INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Laboratorio de Ciencias Cognitivas Computacionales

TESIS

Análisis de texto tóxico
en redes sociales

QUE PARA OBTENER EL GRADO DE
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

Miguel Angel Soto Hernandez

DIRECTORES DE TESIS:

Dr. Francisco Hiram Calvo Castro

Dr. Alexander Gelbukh

CIUDAD DE MÉXICO,DICIEMBRE 2022

SIP-13
REP 2017

INSTITUTO POLITÉCNICO NACIONAL
SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REGISTRO DE TEMA DE TESIS
Y DESIGNACIÓN DE DIRECTOR DE TESIS

Ciudad de México, a 25 de agosto del 2021

El Colegio de Profesores de Posgrado del **Centro de Investigación en Computación** (Unidad Académica) en su Sesión Ordinaria No. 05 celebrada el día 28 del ~~mes~~ mayo de 2021, conoció la solicitud presentada por el (la) alumno (a):

Apellido Paterno:	SOTO	Apellido Materno:	HERNANDEZ	Nombre (s):	MIGUEL ANGEL

Número de registro: A 2 1 0 2 3 8

del Programa Académico de Posgrado: **Maestría en Ciencias de la Computación**

Referente al registro de su tema de tesis; acordando lo siguiente:

1.- Se designa al aspirante el tema de tesis titulado:

"Análisis de texto tóxico en redes sociales"

Objetivo general del trabajo de tesis:

Desarrollar un modelo computacional que a partir de textos escritos por una persona obtenga características que le permitan determinar si el lenguaje utilizado es ofensivo.

2.- Se designa como Directores de Tesis a los profesores:

Director: **Dr. Francisco Hiram Calvo Castro** 2° Director: **Dr. Alexander Gelbukh**

No aplica: ☐

3.- El Trabajo de investigación base para el desarrollo de la tesis será elaborado por el alumno en:

Centro de Investigación en Computación

que cuenta con los recursos e infraestructura necesarios.

4.- El interesado deberá asistir a los seminarios desarrollados en el área de adscripción del trabajo desde la fecha en que se suscribe la presente, hasta la aprobación de la versión completa de la tesis por parte de la Comisión Revisora correspondiente.

Director de Tesis

Dr. Francisco Hiram Calvo Castro

Aspirante

Miguel Angel Soto Hernandez

2° Director de Tesis

Dr. Alexander Gelbukh

Presidente del Colegio

Dr. Marco Antonio Moreno Ibarra

ESTADOS UNIDOS MEXICANOS
INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

SIP-14
REP 2017

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 12:00 horas del día 25 del mes de noviembre del 2022 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Posgrado de: Centro de Investigación en Computación para examinar la tesis titulada:

"Análisis de texto tóxico en redes sociales"

del (la) alumno (a):

Apellido Paterno:	SOTO	Apellido Materno:	HERNANDEZ	Nombre (s):	MIGUEL ANGEL

Número de registro: A 2 1 0 2 3 8

Aspirante del Programa Académico de Posgrado: **Maestría en Ciencias de la Computación**

Una vez que se realizó un análisis de similitud de texto, utilizando el software antiplagio, se encontró que el trabajo de tesis tiene 09 % de similitud. **Se adjunta reporte de software utilizado**.

Después que esta Comisión revisó exhaustivamente el contenido, estructura, intención y ubicación de los textos de la tesis identificados como coincidentes con otros documentos, concluyó que en el presente trabajo **SI** [ ] **NO** [X] **SE CONSTITUYE UN POSIBLE PLAGIO.**

JUSTIFICACIÓN DE LA CONCLUSIÓN: *(Por ejemplo, el % de similitud se localiza en metodologías adecuadamente referidas a fuente original)*
Se repiten fragmentos pequeños de texto o es referenciado

****Es responsabilidad del alumno como autor de la tesis la verificación antiplagio, y del Director o Directores de tesis el análisis del % de similitud para establecer el riesgo o la existencia de un posible plagio.**

Finalmente y posterior a la lectura, revisión individual, así como el análisis e intercambio de opiniones, los miembros de la Comisión manifestaron **APROBAR** [X] **SUSPENDER** [ ] **NO APROBAR** [ ] la tesis por **UNANIMIDAD** [x] o **MAYORÍA** [ ] en virtud de los motivos siguientes:
Cumple con los requisitos para una tesis de maestría

COMISIÓN REVISORA DE TESIS

Dr. Francisco Hiram Calvo Castro
Director de Tesis

Dr. Grigori Sidorov

M. en C. José Eduardo Valdez Rodríguez

Dr. Alexander Gelbukh
2° Director de Tesis

Dr. Luis Manuel Vilches Blázquez

Dra. Olga Kolesnikova

INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Dr. Francisco Hiram Calvo Castro
PRESIDENTE DEL COLEGIO DE PROFESORES

IPN-CIC

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA DE AUTORIZACIÓN DE USO DE OBRA PARA DIFUSIÓN

En la Ciudad de México el día 13 del mes de diciembre del año 2022, el (la) que suscribe Miguel Angel Soto Hernandez alumno(a) del programa Maestría en Ciencias de la Computación con número de registro A210238, adscrito(a) al Centro de Investigación en Computación manifiesta que es autor(a) intelectual del presente trabajo de tesis bajo la dirección de Dr. Francisco Hiram Calvo Castro y Dr. Alexander Gelbukh y cede los derechos del trabajo intitulado "Análisis de texto tóxico en redes sociales", al Instituto Politécnico Nacional, para su difusión con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expresado del autor y/o director(es). Este puede ser obtenido escribiendo a las siguiente(s) dirección(es) de correo msotoh2021@cic.ipn.mx. Si el permiso se otorga, al usuario deberá dar agradecimiento correspondiente y citar la fuente de este.

Miguel Angel Soto Hernandez

Nombre completo y firma autografá del (de la) estudiante

Resumen

Las redes sociales se han convertido en una parte importante de nuestra sociedad y, en particular, de la vida de los jóvenes. Cuando se trata de redes sociales, debemos ser conscientes del poder que tienen. Las redes sociales pueden utilizarse para bien o para mal; pueden ayudar a conectar a las personas convirtiéndose en un lugar donde pueden expresar sus opiniones y compartan sus pensamientos o a separarlas por medio de insultos, amenazas o lenguaje intencionalmente hiriente. Hemos visto una y otra vez cómo el uso de lenguaje tóxico u ofensivo ha arruinado vidas y ha causado un inmenso dolor y sufrimiento. Nadie merece pasar por eso, tenga la edad que tenga. Es relevante que trabajemos para intentar evitar que esto ocurra, y evitar este tipo de comportamiento es un problema al cual hay que hacerle frente.

Esta tesis de maestría propone la revisión de distintos modelos computacionales de aprendizaje automático y aprendizaje profundo, con la finalidad de resolver la problemática de identificar el texto considerado como tóxico, y como parte de la explicabilidad del por qué es tóxico o no, se identificará a que clase es correspondiente. Esta propuesta se divide en dos partes. La primera es crear un ensamble de modelos de aprendizaje automático con la intención de observar que modelos se ajustan mejor a que datos. Y la segunda es ajustar a la problemática algunos de los modelos de aprendizaje profundo del procesamiento del lenguaje natural. Para, al final, realizar una comparativa entre ambos.

Abstract

Social networks have become an important part of our society and, in particular, of young people's lives. When it comes to social media, we need to be aware of the power it has. Social media can be used for good or for bad; it can help connect people by becoming a place where they can express their opinions and share their thoughts, or it can help separate them through insults, threats or intentionally hurtful language. We have seen time and time again how the use of toxic or offensive language has ruined lives and caused immense pain and suffering. No one deserves to go through that, no matter how old they are. It is important that we work to try to prevent this from happening, and avoiding this type of behavior is a problem that needs to be addressed.

This master thesis proposes the review of different computational models of machine learning and deep learning in order to solve the problem of identifying the text considered as toxic, and as part of the explainability of why it is toxic or not, it will be identified to which class it corresponds. This proposal is divided into two parts. The first is to create an ensemble of machine learning models with the intention of observing which models best fit which data. And the second is to fit some of the deep learning models of natural language processing to the problem. In the end, to make a comparison between the two.

Agradecimientos

A CONACyT, por la ayuda económica.

Al CIC, por todo el apoyo durante mi estancia.

A mis asesores, por guiarme en el camino
y por todo el apoyo que me brindaron.

A CMS, por apoyarme a aterrizar las
ideas de este trabajo.

Dedicatorias

A mis padres, por el apoyo incondicional.

A PPC, por todo el amor,
por siempre estar ahí cuando lo
necesité y por alentarme a seguir
adelante en este proyecto.

Contenido

Resumen **i**

Abstract **ii**

Agradecimientos **iii**

Dedicatorias **iv**

Lista de Tablas **viii**

Lista de Figuras **x**

1 Introducción **1**

1.1 ¿Qué es el texto tóxico? 1

1.1.1 Definiciones de diversas fuentes 2

1.1.2 Definición propia 3

1.2 Antecedentes 4

1.3 Justificación 6

1.4 Hipótesis 6

1.5 Objetivos 7

1.5.1 Objetivo General 7

1.5.2 Objetivos Específicos 7

1.6 Contribuciones 8
1.7 Estructura de la tesis 8

2 Marco teórico **10**

2.1 Inteligencia artificial (IA) 10
2.2 Procesamiento del lenguaje natural (PLN) 12
2.3 Algoritmos 15
2.3.1 Regresión logística (RL) 15
2.3.2 Naive Bayes (NB) 16
2.3.3 Naive Bayes multinomial (NBM) 17
2.3.4 Bernoulli Naive Bayes (BNB) 18
2.3.5 Perceptrón 18
2.3.6 Perceptrón multicapa (MLP) 21
2.3.7 Máquinas de soporte vectorial (SVM) 22
2.3.8 Árboles de decisión 23
2.3.9 Bosques aleatorios (RF) 24
2.4 Software y hardware utilizado 24
2.5 Métricas de evaluación 25

3 Estado del arte **27**

3.1 Modelos de aprendizaje automático 28
3.2 Modelos de aprendizaje profundo 30
3.3 Conjuntos de datos 32
3.4 Resumen 33

4 Conjunto de datos **35**

5 Metodología **37**

5.1 Tratamiento de datos 38
5.2 Preprocesamiento 41
5.3 Extracción de características. 43
5.4 Selección del mejor modelo 46
5.5 Obtención de resultados 47

6 Experimentos y resultados **49**

6.1 Unigramas 51
6.2 Unigramas y bigramas 56
6.3 Unigramas, bigramas y trigramas 61

7 Conclusiones **66**

8 Trabajo futuro **70**

Referencias **72**

Lista de Tablas

2.1 Categorías POS. 13

2.2 Grados FRE. 14

3.1 Resumen del estado del arte ordenado de mayor a menor desempeño. 34

4.1 Muestra del conjunto de datos. 36

5.1 Distribución de entrenamiento y prueba del conjunto de datos. 38

5.2 Número de documentos por subconjunto. 40

6.1 Parámetros de los modelos seleccionados. 50

6.2 Modelos generales. 50

6.3 Cantidad de características (unigramas). 51

6.4 Resultados de la validación cruzada para Tf (unigramas). 51

6.5 Resultados de la validación cruzada para Tf-Idf (unigramas). 51

6.6 Resultados de los ensambles para Tf (unigramas). 53

6.7 Resultados de los ensambles para Tf-Idf (unigramas). 53

6.8 Cantidad de características (unigramas). 57

6.9 Resultados de la validación cruzada para Tf (unigramas y bigramas). 57

6.10 Resultados de la validación cruzada para Tf-Idf (unigramas y bigramas). 58

6.11 Resultados de los ensambles (unigramas y bigramas). 58

6.12 Resultados de los ensambles (unigramas y bigramas). 58

6.13 Cantidad de características (unigramas, bigramas y trigramas). 61

6.14 Resultados de la validación cruzada para Tf (unigramas, bigramas y trigramas). 62

6.15 Resultados de la validación cruzada para Tf-Idf (unigramas, bigramas y trigramas). 62

6.16 Resultados de los ensambles (unigramas, bigramas y trigramas). . . . 63

6.17 Resultados de los ensambles (unigramas, bigramas y trigramas). . . . 63

Lista de Figuras

2.1 Función logística estándar. 16

2.2 Neurona biológica. 19

2.3 Neurona artificial. 19

2.4 Perceptrón. 20

2.5 MLP de una capa oculta con una salida escalar. 21

2.6 Ejemplo de una SVM para dos clases. 22

2.7 Ejemplo de un árbol de decisión. 24

2.8 Ejemplo del RF. 25

4.1 Distribución de documentos entre clases. 36

5.1 Bosquejo para la solución del trabajo propuesto. 37

5.2 Conjunto de entrenamiento. 39

5.3 Conjunto de prueba. 39

5.4 Extracción de características. 45

5.5 Selección del mejor modelo. 47

5.6 Ensamble de modelos. 48

6.1 Matriz de confusión para la configuración TF-1G-FM3. 54

6.2 Matriz de confusión para la configuración TF-1G-FM5. 54

6.3 Matriz de confusión para la configuración TFI-1G-FM3. 55

6.4 Matriz de confusión para la configuración TFI-1G-FM5. 55

6.5 Matriz de confusión para la configuración TF-12G-FM3. 59

6.6 Matriz de confusión para la configuración TF-12G-FM5. 59

6.7 Matriz de confusión para la configuración TFI-12G-FM3. 60

6.8 Matriz de confusión para la configuración TFI-12G-FM5. 60

6.9 Matriz de confusión para la configuración TF-123G-FM3. 64

6.10 c 64

6.11 Matriz de confusión para la configuración TFI-123G-FM3. 65

6.12 c 65

7.1 Matriz de confusión de los resultados obtenidos por Davidson et al., 2017. 68

7.2 Matriz de confusión de los resultados obtenidos por Mozafari et al., 2020. 68

7.3 Matriz de confusión de los mejores resultados obtenidos en este trabajo. 69

Capítulo 1

Introducción

Esta investigación se centra en la correcta clasificación entre el texto tóxico u ofensivo y el texto no tóxico, utilizando que se logren encontrar en alguna red social. Además, se presentarán técnicas de aprendizaje automático y aprendizaje profundo, creando ensambles para observar el comportamiento de estos, así como sus ventajas y desventajas.

1.1 ¿Qué es el texto tóxico?

Decidir si un fragmento de texto hace alusión al texto tóxico no es una tarea fácil, ya que como tal no existe una definición. Sin embargo, existen otras definiciones donde la intención que queremos tratar en este trabajo es similar o la misma, algunas pueden ser muy generales y otras pueden llegar a ser más específicas, algunas de ellas son: discurso de odio, lenguaje ofensivo, amenazas, agresividad, etc. El texto tóxico es un fenómeno complejo el cual hace referencia a las relaciones entre personas o grupos de personas en las cuales se tiende a usar un lenguaje derivado de estas problemáticas de la sociedad. Por esta razón es necesario definir que es el texto tóxico para así facilitar en entendimiento de la tarea que se atenderá en este trabajo. Para lograr esto tomaremos inspiración de dos vertientes que son de cierta manera más

generales: el lenguaje ofensivo y el discurso de odio.

1.1.1 Definiciones de diversas fuentes

Lenguaje ofensivo

- **RAE**[1]: Que ofende o puede ofender. Que ataca o sirve para atacar. Situación o estado de quien trata de ofender o atacar.

Discurso de odio

- **Facebook**[2]: "Definimos el lenguaje que incita al odio como un ataque directo a las personas, y no a los conceptos ni a las instituciones, en función de lo que denominamos características protegidas: raza, etnia, nacionalidad, discapacidad, religión, casta, orientación sexual, sexo, identidad de género y enfermedad grave."
- **Twitter**[3]: "Texto que fomente la violencia contra otras personas ni atacarlas o amenazarlas directamente por motivo de su raza, origen étnico, nacionalidad, pertenencia a una casta, orientación sexual, género, identidad de género, afiliación religiosa, edad, discapacidad o enfermedad grave."
- **TikTok**[4]: "Definimos el discurso o el comportamiento de odio como el contenido que ataca, amenaza, incita a la violencia o deshumaniza de alguna manera a un individuo o grupo sobre la base de los siguientes atributos protegidos: raza, etnia, origen nacional, religión, casta, orientación sexual, sexo, género, expresión de género, identidad de género, enfermedad grave, discapacidad, condición de inmigrante."
- **Twitch**[5]: "El acoso se convierte en conducta de odio cuando el ataque se

[1] https://dle.rae.es/ofensivo
[2] https://transparency.fb.com/es-la/policies/community-standards/hate-speech/
[3] https://help.twitter.com/es/rules-and-policies/hateful-conduct-policy
[4] https://www.tiktok.com/community-guidelines?lang=es#38
[5] https://blog.twitch.tv/es-mx/2020/12/09/introducing-our-new-hateful-conduct-harassment-policy/

basa en características de identidad. Consideramos los siguientes atributos basados en identidad como características protegidas: raza, etnia, color, casta, origen nacional, situación migratoria, religión, sexo, género, identidad de género, orientación sexual, discapacidad, condición médica grave y militar en retiro."

A pesar de las similitudes entre las definiciones, concluimos que hay algunos matices que pueden ayudar a definir al texto tóxico:

- **El texto tóxico considera objetivos específicos**. Este tipo de lenguaje tiene objetivos específicos y se basa en características específicas de los grupos, como el origen étnico, la religión, orientación sexual, entre otros.
- **El texto tóxico incita a la violencia u odio**. Las definiciones revisadas anteriormente utilizan diversos términos sobre cuando se produce o no este tipo de comportamiento. Sin embargo, la mayoría hacen referencia que este comportamiento incita a la violencia o al odio hacia una minoría.

1.1.2 Definición propia

El texto tóxico es el lenguaje escrito que ofende o menosprecia, que incita a la violencia o al odio hacia una persona o un grupo de personas basándose en características específicas.

De esta manera, tenemos que, el texto tóxico es la unificación de lo que es considerado como lenguaje ofensivo y de lo que las redes sociales definen como discurso de odio. Creando así una definición aún más general.

1.2 Antecedentes

En los últimos años, se ha visto un crecimiento exponencial respecto al número de personas que utilizan redes sociales. De acuerdo con DataReportal[6], actualmente el 59 % de la población mundial utiliza las redes sociales con un uso promedio de 2 horas y 29 minutos. A pesar de que estos sitios pueden ser una herramienta muy poderosa para las interacciones humanas virtuales, como conectar personas a largas distancias o inclusive para hacer crecer un negocio; estas traen consigo algunos problemas. Uno de ellos es la mala comunicación con los demás usuarios, esto sucede debido a que las personas que interactúan con los contenidos que ofrecen estos sitios no necesariamente tienen las mismas opiniones cuando algún tema surge en una discusión. Sumándole a esto, el poco control de estas plataformas para erradicar su mal uso incita a las personas a usar lenguaje tóxico. Informes recientes del año 2021 sobre la encuesta anual de odio y acoso en redes sociales de ADL[7], realizada en los Estados Unidos, nos da a conocer que el 41 % de los encuestados ha experimentado algún tipo de acoso en línea.

El uso del lenguaje ofensivo ha demostrado tener un impacto negativo en las personas, convirtiéndolo en una de las razones que corrompe la salud mental de los adolescentes (Suzuki et al., 2012) y una de las principales razones de suicidio (Hinduja & Patchin, 2010). Por consiguiente, el detectar esta clase de lenguaje es un problema relevante que necesita ser tomado en cuenta.

La razón principal por la que debemos erradicar el lenguaje tóxico de las redes sociales es para evitar que los usuarios que hacen uso de estas plataformas sigan sufriendo, ya que a menudo las personas que utilizan este tipo de lenguaje juzgan a los demás por sus publicaciones en redes sociales, y a partir de dichas publicaciones más usuarios pueden realizar comentarios ofensivos. De esta manera, las personas que ven comentarios ofensivos hacia otra persona pueden sentirse tentados a tomar represalias sobre la gente que está haciendo daño. Esto genera un círculo

[6] https://datareportal.com/reports/digital-2022-july-global-statshot
[7] https://www.adl.org/online-hate-2021

vicioso, en donde alguien publica comentarios hirientes sobre otro(s), lo que provoca la propagación del texto tóxico. Para contrarrestar el lenguaje ofensivo vive en la Internet, algunas redes sociales como Facebook han implementado medidas para detectar texto ofensivo en imágenes con ayuda de algoritmos de reconocimiento óptico de caracteres (OCR, por sus siglas en inglés) (Borisyuk et al., 2018). Mientras que otros sitios suelen pedirle a la comunidad que denuncien este comportamiento, así como también suelen revisar manualmente o tener moderadores que revisen publicaciones y comentarios con la finalidad de eliminar este tipo de comentarios. Sin embargo, este tipo de revisiones manuales requieren de mucho trabajo, por lo que no son sostenibles ni escalables en el tiempo.

Por otro lado, la comunidad científica no se ha quedado atrás y ha realizado numerosas investigaciones para lograr identificar y clasificar correctamente este tipo de comportamiento. Comenzando desde lo más básico del lenguaje, utilizando características léxicas como los diccionarios (S. Liu & Forss, 2015) o la bolsa de palabras (Burnap & Williams, 2016). Sin embargo, y a pesar de que el uso de las características léxicas han tenido una muy buena aceptación, es necesario entender el contexto o estructura sintáctica del texto a identificar. Por tal motivo, más adelante se comenzaron a utilizar enfoques basados en n-gramas, la parte del discurso e incluso características como pruebas de legibilidad del texto (Davidson et al., 2017).

Con el acelerado crecimiento de las técnicas basadas en el aprendizaje automático, poco a poco se fueron presentando trabajos con arquitecturas más elaboradas como: las redes neuronales recurrentes (Badjatiya et al., 2017a), redes neuronales profundas (Badjatiya et al., 2017b) o las redes neuronales convolucionales (Georgakopoulos et al., 2018).

Basados en el uso de redes neuronales convolucionales y redes recurrentes para el procesamiento del lenguaje natural, nació una nueva corriente en la cual se propone el uso de mecanismos de atención al cual llamaron *transformers* (Vaswani et al., 2017). Desde entonces, se han convertido en la corriente principal de investigación, dando pie a grandes modelos del lenguaje como BERT (Devlin et al., 2018),

RoBERTa (Y. Liu et al., 2019) y XLM (Lample & Conneau, 2019). Dichos modelos, al ser entrenados con millones de datos etiquetados de texto, "comprenden" de una forma muy abstracta el lenguaje que utilizamos los seres humanos, por lo cual solo es cuestión de realizar un ajuste para una tarea específica como la detección del texto tóxico, por mencionar algunos ejemplos: Ozler et al., 2020; Baratalipour et al., 2020; Sivanaiah et al., 2020; Althobaiti, 2022.

1.3 Justificación

Hoy en día, se han efectuado múltiples aproximaciones para la detección del texto tóxico u ofensivo. Recientemente, las soluciones propuestas para ello es el uso del aprendizaje por transferencia. Esta técnica consiste en tomar un modelo que ya ha sido entrenado en una tarea similar y adaptarlo para resolver el problema actual, que en nuestro caso sería la detección de texto tóxico u ofensivo. Sin embargo, hacer un ajuste específico con alguno de estos modelos es computacionalmente caro y por tal motivo no es viable para implementar en sistemas con bajos recursos.

En consecuencia, es esencial disponer de un modelo que pueda detectar con precisión los textos tóxicos u ofensivos y se necesitan técnicas que no consuman tantos recursos computacionales para avanzar hacia su clasificación efectiva. Esta investigación presentará una solución para clasificar este tipo de texto a partir de la creación de un ensamble de modelos de aprendizaje automático que suelen ser más rápidas de llevar a cabo y menos costosas.

1.4 Hipótesis

Es posible crear un ensamble de modelos de aprendizaje automático personalizado para clasificar texto tóxico que tenga un rendimiento similar o mejor a las líneas base encontradas en la investigación del estado del arte, que suelen ser modelos

con arquitecturas más complejas y con una gran cantidad de parámetros. Además, utilizando distintos métodos para el preproceso y equilibrio de los datos, es posible clasificar texto tóxico obteniendo resultados competitivos en comparación con modelos basados en aprendizaje profundo bien conocidos en el estado del arte.

1.5 Objetivos

1.5.1 Objetivo General

Proponer la creación de un ensamble de modelos de aprendizaje automático que se ajuste lo más posible al un conjunto de datos para la clasificación certera del texto tóxico.

1.5.2 Objetivos Específicos

- Realizar un equilibrio del conjunto de datos seleccionado mediante la separación de clases.
- Revisar y seleccionar los mejores modelos de aprendizaje automático a partir del estado del arte.
- Obtener los modelos que mejor desempeño obtienen para las diferentes clases del conjunto de datos seleccionado.
- Desarrollar un ensamble de modelos de aprendizaje automático para resolver la tarea de clasificación de texto tóxico.
- Añadir características al texto para observar el comportamiento de los modelos con más características.
- Validar y evaluar el rendimiento del ensamble de modelos.

1.6 Contribuciones

En este trabajo se propone un nuevo procedimiento para clasificar correctamente el texto tóxico. Este procedimiento consiste en crear un ensamble de modelos de aprendizaje automático tomando en cuenta los mejores clasificadores que identificamos en el capítulo 3. Aunado a esto, el preprocesamiento y extracción de características se llevarán a cabo de una diferente manera, en la cual se realizará un equilibrio del conjunto de datos que se describe en el capítulo 4. Dicha metodología será descrita de forma más detallada en el capítulo 5.

1.7 Estructura de la tesis

En el capítulo 1 se ha presentado una introducción a este trabajo. Además, se han expuesto la justificaión, la hipótesis y los objetivos

En el capítulo 2 contiene el marco teórico, en el cual se podrán encontrar algunas de las definiciones que se utilizaron a lo largo del trabajo. Así como la descripción de los modelos que se emplearon.

El capítulo 3 contiene los trabajos relacionados del estado del arte que son relevantes para esta investigación. Tales como, los modelos de aprendizaje automático y aprendizaje profundo para la clasificación de texto tóxico. Asimismo, muestra los conjuntos de datos que se han usado a lo largo del tiempo.

El capítulo 4 describe el conjunto de datos que se empleó para este trabajo de investigación.

El capítulo 5 detalla el proceso que se llevó a cabo para la resolución de la tarea planteada en esta tesis. Desde la manipulación del conjunto de datos y el equilibrio del mismo hasta como realizamos el ensamble de modelos.

El capítulo 6 presenta la experimentación que se llevó a cabo a través de

las distintas propuestas, así como los resultados obtenidos

El capítulo 7 establece las conclusiones a las que se llegaron tras la realización de la propuesta del ensamble de modelos.

El capítulo 8 describe el trabajo que falta por hacer, así como el trabajo que se estará procediendo en un futuro.

Capítulo 2

Marco teórico

En esta sección se definirán, primero, algunos conceptos que son necesarios para entender el contenido de las secciones posteriores. Más adelante, se definirán los algoritmos y los solucionadores que se utilizarán para lograr cumplir la hipótesis de esta tesis. Por último, se mostrarán las características técnicas con las cuales se creará la propuesta de este trabajo.

2.1 Inteligencia artificial (IA)

A lo largo de los años se han hecho numerosos intentos por definir exactamente que es la IA. Si bien nunca ha habido una definición oficial aceptada, hay algunos conceptos de alto nivel que nos ayudan a dar forma para definir lo que esta significa. Mientras tanto, para los fines de este trabajo, definiremos a la IA como cualquier técnica computacional que permita dar significado a los datos de manera similar a la de un ser humano.

Aprendizaje automático. Es una rama de la IA y de la informática en general, la cual se centra en el uso de datos y algoritmos para imitar la forma en la que los seres humanos aprendemos. Se le atribuye haber acuñado el término a

Samuel, 1959, en su investigación basada en el juego de damas, donde afirma que "el aprendizaje automático es el campo de estudio que brinda a las computadoras la capacidad de aprender sin ser programadas explícitamente".

Existen distintos tipos de modelos de aprendizaje automático, estos se definen por la presencia o ausencia de la influencia humana en los datos, ya sea que se ofrezca una recompensa, se proporcionen comentarios específicos o el uso de etiquetas como método de diferenciación de los datos. Estos modelos son:

- **Aprendizaje supervisado:** para este tipo de modelos se utilizan conjuntos de datos que previamente han sido etiquetados y clasificados por seres humanos para permitir que los algoritmos que hagan uso de estos puedan determinar que tan preciso es su desempeño.
- **Aprendizaje no supervisado:** al contrario que en el aprendizaje supervisado, en este tipo de modelos se utilizan conjuntos de datos sin procesar, es decir, que no se encuentran etiquetados. La finalidad de estos modelos es que por sí mismos logren encontrar patrones y relacionarlos entre si.
- **Aprendizaje semisupervisado:** este tipo de modelos son un tipo de mezcla de los modelos de aprendizaje supervisado y no supervisado, donde se hace uso de datos estructurados y no estructurados. Dichos datos en conjunto guían al modelo en su camino hacia conclusiones independientes. Esto permite que estos modelos aprendan a etiquetar datos sin etiquetar.
- **Aprendizaje por refuerzo:** para estos modelos se utiliza un sistema de recompensa o castigos lo cual ofrece retroalimentación a los algoritmos utilizados para aprender de sus propias experiencias por medio de prueba y error.
- **Aprendizaje profundo:** es el área más reciente del aprendizaje automático, el cual aprende automáticamente de los conjuntos de datos sin la necesidad de introducir reglas o conocimientos humanos. Para lograr esto, se requieren grandes cantidades de datos sin procesar, mientras más se reciban hay más probabilidades de mejora en el modelo.

2.2 Procesamiento del lenguaje natural (PLN)

IBM[1] nos otorga la siguiente definición: "se refiere a la rama de la informática, y más específicamente, la rama de la inteligencia artificial o IA, que se ocupa de brindar a las computadoras la capacidad de comprender textos y palabras habladas de la misma manera que los seres humanos".

El PLN es fundamental para analizar a profundidad los datos de texto extraídos de escritos, audio e inclusive imágenes de manera eficiente. Combina la lingüística computacional (modelos de reglas basadas en el lenguaje humano) con modelos estadísticos, de aprendizaje automático y de aprendizaje profundo, con la finalidad de "comprender" el significado, la intención, el sentimiento de algún texto o audio. Algunas de las tareas que le corresponden al NLP son: el reconocimiento de voz, etiquetado de parte del discurso, desambiguación del sentido de las palabras, análisis de sentimientos, generación de texto, entre otras.

Por lo general, el proceso de PLN comienza con la recopilación y preparación de los datos, los cuales suelen contener una gran cantidad de datos inservibles. Por lo que, el siguiente paso es realizar el preprocesamiento de los datos antes de seleccionar algún modelo de aprendizaje automático, algunos de los pasos que se pueden seguir para realizar el preprocesamiento son los siguientes:

- Eliminar palabras vacías o *stopwords*, las cuales son palabras de uso común en los idiomas, como artículos, pronombres, preposiciones, entre otras.
- Eliminar nombres de usuarios.
- Transformar el texto a minúsculas.

Una vez preprocesado el texto, el siguiente paso es la extracción de características. Algunas de las características que se pueden extraer de los textos son las siguientes:

Lematización. Una vez que el texto se encuentra en su mayoría compuesta

[1]https://www.ibm.com/cloud/learn/natural-language-processing

de oraciones legibles, se utiliza la técnica de lematización, la cual consiste en generar las raíces de las palabras contenidas en los documentos.

POS. De igual manera que en la lematización, el proceso para generar las categorías gramaticales de las palabras en las oraciones es más fácil de llevar una vez que el texto se encuentra más legible. Existen 17 posibles asignaciones para distinguir las propiedades léxicas y gramaticales adicionales de las palabras según la UD[2]. Estas categorías se muestran en la tabla 2.1.

Tabla 2.1: Categorías POS.

Palabras de clase abierta	Palabras de clase cerrada	Otras
ADJ: adjetivo	ADP: adposición	PUNCT: puntuación
ADV: adverbio	AUX: auxiliar	SYM: símbolo
INTJ: intersección	CCONJ: conjunción de coordinacion	X: otra
NOUN: sustantivo	DET: determinante	
PROPN: nombre propio	NUM: número	
VERB: verbo	PART: partícula	
	PRON: pronombre	
	SCONJ: conjunción subordinante	

TAG. Es una parte aún más detallada del POS, la cual contiene cerca de 300 categorías.

Conteo de sílabas, palabras y *stopwords*. Este conteo se lleva a cabo por cada uno de los documentos del conjunto de datos.

Promedio de sílabas. Dado el conteo de sílabas, el promedio de sílabas se llevará a cabo para cada uno de los documentos existentes.

Sentimiento. Se pueden utilizar modelos de aprendizaje automático para detectar sentimientos o lexicones, que son diccionarios que basados en ciertas áreas o problemas del lenguaje pueden o no asignar una clasificación. Un ejemplo de esto es el lexicón llamado VADER[3], el cuál se basa en reglas que se adoptan específicamente a los sentimientos expresados en las redes sociales.

FRE. Esta prueba muestra las puntuaciones de la dificultad de lectura que

[2]https://universaldependencies.org/u/pos/
[3]https://github.com/cjhutto/vaderSentiment

pueda tener un texto. Donde las puntuaciones más altas indican que el texto es más fácil de leer y las puntuaciones más altas indican que es más difícil de leer. Se divide en ocho categorías diferentes, las cuales se describen en la tabla 2.2 y se obtiene de la siguiente manera:

$$206.835 - 1.015(\frac{\text{total de palabras}}{\text{total de oraciones}}) - 84.6(\frac{\text{total de sílabas}}{\text{total de palabras}})$$

Tabla 2.2: Grados FRE.

Puntuación	Nivel de escolaridad (US)	Notas
100.00–90.00	Quinto grado	Muy fácil de leer. Fácilmente comprensible para un alumno medio de 11 años.
90.0–80.0	Sexto grado	Fácil de leer. Inglés conversacional para consumidores.
80.0–70.0	Séptimo grado	Bastante fácil de leer.
70.0–60.0	Octavo y noveno grado	Inglés sencillo. Fácilmente comprensible para estudiantes de 13 a 15 años.
60.0–50.0	Décimo a decimosegundo grado	Bastante difícil de leer.
50.0–30.0	Universitario	Difícil de leer.
30.0–10.0	Universitario graduado	Muy difícil de leer. Lo entienden mejor los universitarios graduados.
10.0–0.0	Profesional	Muy difícil de leer. Lo entienden mejor los universitarios graduados.

FK. Esta prueba de legibilidad es muy usada en el campo de la educación. Mientras que FRE presenta una puntuación basada en el nivel de escolaridad de Estados Unidos, FK mide el número de años de educación generalmente requeridos para entender un texto. Esta prueba es calculada con la siguiente fórmula:

$$0.39(\frac{\text{total de palabras}}{\text{total de oraciones}}) - 11.8(\frac{\text{total de sílabas}}{\text{total de palabras}}) - 15.59$$

A pesar de que estos pasos son algunos de los pasos más utilizados en el PLN, estos pueden variar dependiendo la tarea que se requiera resolver o el formato en el que el texto sea extraído. Una vez obtengamos nuestro conjunto de datos limpio, es necesario representar el texto de forma una computadora lo pueda interpretar, es decir, numéricamente. La forma más sencilla de realizar este procedimiento es por medio de obtener una bolsa de palabras (BoW, por sus siglas en inglés), la cual es una manera de representar el vocabulario que emplearemos en nuestro modelo y

consiste en crear una matriz en la que cada columna es una palabra y se contabiliza la cantidad de aparición de cada palabra en un texto, y por medio de la frecuencia de aparición se asignan probabilidades de pertenencia o no a una clase.

2.3 Algoritmos

2.3.1 Regresión logística (RL)

A pesar de su nombre, es un modelo del aprendizaje supervisado lineal para la clasificación en lugar de regresión. A la RL también se le conoce en la literatura como *logit regression*, clasificación de máxima entropía (MaxEnt) o clasificador logarítmico lineal. La RL es un modelo estadístico que utiliza la función logística, matemáticas como la ecuación entre x e y. La función mapea y como una función sigmoidea de x (Aghdam & Heravi, s.f.).

$$f(x) = \frac{1}{1 + \epsilon^{-x}},$$

Si esta ecuación se representa, se obtendrá una curva en forma de S como la que se muestra en la figura 2.1.

Como se puede ver, la función logística devuelve solo valores entre 0 y 1 para la variable dependiente, al margen de los valores de la variable independiente. Así es como la regresión logística estima él valore de la variable dependiente. Los métodos de RL también puede modelar ecuaciones entre múltiples variables independientes y una variable dependiente.

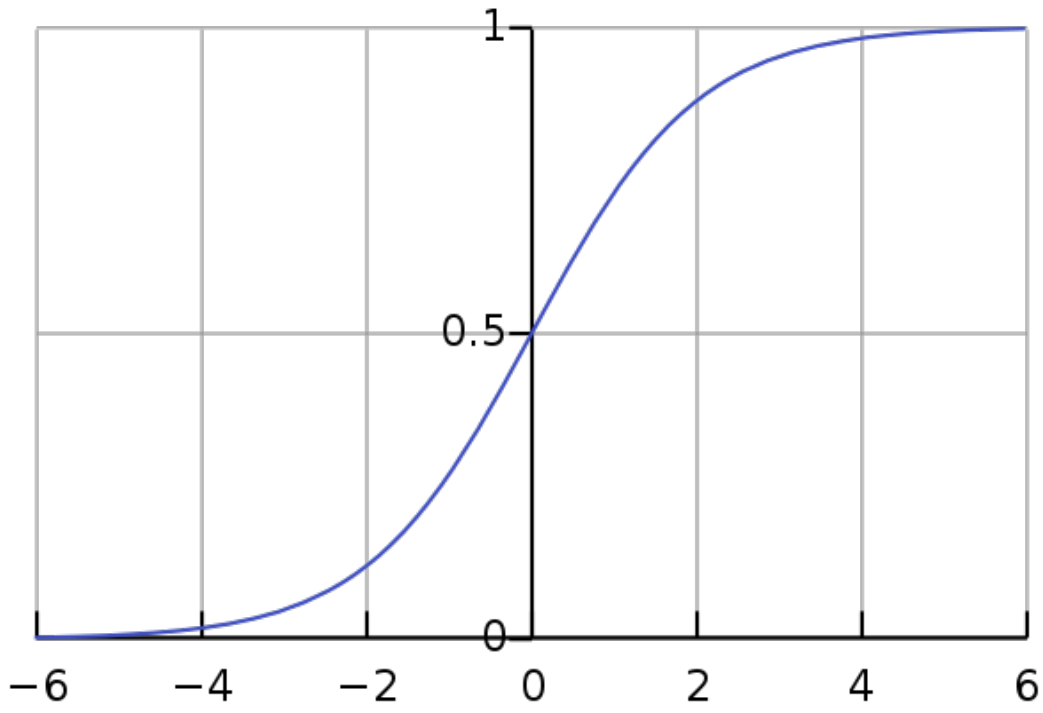

Figura 2.1: Función logística estándar.

2.3.2 Naive Bayes (NB)

Para comprender BNB, es necesario primero comprender Naive Bayes (NB). NB es un algoritmo de aprendizaje automático supervisado, el cual sirve para predecir la probabilidad de diferentes clases en función de numerosos atributos, y nos indica la probabilidad de ocurrencia en un evento (Webb et al., 2011). A NB también se le conoce como probabilidad condicional y se basa en el teorema de Bayes:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)},$$

Donde A es el evento 1, B es el evento 2; $P(A)$ es la probabilidad de que A sea verdadero (probabilidad a priori); $P(B)$ es la probabilidad de que B sea verdadero (probabilidad marginal); $P(A|B)$ es la probabilidad de que A sea verdadero dado que B es verdadero (probabilidad posterior); y $P(B|A)$ es la probabilidad de que B sea verdadero dado que A es verdadero (la probabilidad).

Sin embargo, para el caso del clasificador NB, solamente nos interesamos en la máxima probabilidad posterior, por lo que ignoramos el denominador, es decir, la probabilidad marginal. Este clasificador se basa en dos suposiciones esenciales:

- **Independencia condicional:** todas las características son independientes entre sí. Esto implica que una característica no afecta el desempeño de la otra.
- **Importancia de las funciones:** todas las funciones son igualmente importantes. Es fundamental conocer todas las características para realizar buenas predicciones y obtener resultados más precisos.

2.3.3 Naive Bayes multinomial (NBM)

De igual forma que NB, es un método probabilístico. Sin embargo, este clasificador es adecuado para realizar la clasificación con características discretas. NBM implementa el algoritmo de NB para datos distribuidos, y es ua de las dos variantes de NB usadas para clasificación de texto, donde los datos son representados típicamente como vectores de conteo de palabras. La distribución es parametrizada por vectores $\theta_y = (\theta_{y1}, ..., \theta_{yn})$ para cada clase n, donde n es el número de características (en el caso de clasificación de texto, el tamaño del vocabulario) y θ_{yi} es la probabilidad $P(x_i|y)$ de la característica i apareciendo en una muestra perteneciente a la clase y.

Los parámetros θ_y se estiman mediante una versión suavisada de máxima verosimilitud, por ejemplo, el recuento de la frecuencia relativa:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n},$$

donde $N_{yi} = \Sigma_{x \in T} x_i$ es el número de veces que i aparece en un a muestra de la clase y en el conjunto de entrenamiento T, y $N_y = \Sigma_{i=1}^{n} N_{yi}$ es el total de todas las características de la clase y. Las probabilidades suavizadas a priori $\alpha \geq 0$ tienen en cuenta características que no están presentes en las muestras de aprendizaje y evitan las probabilidades nulas en los cálculos posteriores. Se establece $\alpha = 1$, que

es denominado como suavizado de Laplace, mientras que a $\alpha < 1$ se le denomina suavizado de Lidstone.

2.3.4 Bernoulli Naive Bayes (BNB)

Este algoritmo implementa los algoritmos de clasificación y entrenamiento de NB para datos que se distribuyen de acuerdo con distribuciones multivariadas de Bernoulli; es decir, puede haber múltiples características, pero cada una es una variable con un valor binario. Por lo tanto, este algoritmo requiere que las muestras se representen como vectores de características con valores binarios. La regla de decisión para el BNB se basa en:

$$P(x_i|y) = P(x_i = 1|y)x_i + (1 - P(x_i = 1))(1 - x_i),$$

Que difiere de la regla de NBM en que penaliza explícitamente la no ocurrencia de una característica i que es un indicador de la clase y, donde la variante multinomial simplemente ignoraría una característica que no ocurre.

2.3.5 Perceptrón

Para comprender el funcionamiento del perceptrón, primero es necesario comprender como funciona la neurona artificial, la cual está basada en la neurona biológica, que es una célula especializada y caracterizada por poseer una cantidad indefinida de canales de entrada llamados dendritas y un canal de salida llamado axón. Las dendritas operan como sensores que recogen información de la región donde se hallan y la derivan hacia el cuerpo de la neurona que reacciona mediante una sinapsis que envía una respuesta hacia el cerebro. Un ejemplo de la neurona biológica se muestra en la figura 2.2.

McCulloch y Pitts, 1943, crearon una unidad de cálculo que intenta modela

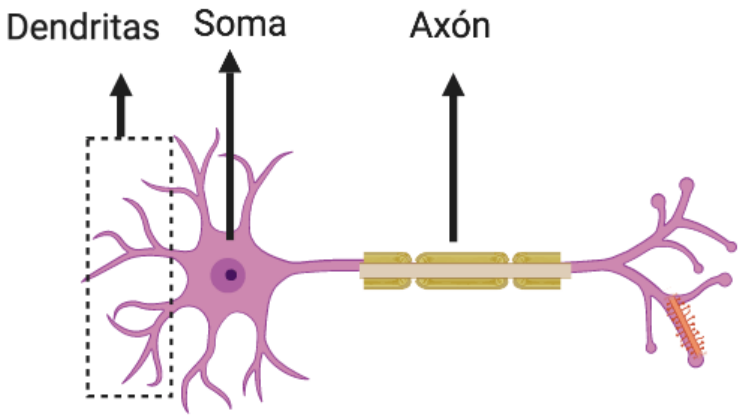

Figura 2.2: Neurona biológica.

el comportamiento de una neurona biológica, la cual es la unidad esencial con la cual se construiría una red neuronal artificial. El resultado obtenido del cálculo en una neurona artificial consiste en realizar una suma ponderada de las entradas, seguidas de la aplicación de una función no lineal. Dicho comportamiento lo podemos observar en la figura 2.3:

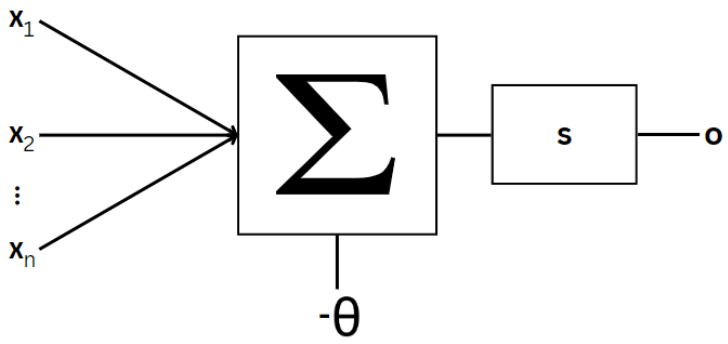

Figura 2.3: Neurona artificial.

La figura 2.3 se puede expresar matemáticamente como $o = s(red)$, donde $red = w_1x_1 + ... + w_nx_n - \theta$ la cual es una suma ponderada; x_i es el valor de la i-ésima entrada; w_i es el peso de la conexión entre la i-ésima entrada y la neurona; θ es el valor umbral; o es la salida de la neurona; y s es la función no lineal conocida como función de activación. La función de activación que se usa es:

$$s(u) = \begin{cases} 1, u \geq 0 \\ 0, u < 0 \end{cases}$$

Más adelante, Rosenblatt, 1958, presenta el perceptrón el cual está basado en la neurona artificial, el cual se puede observar en la figura 2.4. Este algoritmo usa una matriz para representar las redes neuronales y es un discriminador terciario que traza su entrada x a un único valor de salida $f(x)$, a través de dicha matriz.

$$f(x) = \begin{cases} 1, \text{si } w \cdot x - u > 0 \\ 0, \text{en otro caso} \end{cases},$$

Donde w es un vector de pesos reales; $w * x$ es el producto escalar; u es el umbral, el cual representa el grado de inhibición de la neurona, es un término constante que no depende del valor que toma la entrada. El valor de $f(x)$ se utiliza para clasificar x como un caso positivo o un caso negativo, en el caso de un problema de clasificación binario. El umbral puede entenderse como una manera de compensar la función de activación, o una forma de fijar un nivel mínimo de actividad a la neurona para considerarse como activa. La suma ponderada de las entradas debe producir un valor mayor que u para cambiar la neurona de estado 0 a 1.

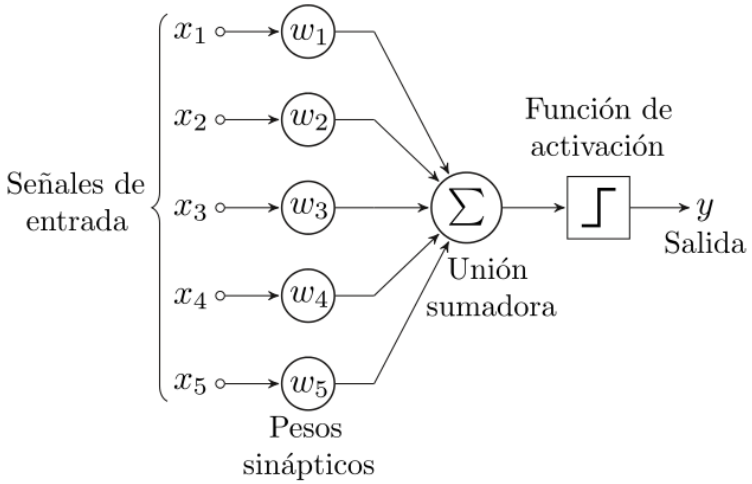

Figura 2.4: Perceptrón.

2.3.6 Perceptrón multicapa (MLP)

Es un algoritmo de aprendizaje supervisado que aprende una función $f(\cdot) = R^m \rightarrow R^0$ entrenando un conjunto de datos, donde m es el número de dimensiones para la entrada y o es el número de dimensiones para la salida. Dado un conjunto de características $X = x_1, x_2, ..., x_m$ y un objetivo y, puede aprender un aproximador de función no lineal para clasificación o regresión. Este algoritmo se diferencia de la RL en que entre la capa de entrada y la de salida puede haber una o más capas no lineales, llamadas capas ocultas. La figura 2.5 muestra un ejemplo de un MLP.

La capa de la izquierda es conocida como capa de entrada, y consta de un conjunto de neuronas representando las entidades de entrada $\{x_i | x_1, x_2, ..., x_m\}$. Cada neurona en la capa oculta (la capa intermedia) transforma los valores de la capa anterior con una suma lineal ponderada $w_1x_1 + w_2x_2 + ... + w_mx_m$, seguida de una función de activación no lineal, $g(\cdot) = R \rightarrow R-$ como lo puede ser una función de tangente hiperbólica. La capa de salida recibe los valores de la última capa oculta y los transforma en valores de salida.

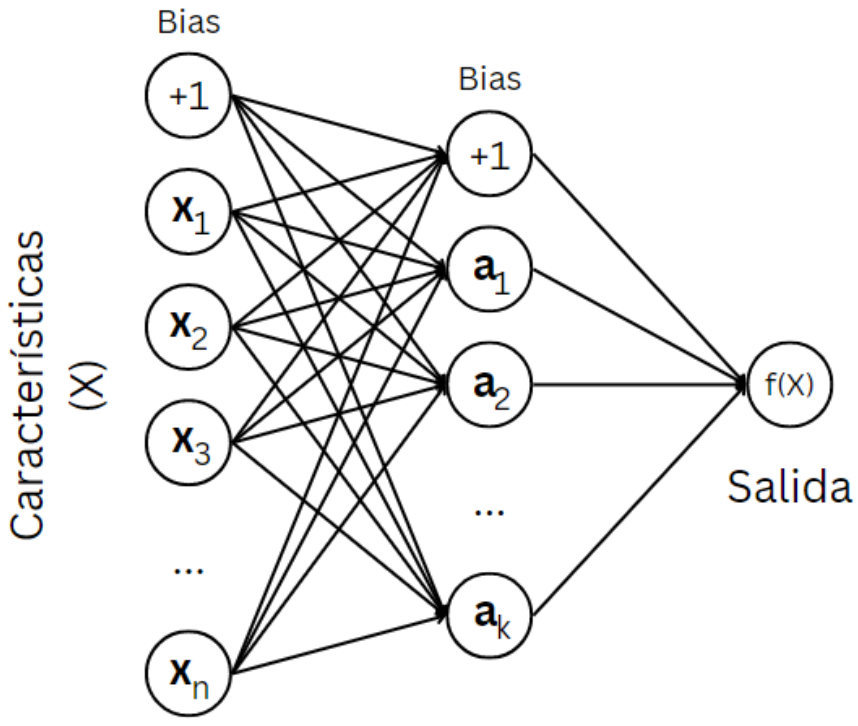

Figura 2.5: MLP de una capa oculta con una salida escalar.

2.3.7 Máquinas de soporte vectorial (SVM)

Son un conjunto de métodos de aprendizaje supervisado que utilizan la clasificación, la regresión y la detección de valores atípicos. Funciona asignando datos a un espacio de características de alta dimensión para que los puntos de datos se puedan clasificar, inclusive cuando los datos no se pueden separar linealmente. Se encuentra un separador entre las categorías, luego los datos se transforman de tal manera que el separador podría dibujarse como un hiperplano, el cual puede ser una o múltiples fronteras de decisión para segregar clases en un espacio n-dimensional (Boser et al., 1992). Después de esto, las características de los nuevos datos se pueden utilizar para predecir el grupo al que debe pertenecer el nuevo registro. La función matemática utilizada para la transformación de los datos se le conoce como kernel, el cual debe su nombre al uso de funciones kernel, que les permiten operar en un espacio de características implícito de alta dimensión, sin calcular nunca las coordenadas de los datos en el espacio, sino simplemente calculando los productos internos de todos los pares de datos en el espacio de las características.

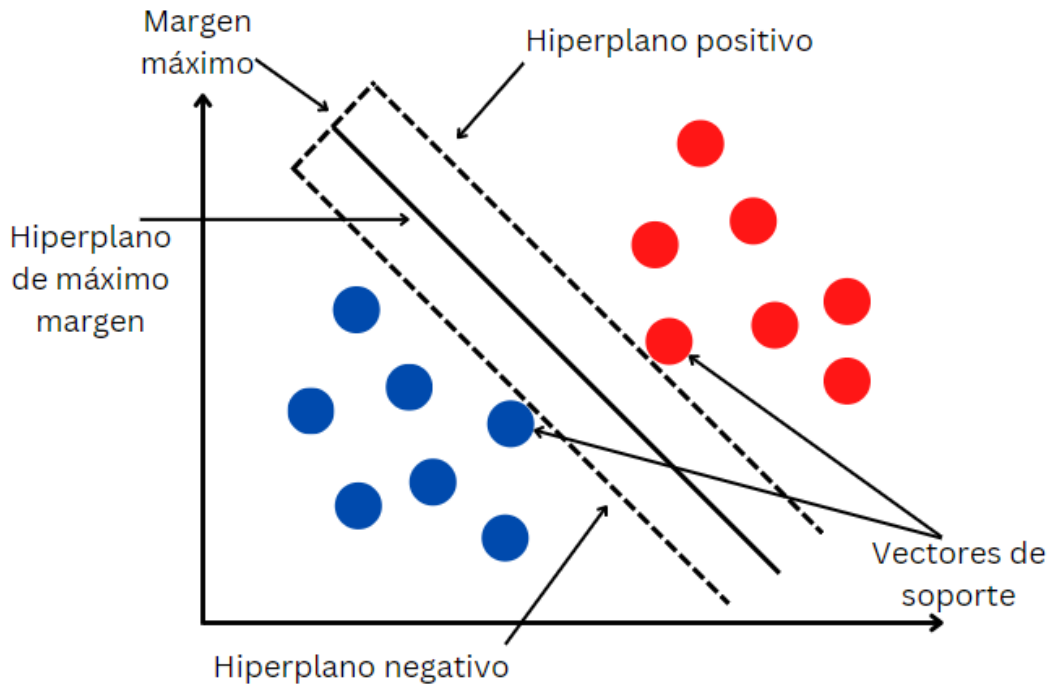

Figura 2.6: Ejemplo de una SVM para dos clases.

En la figura 2.6 podemos observar un ejemplo de una clasificación mediante SVM para dos clases. Las dimensiones del hiperplano dependen de las características presentes en los datos, lo que significa que si hay 2 características (tal y como se muestra en la figura), entonces el hiperplano será una línea recta. Siempre tenemos que crear un hiperplano el cual tenga un margen máximo, es decir, la máxima distancia entre los puntos. Por otra parte, los vectores que se encuentran más cercanos al hiperplano son llamados vectores de soporte.

2.3.8 Árboles de decisión

Dado un conjunto de datos se fabrican diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva para la resolución de un problema. Los árboles de decisión están conformados por:

- Nodos: cada uno de ellos se puede definir como el momento en el que se ha de tomar una decisión de entre varias variables posibles, lo que va haciendo que a medida que aumenta el número de nodos aumente el número de posibles finales a los que puede llegar el individuo. Esto hace que el árbol con muchos nodos sea complicado de analizar debido a la existencia de numerosos caminos que se pueden seguir.
- Vectores: son la solución final a la que se llega en función de las diversas posibilidades que se tienen, dan las utilidades a esa solución.
- Flechas: son uniones entre un nodo y otro, y representan cada acción distinta.
- Etiquetas: se encuentran en cada nodo y cada flecha, y dan nombre a cada acción.

Un ejemplo de esto se puede observar en la figura 2.7.

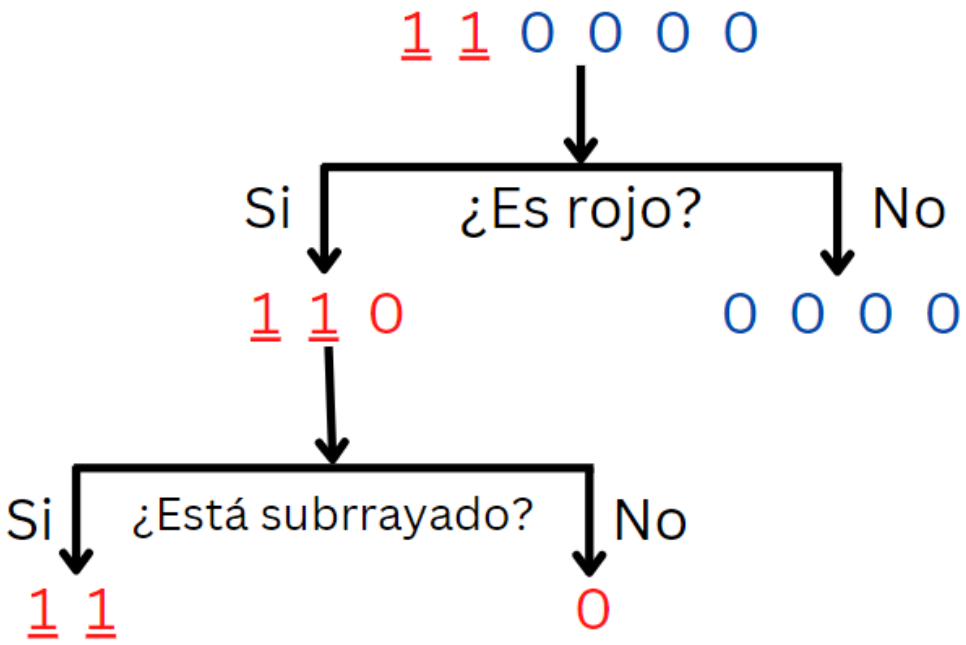

Figura 2.7: Ejemplo de un árbol de decisión.

2.3.9 Bosques aleatorios (RF)

Son un algoritmo de aprendizaje supervisado. Se pueden utilizar tanto para clasificación como para regresión (Breiman, 2001). Como su nombre lo indica, se basa en la idea de un bosque. Un bosque está compuesto por árboles y mientras más árboles tenga un bosque, más robusto será. Los bosques aleatorios crean árboles de decisión en muestras de datos al azar, obtienen predicciones de cada uno de los árboles y seleccionan la mejor solución mediante votación, tal y como se observa en la figura 2.8.

2.4 Software y hardware utilizado

- Para generar la lógica de los *scripts* de la solución implementada, se empleó el lenguaje de programación Python[4] en su versión 3.9.

[4]https://www.python.org/

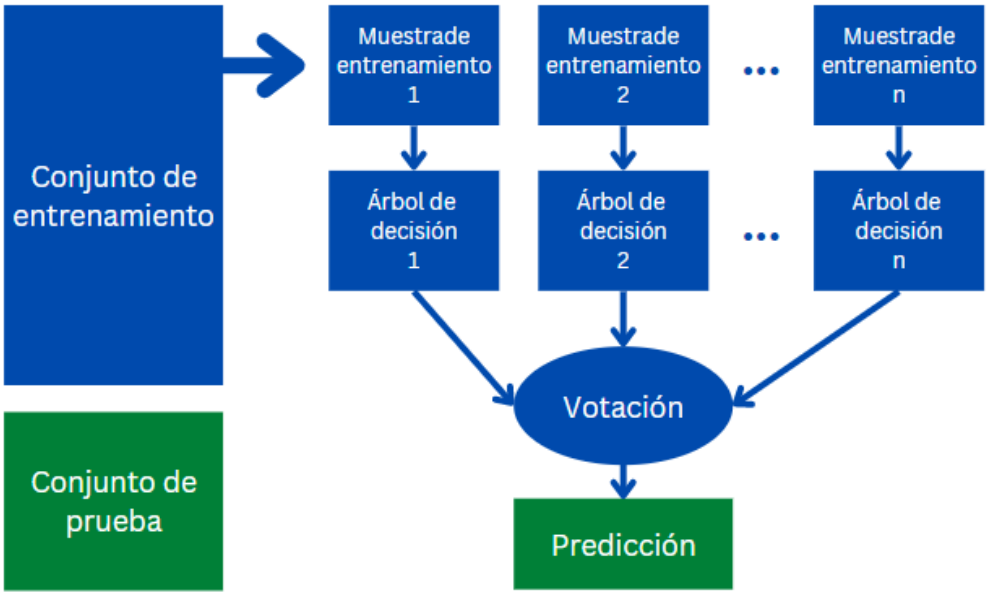

Figura 2.8: Ejemplo del RF.

- En cuanto a la implementación de los algoritmos empleados, se hizo uso de la librería *Scikit learn* (Pedregosa et al., 2011).
- Para la extracción de características del texto se emplearon las siguientes librerías: *spaCy* (Honnibal & Montani, 2017), NLTK (Bird et al., 2009).
- El hardware empleado fueron máquinas virtuales del entorno de Google Colaboratory[5] en su versión gratuita.

2.5 Métricas de evaluación

Las métricas propuestas para evaluar el desempeño de los modelos se enlistan a continuación.

- Precisión (*precision*). La precisión evalúa la calidad del modelo al medir cuántos del total de clasificados como positivos son en verdad positivos y se calcula como: $precisión = \frac{verdaderos\ positivos}{verdaderos\ positivos + falsos\ positivos}$

[5]https://colab.research.google.com

- Exhaustividad (*recall*). La exhaustividad indica la cantidad de verdaderos positivos que el modelo es capaz de recuperar y se calcula con la siguiente expresión: $recall = \frac{verdaderos\ positivos}{verdaderos\ positivos + falsos\ negativos}$
- Valor-F (*F1-score*). El valor-F es una función que relaciona la precisión y la exhaustividad, mide el balance que existe entre estas otras dos métricas y se calcula con la siguiente fórmula: $F1 = \frac{precision \cdot recall}{precision + recall}$

Capítulo 3

Estado del arte

Dado que la problemática de la detección del lenguaje tóxico u ofensivo ha tomado mucha popularidad en los últimos años, la cantidad de trabajos relacionados es bastante amplia, y por ende es muy difícil mencionar todas las aproximaciones que se han realizado. Por este motivo, en esta sección se describirán algunas de las aproximaciones que ha efectuado la comunidad científica con la finalidad de atacar este problema.

Esta sección se encuentra estructurada de la siguiente manera:

- Primero, se revisarán los trabajos con aproximaciones basadas en modelos de aprendizaje automático.
- Posteriormente, se revisarán los trabajos con aproximaciones basadas en modelos de aprendizaje profundo.
- Más adelante, se hará una revisión de los conjuntos de datos que se encuentran disponibles para hacer frente a esta problemática.
- Por último, se presentará una tabla resumiendo los trabajos investigados en la sección.

3.1 Modelos de aprendizaje automático

Greevy (2004), hizo una aproximación para la detección del racismo en internet. Para lograrlo, hizo uso de SVM, y utilizó como características: BoW, n-gramas con secuencias de 2 y 3 palabras y una versión reducida del POS. En este trabajo se obtuvo una precisión de cerca del 90 %, y se utilizó un conjunto de datos propio, el cual consiste en 1,000 documentos con el mismo número de documentos para la clase *racist* y *non-racist*.

Burnap y Williams (2015), emplearon como extracción de características BoW, n-gramas con secuencias de dos palabras, y dependencias tipadas. Dado que trabajan con un conjunto de características de palabras específicas y características sintácticas, optaron por crear un conjunto de resultados y un modelo relacionado que pudiera usarse para informar sobre el riesgo de que el odio cibernético se pueda propagar después de que eventos trágicos sucedan y que probablemente generen una respuesta antagónica o de odio hacia un grupo de personas. Dada la prevalencia de palabras individuales o combinaciones cortas de palabras consideraron apropiado seleccionar un clasificador basado en la probabilidad de ocurrencia de las características, por esta razón seleccionaron un clasificador logístico bayesiano (BLR). Por otra parte, se ha demostrado que los enfoques basados en reglas para la clasificación obtienen resultados prometedores, por lo que también se utilizó un árbol de decisión de bosque aleatorio (RFDT). De igual manera, probaron emplear SVM para determinar si un modelo de clasificación espacial mejoraría en un modelo probabilístico basado en reglas. Por último, generaron un meta-clasificador de votación que produce un resultado de clasificación para cada clasificador base, creando así un ensamble. Su mejor resultado fue con una precisión de 89 %.

Waseem y Hovy (2016), evalúan la influencia de distintas características en la predicción del lenguaje ofensivo, para lograr este cometido, emplearon como modelo la regresión logística con una validación cruzada con 10 iteraciones. Las características que se tomaron en consideración para evaluar el modelo fueron n-gramas

con secuencias hasta de 4 palabras dependiendo la clase que se quería identificar. Este modelo obtuvo una precisión de 72.87 %.

Burnap y Williams (2016), intentan una aproximación con dos modelos del aprendizaje automático: SVM lineales y árboles de decisión de bosque aleatorios. Para el entrenamiento de estos modelos utilizaron como características del texto los n-gramas con secuencias hasta de 5 palabras, las raíces de las palabras, así como términos y frases del lenguaje ofensivo basados en raza, discapacidad y orientación sexual. El mejor resultado obtenido con esta aproximación fue de 79 %.

Davidson et al. (2017), primero proponen usar el modelo de clasificación de regresión logística con una regularización L1 para reducir la dimensionalidad de los datos. Sin embargo, después probaron con otra variedad de modelos como: regresión logística, Naive Bayes, árboles de decisión, bosques aleatorios y SVM lineales. Para estos modelos emplearon una validación cruzada de 5 veces, y reservaron el 10 % del conjunto de datos para la evaluación de los modelos. Las características que extrajeron para los modelos fueron: n-gramas con secuencias de 1, 2 y 3 palabras, POS, lexicones para asignar sentimientos, conteo de *hastags*, menciones, retweets, URL, caracteres, palabras y sílabas, para calcular la calidad de cada tuit se utilizó el grado de nivel de Flesch-Kincaid y las puntuaciones de facilidad de lectura de Flesch. Su mejor modelo fue regresión logística con una regularización L2, obteniendo una precisión de 91 %. Dicho modelo fue entrenado usando en su totalidad el conjunto de datos, y usaron el marco de uno contra el resto, donde se entrena un clasificador separado para cada clase y se asigna la etiqueta de clase con la probabilidad más alta predicha.

Vigna et al. (2017), proponen el uso de dos clasificadores diferentes. El primero basado en SVM y el segundo fue una red de memoria a corto/largo plazo (LSTM, por sus siglas en inglés) donde buscan capturar dependencias de largo alcance en los tuits que puedan desempeñar un papel importante. Para esta aproximación se empleó un enfoque basado en textos no etiquetados morfosintácticamente. Con la finalidad de mejorar la clasificación de los modelos sugeridos, utilizaron recursos

como word2vec (Mikolov et al., 2013), bolsa de palabras continuo (CBoW, por sus siglas en inglés), polaridad de sentimiento, entre otras características como lexicones. Su mejor modelo fue el basado en SVM con una precisión de 80 %.

3.2 Modelos de aprendizaje profundo

Badjatiya et al. (2017b), sugirieron experimentos con múltiples clasificadores como regresión logística, bosques aleatorios, SVM, gradiente de árboles de decisión potenciados (GBDTs), y modelos de redes neuronales profundas (DNN). Para las DNN, eligieron tres arquitecturas de redes neuronales y para cada una de ellas utilizaron *embeddings* aleatorios o GloVe *embeddings* (Pennington et al., 2014). El primer modelo propuesto fue una red neuronal convolucional (CNN, por sus siglas en inglés) con una arquitectura previamente empleada para la clasificación de sentimientos propuesta por Kim (2014). El segundo modelo fue una red LSTM. Por último, el tercer modelo propuesto fue FastText (Bojanowski et al., 2016), donde se representa un documento mediante un promedio de vectores similar a la bolsa de palabras, pero con la diferencia que permite la actualización de los vectores a través de la retropropagación durante el proceso de entrenamiento; esta red además de aprender los pesos, también aprenden *embeddings* de palabras específicas de la tarea a realizar. Su mejor clasificador se logró unificando la DNN y el clasificador GBDT, obteniendo una precisión de 93 %.

Ozler et al. (2020), sugieren el uso de un ajuste fino del modelo BERT (Devlin et al., 2018) para la predicción del lenguaje no civil de múltiples dominios y múltiples clases. Para los modelos de múltiples dominios propusieron tres distintos clasificadores: (1)*Single*, donde ajustan un clasificador para cada dominio; (2)*Joint*, donde un clasificador se ajusta con precisión en los datos de entrenamiento combinados de todos los dominios; (3)*Joint-Single*, donde se ajusta un clasificador conjunto. Luego, los parámetros del clasificador conjunto se utilizan para inicializar n clasificadores individuales, uno para cada dominio. Su mejor modelo obtuvo una precisión

de 83 %.

Mozafari et al. (2020) presentan un marco para la detección de discursos de odio, el análisis y mitigación de sesgos involuntarios. Este enfoque contiene dos módulos principales: (1) módulo de detección del discurso de odio y (2) módulo de mitigación de sesgos; donde el componente BERT(Devlin et al., 2018) se comparte entre los dos módulos. Realizaon múltiples combinaciones de BERT con CNNS y su mejor aproximación obtuvo una precisión de 85 %.

Khan, Fazil et al. (2022), proponen el modelo lingüístico basado en la transferencia, BERT (Devlin et al., 2018), para extraer la representación contextual de las palabras. Aunado a esto, integraron una LSTM bilateral (BiLSTM) con una CNN profunda para incorporar las dependencias de largo alcance entre las palabras semánticamente similares. También aplicaron una atención de alto nivel sobre la representación codificada de la CNN profunda y un mecanismo de atención de bajo nivel sobre la salida de la BiLSTM. Utilizaron los dos niveles del mecanismo de atención para asignar pesos variables a las características tanto de la CNN profunda como de la BiLSTM. Para este trabajo se utilizaron 3 conjuntos de datos: El primer conjunto de datos (HD1) fue proporcionado por el trabajo de Davidson et al. (2017); el segundo conjunto (HD2) lo construyeron a partir de un conjunto de datos de una competición de Kaggle[1]; por último, el tercer conjunto (HD3) fue creado a partir de los tuits de incitación al odio y los tuits normales de los conjuntos HD2 y HD3. La mejor aproximación obutuvo una precisión de 96 % para el conjunto HD2.

Khan, Kamal et al. (2022), proponen un modelo HCovBi-Caps, el cual es un modelo de aprendizaje profundo novedoso de extremo a extremo para la detección del discurso de odio mediante una unidad recurrente bidireccional de entrada (BiGRU), una capa convolucional y una red de cápsulas para incorporar la información contextual en diferentes orientaciones para la detección del discurso de odio. Para este modelo utilizan como características GloVe *embeddings* (Pennington et al., 2014). Su mejor aproximación obtuvo una precisión de 90 % sobre un conjunto de

[1]https://www.kaggle.com

datos obtenido de Kaggle.

3.3 Conjuntos de datos

Burnap y Williams (2015), recolectaron un conjunto de datos de Twitter durante un periodo de tiempo de dos semanas tras el asesinato del baterista Lee Rigby. Se recolectaron un total de 450,000 tuits y se seleccionaron 2,000 para ser anotados por personas. Los anotadores recibieron los tuits y la siguiente pregunta, "¿Es este texto ofensivo o antagónico en términos de raza, etnia o religión?". Asimismo, se les presentó un conjunto ternario de clases: si, no e indeciso. Para lograr esta clasificación se utilizó la plataforma CrowdFlower. Los resultados del ejercicio de anotación produjeron un conjunto de datos de 1,901 tuits con 222 instancias de contenido ofensivo o antagónico (11.68 % de la muestra anotada), y 1,679 instancias de comentarios no ofensivos o antagónicos (88.32 %).

Davidson et al. (2017), extrajeron un corpus de 85.4 millones de tuits haciendo uso de un lexicón que contiene palabras y frases identificadas por los usuarios de internet como discurso de odio y la interfaz de programación de aplicaciones (API, por sus siglas en inglés) de Twitter. De dicho corpus, extrajeron una muestra aleatoria de cerca de 25,000 tuits, el cual fue manualmente anotado por trabajadores de CrowdFlower (después llamado Figure Eight y posteriormente Appen)[2], el cual fue dividido en 3 clases diferentes: *hate speech*, *offensive language* y *neither*. Para lograr esto se les pidió a los etiquetadores que pensaran no solo en las palabras que aparecían en el tuit, sino también el contexto en el que se encontraban. Cada tuit fue anotado por tres o más personas y se utilizó la decisión mayoritaria para asignarle una clase.

Vigna et al. (2017), crearon un corpus de comentarios recuperados de páginas públicas de Facebook de periódicos, políticos, artistas y grupos italianos. En

[2]http://www.appen.com/figure-eight-is-now-appen/

total, se seleccionaron 8 páginas para extraer los datos y se recolectaron poco más de 17,500 comentarios de solo 99 publicaciones de dichas páginas, de estos comentarios, 6,502 fueron anotados al menos una vez y como máximo cinco anotadores. Los etiquetadores fueron 5 estudiantes de licenciatura, a los cuales se les pidió que asignaran una de las tres clases posibles a cada tuit, dichas clases son: *no hate*, *weak hate* y *strong hate*.

Badjatiya et al. (2017b) usan un conjunto de datos con 16 mil tuits anotados distribuidos en 3 clases principales, para las cuales el 21 % corresponde a la clase *sexist*, 12 % a la clase *racist* y 67 % a la clase *neither*. Este conjunto de datos se encuentra disponible gracias al trabajo de Waseem y Hovy (2016).

3.4 Resumen

La tabla 3.1 muestra un resumen del estado del arte mostrado en esta sección contemplando las características que se utilizaron para los trabajos, así como sus mejores modelos y las métricas que obtuvieron.

Tabla 3.1: Resumen del estado del arte ordenado de mayor a menor desempeño.

Año	Algoritmos	Características	Mejor modelo	Precisión	Recall	F1-Score	Artículo
2004	SVM	BoW, n-gramas y POS	SVM	0.90	0.90	0.90	Greevy, 2004
2015	Árbol de decisión de bosque aleatorio, SVM y regresión logística bayesiana	N-gramas y dependencias tipadas	Ensamble de modelos	0.89	0.69	0.77	Burnap y Williams, 2015
2016	Árbol de decisión de bosque aleatorio y regresión logística	N-gramas, raíces y dependencias tipadas	SVM	0.79	0.59	0.68	Burnap y Williams (2016)
2016	Regresión logística	N-gramas	Regresión logística	0.72	0.77	0.73	Waseem y Hovy, 2016
2017	GBDT, DNN, CNN	—	DNN + GBDT	0.93	0.93	0.93	Badjatiya et al., 2017b
2017	Regresión logística, Naive Bayes, árboles de decisión, bosques aleatorios y SVM lineales	POS, sentimiento, menciones, retweets, URL y número de caracteres, palabras y sílabas	Regresión logística	0.91	0.9	0.9	Davidson et al., 2017
2017	SVM, LSTM	POS, sentimiento, word2vec, CBoW, n-gramas y otras características	SVM	0.80	0.78	0.78	Vigna et al., 2017
2020	BERT + CNN	*Word embeddings*	BERT + CNN	0.85	0.88	0.86	Ozler et al., 2020
2020	BERT	*Word embeddings*	BERT	0.83	0.67	0.74	Mozafari et al., 2020
2022	BiLSTM + CNN	*Word embeddings*	BiLSTM + CNN	0.96	0.87	0.91	Khan, Fazil et al., 2022
2022	HCovBi-Caps	*Word embeddings*	HCovBi-Caps	0.90	0.80	0.84	Khan, Kamal et al., 2022

Capítulo 4

Conjunto de datos

Para este trabajo utilizaremos el conjunto de datos recolectado por Davidson et al. (2017). Como se describe en la sección 3.3, este conjunto de datos consta de publicaciones extraídas de Twitter. Asimismo, se encuentra disponible en formato CSV en el repositorio de GitHub del autor, y consta de 6 columnas, las cuales se describen a continuación:

- *count*: se refiere al número de usuarios de CrowdFlower que se encargaron de etiquetar cada tuit. El conteo mínimo para cada tuit fue de 3 personas, sin embargo, existieron algunos casos donde cada tuit fue codificado por más usuarios cuando se determinó que los juicios de la plataforma CrowdFlower no fue viable.
- *hate_speech*: hace referencia al número de usuarios de CrowdFlower que juzgaron el tuit como discurso de odio.
- *offensive_language*: hace referencia al número de usuarios de CrowdFlower que juzgaron el tuit como lenguaje ofensivo.
- *neither*: hace referencia al número de usuarios de CrowdFlower que juzgaron el tuit era neutro.
- *class*: muestra la etiqueta de clase para la mayoría de los usuarios de CrowdFlower, donde 0 corresponde a *hate speech*, 1 a *offensive language* y 2 a *neither*.

- *tweet*: contiene el texto correspondiente al tuit etiquetado.

Para este trabajo, consideraremos las clases *hate speech* y *offensive language* como texto tóxico, ya que ambas etiquetas hace alusión a algún tipo de texto ofensivo, por otra parte, la clase *neither* será considerada como texto "limpio" o no tóxico.

En la tabla 4.1 se puede observar una muestra del conjunto de datos original. Mientras que en la figura 4.1 observamos la distribución de documentos entre clases del conjunto de datos, donde tenemos $1,430$ documentos para la clase *hate speech*, $19,189$ para la clase *offensive language* y $4,162$ para la clase *neither*, dando así un total de $24,781$ documentos.

Tabla 4.1: Muestra del conjunto de datos.

count	*hate_speech*	*offensive_language*	*neither*	*class*	*tweet*
3	0	0	3	2	!!! RT @user: *as a woman you shouldn't...*
3	0	3	0	1	!! RT @user: *boy dats cold...tyga dwn ...*
6	0	6	0	1	!!!!!!!!!!!! RT @user: *the shit you...*

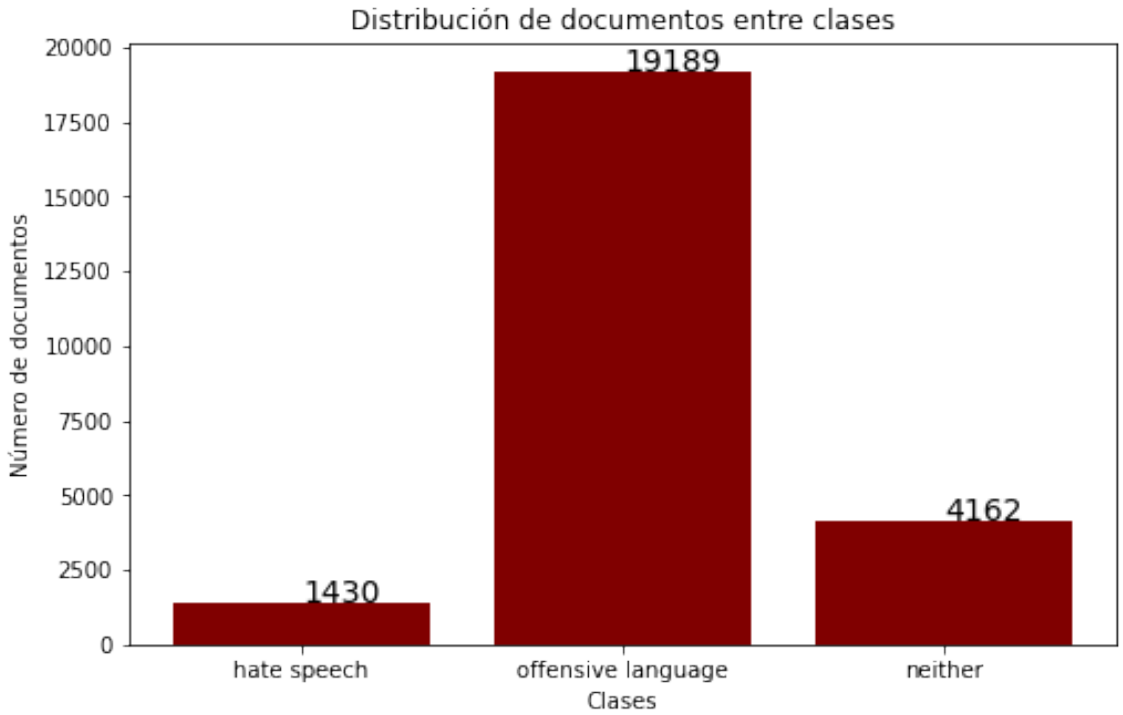

Figura 4.1: Distribución de documentos entre clases.

Capítulo 5

Metodología

En esta sección se describirá de manera concisa el proceso que se llevó a cabo en este trabajo para darle una solución a la hipótesis planteada. En la figura 5.1 podemos observar un bosquejo de la solución propuesta.

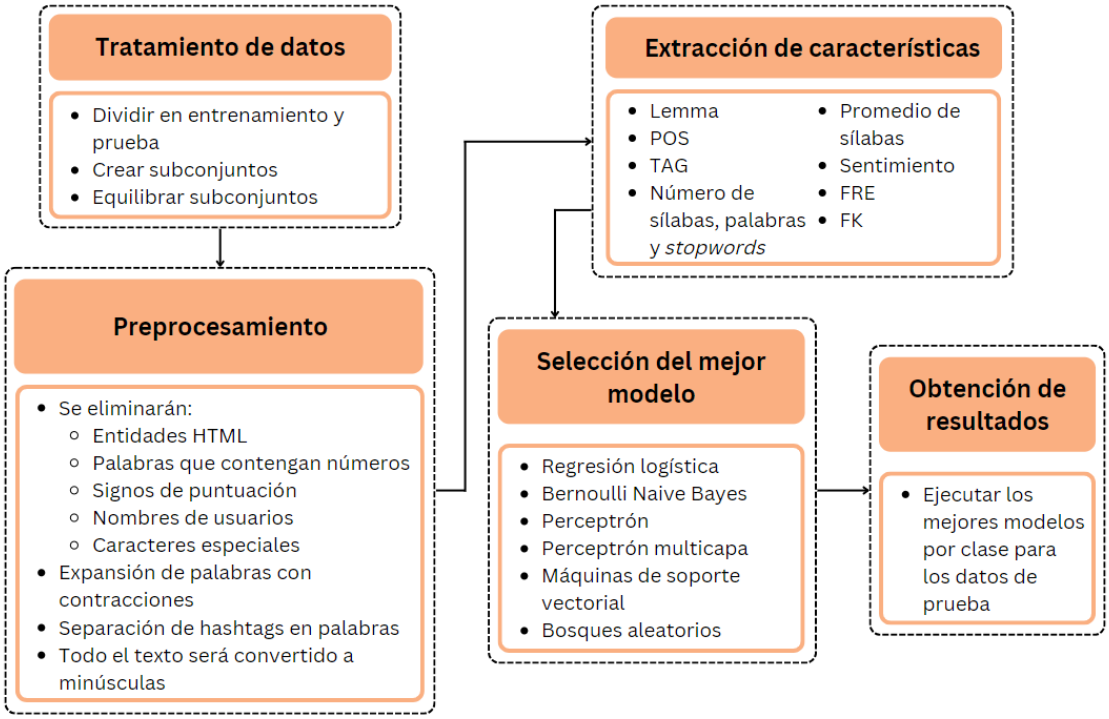

Figura 5.1: Bosquejo para la solución del trabajo propuesto.

5.1 Tratamiento de datos

El primer problema al que nos enfrentamos en la solución de este trabajo son los datos per se. Como se pudo observar en la figura 4.1, nuestra distribución de documentos por clase se encuentra muy desbalanceada. Por otro lado, con la finalidad de comparar este trabajo contra el estado del arte al utilizar este conjunto de datos (Davidson et al., 2017), es necesario utilizar la misma partición de datos que se utilizarán tanto para el entrenamiento como para la prueba de los modelos a implementar. Entonces:

En el trabajo de Davidson et al., 2017, se emplea una partición del conjunto de datos del 90 % para el entrenamiento y del 10 % para pruebas. Por esta razón, el primer paso para la manipulación de los datos será realizar dicha partición del conjunto de datos. El resultado de esta partición la podemos observar en la tabla 5.1, donde obtenemos un total de $22,302$ documentos para el conjunto de entrenamiento y un total de $2,479$ documentos para el conjunto de prueba.

Tabla 5.1: Distribución de entrenamiento y prueba del conjunto de datos.

Conjunto de datos	Número de documentos
Entrenamiento	22,302
Prueba	2,479
Total	24,781

Con esta partición de datos, ahora tenemos que para el conjunto de entrenamiento $1,283$ documentos pertenecen a *hate speech*, $17,305$ pertenecen a *offensive language* y $3,714$ pertenecen a *neither*. Por otro lado, para el conjunto de pruebas 147 documentos pertenecen a *hate speech*, $1,884$ pertenecen a *offensive language* y 448 pertenecen a *neither*. Estas distribuciones las podemos observar en las figuras 5.2 y 5.3.

A partir de que resolvimos el problema de la partición de los datos, ahora solo nos enfocaremos en el conjunto de entrenamiento, y dejaremos a un lado el conjunto de pruebas debido a que este conjunto no se puede manipular.

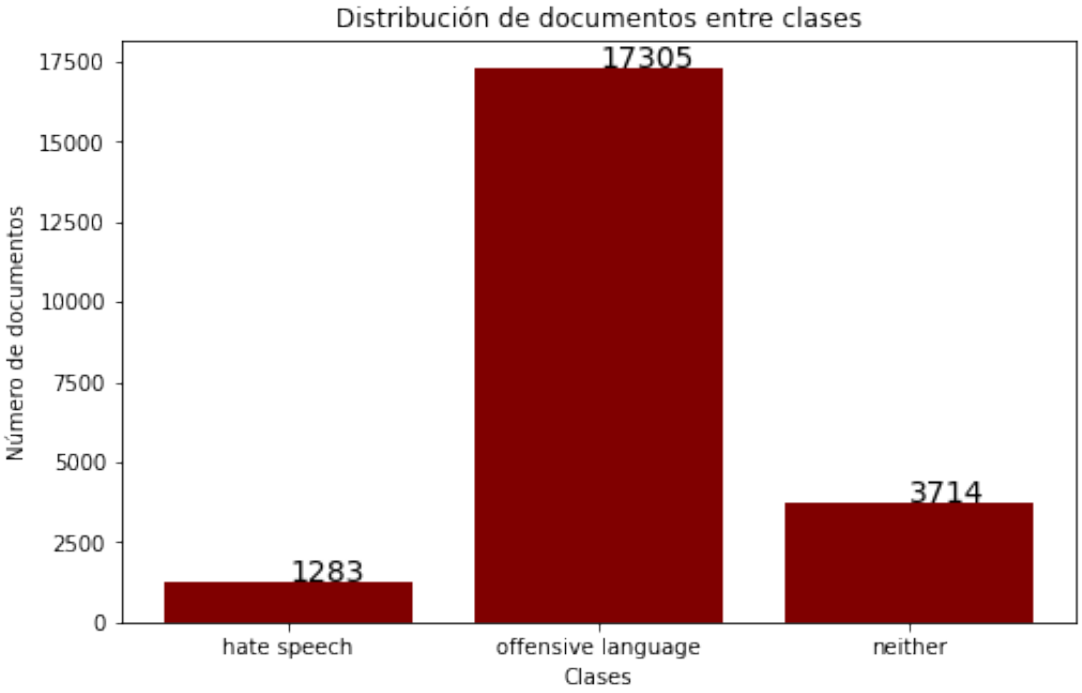

Figura 5.2: Conjunto de entrenamiento.

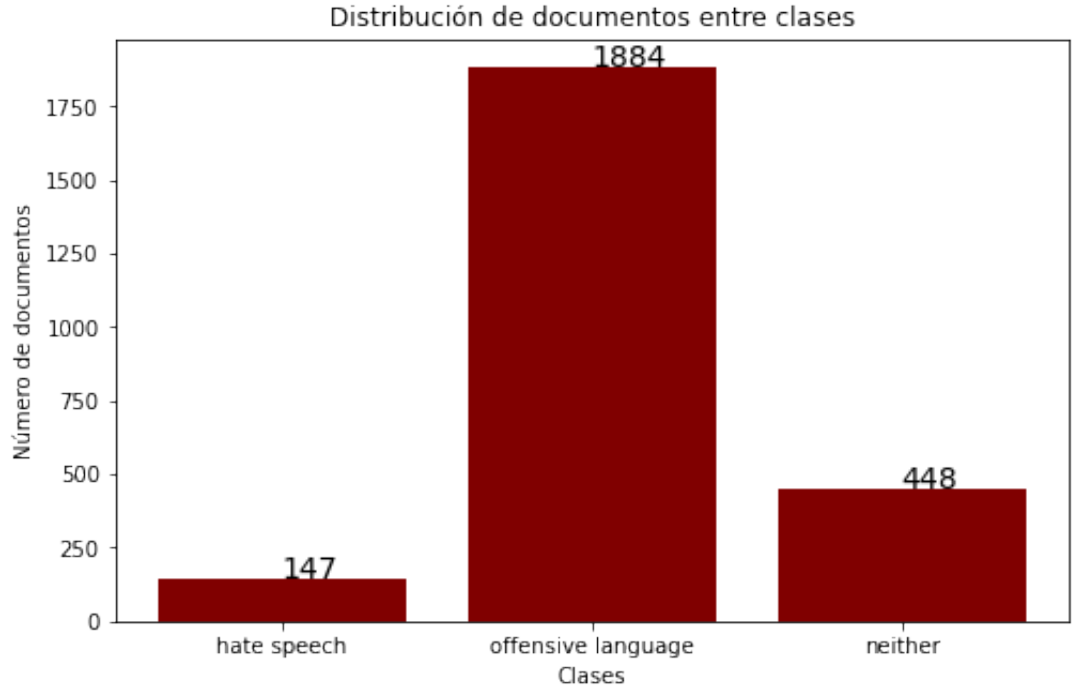

Figura 5.3: Conjunto de prueba.

El siguiente problema a resolver es el desbalance de los datos. El enfoque que daremos para resolver esto será crear subconjuntos de datos equilibrados, es decir, cada subconjunto creado corresponderá directamente a una clase. Para lograr esto es necesario primeramente separar nuestro conjunto de entrenamiento en subconjuntos. Una vez tengamos estos subconjuntos tenemos que considerar como realizar el equilibrio de los datos. Para esto, consideraremos a la clase *neither* como los documentos "no tóxicos" y a las clases *hate speech* y *offensive language* como documentos "tóxicos". Por lo que, la máxima cantidad de documentos "limpios" con la que contamos para equilibrar los subconjuntos "tóxicos" es de $3,714$.

Tabla 5.2: Número de documentos por subconjunto.

	Subconjuto	Documentos tóxicos	Documentos no tóxicos	Total
0	*hate_speech*	1283	1283	2566
1	*offensive_language*	3714	3714	7428
2	*neither*	3714	3714	7428

La tabla 5.2 muestra la cantidad de documentos que tendrán nuestros subconjuntos con los que ya podemos trabajar. Para la clase *hate speech* tendremos un total de $2,566$ documentos, los cuales se encuentran repartidos en 1,283 documentos corrrespondientes a la clase *hate speech* del conjunto de pruebas y 1,283 documentos correspondientes a la clase *neither* que contiene documentos que no corresponden a ninguna de las clases (*hate speech* y *offensive language*); para la clase *offensive language* un total de $7,428$ y se encuentra distribuida de igual manera en $50\,\%$ de documentos correspondientes a la clase *offensive language* y $50\,\%$ a la clase *neither*; por último, para la clase *neither* tenemos un total de $7,428$ documentos. Los cuales ya se encuentran con un $50\,\%$ de documentos "no tóxicos" que corresponden directamente a la clase *neither* y un $50\,\%$ de documentos "tóxicos" que corresponden a una muestra tomada de la combinación de *hate speech* y *offensive language*.

5.2 Preprocesamiento

Una vez equilibrados nuestros conjuntos de datos, el siguiente problema a resolver es el preprocesamiento de nuestros documentos. Para poder realizar de manera correcta este paso, es necesario realizar un análisis de en que forma tenemos estos documentos. A continuación, se muestran algunos ejemplos del conjunto de datos:

- !!! RT @user: *As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...*
- *"" fuck no that bitch don't even suck dick "" 😂😂😂 the Kermit videos bout to fuck IG u*
- *"@user: This bitch was so ungrateful* http://t.co/06e77bGwbx *" fr ..... LULWHORE*
- *"@user: #NewSATQuestions"*

Como se puede observar en la muestra anterior de los documentos del conjunto de datos, hay muchos problemas por resolver con la finalidad de que el texto nos quede lo más legible posible. Por esta razón, el preprocesamiento que se llevará a cabo será el siguiente:

- **Expansión de contracciones.** En el idioma inglés se pueden crear muchas contracciones con la finalidad de juntar dos palabras. Sin embargo, esto es un problema para la creación de vocabularios en PLN, ya que se pueden encontrar las palabras con y sin contracciones. Por tal motivo, palabras como *shouldn't* serán separadas a sus palabras originales en el idioma, *should not*.
- **Separación de *hastags* en palabras.** Uno de los problemas con los que nos enfrentamos al trabajar con documentos de Twitter, son los *hastags*, ya que estos pueden tener información importante acerca de lo que se está hablando. Por tal motivo, los *hastags* serán separados en palabras. Un ejemplo de ello se muestra a continuación: *"#NewSATQuestions"* se separaría de la siguiente manera *New SAT Questions*.
- **Conversión de emojis a texto.** Al trabajar con conjuntos de datos de redes

sociales, nos enfrentamos a distintas formas de expresar nuestras ideas. Un cambio en la escritura habitual a las redes sociales son el uso de emojis, que suelen expresar algún tipo de sentimiento o alguna acción específica. Por tal motivo se utilizó un diccionario de emojis, el cual contiene una lista de emojis y su respectivo significado en texto.

- **Eliminación de diversas entidades.** Se describen a continuación:
  - **HTML.** Entidades como "&amp" o " " serán eliminadas, ya que no aportan valor a nuestro texto.
  - **Enlaces web.** Al igual que las entidades HTML, los enlaces a sitios web, no nos aportan valor a nuestro texto, y lo hace más difícil de leer. Es por esto por lo que serán eliminados.
  - **Palabras que contengan números o caracteres especiales (*Leet speaking*).** Dado que esta es otra corriente de investigación que no se acota a la investigación de este trabajo, este tipo de escritura será descartado. Un ejemplo de estas palabras puede ser: "y0u", "4m4z1ng", "0nly".
  - **Signos de puntuación.** Dado que la mayoría de datos de todas clases contienen una gran cantidad signos de puntuación sin algún sentido. Se optó por retirarlos y conservar solo el texto.
  - **Nombres de usuarios.** Con fines de mantener en el anonimato a los usuarios autores de las publicaciones, los nombres de usuarios no serán parte del entrenamiento de nuestros modelos.
  - **Caracteres especiales.** Como sucede en el caso de los signos de puntuación, el uso excesivo de caracteres especiales en todas las clases no aporta a la lectura y entendimiento del texto.
- **Transformación del texto a minúsculas.** Con la finalidad de evitar palabras iguales, pero con la diferencia del uso o no de mayúsculas, todos los documentos serán transformados a minúsculas.

Al final de nuestro preprocesamiento, obtendremos los siguientes resultados:

- *rt as a woman you should not complain about cleaning up your house as a man*

you should always take the trash out

- *fuck no that bitch do not even suck dick face with tears of joy face with tears of joy face with tears of joy the kermit videos bout to fuck ig u*
- *this bitch so ungrateful fr lulwhore*
- *new sat questions*

Los resultados que obtenemos al realizar el preprocesamiento de nuestros documentos muestra una gran diferencia en cuanto a la lectura de los mismos se refiere. Sin embargo, aún se encuentran algunos sesgos que hay que corregir, como la jerga que se utiliza en Twitter y las redes sociales en general. Asimismo, las palabras mal escritas son un gran problema.

5.3 Extracción de características.

Con una mejor comprensión de los documentos de nuestro conjunto de datos debido al preprocesamiento que se llevó a cabo, podemos proceder a la extracción de características que tomaremos en cuenta para el entrenamiento de los algoritmos de aprendizaje automático que emplearemos. Dado que los documentos en el conjunto de datos seleccionado son bastante similares entre clases, se buscará encontrar alguna diferencia y por esta razón utilizaremos las siguientes características:

- **Lematización.**
- **POS.**
- **TAG.**
- **Otras características:**
  - **Conteo de sílabas, palabras y *stopwords*.**
  - **Promedio de sílabas.**
  - **Sentimiento.**
  - **FRE.**
  - **FK.**

Un ejemplo del proceso de extracción de características se muestra a continuación, donde presentamos un ejemplo dado un texto:

- **Texto original:** *you should not be sad you should be happy smile it be happy smiling face.*
- **Lemas:** *you, should, not, be, sad, you, should, be, happy, smile, it, be, happy, smile, face.*
- **POS:** *PRONOUN, AUXILIARY, PARTICLE, AUXILIARY, ADJECTIVE, PRONOUN, AUXILIARY, AUXILIARY, ADJECTIVE, NOUN, PRONOUN, AUXILIARY, ADJECTIVE, VERB, NOUN.*
- **TAG:** *PRONOUN PERSONAL, VERB MODAL AUXILIARY, ADVERB, VERB BASE FORM, ADJECTIVE ENGLISH, PRONOUN PERSONAL, VERB MODAL AUXILIARY, VERB BASE FORM, ADJECTIVE ENGLISH, NOUN SINGULAR OR MASS, PRONOUN PERSONAL, VERB BASE FORM, ADJECTIVE ENGLISH, VERB GEROUND OR PRESENT PARTICIPLE, NOUN SINGULAR OR MASS.*
- **Otras características:** número de sílabas: 22, número de caracteres: 72, número de palabras: 15, promedio de sílabas: 1.5, palabras únicas: 10, sentimiento: *POSITIVE*, número de *stopwords*: 9, FRE: 8-9, FK: 8.

Para la solución de este problema, se intentó emplear todas las combinatorias posibles de las características seleccionadas. Sin embargo, al final se optó por elegir todas. Para esto, se generaron diferentes vocabularios de palabras: el primer vocabulario se generó transformando el texto original de los documentos de los subconjuntos de datos a sus raíces (lematización); el segundo se creó transformando el texto original a sus respectivas características gramaticales (POS); el tercero se generó con las características gramaticales específicas (POS); el cuarto y último se creó al trabajar con el texto de cada uno de los documentos, cambiando así el texto del documento por las características faltantes (conteo de sílabas, palabras y *stopwords*, promedio de sílabas, sentimiento, FRE y FK). Estos cuatro vocabularios se unificaron para generar uno solo. Para la creación de dichos vocabularios se tomaron en cuenta tres características:

- **N-gramas.** Se consideraron secuencias de 1, 2 y 3, es decir, unigramas, unigramas más bigramas y unigramas más bigramas más trigramas.
- **Frecuencia de aparición mínima.** Con la finalidad de eliminar palabras o secuencias de palabras que pudiesen generar sesgos al estar mal escritas, se optó por que los vocabularios tuvieran existieran al menos 3 veces en todos los documentos. Más adelante, como prueba de robustez, este número subiría a 5.
- **Vectorizador.** Se utilizaron los vectorizadores por conteo y TF-IDF para cada uno de los vocabularios con cada una de las secuencias de n-gramas.

La figura 5.4 muestra gráficamente como es que se extrajeron las características de cada uno de los documentos, así como la cantidad de experimentos finales que se realizaron con sus respectivos nombres que se les asignaron. Dichos nombres tienen la siguiente nomenclatura: TF o TFI corresponden a los vectorizadores, ya sea Tf o Tf-Idf respectivamente; 1G, 12G y 123G corresponden a la secuencia de n-gramas que se usaron; por último, FM3 y FM5 hacen referencia a la frecuencia mínima de aparición de cada n-grama.

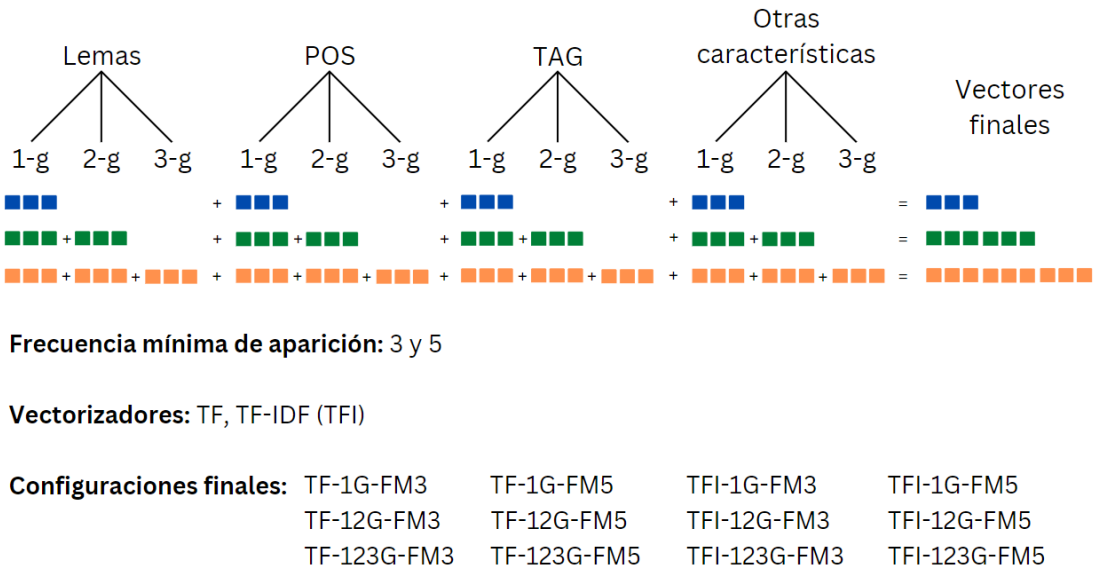

Figura 5.4: Extracción de características.

5.4 Selección del mejor modelo

Para esta fase de la metodología se buscó encontrar el mejor modelo con los mejores parámetros para cada uno de los subconjuntos que tenemos hasta ahora. Dicho procedimiento se llevó como se describe a continuación:

- Se eligieron los modelos que mejor desempeño obtuvieron de acuerdo a la literatura revisada en el capítulo 3. Además, se seleccionaron algunos otros modelos de aprendizaje automático para clasificación. De tal manera que los modelos que se utilizaron como posible mejor clasificador fueron: Regresión logística, Bernoulli Naive Bayes, Perceptrón, Perceptrón multicapa, Máquinas de soporte vectorial, Bosques aleatorios.
- Una vez que obtuvimos nuestra lista de posibles modelos, se empleó una malla de búsqueda, la cual es un algoritmo cuyo objetivo principal es encontrar los mejores parámetros para los modelos otorgados. Para lograr esta selección, es necesario otorgarle al algoritmo una serie de parámetros, los cuales para este trabajo en específico fueron:
  - Para el modelo de regresión logística: el posible solucionador, el número máximo de iteraciones y la penalización.
  - Para el modelo Bernoulli Naive Bayes: las prioridades con las que el modelo se ajusta a las clases.
  - Para el perceptrón: la penalización y el parámetro *alpha*.
  - El perceptrón multicapa es un caso específico, ya que probar múltiples arquitecturas de capas ocultas puede llegar a ser muy complejo. Por esta razón, se optó simplemente por utilizar una arquitectura de dos capas ocultas. La selección de los nodos de estas dos capas fue la opción que se le pidió a la malla que buscara.
  - Para las máquinas de soporte vectorial lineal: el número de iteraciones máximas y la penalización.
  - Por último, para los bosques aleatorios: la profundidad máxima de búsqueda de los árboles.

La selección de modelos se puede observar gráficamente en la figura 5.5.

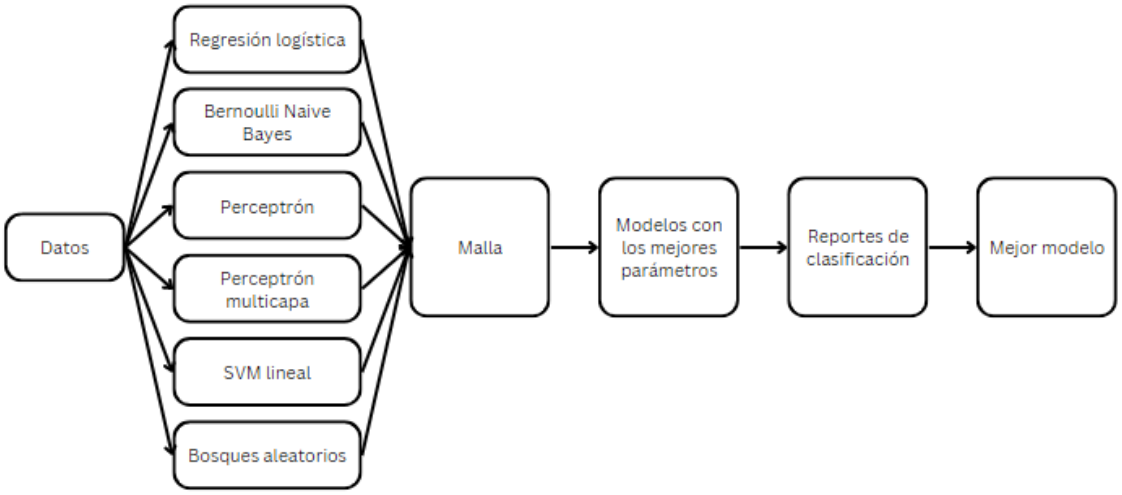

Figura 5.5: Selección del mejor modelo.

5.5 Obtención de resultados

La última fase de la metodología fue la obtención de los resultados. Para esta sección se realizaron dos pasos:

- **Ensamble de modelos.** Primero se seleccionaron los mejores modelos para cada una de las clases. Hasta este punto se retoma el conjunto de datos de prueba para predecir las probabilidades de pertenencia a cada una de las clases con cada uno de los modelos. Para cada uno de los documentos se creó un vector con las tres probabilidades de pertenencia y la mayor de todas es la que se consideró como la clase predicha por el ensamble de modelos. Este proceso se puede observar gráficamente en la figura 5.6.
- **Obtención de resultados.** Los vectores finales dados por el ensamble de

modelos serán comparados con las clases verdaderas del conjunto de datos de prueba. Con esta comparación podemos obtener las matrices de confusión y los reportes de clasificación.

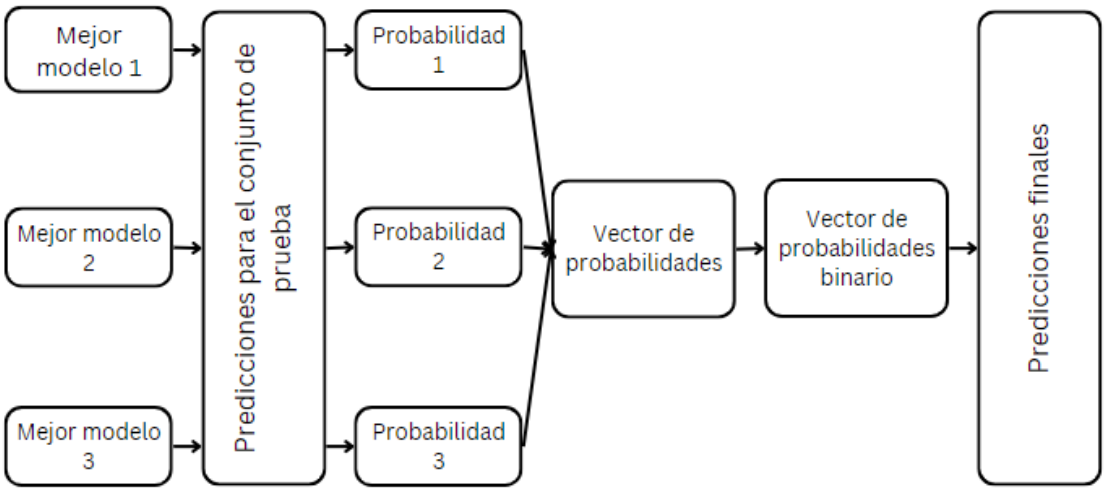

Figura 5.6: Ensamble de modelos.

Capítulo 6

Experimentos y resultados

Al realizar la búsqueda de los mejores modelos a través de la malla, se obtuvieron los resultados mostrados en la tabla 6.1, la cual muestra los mejores parámetros obtenidos para cada uno de los modelos con cada una de las diferentes combinaciones que se realizaron. Los parámetros descritos en la tabla se encuentran en el orden en el que son mencionados en la sección 5.4.

Al hacer una revisión y observar que los parámetros no cambian mucho entre combinaciones de n-gramas ni entre clases, se optó por generalizar los seis modelos de manera tal que en lugar que existan seis modelos para cada una de las clases, existan solamente seis para las tres clases. Estos seis modelos finales los podemos observar en la tabla 6.2. Con estos modelos, el siguiente paso es ejecutar una validación cruzada de 7 iteraciones y obtener los reportes de clasificación de cada uno de los modelos y basándonos en la métrica *precision*, elegir al mejor modelo para cada una de las tres clases.

Tabla 6.1: Parámetros de los modelos seleccionados.

Subconjunto	Modelo	Unigramas	Unigramas + bigramas	Unigramas + bigramas + trigramas
hate speech	RL	'liblinear', 100 iteraciones 'l2'	'liblinear' 500 iteraciones 'l2'	'liblinear' 500 iteraciones 'l2'
	BNB	0.3, 0.7	0.3, 0.7	0.3, 0.7
	Perceptrón	'l2' 0.0001	'l2' 0.0001	'l2' 0.0001
	MLP	(3, 2)	(3, 2)	(5, 2)
	SVM lineal	100 iteraciones 'l2'	500 iteraciones 'l2'	500 iteraciones 'l2'
	RF	10	10	10
offensive language	RL	'liblinear', 500 iteraciones 'l2'	'liblinear' 500 iteraciones 'l2'	'liblinear' 1,000 iteraciones 'l2'
	BNB	0.3, 0.7	0.3, 0.7	0.3, 0.7
	Perceptrón	'l2' 0.0001	'l2' 0.0001	'l2' 0.0001
	MLP	(3, 2)	(5, 2)	(5, 2)
	SVM lineal	100 iteraciones 'l2'	500 iteraciones 'l2'	1,000 iteraciones 'l2'
	RF	10	10	10
neither	RL	'liblinear', 500 iteraciones 'l2'	'liblinear' 500 iteraciones 'l2'	'liblinear' 1,000 iteraciones 'l2'
	BNB	0.3, 0.7	0.3, 0.7	0.3, 0.7
	Perceptrón	'l2' 0.0001	'l2' 0.0001	'l2' 0.0001
	MLP	(3, 2)	(5, 2)	(5, 2)
	SVM lineal	100 iteraciones 'l2'	500 iteraciones 'l2'	1,000 iteraciones 'l2'
	RF	10	10	10

Tabla 6.2: Modelos generales.

Modelo	Solucionador	Iteraciones	Penalización	Prioridades	Profundidad	*alpha*	Capas ocultas
RL	'liblinear'	1,000	'l2'	–	–	–	–
BNB	–	–	–	0.3, 0.7	–	1.0	–
Perceptrón	–	1,000	'l2'	–	–	0.0001	–
MLP	–	1,000	–	–	–	0.0001	(5, 2)
SVM lineal	–	1,000	'l2'	–	–	–	–
RF	–	–	–	–	10	–	–

6.1 Unigramas

La tabla 6.3 muestra la cantidad de características que se extrajeron para cada una de las clases. Asimismo, nos indica a qué categoría de características corresponde. Al trabajar únicamente con unigramas, es de esperar que obtengamos pocas características del texto, ya que los subconjuntos al tener pocos documentos traen por consecuencia que los diccionarios que se crearon sean muy pequeños.

Por otro lado, las tablas 6.4 y 6.5 muestran los mejores modelos seleccionados con sus respectivos reportes de clasificación. Dichos resultados son obtenidos de la validación cruzada que se realizó, donde en cada iteración se obtuvo un reporte de clasificación para cada clase y al final de las siete iteraciones se realizó un promedio, el cual es el resultado final del entrenamiento.

Tabla 6.3: Cantidad de características (unigramas).

Frecuencia mínima	Modelo	Lemas	POS	TAG	Otras características	Total
3	*hate speech*	1,357	14	33	309	1,713
	offensive language	2,901	14	35	318	3,268
	neither	2,921	16	34	317	3,288
5	*hate speech*	789	14	33	299	1,135
	offensive language	1,793	14	34	312	2,153
	neither	1,814	16	34	313	2,175

Tabla 6.4: Resultados de la validación cruzada para Tf (unigramas).

Clase	TF-1G-FM3				TF-1G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	RF	0.85	0.93	0.78	RF	0.85	0.93	0.78
offensive language	RL	0.95	0.97	0.93	RL	0.95	0.97	0.93
neither	RL	0.95	0.93	0.97	RL	0.95	0.93	0.97

Tabla 6.5: Resultados de la validación cruzada para Tf-Idf (unigramas).

Clase	TFI-1G-FM3				TFI-1G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	RF	0.83	0.92	0.76	RF	0.84	0.92	0.77
offensive language	RL	0.94	0.97	0.91	RL	0.94	0.97	0.91
neither	SVM lineal	0.95	0.93	0.96	SVM lineal	0.94	0.93	0.96

Una vez que tenemos los mejores modelos para cada una de nuestras clases, es momento de proceder con el ensamble y obtener los resultados finales al compararlos contra nuestro conjunto de pruebas. Estos resultados se muestran descritos en las tablas 6.6 y 6.7.

En las figuras 6.1 y 6.2 podemos observar con más detalle los resultados obtenidos por las matrices de confusión para las configuraciones donde los subconjuntos fueron vectorizados con Tf con frecuencias mínimas de 3 y 5 respectivamente. Para la figura 6.1, tenemos que la clase *hate speech* tiende a tener una muy mala predicción con un 13 % de clasificación correcta, ya que está clasificando la mayoría de sus datos pertenecientes como si fueran pertenecientes a la clase *offensive language*; sin embargo, es la única clase que falla al clasificar, puesto que las categorías *offensive language* y *neither* tienen un buen porcentaje de clasificación con un 92 % y 95 % respectivamente. Mientras que para la figura 6.4, la clase *hate speech* obtiene una mejora significativa en su correcta clasificación con un 35 %; por otro lado, las clases *offensive language* y *neither* se mantienen en un rango similar.

En cuanto a las configuraciones donde los subconjuntos fueron vectorizados con Tf-Idf, las figuras 6.3 y 6.4 muestran los resultados con frecuencias mínimas de 3 y 5 respectivamente. En la figura 6.3, podemos apreciar que la clase *hate speech* sigue tendiendo a realizar una mala clasificación; sin embargo, su porcentaje sube un 10 % en comparación con su igual en Tf, solo que ahora clasifica los datos como si fueran pertenecientes a la clase *neither* y no a *offensive language*. Mientras que la clase *neither* sigue manteniendo una buena clasificación de sus datos con un 98 %, la clase *offensive language* muestra un porcentaje de clasificación más bajo que en los datos vectorizados por Tf, con una clasificación correcta de 74 %. De igual manera, en la figura 6.4 los resultados obtenidos para todas las clases son bastante similares en comparación con los resultados para la frecuencia mínima de 3.

Hasta este punto de la experimentación, podemos generar una hipótesis, que importa más la cantidad de aparición de las palabras en un documento que el número de documentos en los que aparecen las palabras. Ya que la parte que

diferencia a Tf-Idf del Tf, pareciera que afecta a la correcta clasificación de los documentos, porque los resultados obtenidos con unigramas son más bajos con Tf-Idf que con Tf. Sin embargo, esta intuición puede cambiar al ver el comportamiento de los modelos al considerar más características.

Tabla 6.6: Resultados de los ensambles para Tf (unigramas).

Clase	TF-1G-FM3				TF-1G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	RF	0.18	0.31	0.13	RF	0.30	0.27	0.35
offensive language	RL	0.93	0.95	0.92	RL	0.92	0.96	0.87
neither	RL	0.83	0.74	0.95	RL	0.82	0.73	0.95

Tabla 6.7: Resultados de los ensambles para Tf-Idf (unigramas).

Clase	TFI-1G-FM3				TFI-1G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	RF	0.20	0.18	0.23	RF	0.21	0.21	0.23
offensive language	RL	0.84	0.98	0.74	RL	0.84	0.98	0.73
neither	SVM lineal	0.68	0.52	0.98	SVM lineal	0.65	0.49	0.98

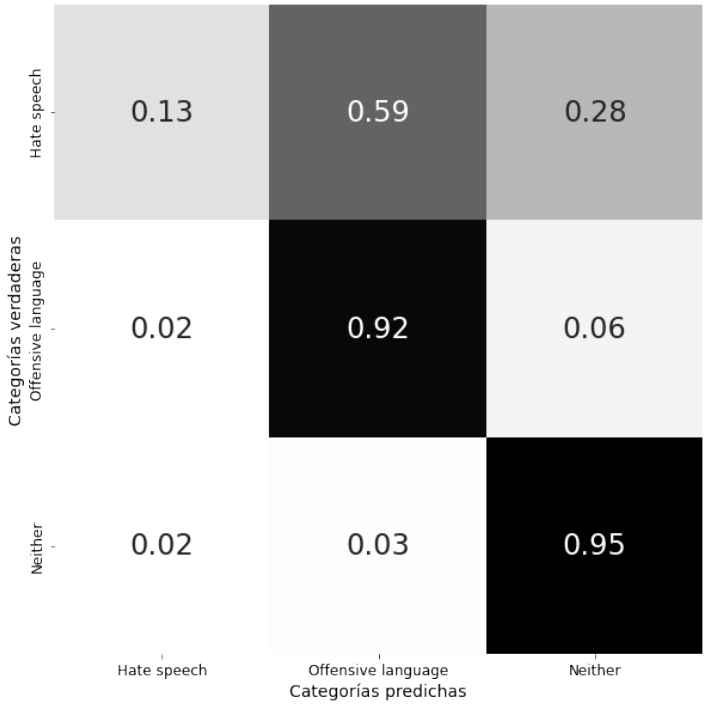

Figura 6.1: Matriz de confusión para la configuración TF-1G-FM3.

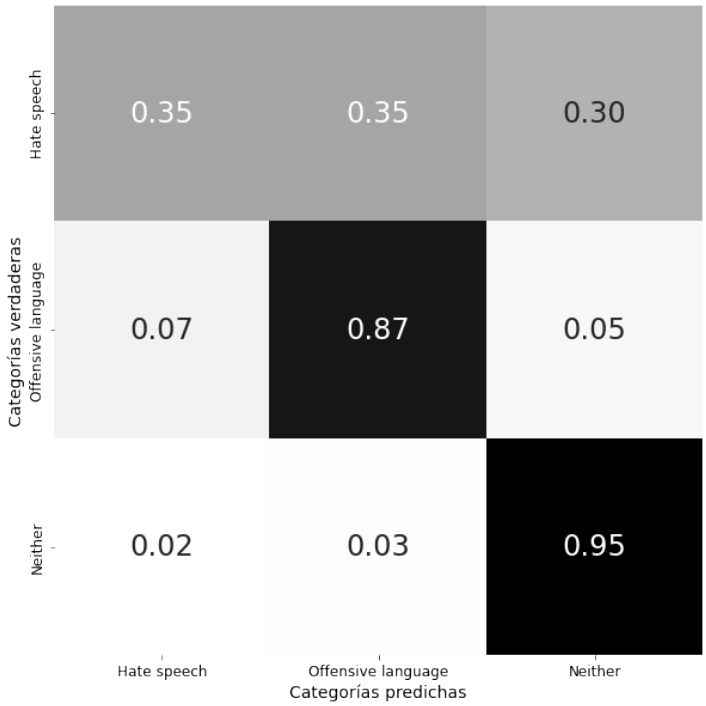

Figura 6.2: Matriz de confusión para la configuración TF-1G-FM5.

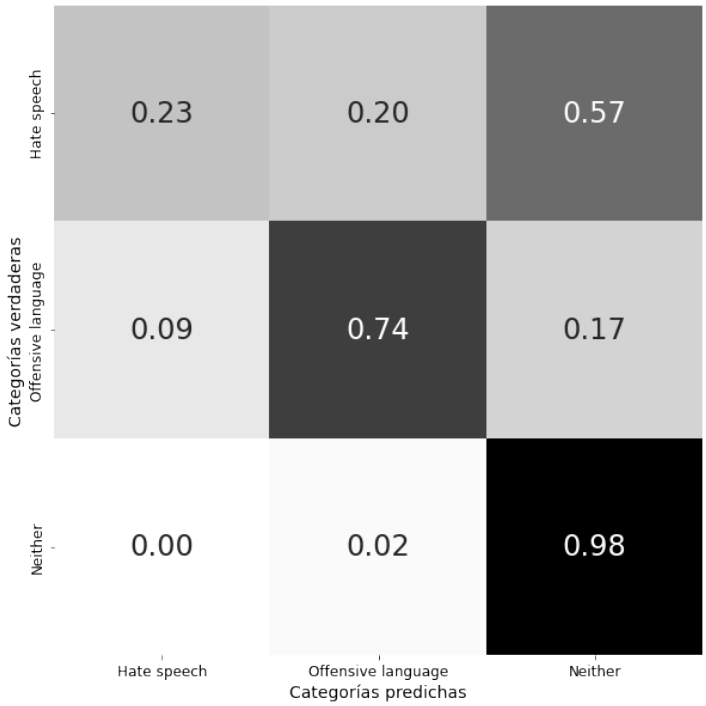

Figura 6.3: Matriz de confusión para la configuración TFI-1G-FM3.

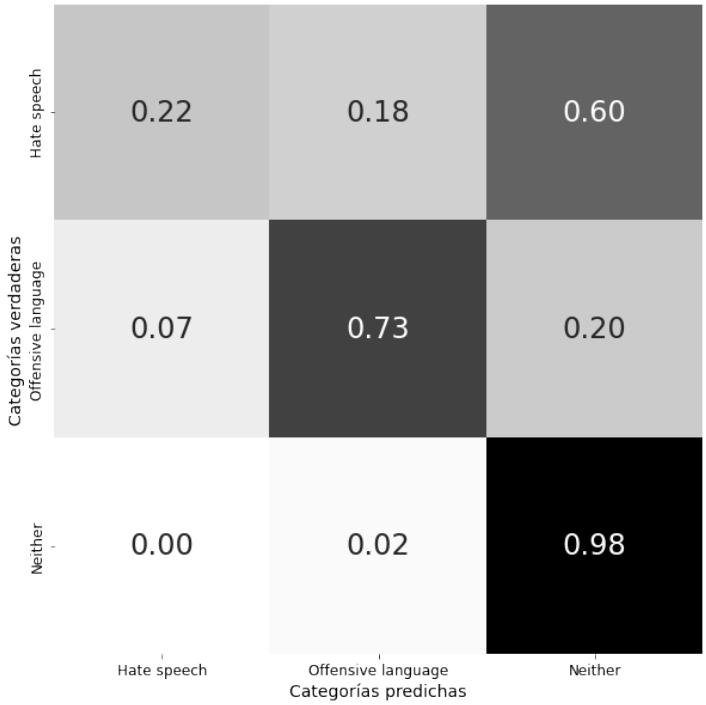

Figura 6.4: Matriz de confusión para la configuración TFI-1G-FM5.

6.2 Unigramas y bigramas

Continuando con la experimentación, ahora veremos cuantas características más obtenemos al no simplemente tener unigramas, sino que ahora también se consideraran los bigramas. En la tabla 6.8 se puede apreciar la cantidad de características que se extrajeron para cada una de nuestras clases. A primera instancia, podemos observar que ahora son más del doble de las características obtenidas solamente con unigramas, lo cual nos podría ayudar a mejorar el rendimiento de los modelos.

Tal y como en el caso de unigramas, las tablas 6.9 y 6.10 nos muestran los mejores modelos obtenidos en el proceso de entrenamiento con sus respectivos reportes de clasificación. Recordemos que estos resultados fueron obtenidos en el proceso de entrenamiento y no son los resultados definitivos, ya que hasta este punto aún no se realiza el ensamble de modelos. También nos podemos percatar que el comportamiento de los modelos durante la etapa de entrenamiento no varía mucho, por el hecho de que independientemente del modelo que se selecciona como el mejor, los resultados son muy similares comparados con los resultados obtenidos en la sección anterior. Para esta sección de experimentos, los mejores modelos que se seleccionaron con frecuencia mínima de aparición de 3 para Tf como extractor de características fueron: regresión logística para *hate speech*, regresión logística para *offensive language* y perceptrón multicapa para *neither*. Por otro lado, los modelos que fueron seleccionados para Tf-Idf como extractor de características fueron: perceptrón para *hate speech*, regresión logística para *offensive language* y perceptrón multicapa para *neither*. De igual manera, los mejores modelos que se seleccionaron con frecuencia mínima de aparición de 5 para Tf fueron: regresión logística para *hate speech*, rperceptrón multicapa para *offensive language* y regresión logística para *neither*. Por otro lado, los modelos que fueron seleccionados para Tf-Idf como extractor de características fueron: regresión logísitica para *hate speech*, perceptrón multicapa para *offensive language* y SVM lineal para *neither*.

Tabla 6.8: Cantidad de características (unigramas).

Frecuencia mínima	Modelo	Lemas	POS	TAG	Otras características	Total
3	*hate speech*	2,587	191	489	309	3,576
	offensive language	7,159	201	614	318	8,292
	neither	7,132	199	620	317	8,268
5	*hate speech*	1,270	183	408	299	2,160
	offensive language	3,701	193	536	312	4,742
	neither	3,714	195	545	313	4,767

Tabla 6.9: Resultados de la validación cruzada para Tf (unigramas y bigramas).

Clase	TF-12G-FM3				TF-12G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	RL	0.89	0.91	0.86	RL	0.88	0.91	0.86
offensive language	RL	0.94	0.96	0.92	MLP	0.89	0.97	0.82
neither	MLP	0.89	0.95	0.85	RL	0.85	0.92	0.96

Al realizar el ensamble de los modelos para el conjunto de pruebas, se obtuvo una mejora significativa para ambos vectorizadores cuando la frecuencia mínima es de 3. En las figuras 6.5 y 6.6 podemos observar las matrices de confusión para los experimentos hechos con el vectorizador Tf con frecuencias mínimas de 3 y 5, respectivamente. Para la frecuencia mínima de 3, obtenemos un 48 % de clasificación correcta para la clase *hate speech*, 86 % para la clase *offensive language* y 94 % para la clase *neither*. Sin embargo, para los experimentos con frecuencia mínima de 5, obtenemos un 68 % de clasificación para la clase *hate speech* aumentando significativamente la correcta clasificación; sin embargo, la clase *offensive language* decae en cuanto a una clasificación certera, ya que obtiene solamente un 12 %; por último, para la clase *neither* obtenemos un buen resultado con un 94 %.

En las figuras 6.7 y 6.8 podemos observar las matrices de confusión para los experimentos hechos con el vectorizador Tf-Idf con frecuencias mínimas de 3 y 5, respectivamente. Para este punto, la matriz de confusión mostrada en la figura 6.7 obtiene resultados similares a los obtenidos en la matriz de la figura 6.5. Sin embargo, la matriz de confusión mostrada en la figura 6.8 obtiene una clasificación casi perfecta para la clase *neither*, y para las clases *hate speech* y *offensive language* se obtienen resultados malos, con un 38 % y 30 %, respectivamente.

Tabla 6.10: Resultados de la validación cruzada para Tf-Idf (unigramas y bigramas).

Clase	TFI-12G-FM3				TFI-12G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	Perceptrón	0.85	0.89	0.82	RL	0.86	0.89	0.83
offensive language	RL	0.92	0.95	0.90	MLP	0.84	0.96	0.75
neither	MLP	0.85	0.94	0.78	SVM lineal	0.94	0.92	0.95

Los reportes de clasificación, los cuales muestran las métricas completas de los ensambles de los modelos realizados en esta sección de experimentos, se muestran en las tablas 6.11 y 6.12.

Con estos resultados podemos seguir sosteniendo la hipótesis que mencionamos en la sección anterior, ya que nuestro vectorizador por Tf sigue obteniendo mejores resultados que Tf-Idf. Sin embargo, cabe mencionar que al agregar más características a nuestro entrenamiento, también mejora Tf-Idf. Esto quiere decir, que el tener más características también está influyendo de manera positiva al entendimiento de los modelos para su correcta clasificación contra los datos de prueba.

Tabla 6.11: Resultados de los ensambles (unigramas y bigramas).

Clase	TF-12G-FM3				TF-12G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	RL	0.36	0.29	0.48	RL	0.11	0.06	0.68
offensive language	RL	0.91	0.97	0.86	MLP	0.21	0.97	0.12
neither	MLP	0.83	0.74	0.94	RL	0.80	0.70	0.94

Tabla 6.12: Resultados de los ensambles (unigramas y bigramas).

Clase	TFI-12G-FM3				TFI-12G-FM3			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	Perceptrón	0.33	0.32	0.33	RL	0.10	0.06	0.38
offensive language	RL	0.91	0.97	0.86	MLP	0.45	0.98	0.29
neither	MLP	0.78	0.65	0.96	SVM lineal	0.63	0.46	0.99

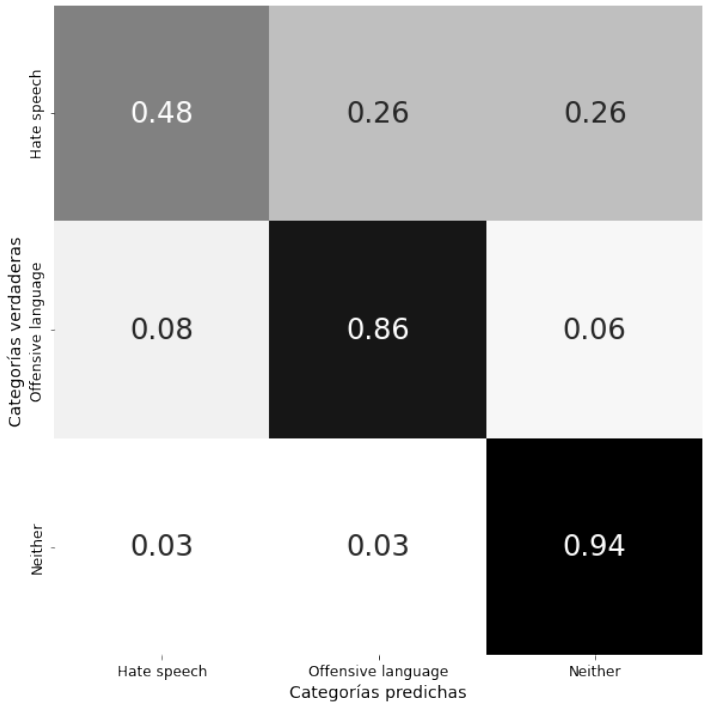

Figura 6.5: Matriz de confusión para la configuración TF-12G-FM3.

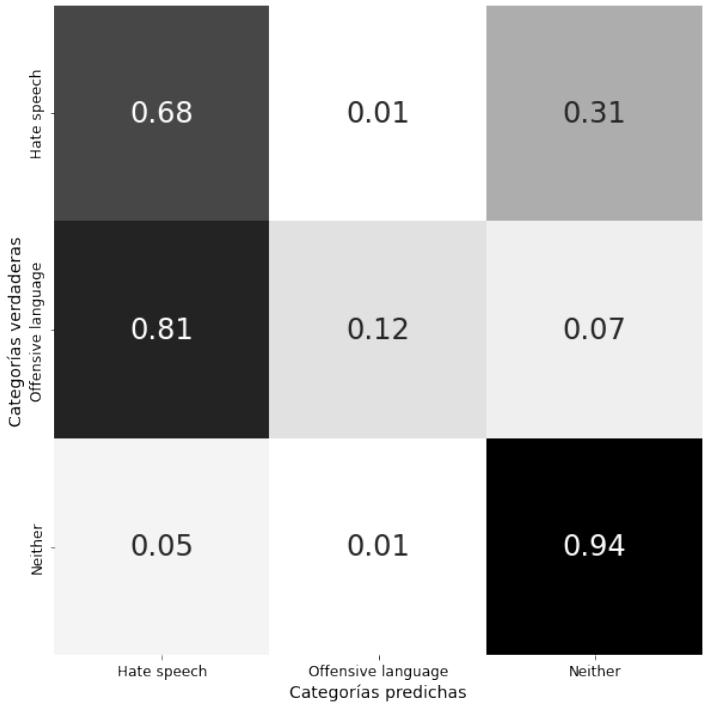

Figura 6.6: Matriz de confusión para la configuración TF-12G-FM5.

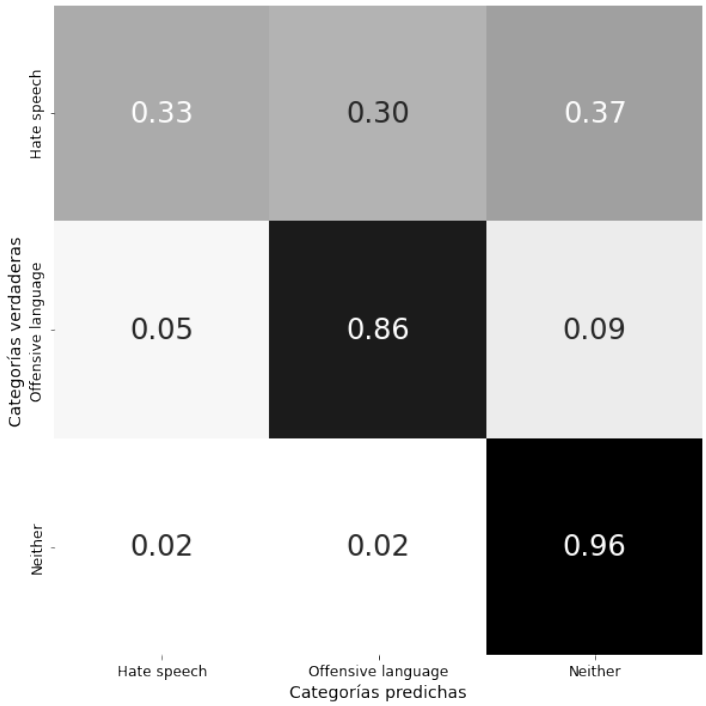

Figura 6.7: Matriz de confusión para la configuración TFI-12G-FM3.

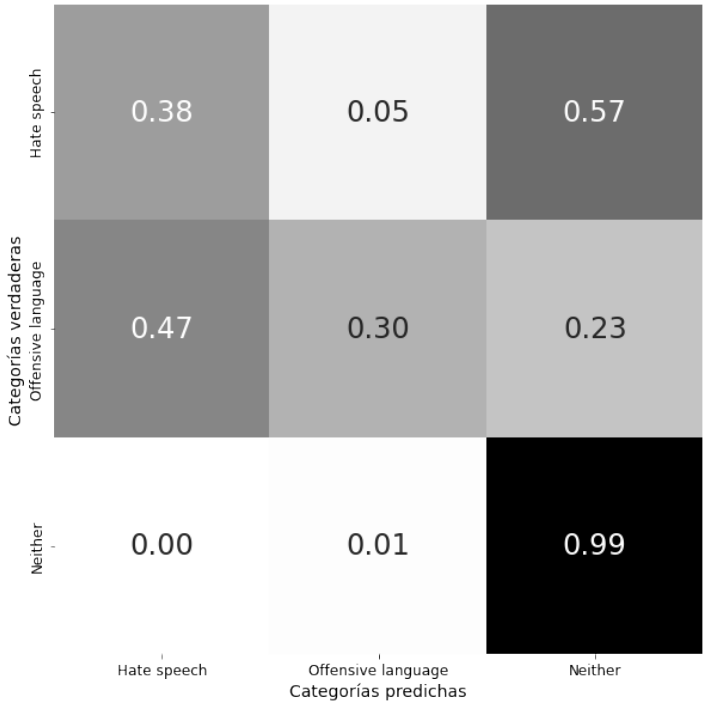

Figura 6.8: Matriz de confusión para la configuración TFI-12G-FM5.

6.3 Unigramas, bigramas y trigramas

Hasta este punto de la experimentación hemos probado distintas configuraciones para los vectorizadores y la cantidad de características que extraemos de nuestros subconjuntos de datos. De esta manera, hemos visto mejoras en algunos casos. Sin embargo, el agregar más características puede que confunda a nuestros modelos durante el entrenamiento y no tengan un desempeño tan bueno o, por otro lado, también está la opción que estos comprendan mejor su tarea al añadirle más características.

En la tabla 6.13 podemos ver como una vez más, al agregar los trigramas, las características que extrajimos suben considerablemente. Asimismo, en las tablas 6.14 y 6.15 podemos observar que modelos fueron los seleccionados para cada una de las clases tras la validación cruzada, así como sus respectivos reportes de clasificación. En esta oportunidad, para el vectorizador por Tf, los mejores modelos para todas las clases y para ambas frecuencias de aparición mínima fue la regresión logística. Mientras que para el vectorizador Tf-Idf, para la frecuencia de aparición mínima de 3 el mejor modelo para *hate speech* fue el perceptrón, para *offensive language* fue regresión logística y para *neither* fue SVM lineal; para la frecuencia de aparición mínima de 5 los mejores modelos fueron SVM lineal, regresión logística y SVM lineal para las clases *hate speech*, *offensive language* y *neither* respectivamente.

Tabla 6.13: Cantidad de características (unigramas, bigramas y trigramas).

Frecuencia mínima	Modelo	Lemas	POS	TAG	Otras características	Total
3	*hate speech*	2,807	1,157	2,218	309	6,491
	offensive language	8,379	1,593	3,777	318	14,067
	neither	8,378	1,584	3,761	317	14,040
5	*hate speech*	1,325	913	1,566	299	4,103
	offensive language	4,070	1,335	2,852	312	8,569
	neither	4,072	1,327	2,807	313	8,519

Tabla 6.14: Resultados de la validación cruzada para Tf (unigramas, bigramas y trigramas).

Clase	TF-123G-FM3				TF-123G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	RL	0.86	0.89	0.84	RL	0.86	0.89	0.84
offensive language	RL	0.94	0.96	0.91	RL	0.93	0.96	0.91
neither	RL	0.93	0.91	0.96	RL	0.93	0.91	0.96

Tabla 6.15: Resultados de la validación cruzada para Tf-Idf (unigramas, bigramas y trigramas).

Clase	TFI-123G-FM3				TFI-123G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	Perceptrón	0.84	0.88	0.81	SVM lineal	0.86	0.89	0.89
offensive language	RL	0.92	0.95	0.89	RL	0.92	0.95	0.90
neither	SVM lineal	0.93	0.92	0.94	SVM lineal	0.93	0.92	0.94

En la tabla 6.16 podemos observar los resultados del reporte de clasificación para el ensamble de modelos de esta sección.

En las figuras 6.9 y 6.10 podemos observar de manera gráfica las matrices de confusión sobre los resultados de clasificación del vectorizador por Tf. Las clases *offensive language* y *neither* siguen obteniendo un buen desempeño en su correcta clasificación, sin embargo, la clase *hate speech* para la configuración **TF-123G-FM5** obtiene el resultado más bajo de todos los experimentos realizados. De esta manera tenemos que, para la configuración **TF-123G-FM3** se tiene una correcta clasificación de los documentos pertenecientes a la clase *hate speech* de 48 %, para la clase *offensive language* un 87 % y para la clase *neither* de 94 %. Para la configuración **TF-123G-FM5** tenemos los siguientes resultados: 7 % para *hate speech*, 92 % para *offensive language* y 95 % para *neither*.

Las matrices de confusión obtenidas sobre el ensamble en el cual se usó el vectorizador Tf-Idf se muestran en las figuras 6.11 y 6.12. Dichas matrices nos muestran un resultado muy similar, ya que para la configuración **TFI-123G-FM3** tenemos un 21 % de correcta clasificación para la clase *hate speech*, un 86 % para *offensive language* y un 97 % para *neither*. Mientras que para la configuración **TFI-123G-FM5** tenemos un 29 %, 80 % y 98 % para las respectivas clases *hate speech*, *offensive language* y *neither*.

Estos resultados siguen manteniendo la hipótesis que el vectorizador por Tf obtiene mejores resultados que Tf-Idf y esto se puede deber al proceso que se realizó para este trabajo, y esto nos dice que es más importante la cantidad de aparición de las palabras en los documentos que el número de documentos en las que aparecen dichas palabras. De igual manera, con estos resultados podemos generar otra hipótesis, la cual diga que a partir de cierto número de características los modelos dejan de mejorar e inclusive llegan a empeorar su desempeño, al menos para el vectorizador Tf-Idf.

Tabla 6.16: Resultados de los ensambles (unigramas, bigramas y trigramas).

Clase	TF-123G-FM3				TF-123G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	RL	0.39	0.33	0.48	RL	0.11	0.36	0.06
offensive language	RL	0.92	0.97	0.87	RL	0.93	0.94	0.92
neither	RL	0.82	0.73	0.94	RL	0.80	0.69	0.95

Tabla 6.17: Resultados de los ensambles (unigramas, bigramas y trigramas).

Clase	TFI-123G-FM3				TFI-123G-FM5			
	Modelo	*F1-score*	*Precision*	*Recall*	Modelo	*F1-score*	*Precision*	*Recall*
hate speech	Perceptrón	0.28	0.40	0.21	SVM lineal	0.30	0.31	0.29
offensive language	RL	0.92	0.97	0.86	RL	0.88	0.98	0.80
neither	MLP	0.75	0.61	0.97	SVM lineal	0.70	0.55	0.98

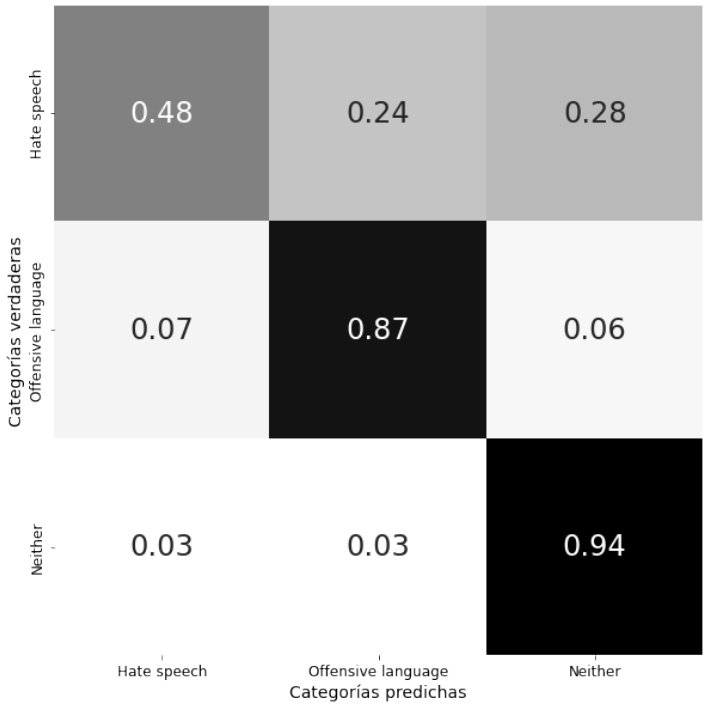

Figura 6.9: Matriz de confusión para la configuración TF-123G-FM3.

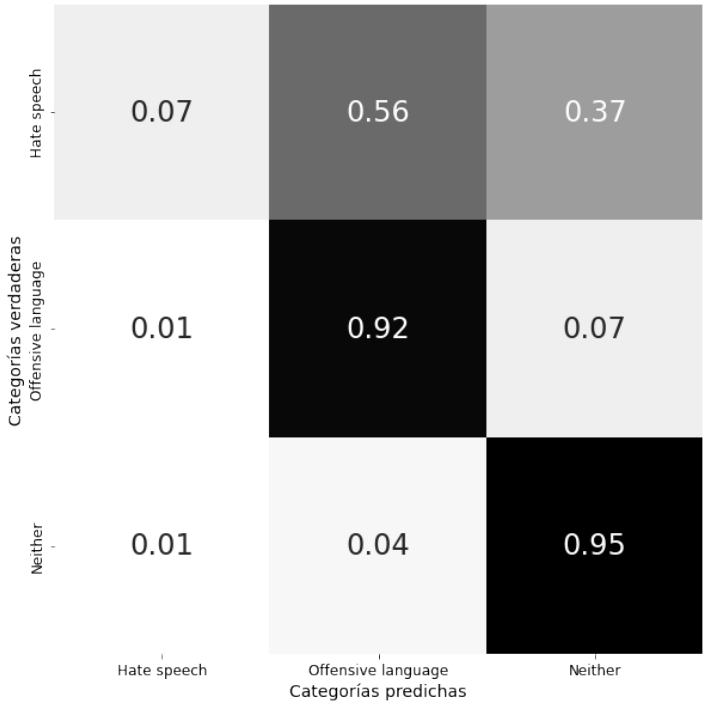

Figura 6.10: Matriz de confusión para la configuración TF-123G-FM5.

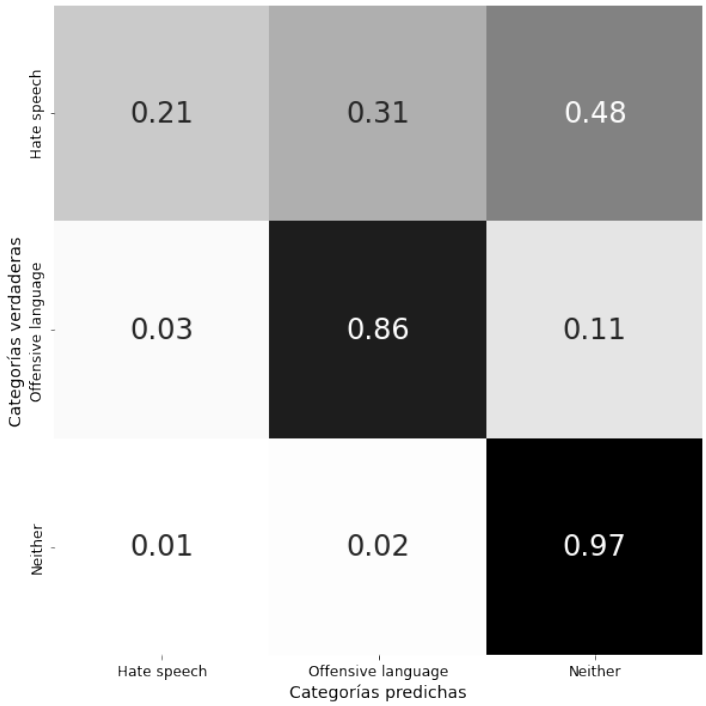

Figura 6.11: Matriz de confusión para la configuración TFI-123G-FM3.

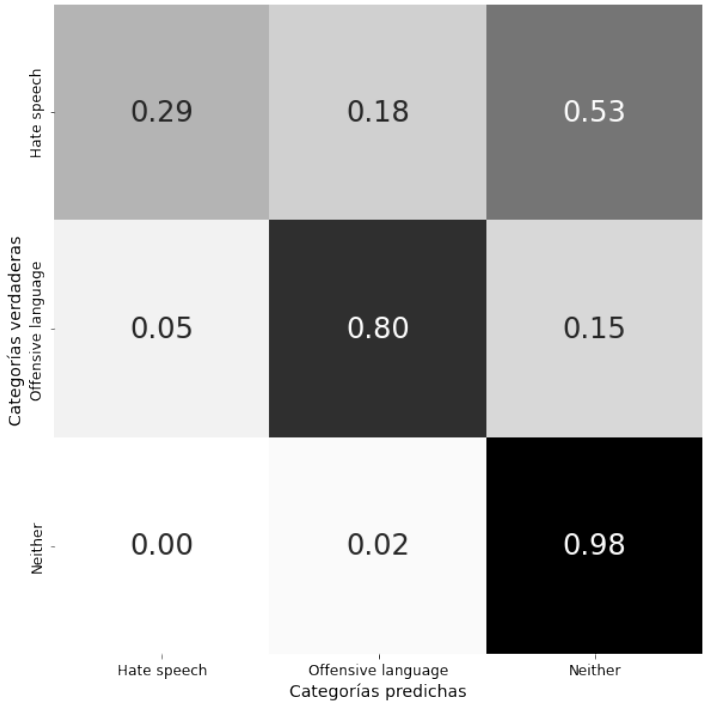

Figura 6.12: Matriz de confusión para la configuración TFI-123G-FM5.

Capítulo 7

Conclusiones

El objetivo general de este trabajo se cumplió de manera exitosa, ya que se encontraron los mejores parámetros para los modelos de aprendizaje automático propuestos. Con los distintos experimentos que se realizaron concluimos que al añadir más características, el desempeño de los modelos tiende a aumentar y así obtener mejores clasificaciones con respecto a aquellos modelos que reciben una menor cantidad de características.

En ensamble de modelos sugerido en este trabajo obtiene buenos resultados cuando se tiene una cantidad considerable de documentos en los conjuntos de datos y una cantidad considerable de características. El mejor modelo obtenido por este ensamble de modelos, fue obtenido por el vectorizador por conteo con todas las características propuestas y con n-gramas con secuencias de palabras de 1, 2 y 3 (unigramas, bigramas y trigramas). Este resultado, lo podemos comparar directamente contra el estado del arte, ya que se siguieron las mismas particiones del conjunto de datos y las mismas semillas para replicabilidad.

La comparación directa contra el estado del arte se puede apreciar en las matrices de confusión que se muestran en las figuras 7.1 y 7.2 en las cuales podemos observar que para el trabajo presentado en el estado del arte por (Davidson et al., 2017) obtiene una correcta clasificación para la clase *hate speech* de 61 %, para

offensive language un 91 % y para *neither* un 95 %. Por otro lado, en el trabajo presentado por (Mozafari et al., 2020) obtiene una correcta clasificación para la clase *hate speech* de 30 %, para *offensive language* un 97 % y para *neither* un 92 %. Asimismo, en el trabajo propuesto en esta tesis la matriz de confusión mostrada en la figura 7.3 nos muestra que se obtuvo una correcta clasificación para la clase *hate speech* de 48 %, para *offensive language* un 87 % y para *neither* un 94 %. Sin embargo, cabe recalcar que en el trabajo de Davidson et al., 2017 a pesar de ser el trabajo que obtuvo mejor desempeño con el mismo conjunto de datos que se uso en este trabajo, extrajeron todas las características del texto, incluyendo las características del conjunto de pruebas, y estas fueron dadas como entrada a los modelos que sugirieron. Por otro lado, para fines de esta tesis, las características dadas como entrada a los modelos solamente fueron las obtenidas de los subconjuntos de entrenamiento, con la finalidad de que los modelos no "conocieran" previamente las características del conjunto de pruebas.

Los mejores resultados obtenidos en el presente trabajo demuestran que el ensamble de modelos finamente ajustados logra clasificar de una manera exitosa el texto tóxico. Asimismo, demuestra un desempeño muy a la par de los modelos presentados en el estado del arte, inclusive con una menor cantidad de documentos, ya que para el entrenamiento de todas las clases se utilizó un total de 17,422 documentos, mientras que en el trabajo de Davidson et al., 2017, se utilizaron 22,303 documentos para el entrenamiento, sin considerar que las características extraídas para el entrenamiento correspondían a todo el conjunto de datos. Con esto podemos concluir que al realizar el equilibrio de datos para cada una de las clases y separar un problema multiclase a distintas tareas binarias, podemos obtener buenos resultados, siempre y cuando el preprocesamiento de los datos y la extracción de características se hayan realizado de manera correcta.

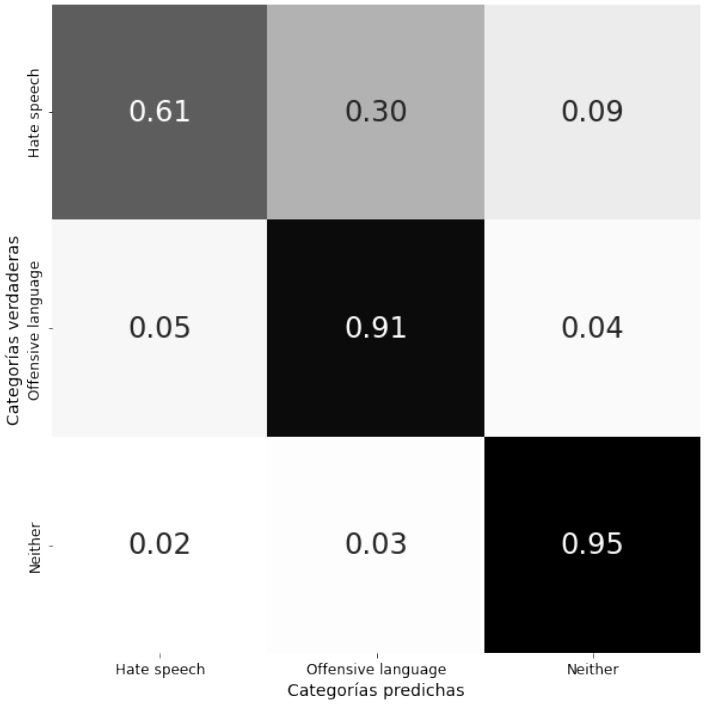

Figura 7.1: Matriz de confusión de los resultados obtenidos por Davidson et al., 2017.

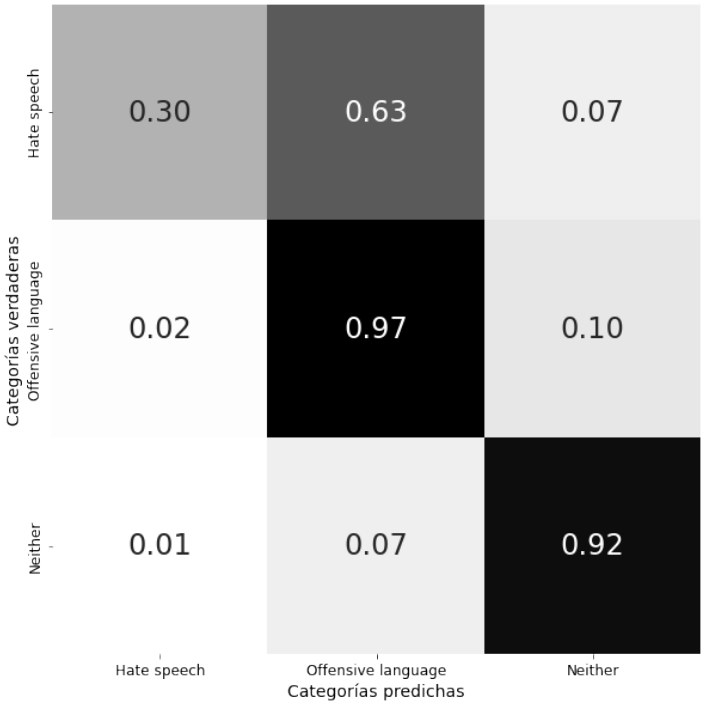

Figura 7.2: Matriz de confusión de los resultados obtenidos por Mozafari et al., 2020.

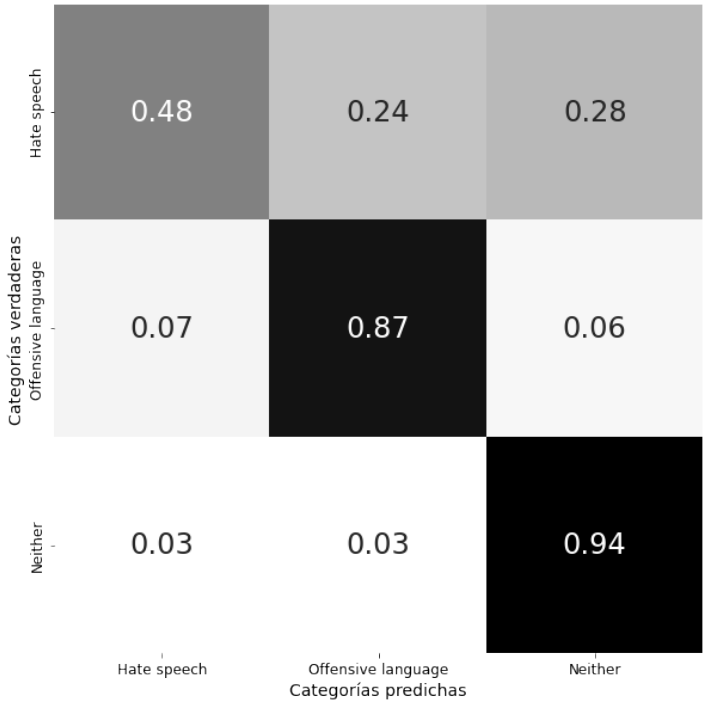

Figura 7.3: Matriz de confusión de los mejores resultados obtenidos en este trabajo.

Capítulo 8

Trabajo futuro

A corto plazo, buscaremos añadir más modelos de aprendizaje automático que nos sirvan para clasificación tales como:

- Las distintas variantes de los algoritmos de Naive Bayes, como Naive Bayes multinomial, Naive Bayes Gaussiano o Naive Bayes per se.
- Modelos lineales como Descenso Gradiente Estocástico (SGD, por sus siglas en inglés), regresión cuantílica, regresión bayesiana, entre otros.

De igual manera, buscaremos mejorar los parámetros del perceptrón multicapa, ya que durante el proceso de experimentación obtuvo resultados muy bajos, y a pesar de que en varios experimentos resulto como el mejor modelo para una clase, este modelo aún tiene mucho de donde se pueda mejorar.

Continuando con el uso de ensambles de modelos, se probarán distintos conjuntos de datos. Para el idioma inglés, se buscarán conjuntos de datos con una mayor cantidad de documentos. Por otra parte, con la finalidad de verificar que el ensamble de modelos no solamente funcione correctamente en datos en inglés, se buscarán conjuntos de datos en español para resolver de igual manera la tarea de la clasificación del texto tóxico. De obtener un resultado positivo del ensamble de modelos para la clasificación de texto tóxico, el siguiente paso sería buscar otras

tareas de clasificación que puedan ser resueltas con este método.

Más adelante, se profundizará en el aprendizaje por transferencia, el cual es el proceso de entrenar un modelo en un conjunto de datos a gran escala y luego usar ese modelo previamente entrenado para llevar a cabo el aprendizaje para otra tarea posterior. Esto debido a que los grandes modelos del lenguaje nos pueden aportar una gran cantidad de características, ya no solamente para tareas de clasificación, sino para el entendimiento del lenguaje en sí. De esta manera podemos predecir de qué manera podrían responder los usuarios dada una publicación en una red social. O basándonos en conversaciones y entendiendo el contexto de la misma conversación, identificar si algún mensaje hace alusión a una ofensa hacia la otra persona.

La tarea de detección del lenguaje ofensivo o lenguaje tóxico no solo aplica para textos, sino que esta tarea se puede escalar aún más, ya que en las redes sociales los usuarios siempre encuentran la manera de burlar a los sistemas que imponen para detener este tipo de comportamiento. Esto lo logran haciendo uso de imágenes o memes, audios e inclusive videos. Por tal motivo, la adición de alguna o algunas de estas corrientes alternas para la detección del lenguaje tóxico serían de gran ayuda para continuar sobre esta línea de investigación que aún tiene muchos matices que cubrir.

Referencias

Suzuki, K., Asaga, R., Sourander, A., Hoven, C., & Mandell, D. (2012). Cyberbullying and adolescent mental health. *International journal of adolescent medicine and health*, *24*, 27-35. https://doi.org/10.1515/ijamh.2012.005

Hinduja, S., & Patchin, J. (2010). Bullying, Cyberbullying, and Suicide. *Archives of suicide research : official journal of the International Academy for Suicide Research*, *14*, 206-21. https://doi.org/10.1080/13811118.2010.494133

Borisyuk, F., Gordo, A., & Sivakumar, V. (2018). Rosetta: Large Scale System for Text Detection and Recognition in Images. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 71-79. https://doi.org/10.1145/3219819.3219861

Liu, S., & Forss, T. (2015). New classification models for detecting Hate and Violence web content. *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, *01*, 487-495.

Burnap, P., & Williams, M. L. (2016). Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science*, *5*(1), 11. https://doi.org/10.1140/epjds/s13688-016-0072-6

Davidson, T., Warmsley, D., Macy, M. W., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. *ICWSM*.

Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017a). Deep Learning for Hate Speech Detection in Tweets. *Proceedings of the 26th International Conference on World Wide Web Companion*, 759-760. https://doi.org/10.1145/3041021.3054223

Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017b). Deep Learning for Hate Speech Detection in Tweets. *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*. https://doi.org/10.1145/3041021.3054223

Georgakopoulos, S. V., Tasoulis, S. K., Vrahatis, A. G., & Plagianakos, V. P. (2018). Convolutional Neural Networks for Toxic Comment Classification. *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*. https://doi.org/10.1145/3200947.3208069

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *CoRR*, *abs/1706.03762*. http://arxiv.org/abs/1706.03762

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, *abs/1810.04805*. http://arxiv.org/abs/1810.04805

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, *abs/1907.11692*. http://arxiv.org/abs/1907.11692

Lample, G., & Conneau, A. (2019). Cross-lingual Language Model Pretraining. *CoRR*, *abs/1901.07291*. http://arxiv.org/abs/1901.07291

Ozler, K. B., Kenski, K., Rains, S., Shmargad, Y., Coe, K., & Bethard, S. (2020). Fine-tuning for multi-domain and multi-label uncivil language detection. *Proceedings of the Fourth Workshop on Online Abuse and Harms*, 28-33. https://doi.org/10.18653/v1/2020.alw-1.4

Baratalipour, N., Suen, C. Y., & Ormandjieva, O. (2020). Abusive Language Detection Using BERT Pre-trained Embedding. En Y. Lu, N. Vincent, P. C. Yuen, W.-S. Zheng, F. Cheriet & C. Y. Suen (Eds.), *Pattern Recognition and Artificial Intelligence* (pp. 695-701). Springer International Publishing.

Sivanaiah, R., Suseelan, A., Rajendram, S. M., & T.t., M. (2020). TECHSSN at SemEval-2020 Task 12: Offensive Language Detection Using BERT Embeddings. *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, 2190-2196. https://doi.org/10.18653/v1/2020.semeval-1.291

Althobaiti, M. J. (2022). Bert-based approach to Arabic hate speech and offensive language detection in Twitter: Exploiting emojis and sentiment analysis. *International Journal of Advanced Computer Science and Applications*, *13*(5). https://doi.org/10.14569/ijacsa.2022.01305109

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, *3*(3), 210-229. https://doi.org/10.1147/rd.33.0210

Aghdam, H. H., & Heravi, E. J. (s.f.). Guide to Convolutional Neural Networks. https://link.springer.com/book/10.1007/978-3-319-57550-6

Webb, G. I., Keogh, E., Miikkulainen, R., Miikkulainen, R., & Sebag, M. (2011). Naïve bayes. *Encyclopedia of Machine Learning*, 713-714. https://doi.org/10.1007/978-0-387-30164-8_576

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133. https://doi.org/10.1007/BF02478259

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386-408. https://doi.org/10.1037/h0042519

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144-152. https://doi.org/10.1145/130385.130401

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5-32. https://doi.org/10.1023/a:1010933404324

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825-2830.

Honnibal, M., & Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing* [To appear].

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. .ºReilly Media, Inc."

Greevy, E. (2004). *Automatic text categorisation of racist webpages.* (Tesis doctoral).

Burnap, P., & Williams, M. L. (2015). Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making. *Policy & Internet*, *7*(2), 223-242. https://doi.org/https://doi.org/10.1002/poi3.85

Waseem, Z., & Hovy, D. (2016). Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. *Proceedings of the NAACL Student Research Workshop*, 88-93. https://doi.org/10.18653/v1/N16-2013

Burnap, P., & Williams, M. (2016). Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science*, *5*. https://doi.org/10.1140/epjds/s13688-016-0072-6

Vigna, F. D., Cimino, A., dell'Orletta, F., Petrocchi, M., & Tesconi, M. (2017). Hate Me, Hate Me Not: Hate Speech Detection on Facebook. En A. Armando, R. Baldoni & R. Focardi (Eds.), *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy, January 17-20, 2017* (pp. 86-95). CEUR-WS.org. http://ceur-ws.org/Vol-1816/paper-09.pdf

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://doi.org/10.48550/ARXIV.1301.3781

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532-1543. https://doi.org/10.3115/v1/D14-1162

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746-1751. https://doi.org/10.3115/v1/D14-1181

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Mozafari, M., Farahbakhsh, R., & Crespi, N. (2020). Hate speech detection and racial bias mitigation in social media based on BERT model. *PLOS ONE*, *15*(8), 1-26. https://doi.org/10.1371/journal.pone.0237861

Khan, S., Fazil, M., Sejwal, V. K., Alshara, M. A., Alotaibi, R. M., Kamal, A., & Baig, A. R. (2022). BiCHAT: BiLSTM with deep CNN and hierarchical attention for hate speech detection. *Journal of King Saud University - Computer and Information Sciences*, *34*(7), 4335-4344. https://doi.org/https://doi.org/10.1016/j.jksuci.2022.05.006

Khan, S., Kamal, A., Fazil, M., Alshara, M. A., Sejwal, V. K., Alotaibi, R. M., Baig, A. R., & Alqahtani, S. (2022). HCovBi-Caps: Hate Speech Detection Using Convolutional and Bi-Directional Gated Recurrent Unit With Capsule Network. *IEEE Access*, *10*, 7881-7894. https://doi.org/10.1109/ACCESS.2022.3143799