorio.uchile.cl/handle/2250/174536>
- Luis Jimenez, J. F. (2017). Reconocimiento de dígitos escritos a mano usando aprendizaje profundo. Obtenido de https://www.academia.edu/40196440/Reconocimiento_de_d%C3%ADgitos_escritos_a_mano_usando_aprendizaje_profundo
- Luz, A. (14 de Mayo de 2020). Twitter. Obtenido de Twitter: <https://twitter.com/analuzsaso/status/1261097809446502400>
- machado, G. (3 de 2 de 2021). kaggle. Obtenido de <https://www.kaggle.com/giovanimachado/hate-speech-bert-cnn-and-bert-mlp-in-tensorflow#Implementation-of-text-classification-with-BERT>
- Magino, J. B. (2020). Obtenido de <http://repositorio.unajma.edu.pe/handle/123456789/588>
- Md Abul Bashar, R. N. (2018). paperswithcode. Obtenido de <https://paperswithcode.com/paper/misogynistic-tweet-detection-modelling-cnn>
- Mujeres, O. (29 de Noviembre de 2017). ONU Mujeres. Obtenido de ONU Mujeres: <https://colombia.unwomen.org/es/como-trabajamos/violencia-contra-las-mujeres>
- Olguín, A. (9 de 2015). Obtenido de https://www.researchgate.net/profile/Alfredo_Olguin/publication/281979580_Que_se_escribe_respecto_al_marxismo_en_redes_sociales_Analisis_de_patrones_de_texto_a_traves_de_Twitter_por_medio_de_Data_Mining/links/5600a93a08aeba1d9f84e48b/Que-se-escribe-respec
- OMS. (29 de 11 de 2017). OMS. Obtenido de WhoInt: <https://www.who.int/es/news-room/fact-sheets/detail/violence-against-women>
- Pedro Angel Ruiz, M. I. (25 de 11 de 2019). dialnet.unirioja. Obtenido de dialnet,unirioja: <https://dialnet.unirioja.es/descarga/articulo/7279807.pdf>

- Raschka, S., & Mirjalili, V. (2019). Python Machine Learning. Aprendizaje automático y aprendizaje profundo con Python, scikit-learn y Tersonflow. Macombo: Madrid.
- Rosa Montañes, R. A. (9 de 2018). Aplicación de un modelo híbrido de aprendizaje profundo para el análisis de sentimiento en twitter. Obtenido de http://ceur-ws.org/Vol-2172/p4_ittainnova_tass2018.pdf
- Saleema Amershi, A. B. (25 de 2 de 2019). Software engineering for machine learning: A case study. Obtenido de https://www.microsoft.com/en-us/research/uploads/prod/2019/03/amershi-icse-2019_Software_Engineering_for_Machine_Learning.pdf
- Sande, J. C. (2018). Obtenido de <http://openaccess.uoc.edu/webapps/o2/bitstream/10609/81435/6/jsobrinostFM0618memoria.pdf>
- SAS. (2020). SAS. Obtenido de https://www.sas.com/es_co/insights/analytics/what-is-artificial-intelligence.html
- Sculley, E. B. (2017). The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction. Obtenido de <https://research.google/pubs/pub46555/>
- Stefan Studer, T. B.-R. (2021). Towards CRISP-ML(Q): A Machine Learning Process Model. Obtenido de <https://www.mdpi.com/2504-4990/3/2/20>
- Torres, J. (2020). Python Deep Learning. Introducción práctica con Keras y Tensorflow 2. Bogotá: AlfaOmega.
- UNAP. (5 de 11 de 2019). UNAP. Obtenido de <https://www.unapiquitos.edu.pe/contenido/actualidades/La-UNAP-presento-en-Panama-importante-investigacion-sobre-IA.php>
- Verma, S. (20 de 06 de 2019). Towardsdatascience. Obtenido de <https://towardsdatascience.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718#:~:text=The%20Loss%20Function%20is%20one,weights%20of%20the%20Neural%20Net.>
- Vimal Shrivastava, S. B. (04 de 2020). Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. Obtenido de https://www.researchgate.net/publication/337816564_Analysis_of_various_optimizers_on_deep_convolutional_neural_network_model_in_the_applicati_on_of_hyperspectral_remote_sensing_image_classification

waki. (2 de 9 de 2017). wakicode. Obtenido de <https://wakicode.com/2017/09/02/distancia-de-levenshtein/>