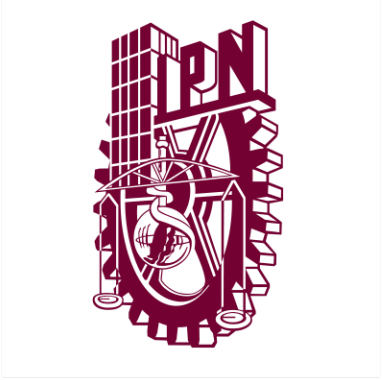


INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS", ZACATENCO

"ANÁLISIS DE CANCIONES INFANTILES APLICANDO ALGORITMOS DE INTELIGENCIA ARTIFICIAL"

T E S I S

PARA OBTENER EL TÍTULO DE

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTA

DAVID SOTO OSORIO

ASESORES

M. EN C. LILIANA CHANONA HERNÁNDEZ
M. EN C. CÉSAR JESÚS NÚÑEZ PRADO
DR. JAIME MORENO ESCOBAR

CIUDAD DE MÉXICO, JUNIO 2022

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TÍTULO DE	**INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**
POR LA OPCIÓN DE TITULACIÓN	**TESIS Y EXAMEN ORAL INDIVIDUAL**
DEBERA (N) DESARROLLAR	**C. DAVID SOTO OSORIO**

"ANÁLISIS DE CANCIONES INFANTILES APLICANDO ALGORITMOS DE INTELIGENCIA ARTIFICIAL"

ANALIZAR LETRAS DE CANCIONES DIRIGIDAS A LOS NIÑOS CON EL PROPÓSITO DE IDENTIFICAR QUE CANCIONES PUEDEN DAR UN MENSAJE NEGATIVO Y PUEDEN INFLUIR DE MANERA NEGATIVA EN SU COMPORTAMIENTO O VOCABULARIO UTILIZANDO ALGORITMOS DE INTELIGENCIA ARTIFICIAL APLICADOS AL PROCESAMIENTO DE LENGUAJE NATURAL.

- **ESTADO DEL ARTE**
- **MARCO TEÓRICO**
- **MARCO METODOLÓGICO**
- **EXPERIMENTOS Y RESULTADOS**

CIUDAD DE MÉXICO, A 21 DE JUNIO DE 2022.

A S E S O R E S

M. EN C. LILIANA CHANONA HERNÁNDEZ

M. EN C. CÉSAR JESÚS NÚÑEZ PRADO

DR. JESÚS JAIME MORENO ESCOBAR

DEPARTAMENTO DE I.C.E

M. EN C. ITZALÁ RABADÁN MALDA
JEFA DE LA CARRERA DE INGENIERÍA
EN COMUNICACIONES Y ELECTRÓNICA

Autorización de uso de obra

Instituto Politécnico Nacional

P r e s e n t e

Bajo protesta de decir verdad el que suscribe **DAVID SOTO OSORIO**, manifiesta ser autor y titular de los derechos morales y patrimoniales de la obra titulada **"ANÁLISIS DE CANCIONES INFANTILES APLICANDO ALGORITMOS DE INTELIGENCIA ARTIFICIAL"**, en adelante "**La Tesis**" y de la cual se adjunta copia, ***en un impreso y un cd*** por lo que por medio del presente y con fundamento en el artículo 27 fracción II, inciso b) de la Ley Federal del Derecho de Autor, otorgo al **INSTITUTO POLITÉCNICO NACIONAL**, en adelante **EL IPN**, autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales o en cualquier otro medio; ***para apoyar futuros trabajos relacionados con el tema*** de **"La Tesis"** por un periodo de **2 años** contado a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a **EL IPN** de su terminación.

En virtud de lo anterior, **EL IPN** deberá reconocer en todo momento mi calidad de autor de **"La Tesis"**.

Adicionalmente, y en mi calidad de autor y titular de los derechos morales y patrimoniales de **"La Tesis"**, manifiesto que la misma es original y que la presente autorización no contraviene ninguna otorgada por el suscrito respecto de **"La Tesis"**, por lo que deslindo de toda responsabilidad a **EL IPN** en caso de que el contenido de **"La Tesis"** o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumo las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Ciudad de México., a 22 de Agosto de 2022.

Atentamente

DAVID SOTO OSORIO

A mis padres y hermanos por haberme forjado, muchos de mis logros se los debo a ustedes que me han brindado un apoyo incondicional y que me han enseñado a ser constante y a llevar una vida de valores, principio y honor, que han sido siempre la clave para alcanzar todas mis metas.
Gracias a todos.

Agradecimientos

Agradezco a mi padre y madre que siempre se han preocupado por mí y me han dado su apoyo incondicional en todos los proyectos e ideas que tengo.

A mis hermanos y demas familiares que siempre me han apoyado y motivado a seguir con mis estudios.

Agradezco a todos mis profesores que contribuyeron a mi desarrollo profesional especialmente a la M. en C. Liliana Chanona Hernández y al Dr. Jaime Moreno Escobar por ofrecerme su apoyo y asesoramiento para el desarrollo de esta tesis.

Agradezco al M. en C. César Jesús Núñez Prado por ofrecerme su asesoramiento técnico en el desarrollo de este proyecto.

Gracias por todo.

David Soto Osorio
Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Zacatenco
Instituto Politécnico Nacional, México, 2022

Contenido

Lista de Figuras **xi**

Lista de Tablas **xiii**

1 Introducción **1**
1.1 Planteamiento del problema 1
1.2 Justificación 3
1.3 Hipótesis 4
1.4 Objetivos 5
1.4.1 General 5
1.4.2 Particulares 5
1.5 Alcance del proyecto 5
1.6 Tesis del Proyecto 6

2 Estado del Arte **8**
2.1 Contexto histórico 8
2.1.1 Canciones Infantiles 8
2.1.2 Lenguajes de programación 9
2.1.3 Inteligencia Artificial 11
2.2 Trabajos relacionados 12
2.2.1 Procesamiento de lenguaje natural en letras 13
2.2.2 Clasificación del estado de ánimo de las canciones hindi basadas en letra 16
2.2.3 Identificación de la polaridad emocional de las letras de las canciones a través del procesamiento de lenguaje 18
2.2.4 Tendencias suicidas: Clasificación automáticas de letristas suicidas y letristas no suicidas usando PLN. 21
2.2.5 Clasificación de la polaridad del sentimiento de las canciones tailandesas basado en la letra. 24
2.3 Discusión 26

3 Marco Teórico **28**
3.1 Procesamiento de lenguaje natural 28
3.2 Python 28
3.3 Corpus lingüístico 29
3.4 Paquetes para PLN y manipulación de datos 30
3.4.1 NLTK 30
3.4.2 Pandas 31
3.4.3 Numpy 31
3.4.4 Scikit-learn 32
3.5 Técnicas de preprocesamiento de PLN. 33
3.5.1 Tokenización 33
3.5.2 Lematización 34
3.5.3 Eliminación de palabras vacías o stop-words 34
3.6 Programas para desarrollo de código en Python 35
3.6.1 Anaconda 36
3.6.2 Jupyter Notebook 37
3.7 Tipos de aprendizaje 38
3.7.1 Aprendizaje supervisado 38
3.7.2 Aprendizaje no supervisado 39
3.8 Metodos de validación 40
3.8.1 Validación simple 41
3.8.2 Validación cruzada 42
3.9 Algoritmo de clasificación 43
3.9.1 Algoritmo K vecinos más cercanos 43

4 Marco Metodológico **45**
4.1 Etapa 1: Búsqueda y acomodo del conjunto de datos 45
4.1.1 Fase 1: Metodología de búsqueda 45
4.1.2 Fase 2: Acomodo de la información y creación del corpus lingüístico 46
4.1.3 Fase 3: Instalación de programas. 46
4.2 Etapa 2: Etiquetado del corpus lingüístico 48
4.3 Etapa 3: Tokenización, eliminación de stopwords y lematización . . 49
4.3.1 Fase 1: Llamado a las librerías 49
4.3.2 Fase 2: Búsqueda de directorio que almacena los archivos . . 50
4.3.3 Fase 3: Inserción de letras en tabla. 50
4.3.4 Fase 4: Eliminación de símbolos no necesarios para el análisis del texto y conversión de mayúsculas a minúsculas. 51
4.3.5 Fase 5: Almacenamiento de stopwords de la librería Spacy y NLTK 52
4.3.6 Fase 6: Tokenización y eliminación de stopwords. 52

4.3.7 Fase 7: Lematización. 53
4.4 Etapa 4: Vectorización y separación del corpus. 54
4.4.1 Fase 1: Vectorización de las letras. 54
4.4.2 Fase 2: Separación en conjunto de entrenamiento y prueba. . 54
4.5 Etapa 5: Algoritmo de clasificación 54

5 Resultados Experimentales **56**
5.1 Condiciones iniciales 56
5.2 Prueba de funcionamiento 57
5.3 Experimento 59
5.4 Evaluación económica. 63

6 Conclusiones **67**
6.1 Conclusiones Finales 67
6.2 Trabajo Futuro 68

A Reporte de Similitud **69**

B **70**

C **71**

Referencias **72**

Lista de Figuras

1.1 Niños cantando en grupo 2
1.2 Lenguajes de programación más usados según el indice TIOBE . . . 3
1.3 Logos de Anaconda y Python 4

2.1 Primeros álbumes musicales de canciones infantiles 9
2.2 Lenguajes de programación 11
2.3 Sistemas Deep Blue y Watson 12
2.4 Metodología de análisis de letras de canciones 15
2.5 Metodología de análisis de letras de canciones hindi 18
2.6 Metodología de análisis de letras de canciones por diccionarios . . . 21
2.7 Metodología de análisis de letras de canciones de artistas suicidas . 23
2.8 Metodología de análisis de letras de canciones tailandesas 26

3.1 Logo oficial de Python 29
3.2 Comparación de código *"Hola mundo"* en JAVA y Python 29
3.3 Logo oficial de NLTK 31
3.4 Logo oficial de Pandas 32
3.5 Logo oficial de Numpy 32
3.6 Logo oficial de Scikit-learn 33
3.7 Tokenización por palabra 33
3.8 Lematización de palabras 34
3.9 Eliminación de stopwords 35
3.10 Logo oficial de Anaconda 36
3.11 Interfaz de desarrollo de Anaconda 37
3.12 Logo oficial de Jupyter 38
3.13 Interface de Jupyter Notebook 38
3.14 Diagrama de funcionamiento del aprendizaje supervisado 40
3.15 Diagrama de funcionamiento del aprendizaje no supervisado 40
3.16 Diagrama de validación simple 42
3.17 Formula distancia euclidiana 43
3.18 Formula distancia euclidiana entre 2 puntos 44

3.19 K vecinos mas cercanos. 44

4.1 Ventana de instalación de Python 47
4.2 Tabla general de clasificación 49
4.3 Llamado a las librerías 50
4.4 Lista de archivos .txt 51
4.5 Inserción de letras en tabla 51
4.6 Eliminación de caracteres no necesarios 52
4.7 Almacenamiento de stopwords 52
4.8 Tokenización y eliminación de stop-words 53
4.9 Lematización de letras 53
4.10 Algoritmo de clasificación K vecinos más cercanos. 55

5.1 Tokenización y eliminación de stopwords en letras 57
5.2 Resultados de la lematización 58
5.3 Separación en conjunto de entrenamiento y prueba 58
5.4 Canciones clasificadas por el algoritmo K vecinos más cercanos. . . 59
5.5 Gráfica de factores que afectan al porcentaje de precisión. 62
5.6 Tabla de análisis de varianza para porcentaje de precisión 63
5.7 Diagrama de flujo de ingresos y egresos mensuales durante un periodo de 1 año 65
5.8 Fórmula para calcular el Valor Promedio Neto (VPN) 65

A.1 Reporte de similitud de Turnitin 69

Lista de Tablas

2.1 Tabla de comparación de trabajos relacionados 26

5.1 Tabla de variables externas que pueden afectar el porcentaje de precisión del algoritmo 59
5.2 Pruebas realizadas para el algoritmo de canciones infantiles 61
5.3 Gastos en servicios mensuales 64

Abstract

Children's songs are part of the childhood of every human being, these songs have great benefits, among them that allows them to develop their vocabulary, helps their intellectual development and also teaches them values and basic principles, however, there are children's songs that have a negative influence on children, as it teaches them to have an aggressive vocabulary and bad behavior towards others.

In response to this problem, a Natural Language Processing (NLP) algorithm was designed to classify nursery rhyme lyrics into positive or negative classes. For the classification of this algorithm, the k-nearest neighbor algorithm was used together with corpus preprocessing techniques such as tokenization, lemmatization, stopword elimination and word vectorization.

A performance test of the k nearest neighbors algorithm was performed considering the 7 nearest neighbors and considering a linguistic corpus of 220 letters, this corpus was divided into a training set of 70 % and a test set of 30 %, as a result of this test an accuracy percentage of 81.4 % was obtained.

An experiment was designed that considers 3 factors that can affect the percentage of accuracy that the k nearest neighbor classification algorithm has, these are i) misspellings, ii) English words and iii) capitalized letters. For these experiments we used ANOVA analysis that allows us to determine which factors affect the system, it was determined that all factors influence the percentage of accuracy, however, the English words affect the system more, since all the letters that are handled for the preprocessing of the corpus and for the training of the algorithm are in Spanish.

Resumen

Las canciones infantiles forman parte la infancia de todo ser humano, estas canciones cuentan con grandes beneficios, entre ellas que permite desarrollar su vocabulario, ayuda a su desarrollo intelectual y también les enseña valores y principios básicos,sin embargo, existen canciones infantiles que influyen de forma negativa en los niños, ya que los enseña a tener un vocabulario agresivo y un mal comportamiento hacia los demás.

Como respuesta a esta problemática en el presente trabajo se diseñó un algoritmo de Procesamiento de Lenguaje Natura (PLN) que realiza una clasificación de letras de canciones infantiles en clases positiva o negativa. Para la clasificación de este algoritmo se usó el algoritmo de k vecinos más cercanos junto con técnicas de preprocesamiento del corpus como son la tokenización, lematización, eliminación de stopwords y vectorización de palabras.

Se efectuó una prueba de funcionamiento del algoritmo de k vecinos más cercanos considerando los 7 vecinos más cercanos y considerando un corpus lingüístico de 220 letras, este corpus fue dividido en un conjunto de entrenamiento del 70% y un conjunto de prueba del 30%, como resultado de esta prueba se obtuvo un porcentaje de precisión del 81.4%.

Se diseñó un experimento que considera 3 factores que pueden afectar al porcentaje de precisión que tiene el algoritmo de clasificación k vecinos más cercanos, estos son i) errores ortográficos, ii) palabras en inglés y iii) letras en mayúsculas. Para estos experimentos se utilizó el análisis ANOVA que nos permite determinar cuáles factores afectan al sistema, se determinó que todos los factores influyen el porcentaje de precisión, sin embargo, las palabras en inglés afectan más al sistema, ya que todas las letras que se manejan para el preprocesamiento del corpus y para el entrenamiento del algoritmo están en español.

Glosario

PLN Procesamiento de Lenguaje Natural

VPN Valor Promedio Neto

TIR Tasa Interna de Retorno

Capítulo 1

Introducción

1.1 Planteamiento del problema

La música es un aspecto muy importante en el desarrollo óptimo de un ser humano, esta tiene grandes beneficios tanto para los que se encuentran en la etapa de desarrollo fetal como en todos los humanos que han nacido y se han desarrollado en un entorno social, la música tiene muchas aportaciones en el desarrollo intelectual, social y de aprendizaje.

Entre los beneficios más importantes cabe denotar que ayuda al desarrollo y estimulo de la inteligencia de todos los niños, por lo que se considera importante que los pequeños escuchen música en casa y en la escuela, interviene en el desarrollo motriz, en el desarrollo de la imaginación y creatividad, también es de mucha ayuda en el desarrollo de la memoria, la concentración y como ultimo beneficio también ayuda al desarrollo de la autoestima y en algunos casos a combatir el estrés.

Diversos investigadores afirman que la música es muy benéfica para el desarrollo del lenguaje, entre ellos Magán Hervás y Gertrudis Barrio que en su artículo "Influencia de las actividades audio-musicales en la adquisición de la lectoescritura en niños y niñas de cinco años" [1] del año 2017 realizan una investigación en España en donde llegan a la conclusión de que la música llega a ser una fuerte influencia en el desarrollo y el aprendizaje óptimo de los niños.

Todos estos beneficios explican el por qué a los niños les resultan muy agradables las canciones infantiles, la gran parte de estas son muy pegadizas y divertidas por lo que les resulta muy fácil de aprender y memorizar. Pero a partir de que son muy divertidas y fáciles de aprender los padres deben ser conscientes y hacerse la

pregunta de que si todas las canciones infantiles tienen letras que pueden transmitir un mensaje positivo para sus niños.[2]

Toda influencia positiva dentro de una canción infantil depende de la letra y del mensaje que la canción transmita a partir de esto nos hemos dado cuenta que algunas canciones tenían una letra poca apropiada con un mensaje negativo, ya que existen canciones infantiles que son racistas, violentas, tristes y machistas que incitan a los niños a tener un cambio de comportamiento hacia todos los demás.[3]

En la actualidad debido al desarrollo de las tecnologías como el internet, los niños son más vulnerables a escuchar este tipo de canciones que incluyen mensajes negativos, esto es un gran problema, ya que dependiendo del entorno en el que se desarrollen, este puede llegar a ser un factor muy importante en el desarrollo de su personalidad.

Las canciones infantiles son una herramienta muy importante no solo en el hogar sino también en los programas educativos, ya que son muy usadas en las etapas escolares iniciales, como se mencionó anteriormente estas canciones ayudan a mejorar el lenguaje y la comunicación, sin embargo, en México no se utilizan mucho estas técnicas de aprendizaje e inclusive las actividades que hacen con música utilizan canciones misóginas y sexistas como el reguetón por lo que es importante trabajar en este campo de formación porque según el informe de resultados de la prueba Excale se determinó que más del 50% de los niños se encuentra en el nivel básico o por debajo del básico en la parte de comunicación y lenguaje.

Figura 1.1: Niños cantando en grupo

1.2 Justificación

Con el surgimiento de la internet los niños tienen un mayor acceso a diferentes plataformas de video y reproducción de música en las cuales pueden ver videos y escuchar canciones que pueden traer un mensaje negativo, con el uso de las tecnologías de Procesamiento de Lenguaje Natural(PLN) y machine learning se desarrolla un algoritmo que realiza una clasificación de que canciones no son aptas, con esto se puede evitar la mala influencia de las canciones que pueden afectar a su desarrollo y comportamiento.

En la actualidad existen una gran cantidad de lenguajes de programación, entre los más usados actualmente según el índice TIOBE (the software quality company)[4] y un reciente ranking elaborado por IEEE Spectrum son C , Python , Java , C++ y C.

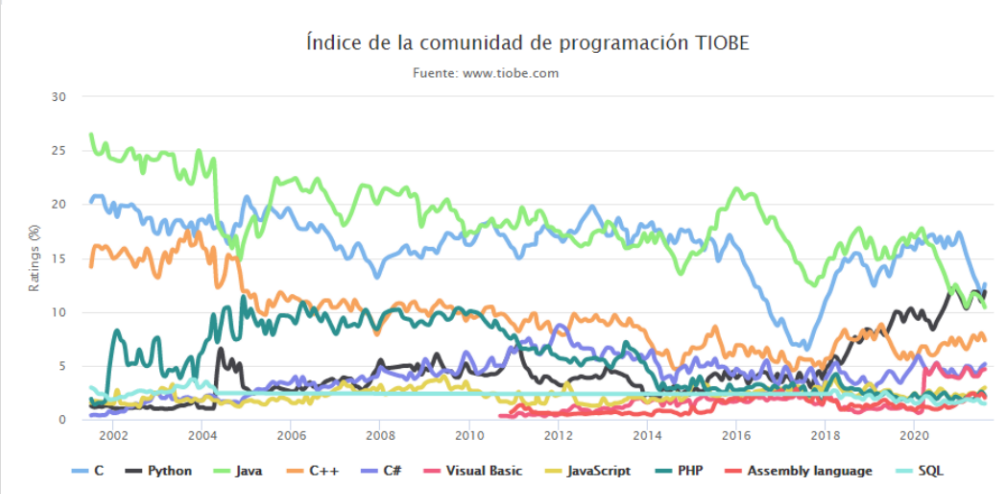


Figura 1.2: Lenguajes de programación más utilizados según el indice TIOBE

El lenguaje de programación Python a vuelto al segundo lugar de los lenguajes de programación más utilizados este 2021 esto se debe a que es un lenguaje interpretado, es decir, traduce y ejecuta el código a la vez, también es multiparadigma porque admite el uso de varios paradigmas de programación y multiplataforma porque puede ejecutarse en diferentes sistemas operativos. Es uno de los lenguajes más utilizados en el desarrollo de Inteligencia Artificial.

Python es uno de los lenguajes más populares para trabajar en el campo de la inteligencia artificial ya que cuenta con una gran comunidad activa y cuenta con

una diversidad de librerías para aplicaciones en diferentes ámbitos, una de las librerías más populares es NLTK , esta es una librería líder para el PLN ya que proporciona interfaces fáciles de usar, más de 50 corpus y cuenta con un conjunto de bibliotecas de procesamiento de texto para la clasificación, tokenización, etiquetado, análisis, y razonamiento semántico.

Existen otros trabajos que usan Python, la librería NLTK y otros paquetes para la clasificación y análisis de textos, entre estos tenemos el análisis de sentimientos dentro de las redes sociales, análisis de textos para detectar plagio, filtros, asistentes inteligentes, etc.

Para el desarrollo de este algoritmo se utiliza el lenguaje de programación python versión 2.0 o 3.0 junto al entorno de desarrollo Anaconda 3.0 y Jupyter Notebook, para el análisis y procesamiento de los textos se utiliza la librería NLTK y para la clasificación se usan algoritmos de machine learning de la libreria Scikitlearn.

Figura 1.3: Logos de la plataforma Anaconda y lenguaje de programación Python

1.3 Hipótesis

H0 : Si se desarrolla un algoritmo con base en el Procesamiento de lenguaje natural, el algoritmo es capaz de identificar las canciones aptas para que un niño las escuche con una precisión mayor o igual al 70%.

H1 : Si se desarrolla un algoritmo con base en el Procesamiento de lenguaje natural, el algoritmo es capaz de identificar las canciones aptas para que un niño las escuche con una precisión menor al 70%.

1.4 Objetivos

1.4.1 General

Analizar letras de canciones dirigidas a los niños con el propósito de identificar que canciones pueden dar un mensaje negativo y pueden influir de manera negativa en su comportamiento o vocabulario utilizando algoritmos de inteligencia artificial aplicados al procesamiento de lenguaje natural.

1.4.2 Particulares

- Analizar el entorno de soluciones existentes para la clasificación de letras musicales mediante una comparación de diferentes trabajos relacionados con el propósito de identificar las diferencias y semejanzas entre cada trabajo y mi propuesta.

- Integrar los conocimientos del Procesamiento de Lenguaje Natural e Inteligencia Artificial por medio de consultas en diferentes páginas web oficiales y libros con el propósito de tener los conocimientos necesarios para desarrollar un algoritmo de clasificación bi-clase de letras de canciones infantiles.

- Presentar por medio de etapas y fases los pasos para desarrollar un programa de clasificación de letras de canciones aplicando el algoritmo de k vecinos más cercanos.

- Desarrollar y analizar los resultados experimentales por medio de un análisis ANOVA para verificar cuáles son las variables externas que afectan al porcentaje de precisión.

1.5 Alcance del proyecto

Para detectar si una canción transmite un mensaje positivo o negativo se aplica procesamiento de textos, en este caso se analizan las letras de las canciones infantiles. Este proyecto se centra en el análisis y procesamiento de letras de canciones infantiles para posteriormente enviar esos datos a un algoritmo K vecinos más cercanos para su clasificación, esto con el propósito de que pueda determinar si una canción trae un mensaje negativo o positivo.

Es por ello que el lenguaje de programación que se usa en el desarrollo de este trabajo es Python ya que ofrece una gran cantidad de módulos y librerías para el Procesamiento de Lenguaje Natural(PLN).Es un lenguaje de programación que

cuenta con una mejor organización de código por lo que se pueden hacer las mismas funciones y programas con una menor cantidad de líneas de código.

El lenguaje de programación Python cuenta con varias librerias para el desarrollo de algoritmos de machine learning, es por ello que este lenguaje de programación es el utilizado en este proyecto, ya que se usa el algoritmo de k vecinos más cercanos para determinar si una canción tiene un mensaje positivo o negativo.

Este proyecto se enfoca principalmente en las canciones infantiles por lo que se hace una investigación de los artistas más populares de canciones infantiles en México, así como de las películas y programas de televisión más populares en la actualidad, esto con el propósito de agregar las letras de sus canciones en el corpus lingüístico.

Para este proyecto se desarrolla una carpeta en donde se tienen varios archivos de texto, en cada uno de estos archivos de texto se tienen la letra de una canción infantil, a partir de estos archivos se realiza todo el preprocesamiento de los textos, para realizar este proceso se usa la libreria de Python NLTK que cuenta con diferentes herramientas para realizar la tokenización, lematización y eliminación de palabras vacías, realizando el preprocesamiento de los textos podemos tener una mayor eficiencia del algoritmo de clasificación y también se reduce el tiempo de procesamiento y los recursos computacionales.

Este algoritmo puede ser empleado en aplicaciones de reproducción de canciones infantiles o inclusive en plataformas de video. Y por otro lado el corpus obtenido puede emplearse en otro tipo de proyectos relacionados al análisis de textos así como a la detección de otros parámetros.

1.6 Tesis del Proyecto

Este trabajo se basa en la inteligencia artificial, la cual es un conjunto de algoritmos que tienen como objetivo que las máquinas tengan las mismas capacidades que el ser humano, es decir, que aprendan de las experiencias [5], para el desarrollo de la inteligencia artificial se usan los lenguajes de programación, estos son un conjunto de reglas lógicas y gramaticales bien definidas que permiten a un ser humano tener la capacidad de escribir un conjunto de instrucciones en formas de algoritmos con el objetivo de controlar un sistema informático [6]. Para el desarrollo de este proyecto se usan las canciones infantiles, estas son un conjunto de

sonidos rítmicos y letras enfocadas principalmente a los niños con propósitos de aprendizaje y desarrollo.

Este trabajo esta integrado por V capítulos y tiene como objetivo principal analizar las letras de las canciones para determinar si transmiten un mensaje positivo o negativo. En este proyecto se presenta un algoritmo escrito en lenguaje de programación Python , utilizando las librerías NLTK, Spacy, Numpy y ScikitLearn para el análisis de textos y también se presenta el algoritmo de k vecinos más cercanos que tendrá la tarea de clasificar las letras de las canciones en dos clases, positivas y negativas.

Es así como en este trabajo se presenta en el capítulo 2 el contexto histórico de las canciones infantiles, la inteligencia artificial y los lenguajes de programación también se describen los trabajos relacionados con nuestro proyecto y las similitudes y diferencias que estos tienen.

Capítulo 2

Estado del Arte

2.1 Contexto histórico

2.1.1 Canciones Infantiles

La música ha estado presente desde los inicios de la humanidad, esta se presentó por el origen del lenguaje y la necesidad que se tenía de la comunicación, la música se remonta a hace más de 40,000 años, desde que el Homo Sapiens era capaz de imitar sonidos de la naturaleza y lograba diferenciarlos de los que eran la estructura de su lenguaje.

La historia de la música abarca a todas las culturas y épocas, la música ha ido cambiando conforme han pasado las diferentes épocas como la edad media y han presentado un gran impacto en la actualidad, centrándonos en las canciones infantiles, esta categoría tiene sus orígenes en el siglo XX, específicamente en los tiempos de la guerra que comprende desde los años 30s hasta los 50s.

Este tipo de música al inicio no era considerado un genero de música comercial en varios países, hasta que en 1954 en México se grabo el primer álbum titulado *Canciones Infantiles* que fue considerado el primer disco conocido en diversos países de Latinoamérica demostrando que las canciones infantiles era un genero comercial como muchos otros.

Debido a que las canciones infantiles tuvieron una buena aceptación de los grupos infantiles se busco aplicar este genero de música a temas educativos, en 1979 un profesor de España se percato de que sus estudiantes estaban reprobando los exámenes y ejercicios de matemáticas, el tema en el que reprobaban era en las multiplicaciones por lo que contacto a la disquera *Hispavox* sugiriendo una grabación que fuera didáctica para los niños, específicamente donde los niños pudieran apren-

der las tablas de multiplicar, el resultado de esta solicitud fue el álbum titulado *Multiplica con Enrique y Ana.*[7]

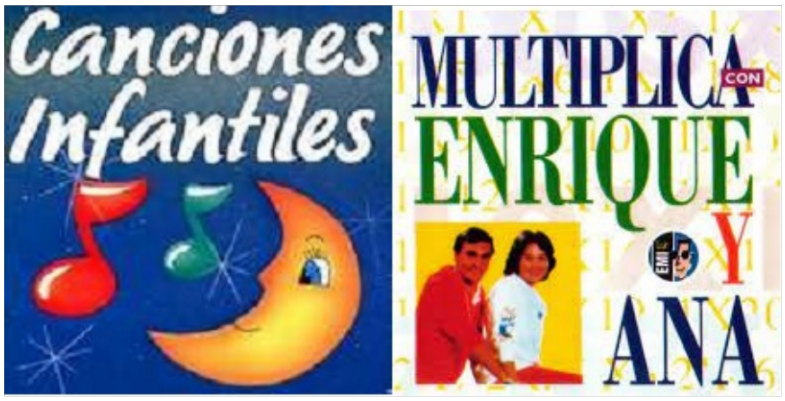


Figura 2.1: Primeros álbumes musicales de canciones infantiles

2.1.2 Lenguajes de programación

En el año de 1957, Joth W. Backus invento el primer lenguaje de programación, que fue utilizado por programadores reales, antes ya se habían hecho pruebas para la creación de un lenguaje de programación con diferentes herramientas como tarjetas perforadas, pero Fortran es considerado el primer lenguaje de programación, este lenguaje se diseño bajo la necesidad de calcular la trayectoria de misiles ya que se necesitaba de una mayor velocidad de cálculo.

En 1959, se inventa el lenguaje de programación COBOL, el objetivo de este lenguaje de programación era que pudiera ser usado por cualquier ordenador esto se debe a que en los años 60 había diferentes ordenadores que no eran compatibles entre sí.

En 1964, se crea el lenguaje de programación BASIC este se creó con propósitos de aprendizaje sin embargo varios sistemas se han escrito en este lenguaje a tal grado que en la actualidad sigue siendo uno de los lenguajes más utilizados para la creación de diferentes algoritmos y es considerado uno de los puntos más importantes dentro de la historia de los lenguajes de programación.

En 1970, Niklaus With creo Pascal que al igual que BASIC su objetivo principal era que fuera una herramienta de enseñanza de programación, este lenguaje de programación se usó para el desarrollo de muchas aplicaciones, sin embargo,

a diferencia de BASIC actualmente ya no es tan ocupado para el desarrollo de aplicaciones, pero aún se sigue viendo en algunas escuelas de programación.

En 1972 y 1979 fueron creados los lenguajes de programación C y C++, primero fue creado C por Dennis Ritchie, este lenguaje se usa para ser un intermediario entre lenguajes, al inicio este lenguaje era de bajo nivel, pero en la actualidad ha sido utilizado en aplicaciones de todo tipo y ha sido un lenguaje base para otros lenguajes más modernos.

Posteriormente a C se creo C++ este es uno de los lenguajes de programación más utilizados la intención de este lenguaje de programación fue la de ampliar las funciones y aplicaciones del lenguaje de programación C, una de las nuevas funciones de este lenguaje es la programación orientada a objetos, actualmente este estilo de programación es el que domina el mercado ya que se pueden ampliar las aplicaciones del lenguaje de programación haciendo más funciones con menos líneas, esto se debe a que el código se reutiliza.

En 1991, se crearon los primeros lenguajes de internet como HTML, pero también se desarrollo el lenguaje de programación Python, este lenguaje fue diseñado por Guido Van Rossum, actualmente es uno de los lenguajes de programación más utilizados en diferentes áreas como la inteligencia artificial, se debe a la simplicidad de su sintaxis.

En 1995, se crearon los lenguajes de programación Java, PHP y JavaScript, estos lenguajes hasta la fecha son muy utilizados en una gran cantidad de páginas web, se estima que más del 50% de las páginas en la red son diseñadas con estos lenguajes de programación.

En el 2001, se crea el lenguaje de programación C por parte de Microsoft, este es un lenguaje multiplataforma, es utilizado en diferentes aplicaciones debido a la facilidad de su codificación y gracias a las herramientas que se diseñaron para trabajar con este código como Visual Studio .NET, actualmente es uno de los lenguajes de programación más utilizados.

En los años posteriores se han desarrollado otros lenguajes de programación que aún no son muy utilizados en el desarrollo de aplicaciones ya que están diseñados para realizar tareas muy específicas, algunos de estos lenguajes son Go de Google, Scratch, Swift y Kotlin.[8]

Figura 2.2: Lenguajes de programación

2.1.3 Inteligencia Artificial

El concepto de inteligencia artificial se remonta a la época de los griegos cuando Aristóteles describió un conjunto de reglas que describen el funcionamiento que tiene la mente para poder dar conclusiones racionales. Sin embargo, se le considera el padre de la inteligencia artificial a Alan Turing ya que en el año de 1936 diseño una máquina que era capaz de hacer cualquier calculo que fuera definido.

En el año de 1956, John McCarthy, Marvin Minsky y Claude Shannon usaron el termino inteligencia artificial para definirla como la ciencia que hace que las maquinas sean inteligentes el problema fue que no se sabia con exactitud cuando llegarían las primeras inteligencias artificiales, ellos tenían previsto que en la decada de 1970 o 1980 estarían rodeados de inteligencias artificiales, pero esto ocurrió hasta la decada de 1990 cuando una gran cantidad de las empresas tecnológicas empezaron a invertir en este campo con el propósito de mejorar la capacidad de procesamiento y análisis de datos que cada vez iba creciendo en el mundo digital.

En 1997, empezó oficialmente la aplicación de la inteligencia artificial, en este año IBM demostró que un sistema informático podía tener la capacidad de vencer al mejor jugador de ajedrez del mundo, este sistema se llamaba Deep Blue, el objetivo de esta prueba fue que la industria tecnología y la sociedad en general cobraran conciencia de las posibilidades que tenían las IA.

Al igual que Deep Blue, IBM diseño a Watson el cual era un sistema informático que participo en el programa televisivo de Jeopardy en donde se realizaban preguntas de cultura general y conocimiento de todo tipo, al final el sistema venció

a los dos campeones de este programa, este sistema pudo ganar ya que tenia la capacidad de comprender la pregunta para posteriormente buscar la respuesta en los 200 millones de páginas almacenadas en su sistema.

Pero no todas las contribuciones a la inteligencia artificial han sido de IBM, también grandes empresas como Facebook y Google están haciendo numerosos progresos dentro de este campo, han desarrollado aplicaciones que usan inteligencia artificial para dar un mejor servicio a los usuarios de sus plataformas, Google ha utilizado inteligencia artificial para sus motores de búsqueda ya que con base en una búsqueda en la web, el mismo buscador te da opciones, han aplicado inteligencia artificial para realizar traductores y también para sus asistentes virtuales.

Actualmente una de las aplicaciones más prometedoras son los chatbots, estos son algoritmos que tienen la capacidad de comprender el lenguaje humano, tienen la capacidad de aprender, procesar e improvisar respuestas de forma autónoma.[9]

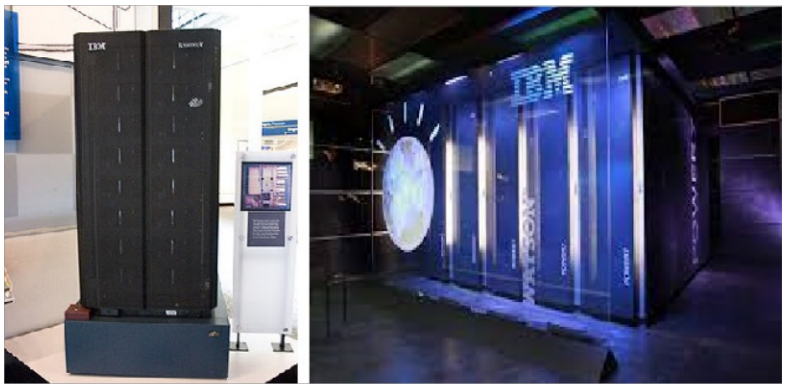


Figura 2.3: Sistemas Deep Blue y Watson

2.2 Trabajos relacionados

Para la búsqueda de trabajos relacionados se ingresó a las plataformas Google Academic y IEEE explore, primero se consultó la plataforma IEEE y se buscaron resultados con las palabras Natural Language Processing, en esta primera búsqueda se obtuvieron 24,177 trabajos, para reducir las opciones se usaron las palabras Songs Natural Language Processing, con estas se obtuvieron 423 trabajos, para reducir aun más las opciones se aplico un filtro para obtener trabajos con máximo 10 años de antigüedad, con este filtro se obtuvieron 332 trabajos, para reducir aun

más las opciones se seleccionaron solo aquellos trabajos que en su título hablaran del análisis de sentimientos dentro de las canciones de cualquier género.

Para la búsqueda en la plataforma Google Academic primero se ingresaron las palabras Natural Language Processing se obtuvieron aproximadamente 4,240,000 resultados, para reducir estas opciones se aplico un filtro para que solo se muestren trabajos con máximo 10 años de antigüedad, con esto se obtuvieron un total de 1,320,000 trabajos, para reducir más las opciones se cambiaron las palabras de búsqueda a Songs Natural Language Processing, se obtuvieron 24,300 trabajos relacionados, de estos solo se consideraron aquellos que en su título tengan análisis de sentimientos en letras de canciones de cualquier género.

Al terminar con la búsqueda se obtuvieron 5 trabajos que se enfocan en el análisis de sentimientos de canciones de diversos géneros, estos trabajos son los siguientes:

1.- *"Procesamiento de lenguaje natural en letras"* por José PG Mahedero, Álvaro Ivaro Martínez, Pedro Cano.[10]

2.- *"Clasificación del estado de animo de las canciones hindi basadas en letras"* por Braja Gopal Patra, Dipankar Das y Sivaji Bandyopadhyay.[11]

3.- *"Identificación de la polaridad emocional de las letras de las canciones a través del procesamiento del lenguaje natural"* por Ahsley M. Oudenne y Sarah E. Chasins.[12]

4.- *"Clasificación automática de letristas suicidas y letristas no suicidas usando PLN"* por Mattew Mulholland y Joanne Quinn.[13]

5.- *"Clasificación de la polaridad del sentimiento de las canciones tailandesas basadas en la letra"* por Chutimet Srinilta , Wisuwat Sunhem, Suchat Tungjunob, Saruta Thasanthiah y Supawit Vatahanavaro.[14]

Estos artículos están relacionados ya que todos aplican el Procesamiento de Lenguaje Natural para el análisis de canciones de diferentes géneros, se aplica el PLN para determinar la precisión de un algoritmo con respecto a otros y para demostrar la eficacia de la PLN para el análisis de las letras.

2.2.1 Procesamiento de lenguaje natural en letras

José PG Mahedero, Álvaro Ivaro Martínez, Pedro Cano

En este trabajo se presentan varios experimentos que usan las herramientas de PLN para el análisis de letras de canciones de diferentes géneros, este trabajo se realizo ya que las letras son una de las partes más importantes dentro de una canción y también es una de las principales partes de análisis ya que estas pueden generar una gran cantidad de datos que pueden ser utilizados para diferentes aplicaciones dentro de la industria musical. Se explica que actualmente el acceder a los contenidos musicales es una tarea de todos los días por lo que las industrias se ven en la necesidad de ofrecer mejores soluciones, estos algoritmos son diseñados para funcionar como filtros para que un reproductor de música pueda buscar canciones que tengan un mismo ritmo o letras parecidas sin necesidad de que el usuario las busque.Actualmente las letras de las canciones de cualquier género son muy fáciles de conseguir, se hizo una investigación y se determina que el 28,9% de las personas buscan las canciones usando algún fragmento de la letra de la canción.

Los experimentos que se desarrollaron en este trabajo son 4, en todos se utilizan herramientas de procesamiento de textos ya que estas son muy poderosas para el análisis de letras, estos experimentos están enfocados a la identificación de idiomas, categorización de textos, detección de estructuras dentro de una canción y a la detección del género de música a partir del texto, para el desarrollo de los proyectos se usaron letras que se obtuvieron de internet.

Para el desarrollo del algoritmo se utilizó el módulo Perl Lingua, para el entrenamiento de los algoritmos se utiliza la técnica de bi/tri gramas que son un conjunto de 2 o 3 palabras que son calculados a partir del texto que se ingresó para el entrenamiento. Para la identificación de idiomas se seleccionaron 500 letras en diferentes idiomas, el resultado obtenido fue de un 92% de identificación del idioma de la letra. Aunque se tiene el problema de que estos porcentajes pueden ser alcanzados solamente en muestras que contienen una gran cantidad de texto, en caso de que las muestras sean más pequeñas como por ejemplo un título el porcentaje de precisión disminuirá.

Para el experimento dos de extracción de estructuras se debió considerar que toda canción y letra tiene estructuras como el verso, coro y puente, para poder diferenciar la estructura se realizaron varios pasos, primero fue la extracción del descriptor, en este se dividió la letra en párrafos posteriormente se utilizó para calcular la similitud entre párrafos ya que dependiendo del tipo de estructura que sea es la cantidad de veces que se repite, el segundo paso se encarga de identificar las partes más repetidas de una canción como el coro principal y como tercer paso se hizo un etiquetado final que consiste en ver la probabilidad de que una canción puede empezar con un coro o un verso. Se hizo la prueba del algoritmo de segmentación con

una muestra de 30 letras en diferentes idiomas y este lanzo un 76.66% de precisión.

Para el experimento de categorización temática se diseñó un clasificador que era capaz de clasificar 5 categorías distintas, estas son Amor, Protesta, Cristiano, Violento y Drogas. Este algoritmo es de mucha importancia ya que dependiendo del contexto de la letra de una canción se puede deducir el genero al que pertenece. El categorizador que se utilizo es un método conocido como Naive Bayes y se implemento usando el módulo Perl.AI, para realizar el entrenamiento del algoritmo se experimentó con 125 canciones de diferentes géneros que fueron divididas manualmente en las 5 categorías, el clasificador tuvo un resultado de precisión de un 82%.

Y por último el experimento número cuatro consistió en la búsqueda de similitud de letras, este algoritmo tiene una gran cantidad de aplicaciones dentro de la industria musical ya que con este algoritmo se pueden evitar los plagios, este algoritmo es más preciso que los algoritmos que detectan la similitud por medio del ritmo ya que este detecta la similitud de letras. Para la prueba de este algoritmo se hicieron varias pruebas con diferentes canciones que tenían letras parecidas pero con diferentes ritmos y en algunos casos con algunos cambios de palabras, una de estas fue "I want to break free" que fue comparada con más de 5000 canciones para determinar si contaba con una similitud con otras canciones, el resultado fue del 0.62% por lo que se puede concluir que el algoritmo no detecto ninguna similitud, este 0.62% se debe a que hay varias canciones que utilizan las palabras como free o break.

Se concluyo que aun se pueden realizar mejoras a estos algoritmos ya que estos pueden llegar a fallar con una mínima modificación, sin embargo, estos pueden ser buenas herramientas para aplicaciones dentro de la industria musical ya que puede complementar diferentes proyectos u otros algoritmos que están enfocados al análisis de los ritmos.

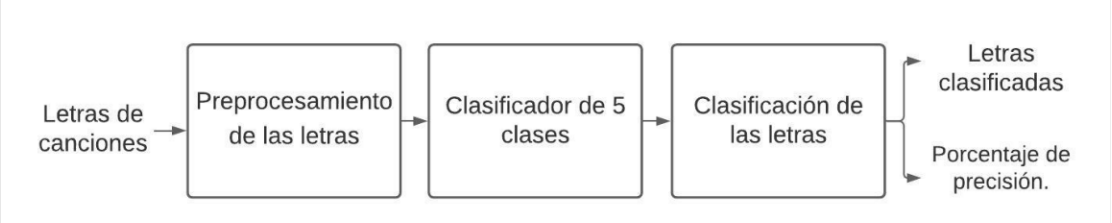


Figura 2.4: Metodología de análisis de letras de canciones

2.2.2 Clasificación del estado de ánimo de las canciones hindi basadas en letra

Braja Gopal Patra, Dipankar Das y Sivaji Bandyopadhyay.

Este trabajo habla de la importancia que ha tenido la digitalización de la música en la actualidad al igual que menciona la importancia de los algoritmos para la reproducción y la obtención de datos para mejorar las experiencias de los usuarios ya que debido al aumento de la presión laboral es complicado que una persona escuche y evalué música para crear una librería o lista de reproducción por lo que se han buscado formas de solucionar estos problemas por medio de motores de búsqueda o sistemas que recomienden canciones basados en el estado de ánimo del usuario. En este trabajo se propone una taxonomía de estados de ánimo para el análisis de las canciones hindi, este trabajo aplica una gran cantidad de herramientas semánticas y estilísticas para el procesamiento de la información, este trabajo se desarrolla debido a que varios estudios demuestras que los estados de ánimo son uno de los puntos más deseables para la creación de colecciones de música. Para el análisis de las letras hindi se considero que la música india se puede clasificar en diferentes géneros, lo clásico y lo popular, en este trabajo se usa el género clásico el cual tiene una subclasificación, esta puede ser indostani que es el ritmo que se escucha más en las partes nortes de la india y el ritmo carnático que es el que se escucha más en el sur de la india.

En este trabajo se considero este idioma porque no se han realizado muchos trabajos que se encarguen de su estudio y además que es una de los idiomas más hablados en todo el mundo. Para el desarrollo del corpus se consideraron 2 fases, en la primera fase se anota la perspectiva de la canción en función del oyente y en la segunda fase el corpus se anota en función del lector.

En el campo de la música india se ha informado de una cantidad de trabajos limitados en detección del estado de ánimo, la gran parte de trabajos desarrollados son para el análisis en función del audio, no de la letra, en los últimos años se han desarrollado muchos algoritmos de clasificación del estado de ánimo en función del ritmo, la intensidad, y el espectro. Para el desarrollo de este algoritmo se consideraron diferentes taxonomías, algunas de estas son el modelo circumplex de rusell y la taxonomía MIREX, se consideraron estas taxonomías ya que son las más utilizadas por varios investigadores en todo el mundo.

Para este trabajo se decidió utilizar el circumplex de rusell ya que si consideramos varias palabras podemos formar una clase con un estado de ánimo especifico, actualmente la taxonomía contiene cinco clases de estados de ánimo diferentes, estos

son contento, asombrado, tranquilo, triste, enfadado. Para desarrollar el algoritmo se consideraron varias fuentes de internet para poder recolectar las letras de las canciones.

Para verificar si el algoritmo estaba teniendo resultados óptimos se debieron anotar los sentimientos obtenidos de las letras al mismo tiempo que se anotaban los sentimientos obtenidos de los archivos de audio, esto es para saber si existen resultados diferentes dependiendo de si solo se lee la letra o se escucha la canción. Para esta parte del experimento fue necesario tener varios anotadores, la mayoría eran estudiantes de pregrado con una edad que varia entre los 18 y 24 años. Cada una de las canciones del corpus fue anotada por cinco anotadores. Se logro una concordancia del 88% para todos los datos obtenidos de las letras, pero se observó que había confusión entre algunas clases ya que los sentimientos de algunas clases eran parecidos o se podrían confundir dependiendo de si el anotador escuchaba la canción o solo leía la letra. Por lo que para evitarse esta clase de confusiones se redujo las clases que se iban a usar para el experimento a dos, solamente positivo y negativo, empleando solo estas dos clases se obtuvo una concordancia del 95%, aunque aún se presentaron algunas fallas ya que algunos anotadores colocaban una letra como positiva, pero en cuanto escuchaban la canción cambiaban de opinión a que era negativa.

Para clasificar las letras se emplearon una gran cantidad de características textuales como los léxicos de sentimiento, las características estilísticas y los n-gramas, estos últimos fueron utilizados para la clasificación del estado de ánimo de la música.

También para el análisis de las letras se consideraron varias características del texto, como las palabras únicas, el número de palabras repetidas, la cantidad de líneas únicas y el número de líneas terminadas con las mismas palabras. Por otra parte, se ha demostrado que los n-gramas funcionan muy bien para la clasificación de los estados de ánimo en las letras.

Para el desarrollo del sistema automático de clasificación se usó el algoritmo de aprendizaje LibSVM que está implementado en la herramienta WEKA, se elijo este debido a que funciona mejor que los otros clasificadores disponibles. Los resultados que se obtuvieron no fueron los deseados, en el caso de las letras hindi se logró un porcentaje del 38.49% en un conjunto de datos de 461 letras utilizando las primeras clases de estado de ánimo. Estos resultados son debido a que las canciones hindi manejan múltiples estados de ánimo y aparte de que las emociones de canciones hindi cambian al momento de leer la letra y escuchar la canción.

En este trabajo concluyeron que se logró un porcentaje de precisión del 38.49% debido a que la perspectiva del oyente y de la emoción del lector es completamente diferente en el caso del audio y las letras. Para trabajos futuros proponen realizar el mismo experimento, pero con un conjunto mayor de características textuales para mejorar el porcentaje, para lograr esto se planea desarrollar un sistema de clasificación híbrido que esté basado tanto audio como en letras.

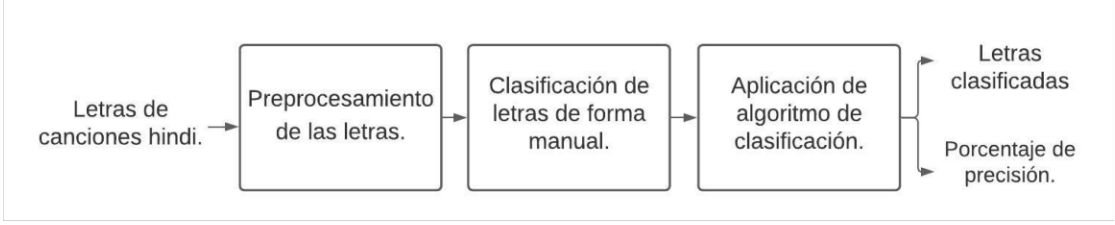


Figura 2.5: Metodología de análisis de letras de canciones hindi

2.2.3 Identificación de la polaridad emocional de las letras de las canciones a través del procesamiento de lenguaje

Ahsley M. Oudenne y Sarah E. Chasins

Este trabajo comienza explicando que las letras de las canciones pueden ser un gran medio para obtener información importante dentro de una canción, con las letras podemos deducir el genero de la canción, emociones, sentimientos y temas, sin embargo, el proceso para obtener esta información puede ser complicada, en este trabajo se demuestra por qué la clasificación basada únicamente en letras y en la polaridad emocional puede ser tan complicada. Se utiliza la clasificación más básica de todas, clasificar si una canción es positiva o negativa, para poder obtener esta información se aplican cuatro grupos de algoritmos de aprendizaje automático que están basados en la frecuencia de ciertas palabras, se utilizo un conjunto de datos de 420 canciones.

Para el análisis y la clasificación de la música, comúnmente la investigación se centra típicamente por el audio, sin embargo, esta puede tener algunas complicaciones debido a que el procesamiento de audio se basa en tener la grabación real de una canción. Las letras de las canciones son una buena fuente de información ya que la mayoría de las letras de las canciones están disponibles gratuitamente en internet. Para demostrar la complejidad del análisis y clasificación a través de letras se evaluará el rendimiento de los léxicos de subjetividad genéricos frente a

los específicos del corpus.

Primeramente, empiezan explicando el análisis de sentimiento, la mayor parte de investigadores hacen uso de la subjetividad para el análisis de sentimientos, uno de los más usados es el de JanyceWiebe que tiene indicadores para el sentimiento positivo y negativo. El análisis de textos no solamente es un tema exclusivo de la música, el análisis de textos se usa en una gran cantidad de plataformas digitales como Twitter para realizar estudios probabilísticos de diferentes temas, también la clasificación de sentimientos se ha usado exitosamente en la clasificación de reseñas de películas, estos procesos usan algoritmos de aprendizaje automático que clasifican las reseñas en positivas o negativas.

En el caso de las letras de canciones se percataron que se pueden tener mejores resultados con un léxico específico que con uno genérico, tuvieron que considerar estos puntos debido a que el análisis y la clasificación de texto de letras es un tema relativamente nuevo.

Como se mencionó anteriormente el análisis y clasificación de letras es más complicado, existen 3 dificultades con el análisis de sentimientos basado en letras, la primera es que las canciones pueden contener una serie de letras que transmitan un lenguaje negativo, pero al momento de terminar la canción, esta termina con una nota positiva o viceversa, la segunda es que una canción puede tener una gran cantidad de palabras negativas y al final el contexto de la canción es positiva y la tercera es que las canciones pueden expresar emociones positivas sobre cosas negativas y viceversa.

Para las pruebas experimentales se usó un corpus con 420 letras de canciones únicas, estas letras están dividas por igual entre las polaridades emocionales positivas y negativas, para la obtención de las letras se consideró la lista de las 100 mejores canciones entre 1980 y 2009. Para la clasificación de los datos se hizo uso del léxico de subjetividad de Jan Wiebe, primero se hizo un recuento de las pistas de subjetividad positivas y negativas y se etiqueto la canción con la mayor emoción. Para la ecualización de los datos se consideró que haya la misma cantidad de letras positivas y negativas para evitar que haya algunas fallas en los sistemas de clasificación por lo que el corpus que se maneja en las pruebas tiene 210 canciones positivas y 210 canciones negativas, para evitar problemas las letras fueron verificadas por un humano.

En este trabajo se aplican 3 algoritmos en el conjunto de datos, el más simple es el de lista de palabras, este algoritmo lo que hace es crear una lista con las pal-

abras que aparecen en cada clase, estas palabras están acomodadas por frecuencia, se hace una comparación con una lista seleccionada por los investigadores por lo que se descartan todas las palabras que no se encuentre en la lista propuesta, posteriormente el algoritmo elimina todas las palabras que aparecen en ambas listas y finalmente las palabras sobrantes son todos los datos necesarios para determinar la clasificación de las letras.

Para determinar la clasificación de la letra de una canción se hace una comparación entre la letra de la canción y las listas obtenidas, si aparece una palabra de la lista positiva, esta empieza un conteo que va incrementando cada vez que aparece una palabra, este mismo proceso se realiza con la lista de palabras negativas, si el conteo de palabras positivas es mayor el sistema puede determinar que la canción es positiva, si el conteo de palabras negativas es mayor, el algoritmo determina que la canción es negativa.

El segundo algoritmo propuesto necesita retener más información, necesita cada palabra que aparece en cada letra y las clases positivas y negativas, para este algoritmo se utiliza el diccionario Present In One, este cuenta con dos diccionarios, uno de palabras positivas y otro de palabras negativas, para la clasificación de una canción, este recorre todas las palabras, este verifica si cada palabra está en ambos diccionarios, en ninguno de ellos o en un solo diccionario, si es alguna de las dos primeras la palabra es ignorada, si es la tercera opción se incrementa el conteo, esta puede ser negativa o positiva, dependiendo de cuál sea la puntuación mayor será la clasificación.

El siguiente algoritmo propuesto también requiere del conocimiento de cada palabra en el conjunto de entrenamiento, pero también necesita tener un porcentaje del numero de veces que aparece una palabra en cada lista, este algoritmo necesita saber el número de veces que aparece una palabra en cada lista. Se usa el algoritmo Prevalent In One, para que funcione este algoritmo se tiene que mandar un valor que se obtiene del número de ocurrencias positivas entre el número de ocurrencias negativas, dependiendo del valor obtenido, este asigna el sentimiento con la puntuación más alta.

Los resultados obtenidos con el algoritmo de listas de palabras es una precisión promedio del 58% con un tamaño de lista máxima de 32 y la precisión máxima registrada fue del 68.6% lograda con una lista máxima de 36. Para el algoritmo de diccionarios de palabras se logró una precisión media del 65.5%, la precisión máxima es de 67%. Para el tercer algoritmo se logró una precisión promedio del 57.1% y el porcentaje más elevado fue del 65%.

Lograron concluir que las herramientas utilizadas actualmente para el procesamiento de otros elementos como mensajes de Twitter y música, aún es muy imprecisa para el análisis de letras, ya que el análisis de las letras es más complejo por el manejo de diferentes estructuras y por el contexto, como trabajos futuros proponen el desarrollo de algoritmos y diccionarios diseñados principalmente para el análisis de letras.

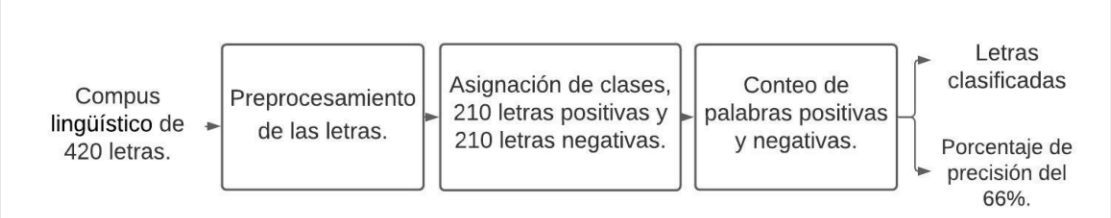


Figura 2.6: Metodología de análisis de letras de canciones por diccionarios

2.2.4 Tendencias suicidas: Clasificación automáticas de letristas suicidas y letristas no suicidas usando PLN.

Mattew Mulholland,Joanne Quinn

En este trabajo se hacen la pregunta si es posible que mediante algoritmos de procesamiento de lenguaje natural se pueda predecir la probabilidad de que un musico o letrista se pueda suicidar o se haya suicidado, para poder responder esta pregunta se desarrolló un corpus que está integrado por canciones de artistas que se han suicidado y por letras de artistas que no se han suicidado, de este corpus se hizo un conjunto para entrenamiento y otro conjunto que sirve de prueba. Esta idea se llevó a cabo debido a las recientes investigaciones de la aplicación del PLN para la detección de enfermedades como la demencia para poder identificar la demencia en escritores se usaron diferentes características sintácticas y de vocabulario, además se consultó un trabajo de Stirman y Pennebaker que utilizan las palabras como indicador de los estados mentales, ellos desarrollaron un programa para analizar más de 70 dimensiones del lenguaje como la polaridad, estados afectivos, sexualidad, muerte, tiempo, etc., este programa se llama Linguistic Inquiry and Word Count, su investigación pudo determinar la relación que existe entre los artistas que se suicidan y las letras que escriben, estos artistas presentan una disminución en el uso de pronombres en primer persona, generalmente usaban más palabras sexuales y referencias a la muerte. Además, existen otro trabajo como el en Neuman del 2010 que creo un campo semántico a partir de las palabras para reconocer el concepto de depresión, pudo identificar con mucha precisión a las personas deprimidas por medio de su escritura. Como se puede observar, se

han hecho varios algoritmos enfocados al análisis de sentimiento por medio de las letras, sin embargo, es un campo muy poco explorado debido a que es complicado analizar las letras debido a varias características que estas poseen.

Para el desarrollo del algoritmo, primeramente se construyó un corpus de letras de artistas que se han suicidado y de artistas que no se han suicidado, está constituido por 533 canciones, de estas 253 fueron escritas por artistas que se suicidaron y 280 por 5 artistas que no se han suicidado. Primero se formó el conjunto de prueba, este está integrado por 63 canciones de artistas suicidas y 46 canciones por artistas que no se suicidaron, después se formó el conjunto de desarrollo que está conformado por 168 canciones en total, 94 de artistas suicidad y 74 de artistas no suicidas. Para la elección de las canciones buscaron artistas que cumplieran con ciertos requisitos, el principal es que el suicido sea inequívoco, se trató de distribuir a los letristas entre los diferentes conjuntos de manera uniforme. Para tener una mayor precisión se desecharon todas aquellas letras que hayan sido escritas por un compañero de banda u otros músicos. El corpus suicida está integrado un 48% por canciones escritas por Bob Dylan por lo que se decidió eliminar el 35% de estas canciones, los otros artistas contribuyeron con un total de 32 a 45 canciones cada uno. Para la obtención de las letras se buscó en bases datos de letras en línea, las líneas se unieron en oraciones o frases para usar un etiquetador, posteriormente las letras se tokenizaron, para hacer este proceso se utilizó el tokenizador de OpenNLP, todas estas características se calcularon usando PYTHON y la herramienta de corpus UAM que se usa especialmente para análisis gramatical.

Para lograr hacer la clasificación, algunas características se basaron solamente en conteos sin procesar de tipos, tokens y tiempo de canción, para lograr una mejor clasificación de las canciones se tomaron a consideración las investigaciones realizadas por Stirman y Pennebaker, por lo que se consideraron que todas las canciones de artistas que se suicidaron deberían tener una mayor cantidad de pronombres en primer personas ya que una de las percepciones que se tienen de los suicidios y de las personas depresivas es que son muy egocéntricas y no se preocupan por los demás, también se consideraron el cambio de uso de los verbos, en las personas con tendencias o que se han suicidado incluyen mucho los verbos pensar y sentir, consideraron que las letras suicidas en comparación con las letras no suicidas serían más negativas y deprimentes por el estado mental del autor.

Para medir el uso de palabras negativas, positivas y neutrales se utilizó el conjunto de palabras de polaridad previa MPQA, se contaron todas las palabras que aparecían en cada texto para poder deducir la polaridad de la canción, también para determinar la polaridad de la canción se usó AFINN que es un diccionario

con casi 2500 términos polares con valores de polaridad asociados que van desde -5 a 5 para determinar el valor de polaridad de cada canción.

Para la creación del clasificador todas las funciones se ingresaron en el paquete de aprendizaje automático Weka y se probaron con una variedad de algoritmos, este clasificador fue entrenado con el conjunto de entrenamiento y posteriormente fue probado con el conjunto de prueba de las canciones de diferentes cantantes. Entre todos los algoritmos probados el más exitoso fue el algoritmo SimpleCart ya que clasifico las canciones como suicidas y no suicidas en un 70,6% de las veces, su nivel estadístico de clasificación aún no es satisfactorio pero se puede deducir que van en el camino correcto ya que han demostrado que esta tarea puede ser solucionada por medio del PLN, muchas características que se calcularon en este trabajo pudieron destacar como los términos neutrales, sustantivos concretos, palabras sensuales y la característica del valor polar total.

Aunque se sabe que van en el camino correcto lograron deducir que la construcción de un corpus para este tipo de proyectos tiene muchos problemas debido a que algunas letras están escritas por artistas de mayor edad que en algunos casos es probable que mueran por causas naturales por lo que esto podría ser un problema para la identificación de las letras y representaría una reducción en el corpus, otro motivo puede ser que un artista tuviera intentos de suicidio y simplemente se mantuvo en secreto por lo que estos podrían ser clasificados como no suicidas, otro gran problema con el corpus es que la cantidad de letras de artistas que se suicidaron es bastante limitado, por lo que es un problema la adquisición de datos.

Este trabajo propone trabajos a futuro, entre estos es perfeccionar los métodos que se utilizan para la clasificación del corpus y expandir el corpus a muchas otras letras no suicidas y por último tratar de extender el análisis a otras características debido a que se ha comprobado que la PLN puede resolver muchos problemas dentro del analisis de letras.

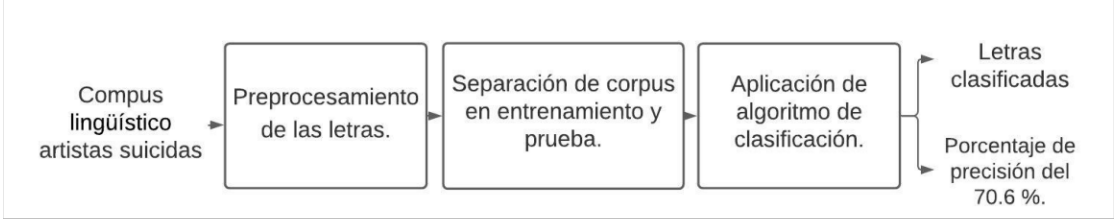


Figura 2.7: Metodología de análisis de letras de canciones de artistas suicidas

2.2.5 Clasificación de la polaridad del sentimiento de las canciones tailandesas basado en la letra.

Chutimet Srinilta , Wisuwat Sunhem, Suchat Tungjunob, Saruta Thasanthiah y Supawit Vatahanavaro.

La música forma parte de la vida humana, produce ciertos tipos de respuestas emocionales, el aprendizaje automático es utilizado comúnmente para la clasificación de los estados de ánimo musical. Los estados de ánimo más básicos para la clasificación de letras de canciones se conforman por dos grupos, los estados de ánimo felices y los estados de ánimo tristes.

El texto de una canción tiene una gran cantidad de información, existen muchas técnicas para la minería de texto, utilizan textos en lenguaje natural para extraer patrones lingüísticos, todas estas técnicas pueden ser aplicables a las letras con el propósito de identificar el estado de ánimo de la canción

Existen diferentes enfoques en la clasificación del estado de ánimo de una canción, la primera de estas se basa en el análisis del audio, por medio del audio se extrae información como el volumen, timbre y ritmo de la canción para poder determinar el estado de ánimo que transmite, el segundo enfoque se basa en la clasificación por medio de las letras y las etiquetas sociales. Existe otro enfoque de clasificación que es hibrido ya se usan en conjunto las dos anteriores para obtener resultados un poco más precisos.

Algunos investigadores han hecho trabajos relacionados con el trabajo descrito en este archivo uno de los más importantes es el trabajo de Dewi y Harjoko que utilizaron patrones rítmicos para determinar el estado de ánimo de las canciones infantiles en inglés e indonesio, otro trabajo relacionado es el de Patra, Das y Bandyopadhyay, ellos usaron características estilísticas de los textos para hacer un clasificador de canciones hindúes.

Para este trabajo se considera el procesamiento de lenguaje natural ya que es fundamental para la extracción de información, en este trabajo se analiza el idioma tailandés, este idioma tiene características específicas que pueden ser un desafío para el PLN, en el PLN es muy común que el texto de entrada se separe en palabras individuales a esto se le conoce como segmentación, este paso es considerado uno de los más importantes ya que dependiendo de la segmentación será el resultado final, para el idioma tailandés se han desarrollado muchas técnicas y programas de segmentación. También se hace uso de la indexación basado en n-gramas, esta técnica es muy utilizada para la recuperación de información, la técnica de n-gramas

presta atención a la probabilidad de una palabra, el beneficio de los n-gramas es que se requieren de conocimientos lingüísticos de la lengua.

Las letras de las canciones tienen una estructura, como AAA, AABA, Verso/Coro y Verso/Coro/Puente, dentro de la estructura de una canción el verso y el estribillo son las partes principales de una canción y es muy probable que el tema de la canción se encuentre en alguna de estas dos partes. El número de palabras dentro de una canción varía entre 100 y 300 palabras, una letra de canción es más larga que un mensaje de una red social.

Para el desarrollo del clasificador se creó un archivo de sentimientos, este archivo está integrado por dos listas, una lista de felicidad y otro de tristeza, estas se crean a partir de los términos del conjunto de características de las letras. Las listas felices tienen los términos que aparecen con más frecuencia en las canciones felices y la lista triste tiene los términos que aparece con más frecuencias en canciones tristes.

Todos los términos del conjunto de características de la letra se añaden a una sola lista de polaridad, en el caso de que una palabra aparezca en las dos listas de polaridad, este término se ignora debido a que no representa un sentimiento fuerte hacia ninguna de las polaridades.

El primer paso es extraer todas las características de la canción, como siguiente paso la letra de la canción se convierte en un conjunto de características de la letra y se realiza un bucle que pasa por todos los términos comparándolos con las dos listas de polaridad, el principal propósito de este paso es determinar la puntuación de polaridad de cada término. Dependiendo de que los términos son igualmente importantes o no son importantes se ponderan con un valor, posteriormente se promedian todas las puntuaciones y el resultado obtenido representa la polaridad de la canción y de esta forma se puede etiquetar como una canción positiva o negativa dependiendo del valor de polaridad que obtuvo.

Para el desarrollo de un clasificador de texto se eligió una red neuronal de percepción multicapa. Todas las letras de canciones que se usan en este trabajo fueron obtenidas por el sitio web de Chord Café, de aquí se obtuvieron diferentes canciones que fueron primeramente clasificados en 34 grupos posteriormente estos 34 grupos fueron clasificados en 2 grupos de polaridad, felices o tristes, en total se obtuvieron 427 canciones felices y 317 canciones tristes

Posteriormente de haber desarrollado el clasificador con diferentes métodos se de-

termino la precisión de cada proceso, la precisión del algoritmo es considerado como todos aquellos datos que fueron clasificados correctamente. Cabe mencionar que los experimentos se desarrollaron con canciones completas y la otra es solamente considerando los versos y los estribillos.

Los resultados obtenidos en todos los experimentos que se utilizaron conjunto de datos de letras completas tuvieron una precisión que se sitúa entre 62% y 68%. En el caso de los experimentos que solamente se consideraron los versos y estribillos se tuvo una precisión muy inferior de un 15%.

Pudieron concluir que el clasificador desarrollado tiene resultados aceptables pero que aún pueden mejorarse al igual que tiene muchas aplicaciones dentro del mundo de la música como el sugerir canciones para una lista de reproducción o también para averiguar el estado emocional actual de una persona.

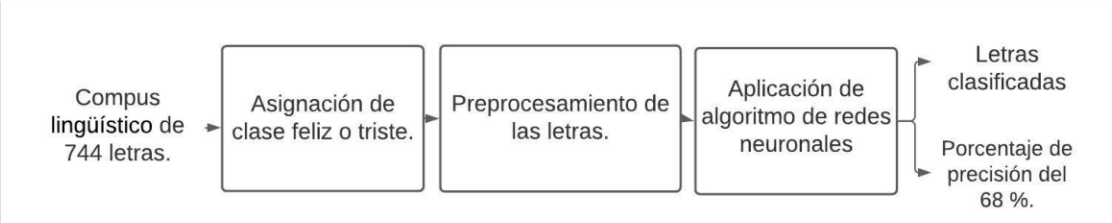


Figura 2.8: Metodología de análisis de letras de canciones de canciones tailandesas

2.3 Discusión

Tabla 2.1: En estas tablas se realiza la comparación de las características de cada trabajo relacionado con el trabajo descrito en este trabajo.

Características	Trabajo 1	Trabajo 2	Trabajo 3	Trabajo 4
Uso de letras de canciones para el analisis	✓	✓	✓	✓
Letras de canciones infantiles	✓	✗	✓	✗
Desarrollo en Python	✓	✓	✓	✓
Libreria NLTK	✗	✗	✗	✗
Tokenización	✓	✓	✓	✓
Diccionarios	✗	✗	✗	✗
Aprendizaje automático	✓	✓	✓	✓
Clasificación de sentimientos	✓	✓	✓	✓

Características	Trabajo 5	Mi propuesta
Uso de letras de canciones para el análisis	✓	✓
Letras de canciones infantiles	✗	✓
Desarrollo en Python	✓	✓
Libreria NLTK	✗	✓
Tokenización	✓	✓
Diccionarios	✗	✓
Aprendizaje automático	✓	✓
Clasificación de sentimientos	✓	✓

Se puede observar en la Tabla 2.1 que todos los trabajos están enfocados al análisis de los sentimientos de las letras de las canciones por medio de algoritmos de aprendizaje automático. El enfoque de casi todos los trabajos es diferente en nuestro caso nos centramos en el análisis de las canciones infantiles mientras que los otros se enfocan en el análisis de letras de canciones hindis o letras de canciones tailandesas, solo el Trabajo 1 y el trabajo 3 abarcan el análisis de letras de diferentes géneros.

Todos los algoritmos son desarrollados en Python y todos aplican la técnica de tokenización para la separación de palabras para su posterior eliminación o análisis.

Solo nuestro trabajo utiliza la librería NLTK, en el caso de los demás trabajos utilizan librerías parecidas pero que se encuentran en su idioma, al igual que solo en este trabajo se usan diccionarios en español para el análisis de las letras.
En todos los trabajos se usan algoritmos de aprendizaje automático, todos usan una parte de su corpus para entrenar su algoritmo para posteriormente hacer una clasificación del sentimiento que quiere transmitir la letra de la canción, con estos resultados determinan la precisión del algoritmo.

Capítulo 3

Marco Teórico

3.1 Procesamiento de lenguaje natural

El Procesamiento de Lenguaje Natural por sus siglas *PLN* es una rama de la ingeniería y de las ciencias de la computación que tiene por objetivo poder facilitar la interacción que hay entre un humano y una máquina, para poder lograr esta interacción se hace uso de las reglas gramaticales y del vocabulario del ser humano, pero debido a que una máquina solo es capaz de entender el lenguaje binario de ceros y unos tiene que pasar por una serie de algoritmos y procesos que la entrenen para poder ser capaz de comprender el lenguaje humano.

Por medio de las fases de entrenamiento la máquina posteriormente es capaz de determinar patrones que la permitan dar una respuesta, la respuesta de estos algoritmos dependen mucho de la cantidad, tipo y calidad de los datos ingresados para el entrenamiento.

Actualmente el *PLN* tiene muchas aplicaciones en el mercado global, ya que por medio de estas técnicas se pueden desarrollar análisis de diferentes patrones que pueden ayudar a la investigación y a la toma de decisiones de diferentes disciplinas o de algunas organizaciones.

3.2 Python

Python es un lenguaje de programación de alto nivel, este lenguaje está orientado a objetos y está diseñado especialmente para el desarrollo de aplicaciones informáticas, pero también es usado para el desarrollo web. Es considerado un lenguaje muy fácil de aprender, debido a su sintaxis sencilla.

Este lenguaje soporta el uso de módulos y paquetes por lo que sus códigos pueden ser reutilizados en varios proyectos, otro de los beneficios de este lenguaje de programación es que tanto el intérprete como la librería estándar están disponibles de forma gratuita.

Este es un lenguaje multiparadigma y multiplataforma, además de que cuenta con una licencia de código abierto que permite utilizarlo en cualquier escenario, esto hace que sea uno de los lenguajes para iniciar en el mundo de la programación, en la Figura 3.1 se presenta el logo oficial de Python y en la Figura 3.2 se hace una comparación entre el lenguaje de programación JAVA y Python.[16]

Figura 3.1: Logo oficial de Python

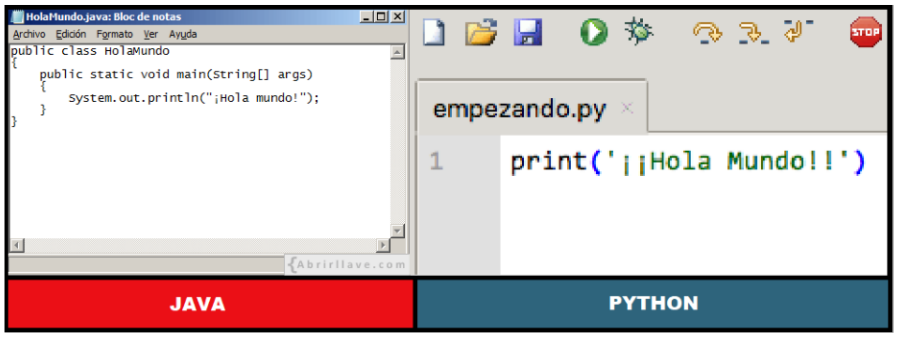


Figura 3.2: Comparación de código *"Hola mundo"* en JAVA y Python

3.3 Corpus lingüístico

Se le conoce como *corpus lingüístico* al conjunto de textos que tienen un mismo origen y que tiene por objetivo almacenar un conjunto de documentos o textos con el fin de usar estos datos para hacer algún análisis.

Todos los *corpus lingüísticos* son creados con base a determinados requerimientos, a nuestras necesidades y al tipo de análisis que queramos llevar a cabo. Existen tres factores básicos que se deben tomar en cuenta al momento de desarrollar un corpus lingüístico, estos son los siguientes:

- Representatividad: Se refiere a que el corpus lingüístico debe abarcar de la manera más amplia posible todas las variedades que existen de algún tema que se esté analizando. Se puntualiza que al tener una mayor variedad de datos es más probable tener un mayor porcentaje de precisión en el algoritmo.
- Tamaño: El corpus lingüístico debe tener un tamaño que permita obtener resultados satisfactorios, al ser demasiado pequeño se tienen mayores probabilidades de que los resultados no sean los esperados y si el corpus es más grande, abarca todos los elementos y variedades del tema a analizar es más probable que los resultados sean los esperados, todos estos factores también dependen del procesamiento que se tengan de los datos.
- Equilibrio: Este se encuentra relacionado con que haya una neutralidad en el corpus, este parámetro se considera difícil de cumplir, sin embargo, se debe de cumplir para que el análisis pueda ser lo más preciso posible.

3.4 Paquetes para PLN y manipulación de datos

Dentro de la *PLN* hay diferentes técnicas que permiten hacer un procesamiento de los corpus lingüísticos con el propósito de agilizar y hacer más eficiente el entrenamiento de nuestros algoritmos, existen diferentes paquetes que permiten aplicar estas técnicas a una gran cantidad de datos, una de estos paquetes es *NLTK*. Hay otros paquetes como *Pandas* y *Numpy* que permiten manipular una gran cantidad de datos y hay otros paquetes como *Scikit-learn* que proporcionan los recursos para aplicar algoritmos de machine learning.

3.4.1 NLTK

Natural Languaje Toolkit *(NTLK por sus siglas en inglés)* es un paquete de *Python* que es usado para el *PLN*, este paquete proporciona una gran cantidad de recursos y herramientas que están orientados al análisis de textos, ademas de que cuenta con una sintaxis poco compleja junto a un conjunto de bibliotecas de procesamiento de texto, con este paquete se aplican técnicas como la tokenización, etiquetado,

lematización, eliminación de palabras vacías (stopwords), entre otras.

Esta herramienta está diseñada para apoyar la enseñanza y la investigación del *PLN* o áreas que este relacionadas con esta. Este paquete está diseñado para poder ser usado de forma estable con Python 2, sin embargo, se está realizando una transición a Python 3 pero algunas funciones del paquete aún son inestables con esta versión.

Por medio de su libro digital de acceso gratuito se puede tener una guía que lleva al programador paso a paso para tener conocimiento de como instalar este paquete en nuestra PC y también proporciona las técnicas más adecuadas para el uso de sus módulos, esto lo realiza por medio de un largo curso en donde se trabajan con diferentes corpus lingüísticos que ya se encuentran incluidos en el paquete.[17]

Figura 3.3: Logo oficial de NLTK

3.4.2 Pandas

Pandas es un paquete que está diseñado para el análisis y la manipulación de datos en *Python*, este paquete ofrece nuevas estructuras de datos y nuevas operaciones para la manipulación de tablas, en estas tablas se pueden almacenar una gran cantidad de datos, las filas representan los datos que se tienen almacenados, mientras que las columnas son las variables.

Este paquete usa el tipo de dato Data Frame para la manipulación de los datos y también incluye estructuras que permiten la búsqueda, modificación y eliminación de datos que ya no sean necesarios, en la Figura 3.4 se presenta el logo oficial de la librería *Pandas*.[18]

3.4.3 Numpy

Es un paquete diseñado para trabajar en *Python*, este paquete da soporte para la creación y manipulación de vectores o matrices de grandes tamaños. Cuenta con funciones que permiten realizar cálculos con las matrices y vectores.

Figura 3.4: Logo oficial de Pandas

Numpy es una de las librerías principales para la informática científica, ya que proporciona estructuras de datos muy poderosas, estos cálculos se desarrollan implementando matrices multidimensionales. Esta clase de datos garantizan cálculos muy eficientes con matrices y es considerado uno de los paquetes básicos para aplicar conocimientos de Machine Learning, en la Figura 3.5 se presenta el logo oficial de la librería *Numpy.*[19]

Figura 3.5: Logo oficial de Numpy

3.4.4 Scikit-learn

Es un paquete de acceso gratuito diseñado para *Python*, cuenta con una variedad de algoritmos para realizar diferentes procesos tales como de regresión, clasificación, entre otros. Esta librería es compatible con otros paquetes que se usan para el manejo y cálculo de grandes cantidades de datos, algunas de estos paquetes son *Pandas* [3.4.2], *Numpy* [3.4.3], SciPy o matplotlib.

Al ser compatible con otras librerias permite manipular datos de diferentes paquetes. Este paquete es una de las herramientas básicas para todos aquellos desarrolladores que quieren incursionar en los sistemas de análisis de datos y de modelado estadístico.La principal ventaja es cuenta con una variedad de algoritmos con los que se realizan análisis estadísticos y algoritmos de aprendizaje supervisado y no supervisado.En la Figura 3.6 se presenta el logo oficial de la librería *Scikit-learn.*[20]

Figura 3.6: Logo oficial de Scikit-learn

3.5 Técnicas de preprocesamiento de PLN.

En este apartado se describen las técnicas más utilizadas para hacer un preprocesamiento de un corpus lingüístico, el propósito de estos procesos es hacer que la estructura del corpus lingüístico sea las más adecuada para obtener resultados precisos y tener un menor tiempo de análisis, en este apartado se explican i) Tokenización ii) Lematización y iii) Eliminación de stop-words.

3.5.1 Tokenización

El proceso de *tokenización* consiste en dividir las cadenas largas de texto en pequeños segmentos que pueden ser palabras u oraciones, a estos elementos que son producto de la tokenización se les conoce como tokens, este procedimiento permite hacer una interpretación del significado del texto al analizar la secuencia de las palabras.

La tokenización es usada para separar largas cadenas de texto, si se realiza una división por palabras a esta se le conoce como tokenización por palabras y si se realiza la separación por oraciones se le conoce como tokenización de oraciones.

En la Figura 3.7 se realiza la tokenización por palabras de la oración *"La muñeca esta muy feliz en su casa de madera"*.

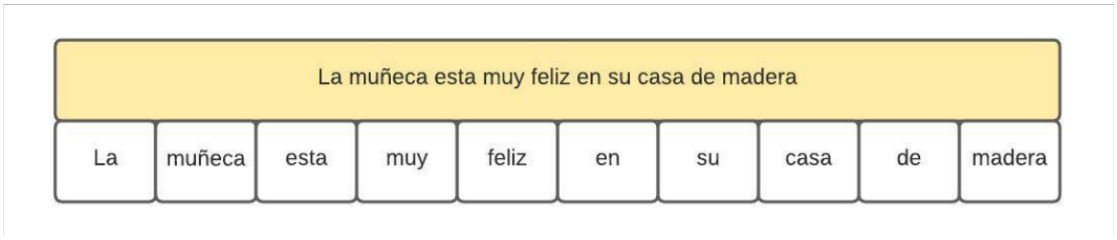


Figura 3.7: Tokenización por palabra

3.5.2 Lematización

El proceso de *lematización* consiste en hacer un análisis de las letras que se tokenizaron con el propósito de simplificar la información que se procesa, para esto la lematización desarrolla una conversión de las palabras a su forma base o de diccionario, a este elemento que fue modificado se le conoce como lema.De esta forma se reducen la cantidad de elementos que se analizan, ya que si no se hiciera la lematización el algoritmo puede considerar a las palabras que tienen la misma forma base como palabras completamente diferentes lo que da como resultado un mayor tiempo de procesamiento.En la Figura 3.8 se realiza la lematización de verbos a su forma base.

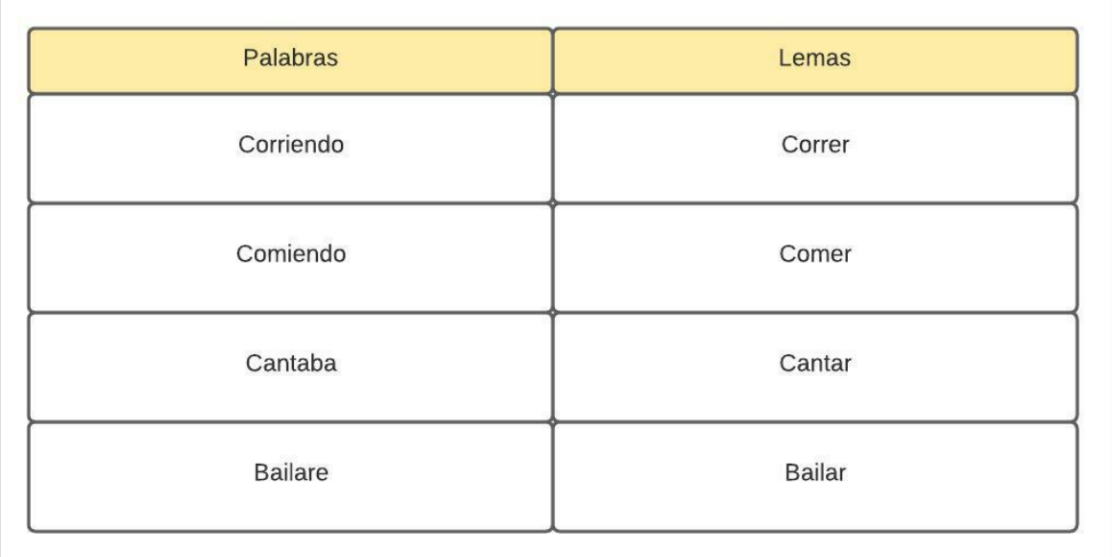


Figura 3.8: Lematización de palabras

3.5.3 Eliminación de palabras vacías o stop-words

Se consideran como palabras vacías a todas aquellas dentro de un texto que no le agregan significado a una oración por lo que pueden ser eliminadas sin arriesgar el contexto de la oración. La eliminación de las palabras vacías depende de la aplicación que se esté desarrollando, en el caso de que se desarrolle un programa para análisis de sentimientos, las palabras vacías no aportan nada en el procesamiento del texto, pero si se desarrolla un clasificador de idiomas las palabras vacías son importantes, ya que se usan junto con otras palabras.

Por un lado la eliminación de estas palabras tiene ventajas, una de estas es que al eliminar las palabras vacías el conjunto de datos es disminuido por lo que el

tiempo de entrenamiento disminuye. También al eliminar estas palabras se puede mejorar el rendimiento, ya que al quedar menos tokens el porcentaje de precisión puede ser mejor.

Por otro lado una de las desventajas que tiene la eliminación de palabras vacías es que si se realiza una eliminación inadecuada se puede cambiar el significado del texto que se esté analizando. En la Figura 3.9 se puede observar la frase *"La muñeca está muy feliz en su casa de madera"* ya tokezanida, después de la tokenización se realiza la eliminación de stop-words y se tiene como resultado *"muñeca feliz casa madera"*[21].

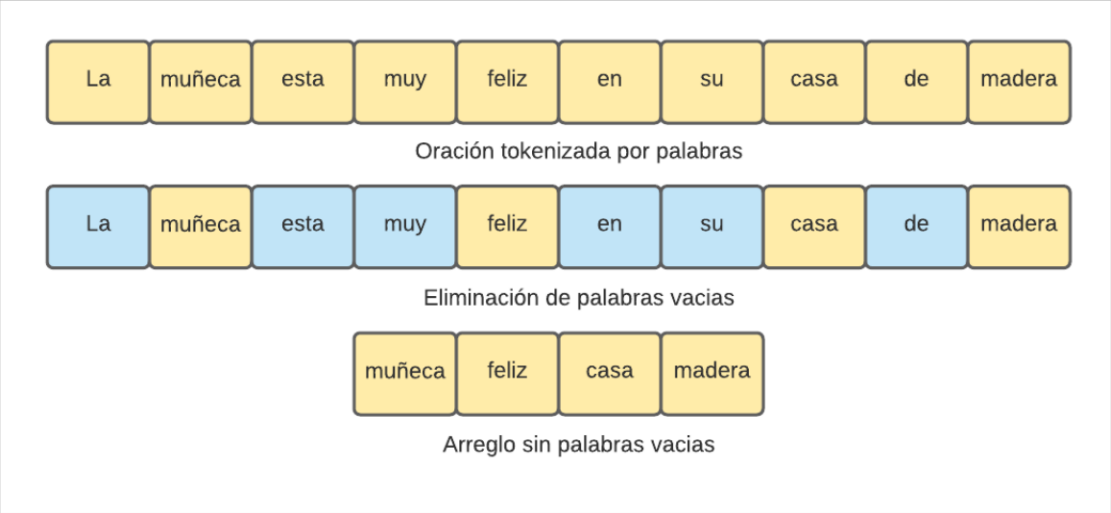


Figura 3.9: Eliminación de stopwords

En el siguiente apartado se describen las plataformas que son utilizadas comúnmente para el desarrollo de algoritmos de PLN en Python.

3.6 Programas para desarrollo de código en Python

En este apartado se describen los programas más populares para el desarrollo de algoritmos en Python, estos son considerados los más utilizados debido a sus características que los convierten en los entornos más cómodos y fáciles de manejar en este lenguaje. En este apartado se describen los entornos de i) Anaconda y ii) Jupyter Notebook.

3.6.1 Anaconda

Anaconda es una plataforma que tiene una variedad de librerías y aplicaciones que están diseñadas específicamente para el desarrollo de la ciencia de datos en Python. La documentación de esta plataforma es bastante detallada además de que cuenta con una gran comunidad activa, otras características de esta plataforma son las siguientes:

- Permite administrar e instalar nuevos paquetes para el análisis y aplicación de las ciencias de datos.
- Cuenta con diversos IDE como Spyder, Jupyter, Jupyter lab, entre otros.
- Cuenta con varias herramientas ya incluidas como los paquetes Numpy y pandas para el análisis de datos.
- Todos los proyectos que se desarrollan son portables, es decir, permite compartir proyectos con otros programadores y también permite ejecutar proyectos de diferentes plataformas.
- Es multiplataforma, es decir, es compatible con sistemas operativos como Linux o macOS.
- Permite el desarrollo de proyectos en diferentes lenguajes tales como Python o C++.

Para la ejecución de los programas proporciona diferentes entornos como Jupyter Notebook que permite ejecutar pequeños segmentos de código sin necesidad de compilar el código completo y Spyder en la cual se necesita compilar todo el programa, pero permite tener una gestión del manejo que se está haciendo de cada una de las variables declaradas.En las Figuras 3.10 y 3.11 se presenta el logo oficial de la plataforma de Anaconda y la interfaz que se utiliza para la selección de diferentes aplicaciones para el desarrollo de código en Python y otros lenguajes de programación.[22]

Figura 3.10: Logo oficial de Anaconda

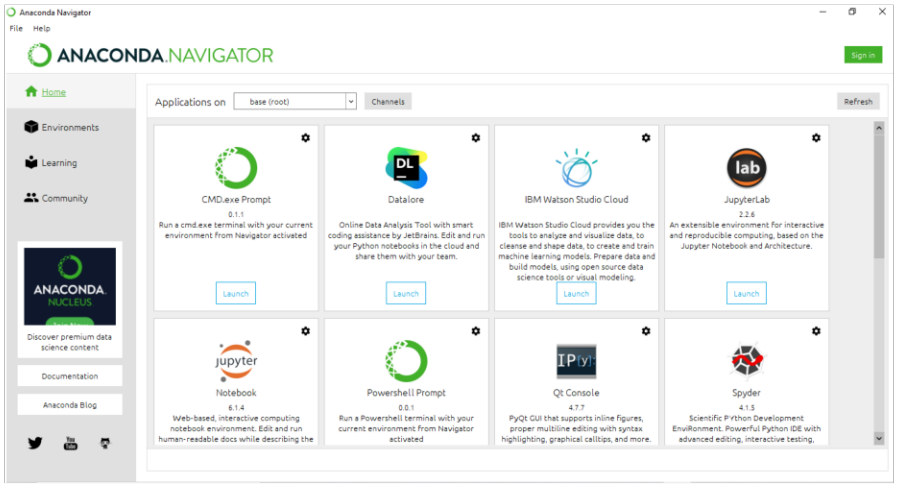


Figura 3.11: Interfaz de desarrollo de Anaconda

3.6.2 Jupyter Notebook

Es una aplicación web de desarrollo de código abierto que permite crear documentos con códigos en vivo, esta aplicación permite trabajar con bloques, además de incluir código fuente, también es posible agregar otros elementos como texto para describir el funcionamiento de un segmento de código e inclusive es posible agregar imágenes para describir el resultado del algoritmo.

Esta aplicación se utiliza para el desarrollo de programas en el ámbito escolar. Además de que tiene la ventaja de escribir solo un segmento de código en un bloque del documento esta aplicación también permite ejecutar ese segmento de código sin necesidad de hacer la compilación del programa completo lo que ocasiona una reducción en el tiempo de desarrollo y compilación.

Los archivos de *Jupyter Notebook* se pueden convertir a varios formatos de salida, como archivos HTML, PDF, Latex, Python, etc.En las Figuras 3.12 y 3.13 se presenta el logo oficial de Jupyter y la interfaz de desarrollo de Jupyter Notebook, en esta interface es posible agregar código y también texto para la descripción de los algoritmos.[23]

Figura 3.12: Logo oficial de Jupyter

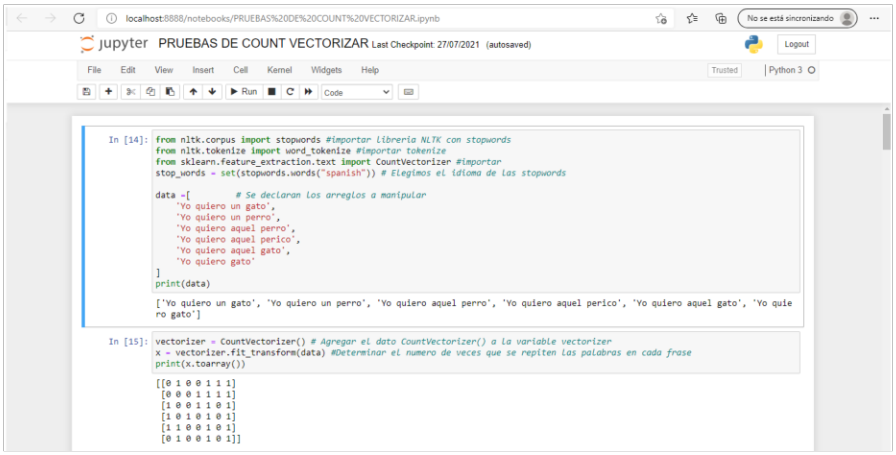


Figura 3.13: Interface de Jupyter Notebook

3.7 Tipos de aprendizaje

En este apartado se explican los diferentes tipos de aprendizaje, este depende de la tarea que se quiera desarrollar, existen 2 tipos , i) Aprendizaje supervisado y ii)Aprendizaje no supervisado.

3.7.1 Aprendizaje supervisado

Los algoritmos de *aprendizaje supervisado* son todos aquellos que basan su aprendizaje en un conjunto de datos que anteriormente ya fueron etiquetados. El etiquetado es el proceso mediante el cual los datos que se van a manejar ya se encuentran clasificados, por lo tanto se conoce el valor de la variable objetivo.

Con los datos que ya fueron etiquetados el algoritmo podrá aprender una función que tenga la capacidad de predecir la variable que se esté analizando para

posteriormente poder ser usado con datos completamente nuevos.

Los algoritmos supervisados se clasifican en dos grandes familias esto depende de la aplicación que se esté desarrollando, el algoritmo se puede clasificar en:

- Algoritmos de regresión: En este tipo de algoritmos se hacen predicciones de valores con base en entradas pasadas, es decir, el algoritmo construye un modelo que está basado en los valores y características de la salida de los datos que se usaron anteriormente para poder entrenarlo, los valores que se obtienen de esta clase de algoritmos son continuos y no discretos. Esta clase de algoritmos tienen aplicación en la predicción de precios, cantidades de compra y para predecir la cantidad de ingresos que se pueden obtener a partir de una campaña de marketing.

- Algoritmos de clasificación: Estos algoritmos intentan etiquetar varios ejemplos eligiendo entre dos o más clases que son diferentes, para lograr esto, esta clase de algoritmos deben crear modelos que puedan predecir resultados por medio de datos que fueron ingresados anteriormente para su capacitación, tras ser capacitados utilizan las características aprendidas para usarlas en datos que son nuevos.

  Cuando un dato se clasifica entre dos clases a eso se le denomina clasificación binaria, en caso de que se clasifiquen en más de dos clases a esto se le conoce como clasificación multiclase. Este tipo de algoritmos son usados en PLN para hacer la clasificación de sentimientos, algunos trabajos utilizan clasificadores multiclase mientras que otros usan clasificación binaria. En la Figura 3.14 se tiene un diagrama de funcionamiento de los algoritmos de aprendizaje supervisado, como entrada al sistema se tienen los datos que anteriormente fueron etiquetados, entran al algoritmo de aprendizaje supervisado y como salida de este sistema se tienen los resultados de la variable objetivo que son comparados con los datos de entrada para determinar el porcentaje de precisión del algoritmo.

3.7.2 Aprendizaje no supervisado

El *aprendizaje no supervisado* son todos aquellos algoritmos que para realizar su proceso de entrenamiento no requieren de datos de entrada que estén etiquetados previamente por lo que la respuesta de salida es desconocida, el propósito de estos algoritmos es aumentar el conocimiento estructural de los datos que estén disponibles.

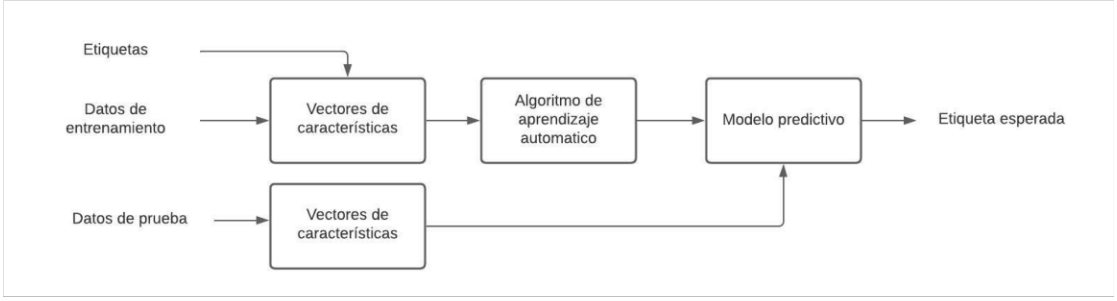


Figura 3.14: Diagrama de funcionamiento del aprendizaje supervisado

Esta clase de algoritmos son de tipo exploratorio, ya que si no se conoce la entrada no se conoce la salida por lo que después de este proceso se debe encontrar algún tipo de organización que puede simplificar el análisis.

En la Figura 3.15, se explica el proceso que realiza este tipo de algoritmos para dar un resultado en base a las caracteristicas de los datos, la entrada del diagrama son los datos, posteriormente entran al algoritmo de aprendizaje y como resultado se tiene una respuesta en base a los datos ingresados en la entrada.[24]

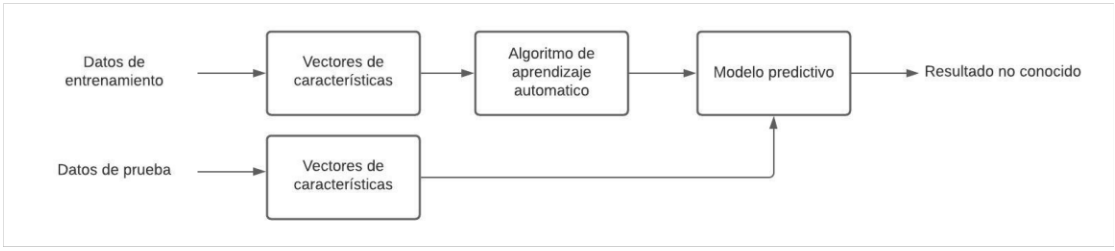


Figura 3.15: Diagrama de funcionamiento del aprendizaje no supervisado

3.8 Metodos de validación

Los *métodos de validación* son un conjunto de estrategias que permiten estimar la capacidad que tiene un sistema para predecir los posibles datos, para esto solamente se hace uso de una pequeña parte del conjunto de entrenamiento. Este proceso se repite n cantidad de veces y los resultados que se obtienen se suman y se promedian, como resultado de repetir el proceso n cantidad de veces podemos hacer una compensación en las desviaciones que puedan surgir por el reparto aleatorio de los datos. La diferencia que existe entre los diferentes métodos de validación es la forma en que generan los subconjuntos que son obtenidos del conjunto de entrenamiento.

Los métodos de validación necesitan de algunos elementos para poder realizar su trabajo, estos son:

- Conjunto de entrenamiento: Estos son los datos con los que se entrena al modelo.
- Conjunto de validación y conjunto de test: Son datos del mismo tipo que el conjunto de entrenamiento, sin embargo, no fueron utilizados para el entrenamiento del algoritmo, mas bien estos son usados después del entrenamiento para verificar si el algoritmo cumple con su objetivo.
- Error de entrenamiento: Es el error que comete el modelo al predecir datos que pertenecen al conjunto de entrenamiento.
- Error de validación y error de test: Es el error que presenta el modelo al predecir datos que son parte del conjunto de prueba o validación, en ambos casos son datos que el modelo no ha visto.

Tras conocer las partes mas importantes para demostrar si un algoritmo está funcionando correctamente, en la siguiente sección se describen los métodos de validación más comunes para saber la capacidad predictiva de un algoritmo.

3.8.1 Validación simple

Este método de validación consiste en repartir de forma aleatoria todos los datos disponibles en dos grupos, el primero se emplea para entrenar al modelo mientras que el segundo se usa para evaluarlo. Este es el método más simple, sin embargo, tiene problemas muy importantes, el primero es la estimación de error que es extremadamente variable, este error de estimación depende de la separación que se hizo de los datos en conjunto de prueba y conjunto de entrenamiento, el segundo problema es que al separar los datos en dos grupos se llega a tener menos información con la se pueda entrenar al modelo y por lo tanto se puede reducir su precisión.

En la Figura 3.16 se muesta el conjunto de datos y la separación de este conjunto en dos partes, el primero es el grupo de entrenamiento mientras que el segundo es el grupo de prueba, el de entrenamiento es el encargado de preparar o entrenar al algoritmo mientras que el de prueba se usa para verificar la precisión y correcto funcionamiento del mismo.

Figura 3.16: Diagrama de validación simple

3.8.2 Validación cruzada

Debido a que con la validación simple se hace una separación y reduccion del tamaño del conjunto principal, la validación cruzada resuelve este problema a cambio de aumentar la complejidad del cálculo, para lograr esto la validación cruzada divide los datos de forma aleatoria en k grupos aproximadamente del mismo tamaño, la gran mayoría de los grupos son usados para entrenar el modelo, pero solo uno de estos se emplea como conjunto de validación, este proceso se va a repetir k veces y en cada iteración se usa un conjunto diferente.

Este tipo de validación presenta principalmente dos ventajas, la primera es los requerimientos computacionales, el número de iteraciones que se van a llevar a cabo depende del valor de k, por lo general se recomienda usar un k con valor de 5 o 10, pero en caso de que el conjunto de datos sea muy extenso se debe considerar realizar mas iteraciones y la segunda es que al realizar una mayor cantidad de iteraciones, las estimaciones de error son menores debido a que se usa el mismo conjunto de entrenamiento para hacer la validación.

La elección del método de validación depende de dos factores.

1. Si el tamaño del conjunto de datos es pequeño, es mejor utilizar el método de validación cruzada, ya que tiene un buen equilibrio en la varianza y por lo mismo de contar con pocos datos, el coste computacional no es muy alto.

2. Si el tamaño del conjunto de datos es muy grande, es más importante la eficiencia computacional, pero se puede usar el método de validación cruzada considerando una mayor cantidad de iteraciones.[25]

3.9 Algoritmo de clasificación

3.9.1 Algoritmo K vecinos más cercanos

El algoritmo de K vecinos más cercanos es un algoritmo de aprendizaje supervisado de machine learning que permite realizar clasificaciones de nuevos valores y también es utilizado para realizar predicciones.

Es considerado uno de los métodos más básicos del aprendizaje automático, en resumen, el algoritmo de k vecinos más cercanos consiste en clasificar valores en base a los puntos más cercanos que fueron aprendidos en la etapa de entrenamiento.

Este tipo de algoritmo de aprendizaje está basado en instancia, con esto se refiere a que memoriza los datos de la etapa de entrenamiento, estos datos son su base de conocimiento para la fase de prueba.

Este algoritmo cuenta con varias desventajas, la primera es que utiliza todo el dataset de datos para poder entrenar cada uno de los datos por lo que requiere de una gran cantidad de recursos de procesamiento, por lo que es recomendable para conjunto de datos pequeños.

Para realizar la clasificación de los datos se basa en los vecinos más cercanos, para determinar los puntos más cercanos este algoritmo usa la distancia euclidiana la cual es un número positivo que indica la separación que se tiene entre dos puntos, la fórmula de la distancia euclidiana se muestra en la Figura 3.17.

$$D(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2)}$$

Figura 3.17: Formula distancia euclidiana

Si consideramos solamente a las dos dimensiones del plano cartesiano la formula queda como en la Figura 3.18.

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Figura 3.18: Formula distancia euclidiana entre 2 puntos

En esta fórmula el dato que queremos clasificar es uno de los puntos mientras que el otro punto es el dato que ya fue clasificado en el entrenamiento del algoritmo.

Para realizar la clasificación se deben considerar el número de K vecinos más cercanos que se tomaran en cuenta para la asignación de una clase, se recomienda que sea un numero impar para evitar empates, como se puede observar en la Figura 3.19 dependiendo del número de vecinos más cercano que se consideren es posible que sea clasificado con la clase correcta o incorrecta.

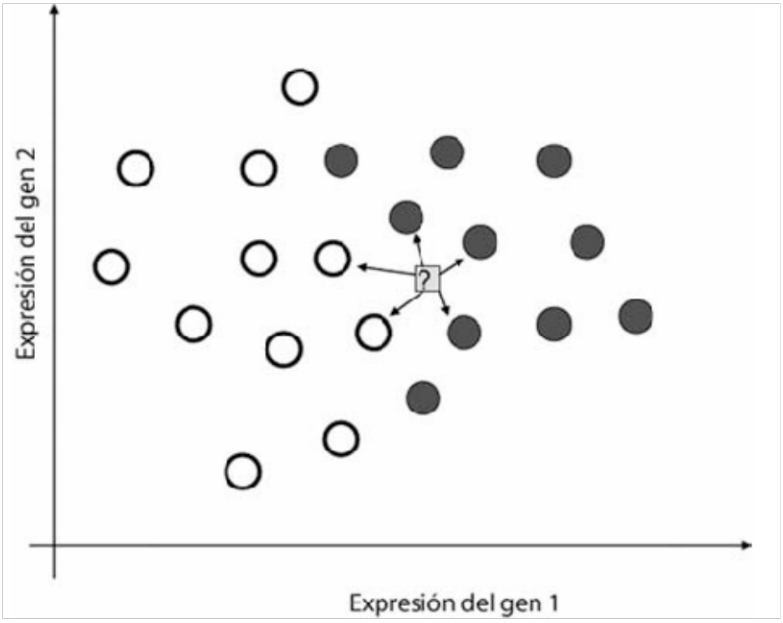


Figura 3.19: K vecinos mas cercanos.

Capítulo 4

Marco Metodológico

En este capítulo, se explica de forma detallada todas las etapas y fases que se llevaron a cabo para el desarrollo de este proyecto.

4.1 Etapa 1: Búsqueda y acomodo del conjunto de datos

En esta primera etapa se desarrollan todas las fases relacionadas con la búsqueda de información, acomodo de la información, desarrollo del corpus lingüístico e instalación de los programas para el desarrollo del código.

4.1.1 Fase 1: Metodología de búsqueda

Para realizar este proyecto primero se realiza una búsqueda en diferentes páginas de internet con el propósito de hacer una captura de las letras infantiles más comunes escritas por diferentes artistas.

Para esta búsqueda se consideraron nombres de cantantes famosos de este género musical como Tatiana, Cri Cri, Chabelo, etc., al igual que se hizo una breve investigación de las películas más populares desarrolladas por diferentes empresas como Disney, Universal Picture, Nickelodeon, etc.

Para la búsqueda de más canciones de este mismo género se consultaron diferentes artículos en la cual explican la influencia de las canciones infantiles en el desarrollo infantil.

4.1.2 Fase 2: Acomodo de la información y creación del corpus lingüístico

Después de la búsqueda de la información para el desarrollo del corpus, todas estas letras son almacenadas en diferentes archivos .txt, la estructura del bloc de nota es primero el título y después la letra de la canción, todos estos archivos fueron almacenados en una carpeta y se les nombra de tal forma que sea una numeración consecutiva.

Al tener las letras almacenadas y numeradas se desarrolla un archivo Excel con el propósito de hacer más fácil la búsqueda de las letras, este archivo Excel está estructurado de la siguiente forma.

- Primera columna: En esta primera columna se almacena la numeración de los archivos, esta numeración esta basada en la numeración que se hizo anteriormente con los blocs de notas.
- Segunda columna: En esta columna se colocó el título de la canción y se hizo un hipervínculo que permite abrir el bloc de notas.
- Tercera columna: Esta columna se utiliza para colocar las etiquetas de todas aquellas canciones consideradas positivas.
- Cuarta columna: Esta se usa para colocar las etiquetas de todas aquellas canciones consideradas negativas.
- Quinta columna: Esta columna se deja para escribir una breve explicación del porqué la letra se considera positiva o negativa.

Al final se obtuvieron en total 220 letras, todas estas son parte del corpus lingüístico de este proyecto.

4.1.3 Fase 3: Instalación de programas.

Para el desarrollo de este proyecto se usa el lenguaje de programación Python y el entorno de desarrollo Anaconda, como primer paso se ingresó a las páginas webs oficiales en donde se descargaron los archivos para la instalación de estos programas. Primero se descargó e instalo Python, los pasos que se siguieron para la instalación son los siguientes:

1. Ir al área de descargas para Windows de la página oficial de Python y descargar el instalador ejecutable, se debe considerar la arquitectura de su ordenador si es de 32 o 64 bits.

2. Abrir el explorador de archivos y buscar el ejecutable descargado para ejecutarlo.

3. Comenzara la instalación del archivo posteriormente en el asistente de instalación se deben seleccionar las opciones que se quieran instalar.

4. En la ventana Advanced Options marque la ruta en donde se desea que se haga la instalación, después de clic en el botón de install y espere a que la instalación finalice.

5. Para verificar que la instalación se haya realizado correctamente puede entrar a símbolo de sistema y escribir Python, debe mostrarle la versión instalada y puede usar esta interfaz para escribir código e instalar nuevos paquetes.

6. Para la instalación de paquetes se puede emplear el instalador PIP, por ejemplo, si se quiere instalar el paquete NLTK solo ejecute el comando pip install –user -U nltk, en caso de querer instalar otros paquetes puede consultar las páginas oficiales de dichos paquetes para ver el comando pip para la instalación.

Figura 4.1: Ventana de instalación de Python

Una vez instalado Python se descargan e instalan los archivos de Anaconda, este programa cuenta con diferentes entornos de desarrollo, para este proyecto se usa Jupyter Notebook que permite programar y ejecutar fragmentos de código específicos y también permite realizar anotaciones para la descripción del funcionamiento de cada fragmento de código.

4.2 Etapa 2: Etiquetado del corpus lingüístico

En esta segunda etapa se desarrollan las fases para el etiquetado de las letras del corpus. Para formar al jurado calificador se contactaron a 5 profesionales, estas 5 personas tienen las siguientes características.

- 5 mujeres.
- Rango de edad de 25 a 32 años.
- Cuentan con Licenciatura en Psicología.
- Cuentan con experiencia en psicología infantil.
- Tiempo de experiencia de entre 1 a 10 años.

La tarea que se le asigna al jurado es la siguiente: Una vez que se recopilaron las 220 canciones en bloc de notas y se hizo el formato del archivo Excel, toda la información se comprimió en un archivo .zip y se mandó por correo a las 5 integrantes del jurado, ellas deben leer la letra de la canción y determinar si la canción transmite un mensaje negativo o un mensaje positivo para los niños, una vez realizada la clasificación de la letra colocan en el archivo Excel su respuesta con un número identificador. Este número está relacionado de la siguiente manera:

- 1 para la clase “Positiva”.
- 0 para la clase “Negativa”.

Para la clasificación de las letras se dio un tiempo de 15 días, pasado este tiempo se tomaron las clasificaciones de cada miembro del jurado y se juntaron en un archivo Excel para realizar el conteo de positivos y negativos, por medio de condiciones de Excel.

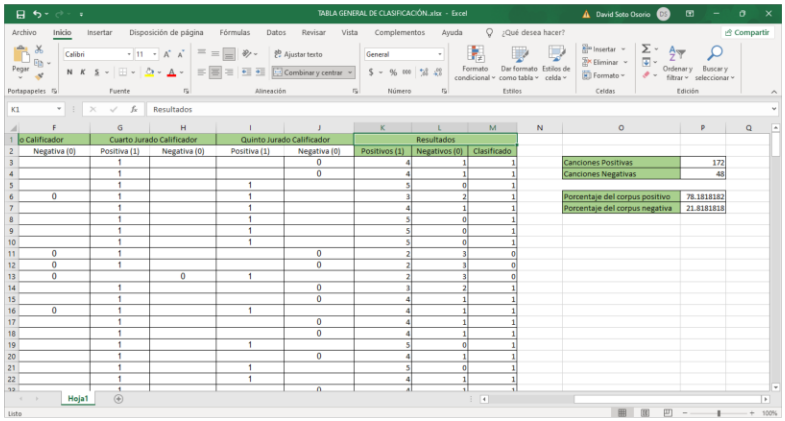
Primero se realiza el conteo de positivos y negativos en cada canción, como máximo puede haber 5 positivos o 5 negativos. Una vez hecho el conteo se determina si hay más negativos o más positivos, en caso de ser más positivos la canción se va a clasificar como positiva, si hay más negativos la canción se va a clasificar como negativa.

A partir de esta métrica, se obtuvo la generación del corpus de las letras de las canciones con las siguientes medidas por clase:

- “Positiva”: 172 letras.
- “Negativa”: 48 letras.

En porcentajes del 100% el 78.18% del corpus está conformado por letras de canciones positivas y el 21.82 está conformado por letras de canciones negativas.

En la Figura 4.2 se presenta el archivo Excel que se utiliza para realizar el conteo de clases positivas y negativas.



o Calificador	Cuarto Jurado Calificador		Quinto Jurado Calificador		Resultados		
Negativa (0)	Positiva (1)	Negativa (0)	Positiva (1)	Negativa (0)	Positivos (1)	Negativos (0)	Clasificado
	1			0	4	1	1
	1			0	4	1	1
	1		1		5	0	1
0	1		1		3	2	1
	1		1		4	1	1
	1		1		5	0	1
	1		1		5	0	1
	1		1		5	0	1
0	1			0	2	3	0
0	1			0	2	3	0
0		0	1		2	3	0
	1			0	3	2	1
	1			0	4	1	1
0	1		1		4	1	1
	1			0	4	1	1
	1			0	4	1	1
	1		1		5	0	1
	1			0	4	1	1
	1		1		5	0	1
	1		1		4	1	1

Canciones Positivas	172
Canciones Negativas	48
Porcentaje del corpus positivo	78.1818182
Porcentaje del corpus negativa	21.8181818

Figura 4.2: Tabla general de clasificación

Con todas las letras etiquetadas se pasa a colocar el número de clase a cada bloc de notas que almacena cada una de las letras. En la próxima etapa se realiza un preprocesamiento de las letras aplicando técnicas como la tokenización, eliminación de stopwords y lematización.

4.3 Etapa 3: Tokenización, eliminación de stopwords y lematización

En esta tercera etapa se explica el preprocesamiento que se hace del corpus para poder ingresarlo al algoritmo de clasificación propuesto.

4.3.1 Fase 1: Llamado a las librerías

En esta primera fase se llaman a las librerías que se usan para realizar el preprocesamiento de los datos. Para llevar a cabo el preprocesamiento de los datos se hace un llamado a las librerías *NLTK*, *Spacy*, *Pandas* y *Numpy*.

Al momento de cargar la librería Spacy se debe seleccionar el diccionario con el que se va a trabajar, esto se debe a que el paquete Spacy cuenta con diferentes diccionarios en diferentes idiomas. Este proceso también se aplica al paquete NLTK para la elección de las stopwords , en ambas librerías se deben seleccionar

los diccionarios en español.

Aparte de los paquetes antes mencionados, también se declara la librería *os*, este paquete sirve para extraer información de diferentes formatos de archivos, en el caso de este trabajo se usa para la extracción de las letras de los blocs de notas con extensión .txt.

En la figura 4.3 se muestra el segmento de código que se usa para hacer el llamado de las librerías para el preprocesamiento de los datos.

```
import spacy  # declaración de la librería Spacy.
import es_core_news_sm
from sklearn.feature_extraction.text import CountVectorizer #importación de la función CountVectorizer.
nlp = spacy.load('es_core_news_sm') # Cargar un modelo mediano de idioma español.
from sklearn.model_selection import train_test_split # Importación para el metodo de validación simple.
from nltk.tokenize import sent_tokenize,word_tokenize # Importación de las funciones sent_tokenize y word_tokenize.
from nltk.corpus import stopwords # Importación de las stopwords de la librería NLTK.
from spacy.lang.es.stop_words import STOP_WORDS #importación de las stopwords de la librería spacy.
import numpy as np # Importación de la librería numpy.
import pandas as pd #Importación de la librería pandas.
import os # Importación de la librería os.
```

Figura 4.3: Llamado a las librerías

4.3.2 Fase 2: Búsqueda de directorio que almacena los archivos

En esta fase se selecciona la carpeta en donde se tienen almacenados los archivos .txt, y se almacena en una variable de tipo string, con esta variable es posible abrir el nombre de todos los archivos .txt que se encuentran en esta ruta, para realizar este proceso se usa la función *os.listdir()*, esta función retorna un dato tipo lista que es almacenado en otra variable, al imprimir la variable se observan los nombres de los archivos .txt de la carpeta seleccionada.

En la Figura 4.4 se presenta la lista de nombres de archivos .txt que se extraen de la carpeta.

4.3.3 Fase 3: Inserción de letras en tabla.

En esta fase primero se declaran dos tablas por medio de la librería Pandas, estas tablas son creadas con el propósito de almacenar la clase de la letra y también para almacenar las letras de las canciones por lo que serán tablas de 220 filas y 2 columnas, en la primera columna se almacena la clase de la letra y en la segunda columna se almacena la letra.

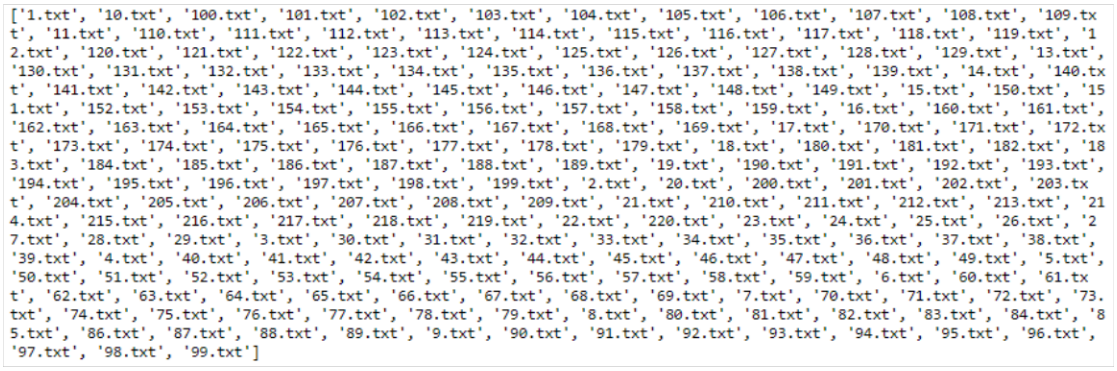


Figura 4.4: Lista de archivos .txt

Una vez que se tiene la letra almacenada se usa la función de pandas *.append* para agregar la letra a la tabla de canciones. Al realizar este procedimiento se utiliza otra variable para almacenar las clases que también serán guardadas en la tabla de pandas.

En la Figura 4.5 se muestra un diagrama de primer orden que presenta el procedimiento de esta fase.

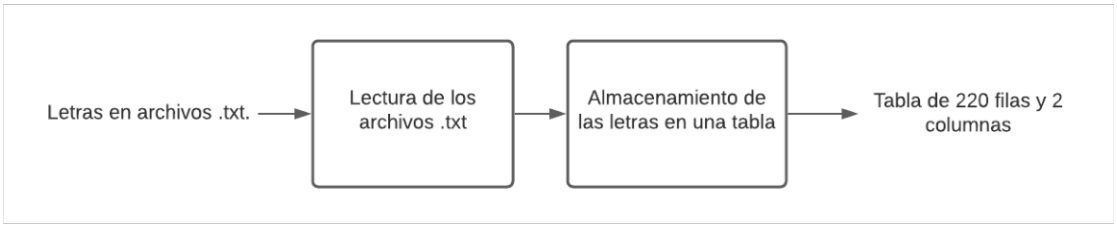


Figura 4.5: Inserción de letras en tabla

4.3.4 Fase 4: Eliminación de símbolos no necesarios para el análisis del texto y conversión de mayúsculas a minúsculas.

En esta fase se hizo una eliminación de todos aquellos caracteres que no aportan ningún contexto o significado a la letra, todos estos símbolos se almacenan en una lista, por medio de ciclos repetitivos se va comparando cada carácter de la letra con los símbolos, si ambos son iguales el carácter es sustituido por un espacio en blanco, este procedimiento se realiza con las 220 letras.

Con los símbolos no deseados eliminados se procede a convertir todas las palabras de las letras a minúsculas, este procedimiento se realiza porque para una computadora son diferentes las mayúsculas a las minúsculas.

En la Figura 4.6 se presenta un diagrama de primer orden que muestra la eliminación de símbolos y la conversión de mayúsculas a minúsculas.

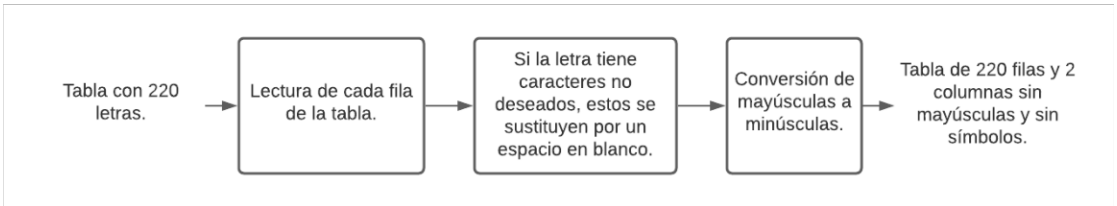


Figura 4.6: Eliminación de caracteres no necesarios

4.3.5 Fase 5: Almacenamiento de stopwords de la librería Spacy y NLTK

Una vez que todos los datos están en letras minúsculas se realiza el llamado a las palabras stopwords de la librería NLTK y Spacy, estas listas de palabras se almacenan en variables diferentes.

En la Figura 4.7 se puede observar un diagrama de primer orden que presenta el proceso para almacenar las stopwords.

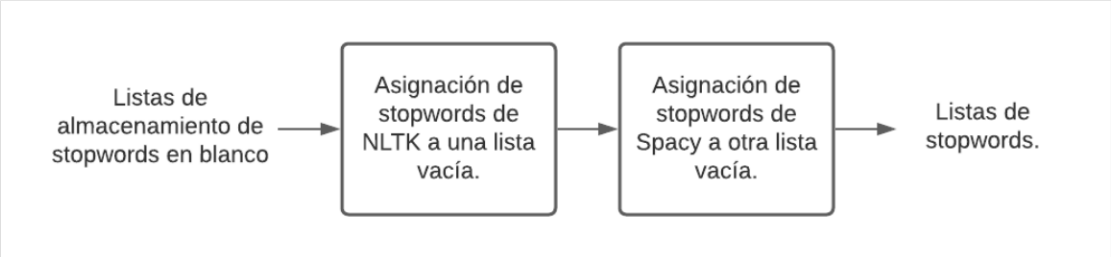


Figura 4.7: Almacenamiento de stopwords

4.3.6 Fase 6: Tokenización y eliminación de stopwords.

En esta fase se realiza la tokenización y eliminación de las stopwords, primero por medio de la función *word tokenize* se realiza la tokenización por palabras, como resultado de este procedimiento se tiene una lista de tokens de cada una de las

palabras de la letra de la canción, posteriormente se realiza la eliminación de las stopwords haciendo una comparación de cada token con la lista de stopwords, si una palabra se encuentra en ambas listas es eliminada de la lista que almacena los tokens de la letra.

En la Figura 4.8 se presenta un diagrama de primer orden que muestra el procedimiento de la tokenización y la eliminación de las stopwords. Todos los elementos que se procesan son almacenados en una tabla de pandas.

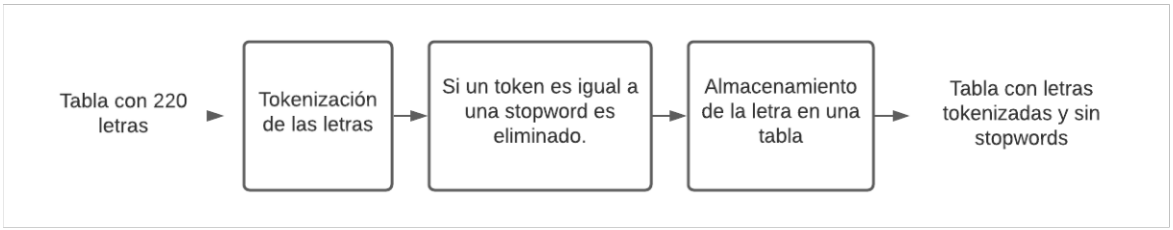


Figura 4.8: Tokenización y eliminación de stopwords

4.3.7 Fase 7: Lematización.

En esta fase una vez que se tiene la letra tokenizada y se eliminaron las stopwords se realiza la lematización de las palabras, para realizar la lematización se usó la librería spacy que por medio de *.lemma*, nos permite convertir a la palabra seleccionada a su forma base.

Al momento de realizar la lematización se hizo una segunda eliminación de caracteres innecesarios que son producto de la conversión de listas a arreglos string, para hacer este proceso se usa un condicional con los caracteres a eliminar, mientras que el dato sea diferente podrá entrar al condicional para ser almacenado.

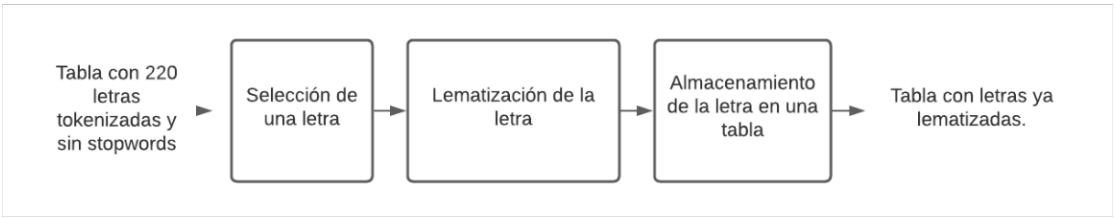


Figura 4.9: Lematización de letras

4.4 Etapa 4: Vectorización y separación del corpus.

4.4.1 Fase 1: Vectorización de las letras.

Para realizar la vectorización de las letras se usa la función *CountVectorizer()*, esta función permite convertir un texto en un vector de palabras, este vector de palabras está conformado por todas las palabras que aparecen en nuestro corpus, cada palabra cuenta con un número de identificación.

Una vez que se tienen a las palabras del corpus con un número identificador se usa la función *.transform()* para realizar el conteo del número de veces que una palabra aparece en una letra.

4.4.2 Fase 2: Separación en conjunto de entrenamiento y prueba.

En esta fase se realiza la separación del corpus lingüístico que anteriormente fue procesado y vectorizado, para esto se realiza una validación simple para hacer la separación en dos conjuntos, un conjunto de prueba y un conjunto de entrenamiento, esta separación se realiza de forma aleatoria, el conjunto de entrenamiento está formado por el 70% del corpus mientras que el conjunto de entrenamiento está conformado por el 30%.

Para realizar la separación se usa la función *train test Split()* el cual retorna dos parámetros que son los conjunto de entrenamiento y prueba, cada uno con sus respectivas clases. Tras que el corpus está formado por 220 letras, el conjunto de entrenamiento queda conformado por 154 letras y el conjunto de prueba queda formado por 66 letras. Todos estos elementos quedan almacenados en dos tablas, la primera almacena las letras de entrenamiento y la segunda almacena las letras de prueba, cabe recordar que cada una de estas letras lleva la clase a la que pertenece por lo que cada tabla debe ser de 2 columnas.

4.5 Etapa 5: Algoritmo de clasificación

Con el corpus ya separado en conjunto de entrenamiento y conjunto de prueba se usa la librería Scikit-learn para el algoritmo de clasificación, el algoritmo que se usa en este trabajo se llama K vecinos más cercanos, primero se usa el conjunto de entrenamiento para entrenar al algoritmo, para este algoritmo consideramos los 7 vecinos más cercanos, una vez que se entrena el clasificador se usa un ciclo

repetitivo que recorre cada uno de los elementos del conjunto de prueba, por cada iteración del ciclo repetitivo se realiza la clasificación de un elemento del conjunto de prueba, una vez clasificado el dato de prueba se imprime la clase a la que fue clasificado y se realiza una comparación con la clase original de la canción, si ambas clases son iguales se imprime un texto que dice que la letra de la canción se clasificó correctamente, si las clases son diferentes se imprime un texto que dice que la letra de la canción no se clasificó correctamente.

Una vez ejecutadas todas las iteraciones se imprime el porcentaje de precisión que tuvo el algoritmo de k vecinos más cercanos y también se imprime el número de letras de canciones clasificadas correctamente y el número de letras clasificadas incorrectamente. En la Figura 4.12 se presenta un diagrama de primer orden que muestra los pasos para la clasificación.

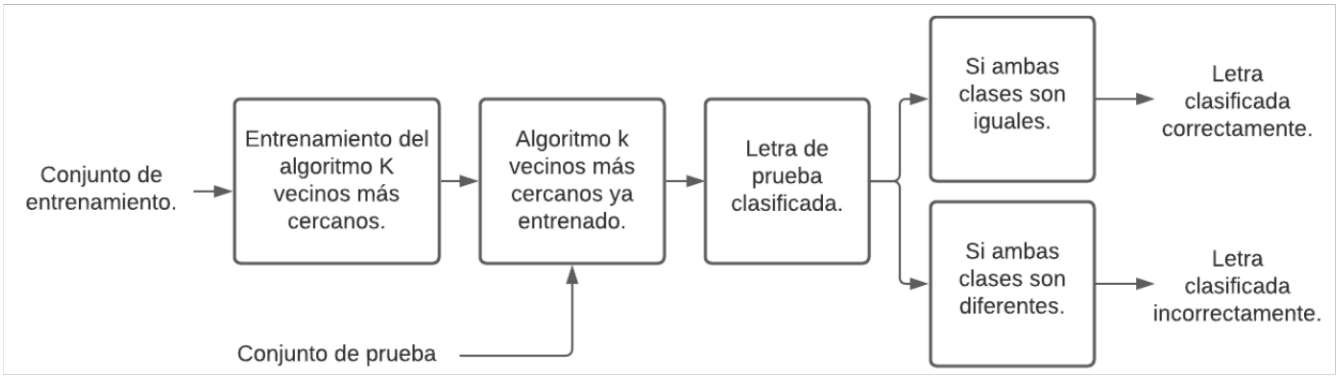


Figura 4.10: Algoritmo de clasificación K vecinos más cercanos.

Capítulo 5

Resultados Experimentales

5.1 Condiciones iniciales

Por un lado, se presentan las características del hardware que se usó para el desarrollo, prueba y experimentos del algoritmo:

- Laptop HP 15-bs0xx.
- Memoria RAM de 8 GB.
- Disco duro de 250 GB.
- Procesador Intel Core i7 de 2 núcleos
- Tarjeta gráfica AMD Radeon.
- Windows 10 64 bits.

Por otro lado, se presentan las características de los software que se utilizan para el desarrollo del proyecto:

- Python versión 3.7.8.
- Anaconda 3
- Jupyter Notebook.
- Microsoft Office 2016.

5.2 Prueba de funcionamiento

Una prueba de funcionamiento se usa para demostrar el correcto funcionamiento del algoritmo, en esta prueba de funcionamiento la variable objetivo es el porcentaje de precisión que tiene el algoritmo para realizar la clasificación de canciones infantiles en positivas y negativas.

Para el desarrollo de la prueba de funcionamiento se usa un corpus lingüístico de 220 letras de las cuales el 70% se utiliza para el entrenamiento y el 30% para las pruebas.

En esta prueba de funcionamiento se va a verificar que los algoritmos de preprocesamiento del corpus trabajen correctamente. Las pruebas de funcionamiento que se efectúan son las siguientes:

- Prueba de funcionamiento del algoritmo de tokenización y eliminación de stopwords.
- Prueba de funcionamiento del algoritmo de lematización.
- Prueba de funcionamiento del método de validación simple.
- Prueba de funcionamiento del algoritmo de clasificación k vecinos más cercanos.

Como primer punto se realizan pruebas del algoritmo de tokenización. Como se observa en la Figura 5.1 el algoritmo de tokenización hace una correcta separación de las letras en tokens de palabras y también realiza la eliminación de las stopwords, para la eliminación se usaron dos diccionarios de la librería NLTK y Spacy.

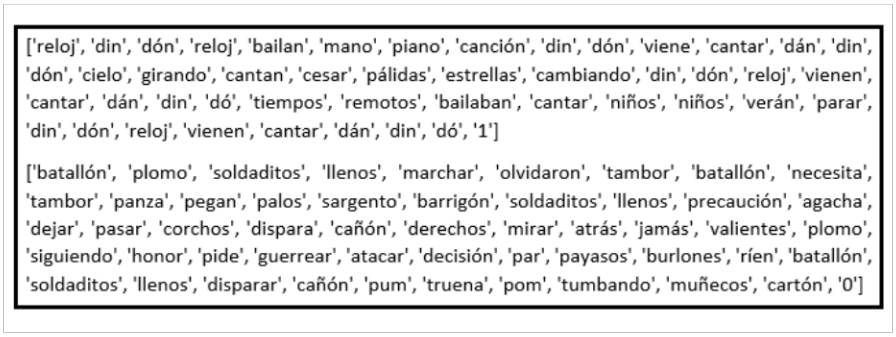
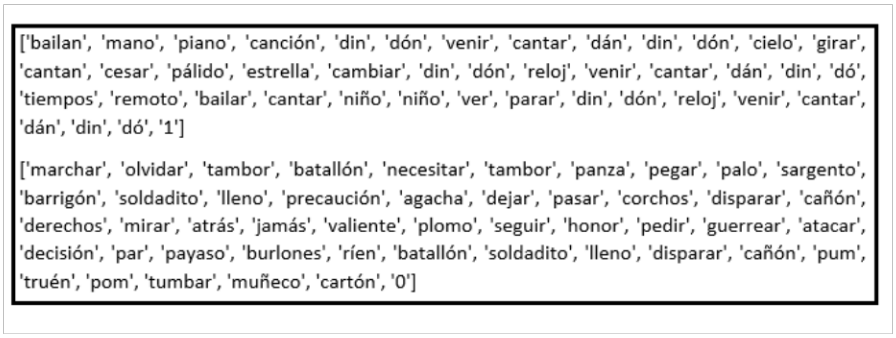


Figura 5.1: Tokenización y eliminación de stopwords en letras

En el segundo punto se va a verificar que el algoritmo de lematización funcione correctamente, en este punto se tiene que verificar que el algoritmo convierta las palabras a su forma base. En la Figura 5.2 se muestra la letra de una canción lematizada aplicando la lematización de la librería Spacy.

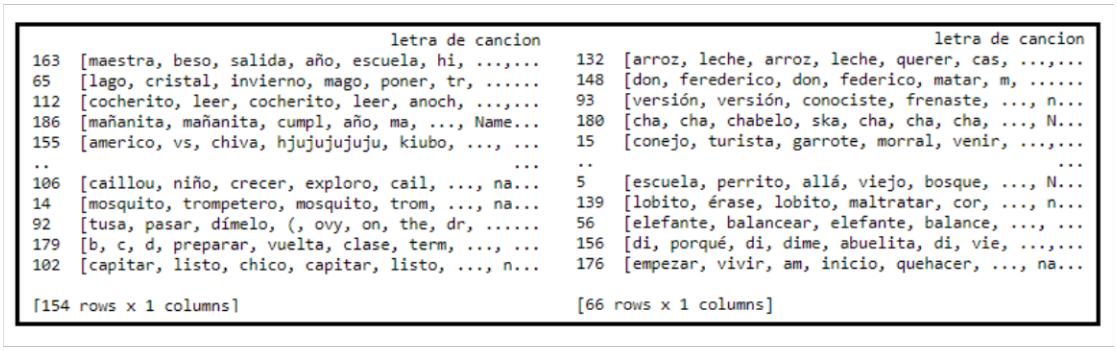


['bailan', 'mano', 'piano', 'canción', 'din', 'dón', 'venir', 'cantar', 'dán', 'din', 'dón', 'cielo', 'girar', 'cantan', 'cesar', 'pálido', 'estrella', 'cambiar', 'din', 'dón', 'reloj', 'venir', 'cantar', 'dán', 'din', 'dó', 'tiempos', 'remoto', 'bailar', 'cantar', 'niño', 'niño', 'ver', 'parar', 'din', 'dón', 'reloj', 'venir', 'cantar', 'dán', 'din', 'dó', '1']

['marchar', 'olvidar', 'tambor', 'batallón', 'necesitar', 'tambor', 'panza', 'pegar', 'palo', 'sargento', 'barrigón', 'soldadito', 'lleno', 'precaución', 'agacha', 'dejar', 'pasar', 'corchos', 'disparar', 'cañón', 'derechos', 'mirar', 'atrás', 'jamás', 'valiente', 'plomo', 'seguir', 'honor', 'pedir', 'guerrear', 'atacar', 'decisión', 'par', 'payaso', 'burlones', 'ríen', 'batallón', 'soldadito', 'lleno', 'disparar', 'cañón', 'pum', 'truén', 'pom', 'tumbar', 'muñeco', 'cartón', '0']

Figura 5.2: Resultados de la lematización

Para el tercer punto se revisa el funcionamiento de la validación simple, en esta validación se usó un 70% para el conjunto de entrenamiento y un 30% para el conjunto de prueba, en la Figura 5.3 se muestra el conjunto de entrenamiento y el de prueba.



	letra de cancion
163	[maestra, beso, salida, año, escuela, hi, ...,...
65	[lago, cristal, invierno, mago, poner, tr, ...
112	[cocherito, leer, cocherito, leer, anoch, ...,...
186	[mañanita, mañanita, cumpl, año, ma, ..., Name...
155	[americo, vs, chiva, hjujujujuju, kiubo, ..., ...
..	...
106	[caillou, niño, crecer, exploro, cail, ..., na...
14	[mosquito, trompetero, mosquito, trom, ..., na...
92	[tusa, pasar, dímelo, (, ovy, on, the, dr, ...
179	[b, c, d, preparar, vuelta, clase, term, ..., ...
102	[capitar, listo, chico, capitar, listo, ..., n...

[154 rows x 1 columns]

	letra de cancion
132	[arroz, leche, arroz, leche, querer, cas, ...,...
148	[don, frederico, don, federico, matar, m, ...
93	[versión, versión, conociste, frenaste, ..., n...
180	[cha, cha, chabelo, ska, cha, cha, cha, ..., N...
15	[conejo, turista, garrote, morral, venir, ...,...
..	...
5	[escuela, perrito, allá, viejo, bosque, ..., N...
139	[lobito, érase, lobito, maltratar, cor, ..., n...
56	[elefante, balancear, elefante, balance, ..., ...
156	[di, porqué, di, dime, abuelita, di, vie, ...,...
176	[empezar, vivir, am, inicio, quehacer, ..., na...

[66 rows x 1 columns]

Figura 5.3: Separación en conjunto de entrenamiento y prueba

Como último punto se prueba el algoritmo de clasificación k vecinos más cercanos se usó un valor K de 7, este K señala el número de vecinos que se van a considerar para clasificar una canción, de preferencia se tiene que considerar un número impar para evitar empates, en la Figura 5.4 se muestran las canciones del conjunto de prueba con un número de clasificación asignado por el algoritmo, esta clasificación se basa en el conjunto de entrenamiento, como resultado se tiene que el algoritmo clasifica correctamente 56 letras mientras que no clasifica correctamente

10 letras, con base en estos resultados el porcentaje de precisión del algoritmo es del 84%.

```
Cancion de prueba numero 55 : clasificacion correcta
Cancion de prueba numero 56 : clasificacion correcta
Cancion de prueba numero 57 : clasificacion correcta
Cancion de prueba numero 58 : clasificacion correcta
Cancion de prueba numero 59 : clasificacion incorrecta
Cancion de prueba numero 60 : clasificacion correcta
Cancion de prueba numero 61 : clasificacion correcta
Cancion de prueba numero 62 : clasificacion correcta
Cancion de prueba numero 63 : clasificacion correcta
Cancion de prueba numero 64 : clasificacion correcta
Cancion de prueba numero 65 : clasificacion correcta
Cancion de prueba numero 66 : clasificacion correcta
total de canciones clasificadas: 66
total acertados: 56
canciones fallidas: 10
porcentaje de precision: 0.8484848484848485
```

Figura 5.4: Canciones clasificadas por el algoritmo K vecinos más cercanos.

5.3 Experimento

Se diseña una serie de experimentos para el algoritmo de clasificación de canciones infantiles con la intención de ver los cambios en el porcentaje de precisión considerando las siguientes variables externas que es posible que afecten al sistema.

- Errores ortográficos en las letras.
- Palabras en inglés.
- Mayúsculas en las letras.

En la Tabla 5.1 se presentan los factores que se usan para realizar los experimentos al igual que sus valores alto y bajo.

Tabla 5.1: En esta tabla se presentan las variables o factores con los que se realizan los experimentos para el análisis de la varianza con el propósito de determinar que factores afectan al sistema.

Factor	Bajo	Alto
Errores ortográficos	Desactivado	Activado
Palabras en inglés	Desactivado	Activado
Mayúsculas en las letras	Desactivado	Activado

Se plantea el siguiente experimento basado en un análisis de varianza (ANOVA, Analysis of Variance) para determinar cuál es la variable que llega a tener una mayor influencia sobre la respuesta del algoritmo. Las muestras que se utilizaran para los experimentos se componen de *letras de canciones infantiles con errores ortográficos, palabras en inglés y letras mayúsculas.*

Con base en el número de factores que se consideran para los experimentos del algoritmo se puede determinar el número de pruebas que se realizan, cada uno de los factores puede tomar un valor de bajo o alto, por lo que el número de combinaciones se calcula como dos elevado al número de factores que tengamos para los experimentos, en este caso contamos con tres factores por lo que se pueden realizar ocho pruebas.

El número de combinaciones para el experimento es de ocho pruebas y cuatro bloques de pruebas por lo que el número total de pruebas que se realizan son 32. En la Tabla 5.2 se presentan todas las posibles combinaciones de estos 3 factores.

Tabla 5.2: En esta tabla se muestran todas las pruebas que se realizan para el algoritmo al igual que se presenta el porcentaje de precisión que se obtiene en cada uno de los experimentos.

Bloque	Ortográfia	Palabras en inglés	Mayúsculas	Porcentaje de precisión
1	0	0	0	84.4%
1	0	0	1	83.2%
1	0	1	0	72.1%
1	0	1	1	71.4%
1	1	0	0	82.5%
1	1	0	1	74.3%
1	1	1	0	68.4%
1	1	1	1	61.2%
2	0	0	0	84.1%
2	0	0	1	83.3%
2	0	1	0	73.2%
2	0	1	1	72.5%
2	1	0	0	81.8%
2	1	0	1	75.2%
2	1	1	0	67.2%
2	1	1	1	60.5%
3	0	0	0	84.2%
3	0	0	1	83.1%
3	0	1	0	74.5%
3	0	1	1	73.1%
3	1	0	0	81.5%
3	1	0	1	74.8%
3	1	1	0	67.1%
3	1	1	1	61.6%
4	0	0	0	83.6%
4	0	0	1	81.8%
4	0	1	0	72.8%
4	0	1	1	71.2%
4	1	0	0	82.4%
4	1	0	1	73.6%
4	1	1	0	66.2%
4	1	1	1	60.2%

Por un lado, el porcentaje máximo de precisión que se obtiene en los experimentos fue del 84%, este valor se obtiene sin considerar los factores externos como la ortografía, palabras en inglés y mayúsculas, en cambio al realizar los experimen-

tos considerando el factor de las letras mayúsculas el porcentaje de precisión bajo muy poco y cuando se considera el factor de palabras en inglés el porcentaje de precisión bajo considerablemente, esto se debe a que todos los diccionarios que se utilizan se encuentran en español.

El gráfico de la Figura 5.5 representa la relación significativa o no significativa para el experimento de las 3 variables a considerar, una distancia corta entre los valores 0 y 1 de los 3 factores indica una relación no significativa y la distancia mayor entre los dos valores de las 3 variables significa una relación significativa. En este caso se puede observar que el factor de palabras en inglés es la variable que llega a afectar más al porcentaje de precisión del clasificador, en cambio los otros 2 factores restantes también afectan al clasificador pero en menor medida.

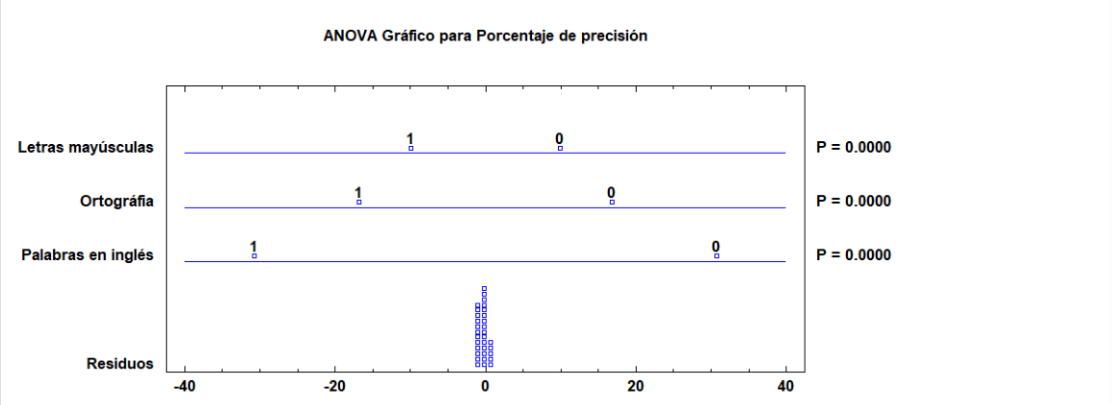


Figura 5.5: Gráfica de factores que afectan al porcentaje de precisión.

La tabla de la Figura 5.6, ANOVA descompone la variabilidad del Porcentaje de precisión en contribuciones debido a varios factores. Puesto que se ha escogido la suma de cuadrados Tipo III (por omisión), la contribución de cada factor se mide eliminando los efectos de los demás factores. Los valores-P prueban la significancia estadística de cada uno de los factores. Puesto que 5 valores-P son menores que 0.05, estos factores tienen un efecto estadísticamente significativo sobre el porcentaje de precisión con un 95.0% de nivel de confianza. Como se puede observar en la tabla la interacción entre los factores de ortografía y palabras en inglés son los que llega afectar más al sistema.

En conclusión, todos los factores afectan al sistema, unos más que otros, ya que al tratarse de un algoritmo de PLN si la letra tiene faltas de ortografía el algoritmo la considera como una palabra completamente diferente y en este caso no será posi-

Análisis de Varianza para Porcentaje de precisión - Suma de Cuadrados Tipo III

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
EFECTOS PRINCIPALES					
A:Palabras en inglés	1257.51	1	1257.51	2298.57	0.0000
B:Ortografía	378.125	1	378.125	691.17	0.0000
C:Letras mayúsculas	132.031	1	132.031	241.34	0.0000
INTERACCIONES					
AB	22.445	1	22.445	41.03	0.0000
AC	0.91125	1	0.91125	1.67	0.2091
BC	67.28	1	67.28	122.98	0.0000
ABC	0.605	1	0.605	1.11	0.3035
RESIDUOS	13.13	24	0.547083		
TOTAL (CORREGIDO)	1872.04	31			

Todas las razones-F se basan en el cuadrado medio del error residual

Figura 5.6: Tabla de análisis de varianza para porcentaje de precisión

ble lematizarlo lo que provocara un mayor tiempo de procesamiento y consumo de recursos computacionales aparte de que el porcentaje de precisión será disminuido.

En el caso de las palabras en inglés, este es el factor que llega afectar más al sistema de clasificación porque el algoritmo no está diseñado para eliminar palabras en inglés y tampoco para lematizarlas, esto se debe a que todos los diccionarios que se utilizan para el preprocesamiento son en español.

5.4 Evaluación económica.

En esta sección se realiza una evaluación económica para determinar si el proyecto es viable, primero se realiza una inversión inicial de $17,000, la inversión inicial abarca el equipo de cómputo que se adquirió para realizar el algoritmo de clasificación, el PC utilizado tiene las siguientes características:

- Laptop HP 15-bs0xx.
- Memoria Ram de 8 GB.
- Disco duro de 250 GB.
- Procesador Intel Core i7 de 2 núcleos
- Tarjeta gráfica AMD Radeon.
- Windows 10.

En la Tabla 5.3 se presenta el gasto que se realiza en los servicios de internet, luz, plan de telefonía, renta y agua, de forma mensual se tiene un gasto total de $10,425.

Tabla 5.3: En esta tabla se presentan todos los gastos que se realizan en pagos de servicios de forma mensual.

Descripción	Por semana	Mensual
Luz	$120	$480
Internet	$105	$420
Celular	$37.5	$150
Renta	$1250	$5,000
Agua	$93	$375
Seguro	$190	$760

Debido a que este trabajo consiste en el desarrollo de un algoritmo de software se debe considerar el sueldo del programador, el salario promedio al año de un programador junior de Python es de $210,000 pesos al año, de forma mensual aproximadamente $17,500 pesos, esta cantidad es antes de impuestos, debido a que el salario se encuentra en un rango de $10,298.36 y $20,770.29 se paga el 21.36% de impuestos por lo que el salario promedio es de $13,762 pesos.

También para el desarrollo de ese proyecto se requiere del trabajo de profesionales en el campo de la psicología para la clasificación del corpus, esto se requiere en el cuarto mes de desarrollo, de esta consulta se hizo un gasto de $5,000 pesos.

Como valor de salvamento se considera vender el equipo de cómputo que se utiliza para el desarrollo el proyecto, este equipo se devaluó un 40% por haber sido usado con fines de programación, de $17,000 pesos se va a vender a un precio de $10,200 pesos.

Con los datos presentados se realiza una evaluación económica para el proyecto análisis de canciones infantiles aplicando algoritmo de inteligencia artificial, para realizar la evaluación se determina el Valor Presente Neto (VPN) y la Tasa Interna de Retorno (TIR) para determinar su viabilidad.

En el diagrama de la Figura 5.7 se muestra el flujo de efectivo que se tiene durante el desarrollo del proyecto, todas aquellas flechas que apunten hacia arriba son consideradas como ingresos mientras que las fechas que apuntan hacia abajo son egresos, este diagrama representa los 12 meses de duración del proyecto.

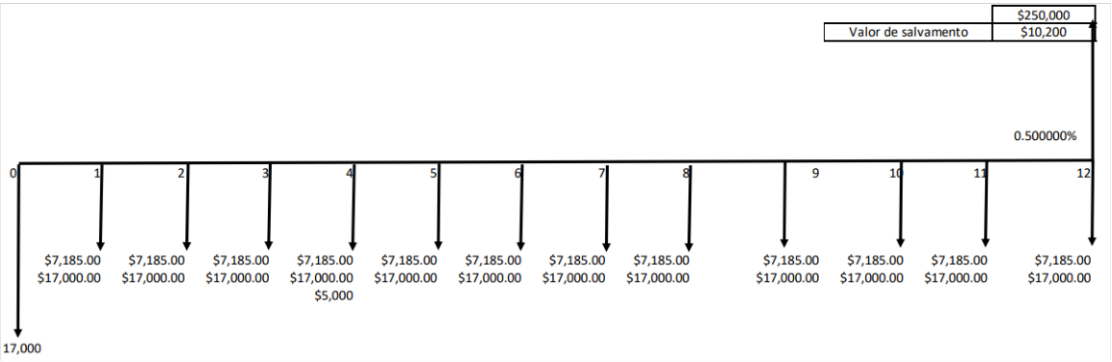


Figura 5.7: Diagrama de flujo de ingresos y egresos mensuales durante un periodo de 1 año

Este proyecto requiere de una inversión inicial de $17,000 pesos con un gasto mensual aproximado de $7,185 pesos por pago de servicios básicos y $17,000 pesos del sueldo del programador, también en el cuarto mes se hace un gasto de $5,000 pesos por el pago al jurado calificador.

Para poder determinar el VPN se usa la fórmula presentada en la Figura 5.8, para el cálculo se considera una tasa de intereses anual de 5% por lo que se considera una tasa mensual del 0.40%.

$$VPN = \frac{-Inversión}{(1+i)^0} + \frac{\sum I_1 - \sum E_1}{(1+i)^1} + \frac{\sum I_2 - \sum E_2}{(1+i)^2} + \frac{\sum I_3 - \sum E_3}{(1+i)^3} + \cdots + \frac{\sum I_{n-1} - \sum E_{n-1}}{(1+i)^{n-1}} + \frac{\sum I_n - \sum E_n}{(1+i)^n}$$

Figura 5.8: Fórmula para calcular el Valor Promedio Neto (VPN)

La VPN de un periodo de tiempo es la suma de las diferencias de ingresos y egresos divididos entre la tasa de interés anual más uno elevado al número del periodo calculado. El VPN de este proyecto es de -52,920.10, esto quiere decir que si se vende el programa al precio que se está proponiendo no se verá ninguna ganancia y habrá perdida por lo que se puede decir que el proyecto no es viable, sin embargo, si se venden varias unidades de este programa se podrá reponer la inversión al segundo programa vendido, si se venden dos programas el VPN cambia a $182,556.23 por lo que se puede decir que el proyecto es viable.

Si el VPN es negativo no es necesario determinar la TIR, ya que por el VPN se puede concluir que no hay una Tasa Interna de Retorno en la que no existan perdidas, la TIR se determina sustituyendo un valor igual a cero como tasa de descuento i para la fórmula del VPN.

Para encontrar el VPN positivo se deben considerar ingresos de venta del proyecto, por lo que es necesario desarrollar una estrategia de venta en donde se estudie un modelo de negocio que pueda hacer al algoritmo viable.

Capítulo 6

Conclusiones

6.1 Conclusiones Finales

Se desarrolló una investigación en donde se encontraron 5 trabajos relacionados con los algoritmos de clasificación de letras musicales, se realizó una comparación entre todos los trabajos y se pudo determinar que las diferencias de los trabajos con la propuesta de este escrito es el idioma al igual que el género de música que se analizó.

Se integraron conocimientos del Procesamiento de Lenguaje Natural e Inteligencia Artificial para conocer las técnicas más utilizadas para el preprocesamiento de las letras, se logró determinar que la tokenización, lematización y eliminación de stopwords son buenas opciones para realizar el preprocesamiento del corpus, también se concluyó que el algoritmo de K vecinos más cercanos es una buena opción para realizar la clasificación de las letras.

Se presentaron los procesos de desarrollo por medio de etapas y fases para describir la metodología que se llevó a cabo para el desarrollo del algoritmo, esta metodología abarca desde la captura de la información para el desarrollo del corpus hasta la aplicación del algoritmo de clasificación k vecinos más cercanos.

Para verificar el funcionamiento del algoritmo de clasificación se realizó una prueba de funcionamiento y se diseñó un experimento de treinta y dos pruebas para determinar los factores que logran afectar al porcentaje de precisión del algoritmo, se concluyó que todos los factores afectan al algoritmo de clasificación, pero sobre todo las palabras en inglés reducen el porcentaje de precisión considerablemente.

6.2 Trabajo Futuro

En este apartado se mencionan algunas ideas que se pueden aplicar para mejorar el sistema desarrollado en este proyecto, estas ideas son las siguientes:

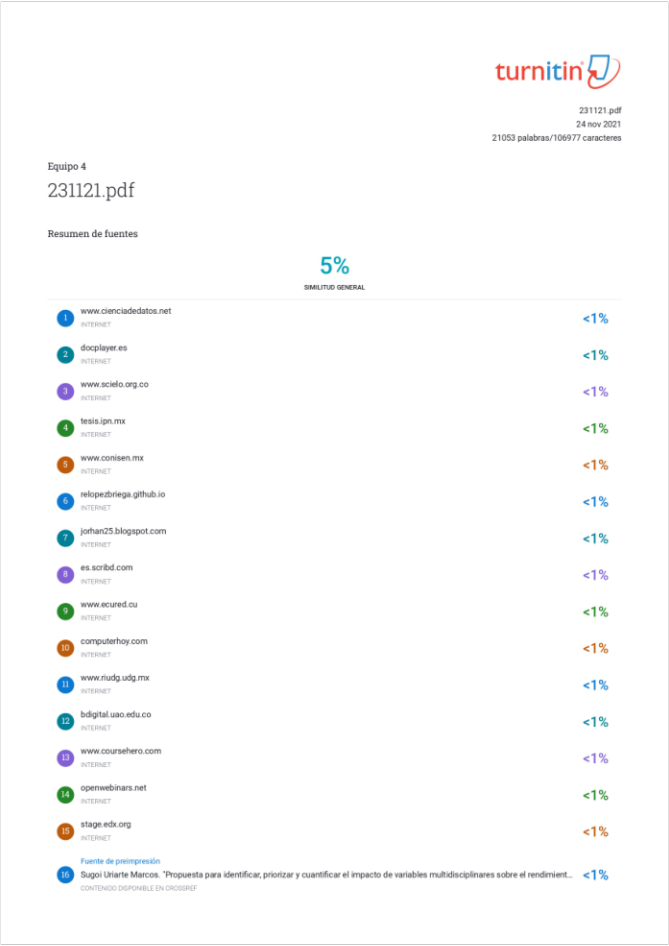
- Aumento del número de letras en el corpus: En este punto se plantea ampliar el corpus lingüístico para tener una mayor cantidad de letras de canciones infantiles teniendo de esta forma un mayor porcentaje en los conjuntos de entrenamiento y prueba. Se tiene que considerar que se deben agregar más canciones con clase negativa para que haya un equilibrio entre ambas clases dentro del corpus.

- Aplicación de otros algoritmos de clasificación: Se plantea desarrollar otros algoritmos para realizar la clasificación de las letras, algunas opciones podrían ser el algoritmo de similitud coseno o de redes neuronales. Al desarrollar otros algoritmos para la clasificación se pueden realizar los experimentos para determinar cuál algoritmo tiene un mayor porcentaje de precisión a pesar de los factores externos que pueden llegar a perjudicar al sistema.

- Ampliar el proyecto a otro idioma: En este punto se propone realizar un corpus lingüístico más extenso considerando letras de canciones infantiles en inglés, por lo que para estas letras se necesitaran de diccionarios de palabras en inglés para la lematización y las stopwords, para este punto aún se pueden usar las mismas librerías de NLTK y Spacy.

Apéndice A

Reporte de Similitud



turnitin

231121.pdf
24 nov 2021
21053 palabras/106977 caracteres

Equipo 4

231121.pdf

Resumen de fuentes

5%
SIMILITUD GENERAL

1	www.cienciadedatos.net INTERNET	<1%
2	docplayer.es INTERNET	<1%
3	www.scielo.org.co INTERNET	<1%
4	tesis.ipn.mx INTERNET	<1%
5	www.conisen.mx INTERNET	<1%
6	relopezbriega.github.io INTERNET	<1%
7	jorhan25.blogspot.com INTERNET	<1%
8	es.scribd.com INTERNET	<1%
9	www.ecured.cu INTERNET	<1%
10	computerhoy.com INTERNET	<1%
11	www.riudg.udg.mx INTERNET	<1%
12	bdigital.uao.edu.co INTERNET	<1%
13	www.coursehero.com INTERNET	<1%
14	openwebinars.net INTERNET	<1%
15	stage.edx.org INTERNET	<1%
16	Fuente de preimpresión Sugoi Uriarte Marcos. "Propuesta para identificar, priorizar y cuantificar el impacto de variables multidisciplinares sobre el rendimient... CONTENIDO DISPONIBLE EN CROSSREF	<1%

Figura A.1: Reporte de similitud de Turnitin

Apéndice B

Apéndice C

Referencias

[1] Magán Hervás y Gertrudis Barrio. (2017), Influencia de las actividades audiomusicales en la adquisición de la lectoescritura en niños y niñas de cinco años. Revista Electrónica Educare. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=5763950

[2] Anayanci Mazariegos Orantes (2001) La influencia de la Música en nuestros niños. [Online]. Available: https://www.depadresahijos.org/educacion_psicologia/influenciamusica.html

[3] Beatriz Martínez (2020, Marzo) Canciones infantiles de siempre con un nefasto mensaje para los niños. [Online] Available:https://www.guiainfantil.com/educacion/valores/canciones-infantiles-de-siempre-con-un-nefasto-mensaje-para-los-ninos/

[4] TIOBE Quality Indicator (2021) Índice TIOBE de los lenguajes de programación más populares en el año 2021. [Online]. Available:https://www.tiobe.com/tiobe-index/

[5] Oracle. ¿Qué es la inteligencia artificial?. [Online]. Available:https://www.oracle.com/mx/artificial-intelligence/what-is-ai/

[6] Universidad Autónoma de México, Lenguajes de programación. [Online].Available:https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html

[7] Carlos Federico Blodek (2013, Octubre). Historia de la música infantil. [Online]. Available:https://musibloglogia.wordpress.com/2013/10/08/historia-de-la-musica-infantil/

[8] Justin Lestal (2021, Agosto) Historia de los lenguajes de programación.[Online].Available:https://devskiller.com/es/historia-de-los-lenguajes-de-programacion/

[9] Fernando Berzal (2017, Febrero) Breve historia de la inteligencia artificial.[Online].Available:https://www.cesce.es/es/w/asesores-de-pymes/breve-historia-la-inteligencia-artificial-camino-hacia-la-empresa

[10] Jose P. G. Mahedero, Alvaro Martínez, Pedro Cano.Natural Language Processing of Lyrics.Universidad Pompeu Fabra.[Online].Available:https://sci-hub.se/10.1145/1101149.1101255

[11] Braja Gopal Patra, Dipankar Das y Sivaji Bandyopadhyay (2015, Dic) Mood Classification of Hindi Songs based on Lyrics. Department of Computer Science Engineering, Jadavpur University. [Online].Available:http://cdn.iiit.ac.in/cdn/ltrc.iiit.ac.in/icon2015/icon2015_proceedings/PDF/14_rp.pdf

[12] Ashley M. Oudenne, Sarah E. Chasins. Identifying the Emotional Polarity of Song Lyrics through Natural Language Processing. Swarthmore College. [Online] Available:https://www.sccs.swarthmore.edu/users/11/aoudenn1/Ashley_M._Oudenne/Research_files/emotionalPolarity.pdf

[13] Matthew Mulholland, Joanne Quinn. Suicidal Tendencies: The Automatic Classification of Suicidal and Non-Suicidal Lyricists Using NLP. Montclair State University. [Online]Available:https://aclanthology.org/I13-1079.pdf

[14] Chutimet Srinilta, Wisuwat Sunhem, Suchat Tungjitnob, Saruta Thasanthiah, and Supawit Vatathanavaro. (2017). Lyric-based Sentiment Polarity Classification of Thai Songs.[Online] Available: http://www.iaeng.org/publication/IMECS2017/IMECS2017_pp364-368.pdf

[15] Oscar Ramirez Jiménez, (2021) Python a fondo, Domine el lenguaje de programación del presente y del futuro. Edit. Alfaomega.

[16] Python [Online].Available:https://www.python.org/

[17] Joana Jabloski. (2021). Procesamiento de lenguaje natural con el paquete NLTK de Python. [Online] Available:https://realpython.com/nltk-nlp-python/

[18] Pandas. [Online] Available:https://pandas.pydata.org/

[19] Numpy [Online] Available:https://numpy.org/

[20] ScikitLearn [Online] Available:https://scikit-learn.org/

[21] Steven Bird, Ewan Klein y Edward Loper. Procesamiento de lenguaje natural con Python. [Online] Available:https://www.nltk.org/book/

[22] Anaconda [Online] Available:https://www.anaconda.com/

[23] Jupyter Notebook [Online] Available:https://jupyter.org/

[24] Juan Francisco Vallarta Rueda. Health Data Miner. [Online]Available:https://healthdataminer.com/data-mining/aprendizaje-supervisado-y-no-supervisado/#:~:text=El%20aprendizaje%20supervisado%20supone%20que,de%20datos%20no%20etiquetados%20previamente.

[25] Joaquín Amat Rodrigo (2020, Noviembre) Validación de modelos predictivos: Cross-validation, OneLeaveOut, Bootstraping.[Online] Available:https://www.cienciadedatos.net/documentos/30_cross-validation_oneleaveout_bootstrap