UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER EN INGENIERÍA Y CIENCIA DE DATOS

Detección de lenguaje ofensivo en redes sociales

JOSÉ MARÍA MOLERO ALONSO

Dirigido por: Jorge Pérez Martín

Álvaro Rodrigo Yuste

Curso: 2021-2022: 1ª Convocatoria

Resumen

Las redes sociales son herramientas que permiten relacionarse con personas de todo el mundo de forma instantánea, compartiendo ideas, opiniones u otro tipo de información como aspectos personales de la vida del usuario. Idealmente esto debería dar lugar a charlas o debates con carácter positivo pero la sensación de anonimato y una sociedad cada vez más polarizada fomentan, en muchas personas, conductas negativas como la discriminación, acoso o, en general, el uso de lenguaje con el solo fin de causar daño o malestar en otras personas.

En las redes sociales los usuarios cuentan con herramientas para evitar o reducir la exposición a este tipo de conductas pero, en muchos casos, estas herramientas no son autónomas, así que el usuario primero se ve expuesto y luego actúa denunciando los mensajes ofensivos o bloqueando usuarios con conductas negativas. Para que las redes sociales sean un lugar menos dañino se está dedicando un esfuerzo continuo para mejorar y desarrollar nuevos métodos de detección de lenguaje ofensivo o ciberacoso. Estos sistema se enfrentan a múltiples retos como tratar grandes volúmenes de datos o tratar con textos informales y con un vocabulario evoluciona muy rápido.

En este trabajo se van a aplicar técnicas de procesamiento del lenguaje para desarrollar modelos de aprendizaje supervisados capaces de detectar comentarios ofensivos en redes sociales. Para obtener el mejor rendimiento en cada modelo se probarán distintos tratamiento de los textos que irán desde no aplicar ningún tratamiento hasta el truncado, eliminación y corrección de palabras. Posteriormente se entrenarán modelos de aprendizaje, empezando por modelos clásicos como clasificadores lineales o bosques aleatorios. Luego se pasarán a modelos neuronales como redes convolucionales y recurrentes y, por último, se entrenarán modelos tipo BERT que representan uno de los últimos avances en el campo del procesamiento del lenguaje natural. Tras el entrenamiento se analizarán y compararán los resultado de estos modelos y los de otros investigadores. Del trabajo realizado se obtienen algunas conclusiones: las librerías empleadas realizan por defecto un procesado de los texto que puede ser suficiente para obtener buenos resultado, por lo que no es necesario dedicar un gran esfuerzo en este tipo de tareas. La corrección de palabras puede aumentar la eficiencia de modelos que emplean incrustaciones de palabras como datos de entrada. Por último, en vista a los resultados queda patente que los modelo BERT han supuesto realmente un salto cualitativo en el campo del procesamiento del lenguaje natural.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción general y objetivos

En este capítulo se explicarán la motivación por la cual se va a desarrollar este trabajo, sus objetivos fundamentales y el contexto en el que se desarrolla.

1.1. Motivación

En Wiegand et al. (2018) definen el lenguaje ofensivo como los comentarios hirientes, despectivos u obscenos realizados por una persona a otra o a un grupo de personas y está relacionado con otros conceptos como el lenguaje abusivo, discurso de odio, ciberacoso o lenguaje tóxico. Aunque existen matices entre ellos, son compatibles con la definición de lenguaje ofensivo dada. En la figura 1.1 se muestra la relación entre los distintos tipos de lenguajes según Poletto et al. (2021), donde se puede comprobar que existe relación entre todos ellos.

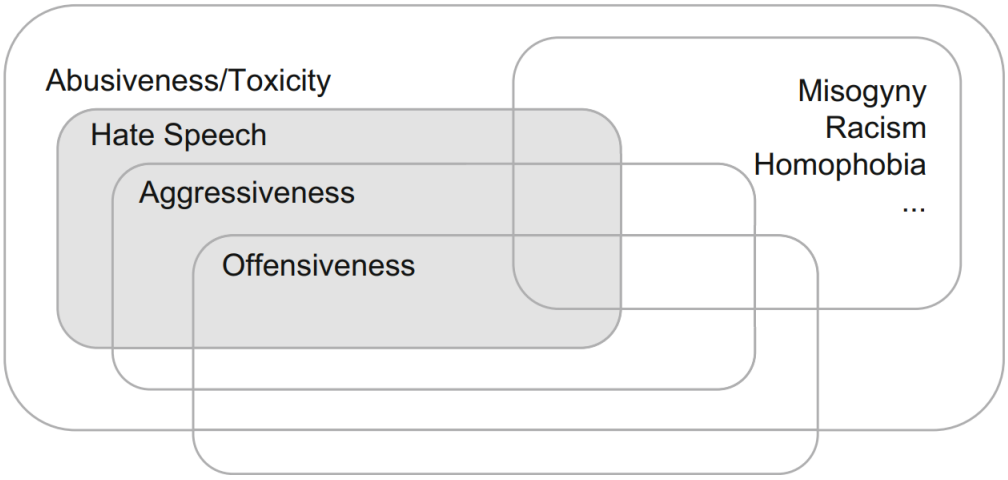

Figura 1.1: Relación entre el discurso de odio y los conceptos relacionados

Con el auge de las redes sociales estamos más conectados pero también más expuestos a recibir comentarios ofensivos por nuestras ideas, sexo, raza o condición física. En los informes de las propias

redes o en la prensa, se pueden encontrar estadísticas que evidencian este problema. Instagram[1] reportó en 2021 el borrado de 6,5 millones de mensajes con discurso de odio entre los meses de julio y septiembre, el 95 % de ellos detectados antes de ser reportados por los usuarios. Entre julio y diciembre de 2020 Twitter[2] eliminó 1,2 millones de cuentas por infringir su política de conducta de odio. Por otro lado, YouTube[3] eliminó 500 millones de comentarios con discursos de odio y varios miles de videos y canales.

Cada red social aborda el problema de los comentarios de distinta forma:

- Twitter permite denunciar comentarios, bloquear e ignorar usuarios[4]. También avisa a los usuarios antes de publicar un comentario con contenido potencialmente dañino para que reconsidere su publicación pero no aplica un filtrado automático de estos comentarios.

- En YouTube existe una funcionalidad[5] activable por el dueño del canal bloquear automáticamente mensajes con contenido inapropiado. Con esta funcionalidad el dueño del canal decidirá la publicación de los comentarios tras su revisión.

- En Instagram los usuarios pueden aplicar filtros para evitar que sus seguidores publiquen comentarios con términos prohibidos por el mismo. Para las peticiones de mensajes privados implementa un filtrado[6] automático de palabras, frases o emoticonos ofensivos.

Los sistemas propuestos por YouTube e Instagram pueden considerarse como "activos" ya que por defecto no exponen a los usuarios a los comentarios potencialmente ofensivos, mientras que los sistemas propuestos por Twitter y WhatsApp son sistemas "reactivos" porque el usuario debe ver el mensaje y luego decidir qué hacer con él. Con este último tipo de sistema, los usuarios siguen expuestos al lenguaje ofensivo y a sus consecuencias.

El uso del lenguaje para causar daño a terceras personas no afecta solo a estas redes sociales, también afecta a aplicaciones de mensajería instantánea, páginas de reseñas o a medios más tradicionales como foros. En resumen, es un problema que afecta a muchas personas en plataformas muy diversas y que implica un volumen de dato muy alto. Dada la magnitud del problema que se plantea, se está dedicando un gran esfuerzo en la investigación y desarrollo de sistemas que detecten de forma automática comentarios con lenguaje ofensivo, ya sea discurso de odio, ciberacoso o similar.

En este trabajo se analizará el estado del arte sobre estas investigaciones, se emplearán técnicas del Procesamiento del Lenguaje Natural para tratar los textos y se entrenarán y compararán modelos de aprendizaje automático y profundo con capacidad de detección de lenguaje ofensivo.

[1] https://about.instagram.com/blog/announcements/an-update-on-our-work-to-tackle-abuse-on-instagram
[2] https://time.com/6080324/twitter-hate-speech-penalties/
[3] https://edition.cnn.com/2019/09/03/tech/youtube-hate-speech/index.html
[4] https://help.twitter.com/en/safety-and-security/offensive-tweets-and-content
[5] https://support.google.com/youtube/thread/8830320/
[6] https://about.instagram.com/blog/announcements/introducing-new-tools-to-protect-our-community-from-abuse

1.2. Objetivos

1.2.1. Objetivo General

El objetivo principal de este trabajo fin de máster consiste en implementar un sistema con capacidad para detectar y clasificar comentarios ofensivos en redes sociales.

1.2.2. Objetivos específicos

Para alcanzar el objetivo principal se platean los siguientes objetivos específico:

- Aplicar y desarrollar métodos para recuperar palabras mal escrita que quedan fuera del vocabulario, con la consecuente pérdida de información. También ser formalizarán los textos mediante la eliminación de textos duplicados y de contenido sin significado.

- Entrenar y comparar modelos de aprendizaje automático y profundo para obtener el modelo que mejor cumpla el objetivo general propuesto.

- Estudiar como afectan distintas formas de procesar los textos en los resultados de los modelos.

1.3. Contexto

En 2021 del Arco et al. (2021) publicaron un corpus de lenguaje ofensivo en español creado a partir de comentarios extraídos de redes sociales, este corpus recibe el nombre "OffendES". Además de crear el corpus, entrenaron dos modelos de aprendizaje:

- Modelo de clasificación multietiqueta: este modelo clasifica los comentarios en las categorías "NO" (no ofensivo), "NOM" (no ofensivo pero usa de lenguaje soez), "OFP" (ofensivo hacía persona) y "OFG" (ofensivo a hacia grupos).

- Modelo de clasificación binaria: este modelo clasifica los comentarios como "NO" (no ofensivo) o "OF" (ofensivo).

En el mismo año de publicación del corpus se celebró la tarea MeOffendES[7] en la *Iberian Languages Evaluation Forum 2021*[8] (IberLef 2021) donde se utilizó una versión refinada de este corpus. En este trabajo se va a emplear esta versión refinada del corpus para evaluar los modelos desarrollados y poder compararlos con los resultados obtenidos por los participantes de la tarea indicada.

[7] http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6388
[8] http://ceur-ws.org/Vol-2943/

1.4. Estructura de la memoria

La memoria de este trabajo se estructura en los siguientes capítulos:

1. **Introducción y objetivos**: se exponen los objetivos del trabajo, su contexto y la organización de la memoria.
2. **Marco teórico**: se definen los conceptos claves para comprender el trabajo desarrollado.
3. **Estado del arte**: se realiza una revisión de trabajos y tareas de evaluación sobre detección y clasificación del lenguaje ofensivo.
4. **Marco de trabajo**: se describe el conjunto de datos, las métricas de evaluación de los modelos y las herramientas empleadas.
5. **Preparación de experimentos**: se detalla el trabajo realizado para la preparación de los datos de entrenamiento.
6. **Experimentos y resultados**: se detallan los modelos entrenados y los resultados obtenidos.
7. **Análisis y discusión**: se analizan y comparan los resultados, se comentan las ventajas y limitaciones encontradas en los modelos.
8. **Conclusiones y trabajos futuros**: se exponen las conclusiones y nuevas líneas de trabajo.

Capítulo 2

Marco teórico

En este capítulo se van a exponer las definiciones y conceptos claves para comprender el trabajo desarrollado. Primero se darán las definiciones aprendizaje automático y aprendizaje profundo. Luego se pasará a explicar los métodos para convertir los textos a datos que puedan manejar los modelos de aprendizaje. Por último se explicarán las métricas utilizadas para evaluar la bondad de los resultados obtenidos.

2.1. Aprendizaje automático

El aprendizaje automático se refiere al campo de investigación dedicado a construir métodos que aprenden a dar una respuesta a partir de los datos que reciben como entradas. Los métodos de aprendizaje pueden ser supervisados o no supervisados. En los métodos supervisados los datos de entrenamiento necesitan tener una etiqueta que será lo que el modelo trate de inferir. La etiqueta es un elemento que no siempre está disponible, en esos casos deberá asignarse tras la recogida de datos. El etiquetado es generalmente muy costoso en comparación al de obtención de datos, ya que puede requerir de procesos manuales. En los métodos no supervisados no se tienen etiquetas, estos métodos tratan de encontrar estructuras en los datos para realizar agrupaciones más o menos homogéneas de los datos. Los modelos que se van a entrenar en este trabajo se encuentran dentro de la categoría de métodos supervisados ya que los datos de entrada cuentan con etiquetas.

Dentro del campo del aprendizaje automático se encuentra el **aprendizaje profundo** que se basa en el uso redes neuronales de múltiples capas. Una red neuronal es un sistema bioinspirado que trata de simular el funcionamiento del cerebro. La red neuronal se compone de unidades llamadas neuronas que se conectan entre sí, a través de estas conexiones fluyen los datos de entrada hasta producir una salida. El proceso de aprendizaje de las redes neuronales consiste en ajustar la conexiones entre las neuronas hasta obtener el mejor rendimiento en sus respuestas.

La ventaja del aprendizaje profundo frente a otros métodos es que no solo aprende a predecir el valor de salida en base a las entradas, también es capaz de extraer automáticamente características

de los datos de entrada y hacer predicciones en base a ellas, es decir, ofrecen un mayor nivel de abstracción que los métodos de aprendizaje automático y las redes neuronales de pocas capas.

2.1.1. Modelos de aprendizaje automático

En este trabajo se han aplicado los siguientes modelos de aprendizaje:

RandomForest

Los modelos *RadomForest* o bosques aleatorios, forman parte de los modelos que funciona mediante ensamblaje de predictores. Estos modelos consisten en combinar un número fijado de árboles de decisión que se entrenan mediante una técnica llamada *bagging*. Esta forma de entrenar consiste en que cada árbol se entrena con un conjunto de muestras distinto, la muestras son tomadas de forma aleatoria del conjunto de datos de entrenamiento. Con esta técnica, al combinar los resultados de todos los árboles los errores de unos árboles se compensan con los de otros lo cual mejora la capacidad de generalización del método. Para realizar la predicción se combinan las predicción de todos los árboles y se elegirá la opción con más votos, ponderando el voto según la probabilidad que da cada árbol. En la figura 2.1 se muestra de forma esquemática el funcionamiento de este modelo.

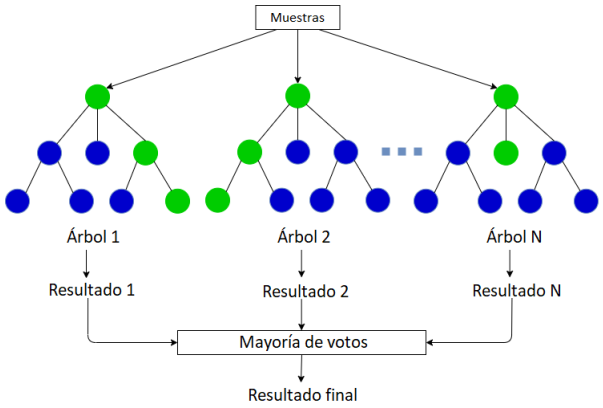

Figura 2.1: Bosque aleatorio

GrandientTreeBoosting

Es otro modelo de ensamblaje como los bosques aleatorios pero aplicando una técnica de entrenamiento conocida como *boosting*. Mientras que en los bosques aleatorios cada árbol de decisión se entrenada de forma independiente mediante *bagging* en los modelos basado en *boosting*, como este y Adaboost, cada árbol de decisión es construido sobre el árbol anterior, es decir, es un proceso aditivo (figura 2.2). En el modelo GradientBoosting cada nuevo árbol trata de corregir el error residual del anterior.

Adaboost

Este el último método de ensamblaje empleado, su entrenamiento también se realiza mediante *boosting* pero con una filosofía distinta al del modelo anterior: se añaden predictores que presten más atención sobre las instancias que el anterior clasificó de forma incorrecta Para aumentar la atención

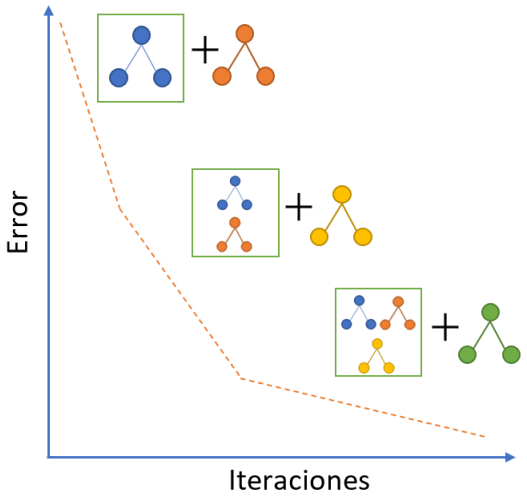

Figura 2.2: GradientBoosting

sobre los ejemplos mal clasificados el algoritmo aumenta los pesos de las instancias mal clasificadas y entrena un nuevo clasificador usando los nuevos pesos. Al contrario de los métodos anteriores se pueden emplear otros predictores bases a parte de los árboles de decisión.

Support Vector Machine

El modelo *SVM* trata de separar las muestras de cada clase en espacios lo más separado posibles, esta separación se llama superficie de decisión y se delimita con muestras situadas en los bordes, a las que se les denominada vectores de soporte (figura 2.3). Para espacios que no son linealmente separables se emplean *kernels,* que son funciones que aumentan la dimensionalidad del problema de forma que un problema no linealmente separable en un espacio de N dimensiones puede serlo en un espacio N+1 dimensional.

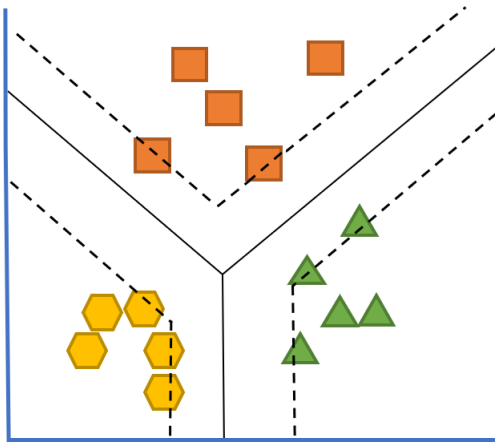

Figura 2.3: Support Vector Machine

Clasificador SGD

Es un tipo de clasificador lineal que usa el método del descenso del gradiente estocástico para

realizar el aprendizaje. Este clasificador tiene la ventaja de ser capaz de manejar eficientemente conjuntos de datos grandes y con alta dimensionalidad. Este modelo tiene una cantidad enorme de hiperparámetros pero gracias a que su entrenamiento es muy rápido se pueden realizar una enorme cantidad combinaciones de ellos en tiempo razonable.

2.1.2. Modelos de aprendizaje profundo

Como modelos de aprendizaje profundo se han empleado

RNN

RNN o *Recurrent Neural Network* es un tipo de red que utiliza datos secuenciales o basados en series temporales esto lo convierte en una herramienta muy interesante en el procesamiento del lenguaje natural ya tiene carácter secuencial. Estas redes se distinguen por usar un tipo de neuronas llamadas "células de memoria". En estas células la salida de cada una de ellas depende de sus salidas, esta dependencia es lo que se define como "memoria", que es la clave que convierte a este tipo de redes en las ideales para trabajar con secuencias. Aunque no son las únicas que pueden trabajar con secuencias.

Las células de memorias simples presentan algunas limitaciones respecto al tamaño de las secuencias que son capaces de memorizar y también problemas en el aprendizaje debido al desvanecimiento o explosión del gradiente durante su entrenamiento. Afortunadamente se han desarrollado nuevas células de memoria que suavizan estos problemas, como son las LSTM *(Long Short Term Memory*) y GRU (*Gated Recurrent Unit*).

La principal diferencia entre una célula recurrente simple y una LSTM es que las LSTM en lugar de tener una sola capa oculta tiene cuatro que interactúan entre sí a través de unas conexiones llamadas puertas. Los parámetros de estas puertas son aprendidos durante el entrenamiento. Las GRU son una simplificación de las LSTM con menos puertas, son más rápidas de entrenar pero su resultado puede ser peor.

CNN

Las CNN o *Convolutional Neural Networks* son un tipo de red que explotan la información espacial y, por tanto, presentan muy buen rendimiento en problemas con imágenes como datos de entrada pero pero también se pueden aplicar en el procesamiento del lenguaje natural. Una red de este tipo se construye apilando capas: las capas del nivel inferior son capaces de detectar características de bajo nivel mientras que las superior detectan las de alto nivel. En cada capa se realiza una operación de convolución y filtrado: la convolución consiste en asociar una submatriz de entrada con una sola neurona de la capa (figura 2.4), mientras que el filtrado es una operación que permite resaltar alguna característica de los datos. Por ejemplo, en una imagen se puede aplicar un filtro que realce las línea verticales de una imagen y que suavice el resto.

Modelos basados en BERT

Las siglas BERT vienen de *Bidirectional Encoder Representations from Transformers* (Vaswani

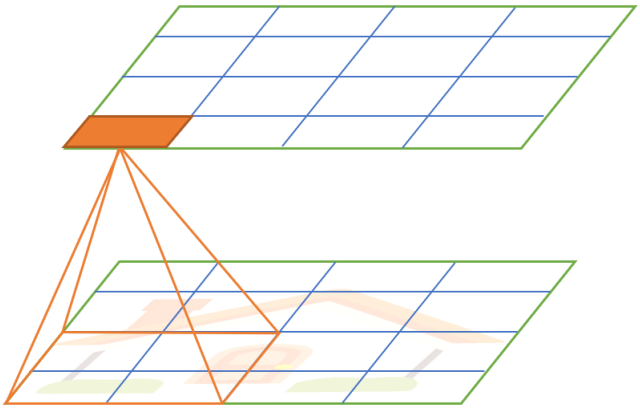

Figura 2.4: Convolución

et al. (2017)) y es uno de los avances más significativos en el campo del NLP. En los modelos BERT se emplea un mecanismo de atención llamado "Transformer", este mecanismo permite aprender la relación contextual de las palabras. A diferencia de los modelos direccionales, en los modelos BERT los textos se evalúan de izquierda a derecha y de derecha a izquierda, por este motivo se dicen que son modelos bidireccionales aunque también podrían denominar "no direccionales". Este funcionamiento hace que el modelo adquiera mayor conocimiento del contexto de las palabras lo que habilitar que una palabra polisémica tenga distinta representación según el contexto en el que se encuentra. Al contrario de lo que ocurre con las incrustaciones de palabras.

Los modelo BERT tienen dos características muy interesantes:

- Se **preentrenan de forma no supervisadas** mediante dos tareas diferentes, estas son *Masked Language Modeling* y *Next Sentence Prediction*. La primera consiste en ocultar una palabra en una frase y luego predecir qué palabra se ha ocultado (o enmascarado) basándose en el contexto de la palabra. La segunda tarea consiste en predecir si dos frases dadas tienen una conexión lógica y secuencial o si su relación es simplemente aleatoria.

- Los modelos preentrenados pueden ser sometidos a un **ajuste fino** mediante el aprendizaje por transferencia. Con esto se puede ajustar el modelo a un problema específico para mejorar su rendimiento. El aprendizaje por transferencia ya ha demostrado ser muy potente en tareas relacionadas con la visión por computador y ahora también en el procesamiento de lenguaje natural

Desde la aparición del primer modelo BERT en 2018 se ha publicado nuevos modelos en distintos idiomas y corpus (formales e informales). En este trabajo se entrenan dos modelos BERT en español: BETO y RoBERTuito. BETO es un modelo BERT entrenado con un corpus de unos 3.000 millones de palabras en español extraídas de artículos de Wikipedia[1] y fuentes del proyecto OPUS[2] (noticias,

[1] https://www.wikipedia.org/
[2] https://opus.nlpl.eu/

revistas, charlas TED...), fuentes cuyos textos tienen un carácter formal.

RoBERTuito (Pérez et al. (2021)) es un modelo preentrenado con textos en español extraídos de Twitter. El corpus de entrenamiento lo componen 500 millones de tweets. A diferencia que en BETO, el carácter de los textos de entrenamiento es informal.

2.2. Representaciones de documentos

Cuando los datos de entrada son de tipo texto hay que aplicar técnicas de representación de documentos para transformarlos en un estructura de datos que los modelos sean capaces de interpretar para realizar el entrenamiento. Las representaciones empleadas en este trabajos se explican a continuación

2.2.1. TF-IDF

Las siglas de este método corresponden a *Term Frequency - Inverse Document Frequency* y es un método que convierte cada documento en un vector con tantas posiciones como palabras en el vocabulario del corpus. A cada palabra se le asigna un valor que refleja su importancia en el corpus. El cálculo de este valor consiste en dividir el número de veces que aparece una palabra en un documento por el número de documentos que contiene esa palabra. Así una palabra que aparece en un solo documento con una frecuencia alta tendrá más importancia que una palabra con la misma frecuencia pero que se repite en todos los documentos.

Unos los problemas de esta representación es que generan matrices de muy alta dimensionalidad llenas de cero (matrices dispersas) que impactan en la velocidad de entrenamiento de los modelos.

2.2.2. Word embeddings

Word embeddings o incrustaciones de palabras en español, es una técnica que consiste en representar cada palabra como un vector de números con longitud fija. Estos vectores se aprenden mediante aprendizaje no supervisado empleando redes neuronales. Existen diversos métodos para realizar el aprendizaje de los vectores como son Word2Vec o FastText. Word2Vec (Mikolov et al. (2013)) propone dos métodos de aprendizaje:

- *Continuous Bag of Words* (C-BOW) donde hay que predecir las palabras centrales dadas las palabras vecinas.
- *Skip-gram* donde se trata de predecir las palabras vecinas dada una palabra central.

La idea detrás de estos métodos es que si se puede predecir el contexto en los que aparece la palabra entonces se puede entender el significado de la palabra en ese contexto. Si se proyectasen en el espacio

vectorial palabras de significado similar estas aparecerían muy próximas, mientras que palabras con semántica muy diferente aparecerían muy distanciadas, como puede verse en la figura 2.5.

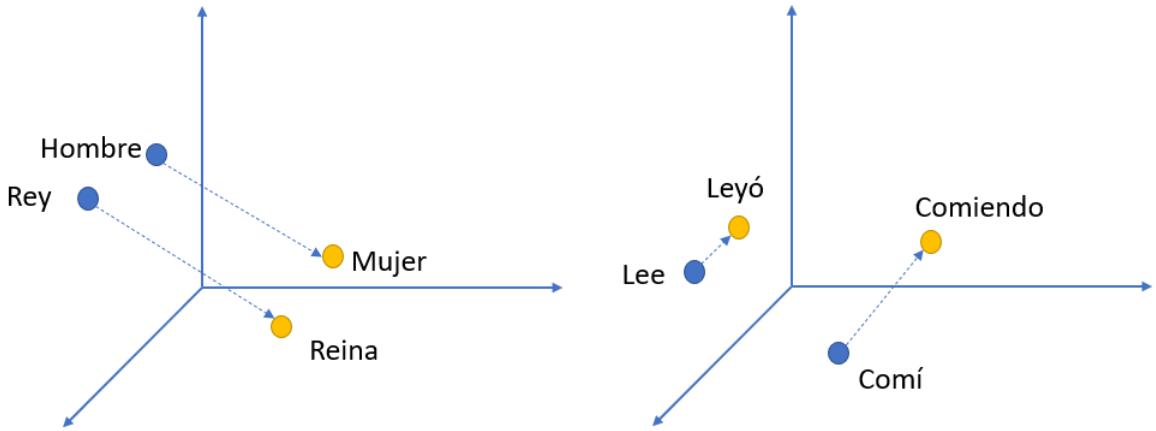

Figura 2.5: Representación de palabras en espacio vectorial

Word2Vec tiene cierta limitaciones como no poder tratar con palabras que no están dentro del vocabulario o que palabras que comparten raíz (por ejemplo "corren" y "corriendo") se aprende de forma separada basándose en el contexto en el que aparece. No existe un mecanismo para utilizar la estructura interna de la palabra para hacer el proceso más eficiente.

El método FastText (Mikolov et al. (2017)) explota la estructura interna de las palabras. En este método el vector de cada palabra se componen de combinaciones de n-gramas de longitud 3 a 6. Por ejemplo para "apple" se generaría "<ap, app, appl, apple>, ppl, pple, pple>, ple, ple>, le>". Este método permite generar vectores para palabras que no están dentro del vocabulario de entrenamiento y mejora las representaciones de palabras poco frecuente. Estos dos aspectos son los que han propiciado el uso de vectores preentrenados con este método en el trabajo, porque al tratar con textos informales es necesario emplear un conjunto de vectores que pueda cubrir un mayor número de palabras fuera del vocabulario de entrenamiento de estos vectores.

Una limitación de las incrustaciones de palabras preentrenadas es que palabras polisémicas solo tienen una representación: la palabra "banco" tendrá la misma representación en la frase "Estoy sentado en un banco" y "Saco dinero del banco". Estas dos frases guardan muy poca similitud. Sin embargo, al usar un mismo vector para la misma palabra un método que evalúe la similitud entra ambas frase podría dará un valor más alto del esperado. Este hecho afectaría a la precisión de los modelos. Este es uno de los problemas que los modelo BERT son capaces de resolver porque para asignar la representación de las palabras tienen en cuenta el contexto en que aparecen.

Capítulo 3

Estado del arte

En esta sección se exponen trabajos relacionados con la temática de este trabajo fin de máster. En la primera sección se presentan distinta soluciones para la detección de lenguaje ofensivo. En la segunda sección se expondrán los mejores trabajos presentados en distintas tareas de evaluación sobre detección de lenguaje abusivo.

3.1. Detección de lenguaje ofensivo

Desde la irrupción de los modelos *BERT* en 2018 muchas de las propuesta para detectar lenguaje abusivo emplean modelos *BERT* entrenados sobre corpus formales o sobre textos de redes sociales. En Caselli et al. (2020) se propone un nuevo modelo *BERT*, llamado *HateBERT*, entrenado sobre comentarios recolectados de comunidades de Reddit que fueron expulsadas por ser ofensivas y promover el odio, así que en el conjunto de datos existen abundantes muestras con con todo tipo de lenguaje abusivo. Los autores reentrenaron el modelo BERT base sobre un total de 1,5 millones de comentarios y posteriormente probaron el rendimiento del modelo sobre conjuntos de datos empleados en tres tareas de detección de lenguaje abusivo. Los resultado obtenidos por HateBERT (tabla 3.1) superaron a los obtenidos en esas tareas de evaluación.

Tarea	HateBert	BERT	Diferencia
OffensEval	**0,809**	0,803	0,6
AbusEval	**0,765**	0,727	3,8
HatEval	**0,516**	0,480	3,2

Tabla 3.1: Comparativa HateBERT con BERT

Además de sistemas para de la detección de lenguaje ofensivo otros autores como Ruiter et al. (2022) proponen un sistema de transferencia de texto para traducir comentarios ofensivos a no ofensivos. El sistema consiste en un codificador-decodificador, con la particularidad de que en su entrenamiento no se puede emplear un corpus paralelo (puesto que no existe) que les de la máxima

verosimilitud de del texto transferido. Este problema se trata de solventar mediante un clasificador colaborativo y además se incorpora un mecanismos de atención para reducir la perdida de contenido al realizar la transferencia de estilo. En este sistema los codificadores y decodificadores consisten en redes recurrentes (GRU+RNN) y el clasificador en redes convolucionales (CNN). Para realizar la evaluación se aplicó el método sobre comentarios de redes sociales y los textos resultantes fueron clasificados como ofensivo o no ofensivos por otro sistema. El resultado fue prometedor, con un 99 % de los comentarios clasificados como no ofensivos. Aún así el sistema presenta algunos problemas, por ejemplo, al sustituir palabras ofensivas por otras no ofensivas, sin tener en cuenta el contexto, provoca en ocasiones que se pierda el sentido de la frase.

En los trabajos anteriores se emplearon redes neuronales profundas, que es la tendencia actual pero antes de la proliferación de estas técnicas se empleaban otros enfoques basados en aprendizaje automático clásico como por ejemplo los usados en Nand et al. (2016). En este trabajo se extrajeron comentarios de Twitter que contenían ciertas palabras claves ofensivas y se realizó una clasificación manual de los comentarios en varias categorías. Para realizar la clasificación emplearon los métodos de bosques aleatorios, SMO, perceptrón multicapa de la librería Weka. Los modelos fueron entrenados con características extraídas mediante la librería LIWC[1], diseñada para el estudio de emociones en textos. Este librería convierte los texto en 67 valores numéricos que representan 67 categorías de palabras. El mejor resultado los obtiene el modelo de bosques aleatorios con un valor f1 de 94,47 %.

En Chen et al. (2012) se propone una arquitectura denominada LFS (*Lexical Syntactic Feature*) para la detección tanto de comentarios ofensivos como de usuarios potencialmente ofensivos. Proponen un método con dos fases: en la primera se trata obtener características léxicas y sintácticas de cada frase mediante técnicas de minería de textos y procesamiento del lenguaje natural. En la segunda fase incorporan características a nivel de usuario calculadas mediante el análisis de patrones de comportamiento del autor. Para medir el nivel de ofensa de una frase emplean un lexicón de palabras ofensivas y reglas sintácticas que regulan la intensificación de la ofensa. Para medir como de ofensivo es un usuario hacen una agregación del nivel de ofensa de su historial de mensajes y la combina con otras características como el estilo de escritura. Con este conjunto de características realizaron diversos experimentos con distintas variaciones del conjunto de datos. En el primer experimento se incluyeron palabra ofensivas fuertes y débiles, el método desarrollado no resultó ser tan efectivo como usar un método basado únicamente en palabras ofensivas. En el segundo experimento solo se emplearon palabras ofensivas débiles, en esta ocasión el método propuesto resultó el mejor. Los autores concluyen que a falta de palabras ofensivas es necesario interpretar el contexto para detectar lenguaje ofensivo y esta capacidad es la que han conseguido desarrollar con su método.

[1] http://www.liwc.net/liwcespanol/

3.2. Tareas de evaluación

En esta sección se van a analizar las principales tareas de evaluación sobre la detección de lenguaje abusivo en redes sociales. Es de especial interés realizar este análisis porque gran parte de la investigación sobre la detección de lenguaje abusivo se ha realizado en este contexto. De cada tarea se expondrán los métodos de clasificación más destacables, con lo que se podrá la evolución en los enfoques empleados para tratar este problema a lo largo del tiempo.

3.2.1. SemVal

Dentro del Taller Internacional de Evaluación Semántica (SemVal)[2] de 2019 y 2020 se han realizados tareas de detección de lenguaje ofensivo en redes sociales. Estas tareas reciben el nombre OffensEval[3] y se han celebrado hasta el momento dos ediciones.

3.2.1.1. OffensEval 2019

En OffensEval 2019 (Zampieri et al. (2019)) se realizaron 3 tareas empleando un conjunto de datos compuesto por comentarios de Twitter en inglés etiquetados a mano. Las 3 tareas realizadas fueron: clasificar los mensajes como "ofensivos" o "no ofensivos", clasificar el tipo de ofensa en "dirigida" (hacia una persona o grupo específico) o "no dirigida" e identificar a qué tipo de objetivo va dirigida la ofensa: a un individuo, grupo u otro objetivo.

En la primera tarea el ganador empleó un modelo BERT ajustado para tratar las clases de etiquetas menos representadas (Liu et al. (2019)). Las siguientes posiciones las ocuparon equipos que también emplearon modelos BERT con distintos ajustes. El primer equipo en usar un modelo no basado en BERT se posicionó en sexta posición empleando un ensamblaje de redes CNN y Bi-LSTM+GRU junto con vectores Word2vec preentrenados sobre Twitter (Mahata et al. (2019)).

En la segunda tarea predominaron los modelos basados en ensamblaje de clasificadores. El trabajo de Han et al. (2019) quedó como primer clasificado construyendo un modelo probabilístico basado en el cálculo del nivel de ofensa de los comentarios aplicando un diccionario con palabras claves y *hash-tags*. El segundo clasificado construyó una red neuronal basada en el ensamblaje de redes CNN junto con BERT (Rozental and Biton (2019)). El tercer clasificado combinó, mediante regresión logística, la salida de una red LSTM, cuyas entradas eran vectores contextuales ELMo, con características de los textos a nivel de palabras como unigramas y bigramas (Oberstrass et al. (2019)).

En la tercera tarea el mejor modelo se basó en un modelo BERT aplicando menos peso en las clases más representadas (Nikolov and Radivchev (2019)). El segundo equipo combinó modelos como OpenAI Finetune, LSTM, Transformers y otros modelos no basados en redes neuronales como SVM

[2] https://semeval.github.io
[3] https://sites.google.com/site/offensevalsharedtask/

y RandomForest y para asignar la etiqueta a cada instancia se utilizó un sistema de elección basado en la mayoría de votos (Seganti et al. (2019)).

3.2.1.2. OffensEval 2020

En OffensEval 2020 (Zampieri et al. (2020)) las tareas fueron idénticas a las de OffensEval 2019 pero el conjunto de datos en esta ocasión constaba de comentarios de Twitter en 5 idiomas: inglés, árabe, danés, griego y turco. En esta edición las etiquetas del conjunto de datos en inglés estaban asignada en gran medida mediante un proceso semisupervisado.

En la primera tarea el ganador utilizó un ensamblaje de modelos ALBERT de distintos tamaños (Wiedemann et al. (2020)). El segundo clasificado empleó un modelo RoBERTa-large (Zampieri et al. (2020)) y el tercero usó un modelo basado en el ensamblaje de modelos basados en XLM-RoBERTa (Wang et al. (2020)). En esta primera tarea los 10 primeros clasificados emplearon modelos BERT, algunas veces como parte de un ensamblaje con otras redes de tipo CNN o LSTM.

En la segunda tarea el primer puesto lo obtuvo el modelo desarrollado por Wang et al. (2020), que propuso un modelo basado en XLM-RoBERTa . El segundo clasificado combinó un modelo BERT con capas LSTM , cuyo entrenamiento empleó el método *Noisy Student* para reducir el ruido que puede provocar el etiquetado semisupervisado (Pham-Hong and Chokshi (2020)). El equipo en tercera posición creó una arquitectura que le permitía realizar las tres subtareas de forma jerárquica empleando modelos BERT (Chen et al. (2020)).

En la tercera Wang et al. (2020) obtiene el primer puesto como hizo en la tarea anterior, proponiendo de nuevo un modelo XLM-RoBERTa. El segundo clasificado empleó un modelo BERT con sobremuestreo para mejorar las clases desbalanceadas (Pàmies et al. (2020)). El tercero combinó BERT con algunas características de los textos como longitud de los tweets, palabras mal escritas o uso de emojis (De la Pena Sarracén and Rosso (2020)).

3.2.2. OSACT4

En la tarea OSACT4 celebrada en 2020 se aborda la tarea de detección de lenguaje ofensivo y de discurso de odio en comentarios de Twitter en árabe. El primer clasificado formó un sistema con dos modelos SVM, una red CNN+BiLSTM y un modelo M-BERT. La etiqueta de cada comentario se decidía por el voto mayoritario (Hassan et al. (2020)). El segundo clasificado empleó el modelo AraBERT con un ajuste fino (Djandji et al. (2020)). El tercero empleó un modelo de aprendizaje tradicional SVM pero aplicó un preprocesado intenso: tratamiento de emoticonos, categorización de palabras, normalización de letras, segmentación de hashtag y conversión de emojis a texto.

3.2.3. IberLeF

En 2019 los foros de evaluaciones TASS e IberEval se unen para formar un único foro llamado IberLef. Entre 2018 y 2020 se celebraron las tareas MEX-A3T que consistían en la detección de lenguaje agresivo en comentarios de Twitter en español de México. Por otro lado, en 2021 se celebra la primera edición de MeOffendES, donde se trata la detección de lenguaje ofensivo en comentarios en español extraídos de diversas plataformas (Instagram, Twitter y Youtube) sin diferenciar la variante del idioma.

3.2.3.1. MEX-A3T 2018

En la edición de 2018 de MEX-A3T (Álvarez-Carmona et al. (2018)) el ganador de la tarea empleó un método basado en el ensamblaje de distintos clasificadores junto con un lexicón de palabras afectivas y agresivas (Sánchez-Gómez (2018)). El segundo clasificado empleó una red LSTM y el tercero propone un método formado por 4 etapas donde se generan distintas representaciones de n-gramas y emplean un modelo SVM para la clasificación (Aragón and López-Monroy (2018)).

3.2.3.2. MEX-A3T 2019

En MEX-A3T 2019 (Aragón et al. (2019)) el trabajo de Casavantes et al. (2019) quedó primero utilizando una red neuronal perceptrón multicapa junto con vectores FastText, mientras que el segundo clasificado, y baseline de la tarea, es el ganador de la tarea del año anterior. El tercer clasificado utilizó una red perceptrón multicapa con los textos representados con TF-IDF (De la Peña Sarracén and Rosso (2019)).

3.2.3.3. MEX-A3T 2020

En esta edición de MEX-A3T (Aragón et al. (2020)), el ganador crea un modelo basado en el ensamblaje de modelos BERT en español (BETO) y además aplica aumento de datos cambiando palabras por sinónimos o intercambiando posiciones de palabras (Guzman-Silverio et al. (2020)). El segundo clasificado empleó también un modelo BETO y aumentó el conjunto de entrenamiento añadiendo muestras del conjunto de datos de HatEval Spanish[4] que contiene comentarios con discurso de odio extraídos de Twitter (Tanase et al. (2020)). El tercer clasificado al igual que los anteriores también usó BETO y añaden al modelo metadatos de los comentarios y usuarios con la librería GetOldTweets3 (Casavantes et al. (2020)).

3.2.3.4. MeOffendEs 2021

En esta tarea, celebrada también dentro de los foros IberLef, se propone la identificación de lenguaje ofensivo extraídos de tres redes sociales y su clasificación en 4 categorías (Plaza-del Arco et al.

[4] https://competitions.codalab.org/competitions/19935

(2021)). El primer clasificado lo hizo con un modelo multilenguaje XLM-RoBERTa preentrenado con textos de Twitter y con análisis de sentimiento. El segundo clasificado empleó una combinación del modelo BERT con características del lenguaje, como por ejemplo el uso de las negaciones (Garcá-Dıaz et al. (2021)). El tercer clasificado también empleó un modelo preentrenado BERT al que aplicaron *pseudo-labeling* para ampliar el conjunto de etiquetado y *focal-loss* para tratar el desequilibrio en el número de muestras en cada etiqueta.

3.3. Conclusiones

La primera tarea analizada es de 2018 que coincide con la publicación del primer modelo basado en Transformers, al tratarse de un modelo muy reciente no aparecen referencias a el en esta tarea. En las tareas de 2019 aparecen algunos modelos basados en Transformers como BERT, aparecen en la tarea OffensVal pero no en MEX-A3T 2019. Esto sucede porque hasta 2020 no aparece la primera versión del modelo BERT en español (Canete et al. (2020)). En MEX-A3T 2020 se empleó el primer modelo en español, llamado BETO. En la tarea OffensVal del mismo año se utilizan variantes del modelo BERT, como RoBERTa para tareas multilenguaje. En la edición de 2021 se ve un aumento del uso de modelo BERT, donde se aplican nuevas variantes como XLM-RoBERTa o ALBERT.

Con los datos expuestos es evidente que la introducción de los modelos BERT ha supuesto un cambio en la forma de trabajar con datos basado en textos. Por este motivo, en este trabajo también se probarán estos modelos y se hará uso de una variante entrenada sobre textos procedentes de Twitter. A priori esto debe suponer una mejora puesto que el corpus y el conjunto de datos procederán del mismo contexto.

Capítulo 4

Marco de trabajo

En este capítulo se detallará el conjunto de datos utilizado en el trabajo, las métricas empleadas para evaluar los modelos y las herramientas empleadas para el tratamiento de los datos.

4.1. Conjunto de datos

El conjunto de datos empleado procede de la tarea MeOffendES celebrada en el *Iberian Languages Evaluation Forum* celebrado en 2021 (IberLEF (2021)). Está formado por un total de 30 416 comentarios recogidos entre febrero y marzo de 2020 de tres plataformas distintas: Twitter, Instagram y YouTube. Concretamente se recolectaron comentarios de las cuentas de 12 *influencers* españoles que generan gran controversia y tienen un número significativo de seguidores, cuyas edades están entre los 14 y 24 años.

Los comentarios se etiquetaron manualmente a través de la plataforma Amazon Mechanical Turk[1]. Las etiquetas disponibles son:

- NO: comentario no ofensivo y sin lenguaje soez.
- NOM: comentario no ofensivo pero que contiene lenguaje soez.
- OFG: comentario ofensivo hacia un grupo de personas que pertenecen a una misma orientación sexual, religión o seguidores de un determinado *influencer*.
- OFP: comentario ofensivo hacia una persona.

La calidad del etiquetado se evaluó mediante el coeficiente kappa. Este índice mide el grado de concordancia entre varios evaluadores al asignar etiquetas en un conjunto de datos. El valor máximo del índice es uno, que indica un grado de acuerdo casi perfecto, hasta valores menores de cero cuando no existe acuerdo. En la tabla 4.1 se muestra el grado de acuerdo según el valor de este índice (Abraira

[1]https://www.mturk.com/worker

(2001)).El coeficiente kappa del conjunto de datos es del 39,37 %, un valor no demasiado alto que se debe principalmente a las discrepancias entre los anotadores en la clasificación de los comentarios como "OFP" o "OFG".

Kappa	Grado de acuerdo
<0	Sin acuerdo
0,00-0,20	Insignificante
0,21-0,40	Mediano
0,41-0,60	Moderado
0,61-0,80	Sustancial
0,81-1,00	Casi perfecto

Tabla 4.1: Grados de acuerdo según índice kappa

En la tabla 4.2 se muestra el número comentarios por etiquetas y conjunto de datos.

Etiqueta	Entrenamiento	%	Validación	%	Pruebas	%	Total	%
OFP	2051	12,27	10	10	1404	13,32	3465	**11,39**
OFG	212	1,27	4	4	211	1.55	427	**1,40**
NOM	1235	7,39	22	22	2340	17,20	3597	1183
NO	13 212	79,07	64	64	9651	70,93	22 927	75,38
Total	16 710	100	**100**	100	13 606	100	30 416	100

Tabla 4.2: Distribución de las etiquetas del conjunto de datos

Como puede apreciarse en la tabla 4.2 las etiquetas se encuentran muy desbalanceadas, los comentarios etiquetados como ofensivos suponen solo el 12 % del conjunto de datos completo. Este aspecto se tendrá en consideración en los entrenamientos de los modelos, se tendrán que aplicar los mecanismos disponibles en cada modelo para entrenar con conjuntos de datos con etiquetas desbalanceadas. Otro aspecto destacable del conjunto de datos es que el tamaño del conjunto de validación es muy inferior al del conjunto de entrenamiento. La ratio entre ambos es del 0,59 % mientras que lo normal estaría entre el 10 % y el 20 %. En el artículo de la tarea no se ofrece ninguna explicación sobre la elección de esta proporción en los datos. Los participantes de la tarea adoptaron distintas estrategias, por ejemplo, Garcıa and Bedmar (2021) ignoró el conjunto de validación y realizó un división típica del 80/20 del conjunto de entrenamiento, Garcá-Dıaz et al. (2021) unió los conjuntos de entrenamiento y validación antes de realizar una partición 80/20 y, por último, el primer clasificado de la tarea trabajó con el conjunto de validación original.

Los dos aspectos comentados arriba van a afectar directamente a la calidad de los modelos, ya que las dos etiquetas que resultan más interesantes para entrenar el modelo están poco representadas y, además, la fiabilidad de las mismas no es todo lo alta que se podría desear.

4.2. Métricas de evaluación

Las métricas son las medidas que ayudan a evaluar el desempeño de los modelos. Se van a emplear las siguientes métricas:

- Precisión: número de elementos positivos clasificados correctamente entre el número total de elementos clasificados.
- Exhaustividad: número de elementos positivos clasificados correctamente entre el total de elementos positivos reales.
- f1: es la media armónica de la precisión y la exhaustividad.
- Exactitud: ratio entra las predicciones correctas y las totales.

Las 3 primeras métricas se calcularán para cada etiqueta y también de forma global, mientras que la exactitud solo se calcula de forma global. El cálculo global de las 3 primeras métricas puede realizarse de 3 formas distintas:

- Macro: media aritmética de los valores obtenidos en la métrica.
- Micro: corresponde a la métrica exactitud.
- Ponderada: media ponderada al número de instancias de etiqueta de los valores obtenido en la métrica calculada.

Al trabajar con etiquetas desbalanceadas se prestará atención fundamentalmente a las métricas macro pues será la que mejor refleje el rendimiento de los modelos en las clases menos representadas, que al tratarse de las clases ofensivas requieren especial atención. Por defecto, siempre que se haga referencia a una métricas se estará refiriendo, por defecto, a su valor macro.

4.3. Entorno de trabajo

En el trabajo se ha empleado exclusivamente Python como lenguaje de programación que se ha combinado con el uso de librerías y otras herramientas como base de datos y entornos de programación. En esta sección se dará una visión general de estas librería y herramientas.

4.3.1. Librerías

- **Pandas**[2]**:** es la herramienta para el análisis y manipulación de datos más popular de Python. Se integran con otras librerías como Sciki-learn o Keras.

[2] **https://pandas.pydata.org/**

- **Spacy:** librería de procesamiento de lenguajes natural basada en el uso de redes neuronales.
- **NTLK:** es otra librería de procesamiento de lenguajes natural simbólico y estadístico.
- **Scikit-learn** [3]: es una las principales librerías de Python para aprendizaje automático incluye algoritmos para clasificación, regresión y agrupamiento. También incluye funciones para tratamiento de texto como por ejemplo la vectorización con TF-IDF.
- **Imbalanced-learn**[4]**:** proporciona métodos para aplicar sombremuestreo y submuestreo en conjuntos de datos. Tiene buena integración con Scikit-learn.
- **Keras/Tensorflow**: Keras[5] es una librería que ofrece una interfaz para la creación de redes neuronales. Es capaz de trabajar sobre Tensorflows y otros backend como Microsoft Cognitive Toolkit o Theano.
- **Huggingface**[6]: es una librería que da acceso a través de una interfaz común a una gran cantidad de modelos preentrenados como BETO o RoBERTuito.
- **Pyspellchecker**[7]**:** ofrece corrección de palabras aplicando el algoritmo de distancia Levenshtein. Como corpus emplean subtítulos de películas, estos son obtenidos de la plataforma OpenSubtitle.

4.3.2. Entorno de desarrollo

Para el desarrollo del trabajo se ha combinado el uso de un entorno local y otro en la nube. En el entorno local se ha utilizado como IDE JupyterLab[8] y como entorno en la nube se ha empleado Google Colab[9] y Google Drive[10] como sistema de almacenamiento.

Tanto JupyterLab como Google Colab son entornos de desarrollo interactivos basados en la web en los que se trabaja en celdas donde se pueden ejecutar código, mostrar gráficos o escribir texto en formato Latex. En el entorno local se realizó el preprocesado de los textos, cálculos de métricas y el entrenamiento de modelos de aprendizaje con Scikit-Learn. En Google Colab, se entrenaron los modelos basados en redes neuronales y aprendizaje profundo, la ventaja de esta plataforma es la disponibilidad de GPUs para ejecutar estos modelos en menos tiempo que empleando CPU.

[3] https://scikit-learn.org/
[4] **https://imbalanced-learn.org/**
[5] https://keras.io/
[6] https://huggingface.co/
[7] **https://pypi.org/project/pyspellchecker/**
[8] https://jupyter.org/
[9] https://colab.research.google.com/
[10] https://drive.google.com/

4.3.3. Persistencia de datos

El sistema para dar persistencia a los datos generados debía cumplir dos requisitos fundamentales, el primero es ofrecer flexibilidad para modificar la estructura de los datos en cualquier momento, por ejemplo, añadir campos con cualquier tipo de valor, como texto, arrays o diccionarios. El segundo requisito era tener comunicación con el entorno local y con el entorno en la nube, de esta forma un dato creado en local puede leerse directamente desde la nube. El sistema que se implantó fue una base de datos MongoDB en su versión Cloud, MongoDB Atlas[11]. MongoDB es una base de datos NoSQL orientada a documentos y código abierto. Este sistema de almacenamiento guarda los datos en un formato similar a JSON con un esquema dinámico que se puede modificar a medida que se insertan los datos sin necesidad de crear un esquema nuevo.

[11] https://www.mongodb.com/atlas/

Capítulo 5

Preparación de experimentos

En este capítulo se presentan los trabajos realizados para la preparación de los datos de entrenamiento de los modelos que se desarrollarán en la fase de experimentación. En la sección 5.1 se detalla el procesado realizado a los textos con el que se obtendrá una versión limpia de estos. En la sección 5.2 se indican las características y métricas del lenguaje obtenidas sobre los textos limpios. Por último en la sección 5.3 se explica como se obtienen variaciones del conjunto de datos combinando distinta información lingüística.

5.1. Procesado de datos

Los comentarios en las redes sociales son informales así que contienen muchos elementos que introducen ruido y que reducen la efectividad de las herramientas de tratamiento de texto que trabajan frecuentemente con corpus basados en texto formales como noticias, libros... En esta tarea se han implementado métodos para limpiar los comentarios y hacerlos los más formales posible. El resultado de la limpieza es un nuevo texto limpio que se identificará con el literal "comment_c" . Este texto será empleado para el cálculo de métricas del lenguaje, que se describen en la sección 5.2, y también como referencia para generar distintas representaciones de los textos como se detallará en la sección 5.3.

Los métodos de procesado implementados son:

- **Eliminar frases repetidas**: es frecuente encontrar comentarios en los que se repiten de forma consecutiva una misma palabra o frases completas. Este método reduce el texto para dejar solo la primera aparición de la palabra o frase, en la tabla 5.1 puede verse el efecto de este método. Con una única repetición se tiene la misma información que con el comentario completo. Además de eliminar las repeticiones, se calcula una métrica con la ratio entre el tamaño del texto procesado y el texto original.

- **Eliminar repeticiones de caracteres**: en los comentarios es frecuente encontrar repeticiones del mismo carácter en una palabra como forma de enfatizarla. Esto provoca que las palabras

Sin procesar	Procesado
"Correr es vivir, Correr es vivir, Correr es vivir, Correr es vivir,"	*"Correr es vivir,"*

Tabla 5.1: Ejemplo de frase repetida

queden fueran del vocabulario haciendo que librerías como Spacy o NTLK no puedan analizarlas o que no se les pueda asignar un vector cuando se usan embeddings preentrenados. El método implementado elimina las repeticiones de caracteres pero tiene en cuenta que existen repeticiones válidas como las consonantes "r", "l", "c" ó "n". En la tabla 5.2 puede verse le resultado de aplicar este método. Gracias a el se han podido recuperar 2 palabras ("curro" y "año") que en el comentario original estarían fuera del vocabulario.

Sin procesar	Procesado
*"El -13 tiene un **currrrooooo** hace **aaaños**"*	*"El -13 tiene un **curro** hace **años**",*

Tabla 5.2: Ejemplo de frase repetida

- **Tratamiento de escritura *leet:*** la escritura *leet* (pronunciado *lit*) consiste en sustituir determinadas letras por números cuya forma tiene cierto parecido con la letra a la que sustituyen. El objetivo es dificultar la lectura a los usuarios ajenos a este tipo de escritura[1] pero también dificultan que los sistemas informáticos puedan interpretar los mensajes. Se ha implementado un método que trata las palabras donde se encuentran mezclas de números y letras, en estos casos se realizaría la conversión de letras a número indicada en la tabla 5.3. En la tabla 5.4 se muestra un ejemplo del resultado de aplicar este método. En el se recuperarían 4 palabras que estaban fuera del vocabulario.

Número	Letra
0	O
1	I
3	E
4	A
5	S
7	T
8	B

Tabla 5.3: Tabla de conversión número a letra

- **Limpieza de número**: todos los números se han sustituido por un mismo número, de esta forma se reduce el número de tokens distintos y en consecuencia se reduce la dimensionalidad de los textos vectorizados.

[1] https://es.wikipedia.org/wiki/Escritura_leet

Sin procesar	Procesado
"el que da **m3 gu5t4** (**m1ra** mi **nombr3**)"	"el que da **me gusta** (**mira** mi **nombre**)"

Tabla 5.4: Ejemplo de lenguaje leet

- **Limpieza de emojis:** se han eliminado los emojis de los comentarios pero se han almacenado en una lista asociada al comentario dentro de la base de datos. Para el tratamiento de los emoticonos se ha empleado la librería "emoji[2]". No se debe confundir los emojis con los emoticonos. Los emojis son pictogramas que se incrustan en los textos y para los que existe un estándar definido en el *unicode consortium*[3], gracias a esta estandarización, librerías como Emoji, pueden detectar los emojis en los textos y también sustituirlos por su descripción textual. En cambio, los emoticonos son una combinación de caracteres que tratan de representar expresiones faciales para expresar sentimientos. Para estos no existe un estándar, sí existen distintos estilos (occidental, japonés, coreano...) y un número indefinidos de emoticonos, esto hace complejo su tratamiento y, por ello, no se aplicó ninguna técnica para tratarlos.

- **Limpieza de URL**: las URL detectadas se han sustituido por la palabra "dirección".

- **Limpieza de hashtag o etiquetas**: se sustituyen los hashtags por la palabra "etiqueta", a cada comentario se le asocia una lista con los hashtags encontrados y la cantidad encontrada.

- **Limpieza de usuarios:** se sustituyen los nombres de usuarios por la palabra "usuario". Se considera que una palabra es un nombre de usuario cuando empieza por "@" y va seguido por una letra mayúscula. Se almacenan en una lista los nombres de usuarios encontrados junto con el número total de usuarios encontrados.

- **Estandarización de risas**: las formas de representar risas en un texto son tremendamente variadas, se ha tratado de detectar el mayor número de representaciones posibles. Todas ellas se han sustituido por la palabra "risas" y se guarda la cantidad encontrada.

- **Ajustes de espacio**s: se ha aplicado un tratamiento específico para los espacios en blanco:

  - Las secuencias de 2 o más espacio se sustituye por un solo espacio
  - Se introduce espacio detrás de los símbolos "?" y "!" para mejorar la efectividad del tokenizado de Spacy. En ocasiones Spacy considera que estos símbolos forman parte de la palabra, por ejemplo, "palabra!" no la separan en "palabra" y "1" y provoca que el primer token quede fuera del vocabulario.

[2]https://pypi.org/project/emoji/?msclkid=b2f90091ceac11ecbeca378d9a302f28
[3]http://www.unicode.org/emoji/charts/full-emoji-list.html

5.2. Cálculo de características lingüísticas

A cada comentario se le ha asociado una serie de características, calculadas a partir del textos, con las que se entrenarán modelos de aprendizaje para etiquetar los comentarios. Las métricas se han obtenido de dos formas distintas. La primera, mediante métodos de cálculo implementados en paralelo a los de procesamiento de los textos. Por ejemplo, al mismo tiempo que se limpian *emojis*, se elabora una lista con ellos y se calcula la longitud de la misma. A estas características se le denominará "características propias" y la lista completa es la siguiente:

- comment_len: longitud del comentario sin procesar.
- clean_comment_len: longitud del comentario procesado.
- uppers_count: cantidad de palabras escribas con todas sus letras en mayúsculas.
- oovs_len: cantidad de palabras fuera del vocabulario. Se considera el vocabulario de Spacy y el de la librería Pyspellchecker[4].
- adjs_len: cantidad de adjetivos.
- nouns_len: cantidad de nombres .
- propns_len: cantidad de nombres propios .
- corrections_len: cantidad de palabras corregidas por Pyspellchecker.
- emojis_len: cantidad de emojis.
- laughs_len: cantidad de "risas".
- users_len: cantidad de referencias a usuarios.
- verbs_list: cantidad de verbos.
- hashtags_len: cantidad de hashtags.
- polaridad del comentario: puntuación que recibe la polaridad del texto dividido en negativo, positivo y neutro.

El otro método de cálculo de características es mediante la librería TextComplexity (y Jaime Collado-Montañez y Arturo Montejo-Ráez (2020)). Las características obtenidas con esta librería pueden agruparse en tres categorías:

- Conteos de letras, sílabas o palabras raras.

[4]https://pypi.org/project/pyspellchecker

- Basadas en cálculo medios como número de sílabas por palabras, de palabras por frase o proporción de palabras léxicas por frase.
- Métricas de complejidad que miden como de compleja es la lectura del texto en base al tipo de palabras o longitud de las frases

La lista detallada puede consultarse en el anexo A A.1.

5.3. Creación de conjuntos de datos

El resultado del procesado realizado en la sección 5.1 es un texto donde se han recuperado palabras y eliminado elementos que pueden introducir ruido (número, nombres de usuarios, hashtag, etc). Sobre este resultado se van a generar nuevos conjuntos de datos con distinta información textual. Para obtener estos conjuntos se aplicarán las siguientes operaciones:

- Truncando de las palabras (TR): las palabras son convertidas a su raíz
- Lematizado (LE): se obtiene el lema de las palabra. Como por ejemplo la forma singular para sustantivos, masculino singular para adjetivos e infinitivo para verbos.
- Borrado de todas las palabras vacías (SW): se eliminan las palabras que consideran que no tiene significado (artículos, pronombres, preposiciones). Estás palabras vienen en listados predefinidos según el idioma.
- Borrado de palabras vacía*s* pero manteniendo los pronombres personales (SWP).
- Borrado de palabras que no tengan significado propio (CO): las palabras se etiquetan según su función en el texto y luego se eliminan aquellas que no han sido etiquetadas como sustantivos, verbos, adjetivos o marcada como "sin etiqueta". A diferencia del método anterior, aquí no se parte de una lista fija si no que se analiza cada palabra para asignar las etiquetas según según su función en la frase.

En la tabla 5.5 se resumen los conjuntos de datos disponibles. Estos conjuntos de datos serán evaluados en la sección 6.4, con el objetivo de encontrar la información textual que mejor rendimiento aporta a cada modelo.

conjunto	Descripción	TR	LE	SW	SWP	CO
comment_s	Truncado sin palabras vacías	X		X		
comment_cs	Truncado solo con palabras léxicas	X				X
comment_at	Sin palabras vacías			X		
comment_sp	Truncado con pronombres personales	X			X	
comment_atp	Sin palabras vacías y con pronombres				X	
comment_cat	Truncado solo con palabras léxicas					X
comment_l	Lematizado con pronombres		X		X	
comment_ls	Lematizado y truncado con pronombres	X	X		X	

Tabla 5.5: Conjunto de datos generados

Capítulo 6

Experimentos y resultados

En este capítulo se van a detallar los experimentos realizados. En este trabajo se han abordado dos líneas de experimentación, la primera consiste en la búsqueda del mejor modelo de aprendizaje para la detección de comentarios ofensivo y su clasificación, esto se verá en la sección6.3. La segunda línea, desarrollada en paralelo con la anterior, consiste en estudiar como afectan distintos preprocesado de los texto en los resultados de los modelos, esto se verá en la sección 6.4.

6.1. Modelos de referencia

Como se comentó en la sección 1.3 el conjunto de datos que se emplea en este trabajo fue usado en la tarea de evaluación MeOffendES 2021 (Plaza-del Arco et al. (2021)). Los resultados de los participantes se muestran las tablas 6.1 y 6.2. Estos resultados serán tomados como referencias para medir el rendimiento de los modelos desarrollados en este trabajo en la clasificación multietiqueta.

En la tarea MeOffendES 2021 no se dispone de resultados de clasificación binaria con el conjunto de datos empleado en este trabajo. Para tener una referencia se ha tomado el resultado obtenido en del Arco et al. (2021) donde emplean una versión menos refinada del conjunto de datos. Este resultado se muestra en la tabla 6.3.

Equipo	Precisión	Exhaustividad	F1
NLP-CIC	0,7679	0,7093	**0,7324**
UMUTeam	**0,7861**	0,6919	0,7301
GDUFS_DM	0,7565	0,7002	0,7239
Marta_Isabel	0,5781	0,5451	0,5595
baseline-svm	0,6278	0,4831	0,5236

Tabla 6.1: Resultados clasificación multietiqueta en MeOffendES 2021. Los resultados se refieren a las puntuaciones de Macro-precision, Macro-Recall y Macro-F1.

Equipo	Precisión	Exhaustividad	F1
NLP-CIC	0,8815	0,8815	0,8815
UMUTeam	0,8782	0,8782	0,8782
GDUFS_DM	0,8732	0,8732	0,8732
Marta_Isabel	0,8416	0,8416	0,8416
baseline-svm	0,8285	0,8285	0,8285

Tabla 6.2: Resultados clasificación multietiqueta en MeOffendES 2021. Los resultados se refieren a las puntuaciones de Micro-precision, Micro-Recall y Micro-F1.

Equipo	Precisión	Exhaustividad	F1
Offendes	0,8042	0,7674	0,7839

Tabla 6.3: Resultado clasificación binaria. Los resultados se refieren a las puntuaciones de Macro-precision, Macro-Recall y Macro-F1.

6.2. Ajuste de hiperparámetros

Para la búsqueda de los mejores hiperparámetros de cada modelo se han empleado dos tipos de búsquedas: la exhaustiva y búsqueda aleatoria. Ambos métodos reciben una lista de valores posibles para cada hiperparámetro y con ellos realizan entrenamientos de forma masiva de modelos con distintas combinaciones de valores. La diferencia entre ambos métodos es que en la búsqueda exhaustiva se entrenan tantos modelos como combinaciones de hiperparámetros sean posibles, mientras que en la búsqueda aleatoria se entrena un número fijo de modelos con combinaciones aleatoria de hiperparámetros, no se prueban todas las combinaciones.

La búsqueda exhaustiva se ha aplicado a los modelos de aprendizaje automático porque al ejecutarse en un entorno local no se tenían limitaciones de tiempo de uso de los recursos como sí ocurría en el servicio Google Colab. En algunos modelos se realizaron hasta 3 iteraciones, cada vez con unos parámetros más ajustados. Para generar la lista de valores numéricos se emplearon dos métodos distintos:

- Generación de valores mediante escala logarítmicas. Se aplicó principalmente en el primer ajuste porque permite obtener valores con distintos órdenes de magnitud de forma fácil.

- Generación de valores mediante incrementos fijos. Para ello, se establece un valor de referencia y se van creando nuevo valores sumando o restando una cantidad fija al valor anterior. Se aplicó cuando ya tenía un valor de referencia y se quería buscar un mejor valor en un intercalor alrededor suya.

Por otro lado, la búsqueda aleatoria se ha aplicado a los modelos de aprendizaje profundo porque sus tiempos de entrenamiento son más prolongados y por las limitaciones de tiempo del servicio Colab. En estos casos primero se entrenó un modelo de referencia con parámetros establecidos manualmente

y luego se aplicó la búsqueda aleatoria. En algunos casos la búsqueda aleatoria no consiguió superar el resultado del modelo de referencia.

En las búsquedas la métrica empleada para evaluar el rendimiento de la validación cruzada fue balanced_accuracy_score[1], que se define como la media de la exhaustividad obtenida en cada clase. Se eligió esta métrica porque puede trabajar de forma adecuada con problemas de clasificación con etiqueta desbalanceadas como ocurre en el conjunto de datos de entrenamiento.

6.3. Experimentos de clasificación

En la experimentación sobre la clasificación de los comentarios se han desarrollado experimentos para realizar la clasificación de los comentarios en una de las cuatro etiquetas posibles y también experimentos para realizar clasificación binaria, en la que los comentarios son etiquetados como ofensivo o no ofensivo. Para la clasificación multietiqueta se probaron dos grupos de modelos diferenciados por el tipo de entrada:

- Características lingüisticas: se emplean las características del lenguaje vistas en las sección 5.2.
- Palabras: se toman los textos como entradas.

En la clasificación binaria además se realizaron pruebas con un modelo que combina métricas del lenguaje y palabras.

6.3.1. Clasificación multietiqueta basadas en características del lenguaje

En esta clasificación se emplearon las métricas calculadas durante el preprocesado y las calculadas por la librería TextComplexity, en total se obtienen 46 métricas del lenguaje. Algunas de estas métricas están correlacionadas, como por ejemplo "número de caracteres" y "longitud del texto" o las diversas métricas de complejidad de lectura como SSR, ARI o Huerta. Para reducir el número de variables y emplear las de mayor capacidad predictiva se abordó una tarea de selección de variables antes de pasar al entrenamiento de los modelos. En esta tarea de selección se aplicaron 3 métodos distintos: búsqueda recursiva, búsqueda secuencial y el método k-Best. Estos métodos pueden devolver resultados dispares así que para seleccionar las variables se aplicó el criterio de elegir las de mayor consenso entre los métodos. Tras este proceso las variable seleccionadas fueron:

- Cantidad de emojis
- Cantidad de sustantivos
- Polaridad negativa

[1] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html

- Polaridad positiva
- Complejidad de las frases
- Palabras distintas
- Cantidad de palabras de contenido (aquellas con significado propio, como nombres o verbos)
- Cantidad de sílabas
- *ILFW:* Número de palabras de baja frecuencia
- *ARI*: Dificultad para leer el texto
- *SOL*: Edad mínima para entender el texto

Con estas variables se entrenaron los modelos que se presentan en las siguientes secciones.

6.3.1.1. RandomForest

El modelo se entrena con los hiperparámetros:

- criterion: gini
- max_features: sqrt
- n_estimators: 300

En tabla 6.4 se puede apreciar que el modelo sufre de un sobreajuste excesivo porque obtiene un 100 % en todas las métricas durante el entrenamiento. Sin embargo, en pruebas las puntuaciones son significativamente más baja. Destaca que el modelo **no tiene ningún acierto en la etiqueta "OFG**", todas las métricas relacionadas a dicha etiqueta están a cero.

	Entrenamiento		
	Precisión	Exhaustividad	f1
NO	1,00	1,00	1,00
NOM	1,00	1,00	1,00
OFG	**1,00**	**1,00**	**1,00**
OFP	1,00	1,00	1,00
macro	1,00	1,00	1,00
micro	1,00	1,00	1,00

	Prueba		
	Precisión	Exhaustividad	f1
NO	0,7973	0,9751	0,8773
NOM	0,5763	0,1186	0,1967
OFG	**0,0000**	**0,0000**	**0,0000**
OFP	0,6389	0,4117	0,5007
macro	0,5031	0,3763	0,3937
micro	0,7751	0,7751	0,7751

Tabla 6.4: Randomforest basado en métricas

6.3.1.2. GradientBoosting

El modelo se entrena con los hiperparámetros:

- n_estimators: 500
- learning_rate: 0,1
- max_features : sqrt
- max_depth: 3
- criterion : friedman_mse

Como se ve en la tabla 6.5, este modelo no sufre tanto sobreajuste como el modelo basado en RandomForest excepto en la etiqueta OFG donde se produce una diferencia notable con el resultado en entrenamiento.

	Entrenamiento				**Prueba**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9049	0,9803	0,9411	NO	0,8109	0,9647	0,8811
NOM	0,8113	0,3587	0,4974	NOM	0,5101	0,1443	0,2250
OFG	**1,00**	**0,5094**	**0,6750**	OFG	**0,0454**	**0,0047**	**0,0085**
OFP	0,7502	0,6372	0,6891	OFP	0,6268	0,4551	0,5273
macro	0,8666	0,6214	0,7006	macro	0,4983	0,3922	0,4105
micro	0,8863	0,8863	0,8863	micro	0,7779	0,7779	0,7779

Tabla 6.5: GradientBoosting sobre métricas

6.3.1.3. SGDClassifier

El modelo se entrena con los hiperparámetros:

- loss: log
- penalty :l2
- alpha: 0,0001
- epsilon: 0,001
- tol: 0,0001

Este modelo presenta (tabla 6.6) en pruebas valores muy similares a los obtenidos en entrenamiento lo que indica que no sufre sobreajuste pero al igual que los anteriores modelos los resultado en la etiqueta “OFG” siguen siendo bajos.

	Entrenamiento			Prueba		
	Precisión	Exhaustividad	f1	Precisión	Exhaustividad	f1
NO	0,9065	0,8438	0,8740	0,8512	0,8424	0,8468
NOM	0,5200	0,0421	0,0779	0,5714	0,0400	0,0748
OFG	0,0378	0,3632	0,0684	0,0378	0,3270	0,0678
OFP	0,4316	0,4787	0,0684	0,4681	0,4255	0,4457
macro	**0,4740**	**0,4319**	**0,3686**	**0,4821**	**0,4087**	**0,3588**
micro	0,7336	0,7336	0,7336	0,6803	0,6803	0,6803

Tabla 6.6: SGDClassifier sobre métricas

6.3.1.4. Resumen de resultados

Los resultados obtenidos se muestran en la tabla 6.8 junto con los resultados del ganador de la tarea, *NLP-CIC*, y con el *baseline* de la misma. Todos los resultados obtenidos tienen una diferencia superior a los 10 puntos respecto al baseline en la tarea.

Modelo	Precisión	Exhaustividad	f1
NLP-CIC	**0,7679**	**0,7093**	**0,7324**
baseline-svm	0,6278	0,4831	0,5236
GradientBoosting	0,4983	0,3922	***0,4105***
RandomForest	*0,5031*	0,3763	0,3937
SGDClassifier	0,4821	*0,4087*	0,3588

Tabla 6.7: Resultados macro de clasificación con métricas

Modelo	Precisión	Exhaustividad	f1
NLP-CIC	0,8815	0,8815	0,8815
baseline-svm	0,8285	0,8285	0,8285
GradientBoosting	0,7779	0,7779	0,7779
RandomForest	0,7751	0,7751	0,7751
SGDClassifier	0,6803	0,6803	0,6803

Tabla 6.8: Resultados micro de clasificación con métricas

6.3.2. Clasificación multietiqueta basada en palabras

En la clasificación basada en palabras se han entrenado modelos de aprendizaje automático y de aprendizaje profundo. Estos modelo reciben como entrada una representación de los textos como puede ser TF-IDF o vectores preentrenados. En cada modelo se empleó distintas versiones del conjunto de datos con distintas cantidad de información textual, en la sección 6.4 se detallará como afecta cada conjunto de datos en cada modelo.

Inicialmente en las pruebas con modelos de aprendizaje automático se selección un conjunto de datos con una cantidad de información textual reducida para generar una representación vectorial

con baja dimensionalidad. Con una dimensionalidad reducida se buscaba acelerar la obtención de resultados con el objetivo de encontrar mejor modelo y tomarlo como referencia para realizar pruebas con otras versiones del conjunto de datos con mayor dimensionalidad.

A continuación, se verán todos los modelos entrenados, se indicará las características de los datos de entrada y los hiperparámetros que resultan de las búsquedas exhaustivas o aleatorias.

6.3.2.1. RandomForest

Se emplea un texto con palabras truncadas y sin palabras vacías. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. Los hiperparámetros aplicados son:

- criterion : gini
- max_features: sqrt
- n_estimators: 2000

El ajuste del número de estimadores se vio limitado por la cantidad de memoria consumida por el algoritmo, al aumentar el número de estimador el consumo se disparaba hasta consumir la memoria por completo y producir errores en las ejecución.

Los resultados se muestran en la tabla 6.9. Se aprecia un sobreajuste elevado en todas las etiquetas y especialmente elevado en la etiqueta "OFG". A pesar del sobreajuste, se supera a los resultados obtenidos por el mismo modelo entrenado mediante métricas del lenguaje (visto en la sección 6.3.1.1)

	Entrenamiento				**Prueba**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9991	0,9983	0,9987	NO	0,8437	0,9824	0,9078
NOM	0,9855	0,9935	0,9895	NOM	0,7582	0,6207	0,6826
OFG	**0,9952**	**0,9811**	**0,9881**	OFG	**0,00**	**0,00**	**0,00**
OFP	0,9902	0,9921	0,9912	OFP	0,7787	0,4018	0,5301
macro	0,9925	0,9912	0,9919	macro	0,5951	0,5012	0,5301
micro	0,9970	0,9970	0,9970	micro	0,8304	0,8304	0,8304

Tabla 6.9: Resultado RandomForest sobre palabras

6.3.2.2. GrandientBoosting

Se emplea un texto con palabras truncadas y sin palabras vacías. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. Los hiperparámetros aplicados son:

- criterion: squared_error
- learning_rate: 0,1

- max_depth: 16
- max_features: sqrt
- n_estimators: 500

Los resultados obtenidos se muestran en la tabla 6.10. Este modelo presenta también sobreajuste pero, a diferencia del modelo RandomForest anterior, muestra algún resultado en la etiqueta "OFG" aunque con valores muy bajos.

	Entrenamiento				**Prueba**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9989	0,9984	0,9986	NO	0,8600	0,9708	0,9120
NOM	0,9887	0,9919	0,9902	NOM	0,7471	0,6121	0,6729
OFG	**0,9859**	**0,9905**	**0,9882**	OFG	**0,1372**	**0,0331**	**0,0534**
OFP	0,9902	0,9912	0,9907	OFP	0,7222	0,4645	0,5654
macro	0,9909	0,9930	0,9919	macro	0,6166	0,5201	0,5509
micro	0,9969	0,9969	0,9969	micro	0,8325	0,8325	0,8325

Tabla 6.10: Resultado GradientBoosting sobre palabras

6.3.2.3. AdaBoost

Se emplea un texto con palabras truncadas y sin palabras vacías. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. Los hiperparámetros aplicados son:

- base_estimator: RandomForestClassifier con max_depth=3
- learning_rate: 0,01
- n_estimators: 200

Los resultados entrenamiento y pruebas son similares (tabla 6.11), incluso superiores en las etiquetas "NOM" y "OFP". El valor alcanzado en "OFG" sigue siendo bajo, tan solo un 16.60 %, aunque supone una mejora de aproximadamente 10 puntos respecto al modelo anterior.

	Entrenamiento				**Prueba**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9598	0,7468	0,8400	NO	0,9130	0,7441	0,8199
NOM	0,6046	0,7651	0,6754	NOM	0,6668	0,7335	0,6986
OFG	0,2337	0,7971	**0,3614**	OFG	0,1130	0,3127	**0,1660**
OFP	0,3415	0,6899	0,4568	OFP	0,4004	0,6191	0,4863
macro	0,5349	0,7497	0,5834	macro	0,5233	0,6024	0,5427
micro	0,7418	0,7418	0,7418	micro	0,7149	0,7149	0,7149

Tabla 6.11: Resultado AdaBoost sobre palabras

6.3.2.4. Support Vector Machine

Se emplea un texto con palabras truncadas y sin palabras vacías. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. Los hiperparámetros aplicados son:

- C: 0,3
- gamma: scale
- kernel: sigmoid

Este modelo fue sometido a 3 búsquedas de parámetros, afinándolos más en cada una de ellas.

Las métricas obtenidas en prueba son cercanas a las de entrenamiento (tabla 6.12), lo que indica que no sufre sobreajuste excepto en la etiqueta "OFG" como también ocurre en los anteriores modelos.

	Entrenamiento				**Prueba**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9624	0,8821	0,9205	NO	0,9190	0,8867	0,9025
NOM	0,6539	0,8048	0,7215	NOM	0,6959	0,7357	0,7152
OFG	0,2047	0,9292	**0,3356**	OFG	0,0982	0,4075	**0,1583**
OFP	0,6355	0,6562	0,6457	OFP	0,6663	0,5504	0,6028
macro	0,6141	0,818	0,6558	macro	0,5948	0,6451	0,5947
micro	0,8493	0,8493	0,8493	micro	0,8060	0,8060	0,8060

Tabla 6.12: Resultado SVC sobre palabras

6.3.2.5. Support Vector Machine con sobremuestreo

En este modelo se aplica sobremuestreo en las etiquetas ofensivas "OFG" y "OFP". Se realizaron pruebas con distintos valores de sobremuestreo (3000, 7000 y 14 000) tanto para todas las etiquetas como para solo las ofensiva. Se comprobó que el mejor resultado se obtenía con sobremuestreo en las etiquetas ofensivas, incrementando el número de instancias hasta las 7000.

En el entrenamiento se emplea un texto con palabras truncadas y sin palabras vacías. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. Los hiperparámetros aplicados son:

- C: 0.3
- gamma: scale
- kernel: sigmoid

En tabla6.13 se puede aprecia que al aplicar sobremuestreo las etiquetas "OFG" y "OFP" no mejoran respecto al modelo sin sobremuestreo y que además la métrica f1 empeora.

	Entrenamiento			Pruebas		
	Precisión	Exhaustividad	f1	Precisión	Exhaustividad	f1
NO	0.9154	0.8810	0.8979	0.9218	0.8787	0.8997
NOM	0.4573	0.8340	**0.5907**	0.6709	0.7528	**0.7095**
OFG	0.8769	0.9488	**0.9114**	0.0998	0.3554	**0.1559**
OFP	0.82255	0.6940	0.7528	0.6292	0.5590	0.5920
macro	0.7680	0.8394	0.7882	0.5804	0.6365	0.5893
micro	0.8496	0.8496	0.8496	0.8028	0.8028	0.8028

Tabla 6.13: Resultado SVM sobre palabras con sobremuestreo

6.3.2.6. Stochastic Gradient Descent

Se emplea un texto con palabras truncadas y sin palabras vacías. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. Los hiperparámetros aplicados son:

- alpha: 0.0001
- epsilon: 0.001
- loss: hinge
- penalty: l1
- tol: 0.01

Los resultados de este modelo (tabla 6.14) son similares a los obtenidos por el modelo SVM sin sobremuestreo (sección 6.3.2.4), presentan los mismo problemas en la etiqueta "OFG".

	Entrenamiento			Pruebas		
	Precisión	Exhaustividad	f1	Precisión	Exhaustividad	f1
NO	0.9502	0.9256	0.9378	0.9012	0.9260	0.9134
NOM	0.6429	0.7959	0.7112	0.6816	0.7492	0.7138
OFG	0.3141	0.9245	**0.4689**	0.1114	0.3127	**0.1643**
OFP	0.7042	0.5792	0.6356	0.7221	0.4787	0.5757
macro	0.6528	0.8063	0.6884	0.6041	0.6167	0.5918
micro	0.8735	0.8735	0.8735	0.8216	0.8216	0.8216

Tabla 6.14: Resultado SGD sobre palabras

6.3.2.7. Stochastic Gradient Descent con sobremuestreo

En este modelo se aplica sobremuestreo en las etiquetas ofensivas "OFG" y "OFP". Se realizaron pruebas con distintos valores de sobremuestreo (3000, 7000 y 14 000) tanto para todas las etiquetas como para solo las ofensiva. Se comprobó que el mejor resultado se obtenía con sobremuestreo en las etiquetas ofensivas, incrementando el número de instancias hasta las 7000.

En el entrenamiento se emplea un texto con palabras truncadas y sin palabras vacías. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. Los hiperparámetros aplicados son:

- alpha: 0,0001
- epsilon: 0,001
- loss: hinge
- penalty: l1
- tol: 0,01

Los resultados se muestran en la tabla 6.15. El valor macro-f1 aumenta 2,55 puntos respecto al modelo sin sobremuestreo, pasando del 59,18 % al 61,73 %. El incremento de muestras en la etiqueta "OFG" produce un fuerte sobreajuste en dicha etiqueta pero a pesar de ello obtiene un valor de macro-f1 más alto que el modelo sin sobremuestreo.

	Entrenamiento				**Pruebas**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9011	0,9137	0,9074	NO	0,9144	0,9206	0,9175
NOM	0,5330	0,7846	0,6347	NOM	0,6884	0,7135	0,7008
OFG	0,9516	0,9935	**0,9721**	OFG	0,1703	0,3270	**0,2240**
OFP	0,8652	0,7322	0,7932	OFP	0,6750	0,5852	0,6269
macro	0,8127	0,8560	0,8269	macro	0,6121	0,6366	0,6173
micro	0,8831	0,8831	0,8831	micro	0,8325	0,8325	0,8325

Tabla 6.15: Resultado SGD sobre palabras con sobremuestreo

6.3.2.8. Stochastic Gradient Descent con ajuste fino

El modelo se entrena con sobremuestro en las etiqueta "OFG" y "OFP", incrementando las muestras de cada una hasta las 7000.

Se emplea un texto sin truncar y sin palabras vacías excepto pronombres personales. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. Los hiperparámetros aplicados son:

- alpha: 0,0001
- epsilon: 0,001
- loss: hinge
- penalty: l1

- tol: 0,01

En la tabla 6.16 puede verse el valor de la métrica macro f1 de este modelo es 3,96 puntos superior al del modelo SGD anterior. Esta mejora se debe al incremento de las métricas de las etiquetas "OFG" y "OFP".

	Entrenamiento				**Pruebas**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9019	0,9206	0,9111	NO	0,9103	0,9264	0,9183
NOM	0,5382	0,7805	0,6371	NOM	0,7078	0,7042	0,7060
OFG	0,9624	0,9934	0,9777	OFG	0,2821	0,4265	**0,3396**
OFP	0,8864	0,7530	0,8143	OFP	0,7074	0,6251	0,6637
macro	0,8222	0,8618	0,8350	macro	0,6519	0,6706	**0,6569**
micro	0,8912	0,8912	0,8912	micro	0,8441	0,8441	0,8441

Tabla 6.16: Resultado SGD sobre palabras con ajuste fino

6.3.2.9. BI-LSTM 1 capa

Se emplea el texto sin truncar y sin eliminar palabras. La palabras son transformadas a vectores FastText aplicando previamente corrección automática de palabras. Se emplea un optimizador RMSprop con ratio de aprendizaje de 0.0003. A la capa LSTM se le aplica un ratio de abandono (dropout) del 10 % . La estructura de la red se muestra en al figura 6.1.

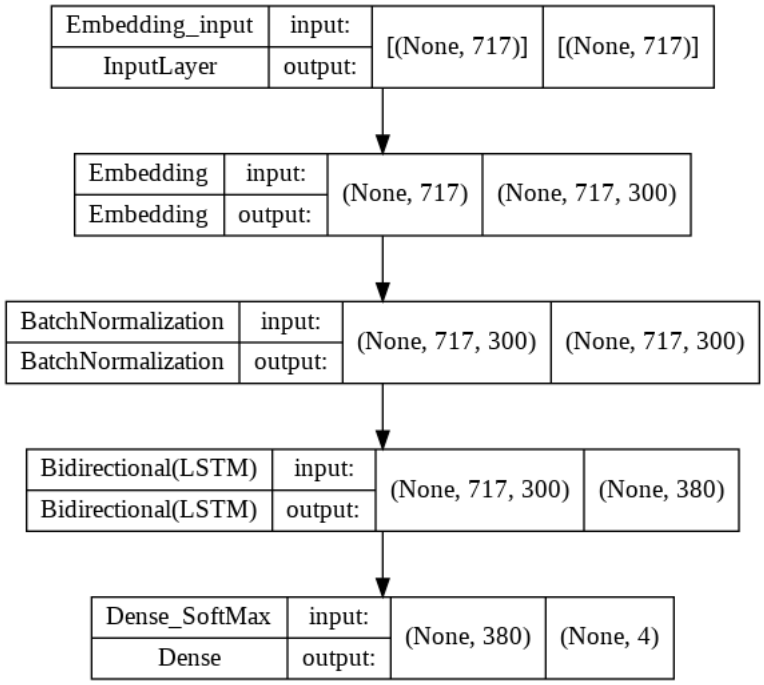

Figura 6.1: Red BI-LSTM 1 capa

En la tabla 6.17 se muestra que el valor de f1 superar en 1,48 puntos al modelo SGD visto en

la sección 6.3.2.8. Las diferencias entre los valores de entrenamiento y prueba denotan que existe sobreajuste en el modelo, focalizado en las etiquetas ofensivas.

	Entrenamiento		
	Precisión	Exhaustividad	f1
NO	0.9995	0.8952	0.9445
NOM	0.7796	0.9967	0.8749
OFG	0.6235	1.00	0.7681
OFP	0.6869	0.9907	0.8113
macro	0.7724	0.9706	0.8497
micro	0.9157	0.9157	0.9157

	Pruebas		
	Precisión	Exhaustividad	f1
NO	0.9428	0.8801	0.9104
NOM	0.6720	0.7785	0.7213
OFG	0.3023	0.4786	**0.3706**
OFP	0.6460	0.7282	**0.6846**
macro	0.6408	0.7164	**0.6717**
micro	0.8374	0.8374	0.8374

Tabla 6.17: Resultado BI-LSTM 1 capa

6.3.2.10. BI-LSTM 2 capas

Se emplea el texto sin truncar y sin eliminar palabras. La palabras son transformadas a vectores FastText aplicando previamente corrección automática de palabras. Se emplea un optimizador RMSprop con ratio de aprendizaje de 0,002. A la capa segunda capa LSTM se le aplica un ratio de abandono del 10 %. La estructura de la red se muestra en al figura 6.2.

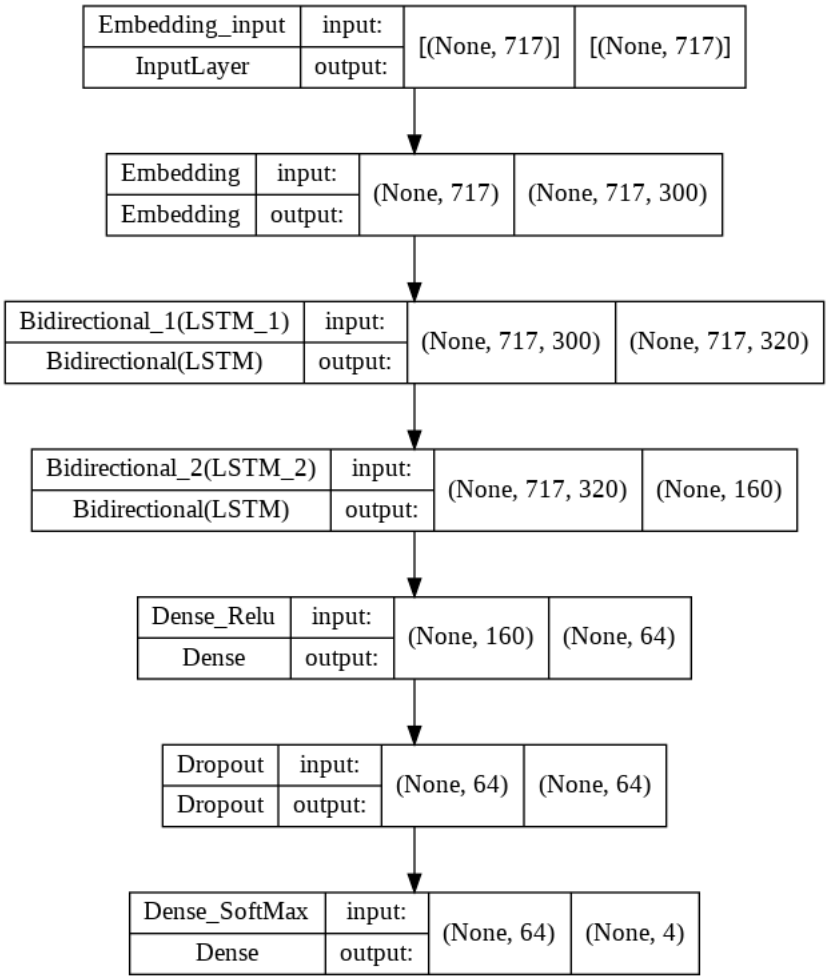

Figura 6.2: BI-LSTM 2 capas

Comparando los resultados de entrenamiento y prueba mostrados en la tabla 6.18 se comprueba que este modelo tiene menor sobreajuste que el modelo Bi-LSTM de una sola capa. Además hay una leve mejora de las métricas, tanto en las de la etiqueta "OFG" como en la macro f1.

	Entrenamiento		
	Precisión	Exhaustividad	f1
NO	0,9902	0,8277	0,9017
NOM	0,6493	0,9433	0,7692
OFG	0,3840	1,00	0,5549
OFP	0,5557	0,8995	0,6870
macro	0,6448	0,9176	**0,7282**
micro	0,8472	0,8472	0,8472

	Pruebas		
	Precisión	Exhaustividad	f1
NO	0,9596	0,8402	0,8960
NOM	0,6517	0,8207	0,7265
OFG	0,2934	0,6398	**0,4023**
OFP	0,6133	0,7681	0,6820
macro	0,6295	0,7672	**0,6767**
micro	0,8227	0,8227	0,8227

Tabla 6.18: Resultado BI-LSTM 2 capas

6.3.2.11. CNN 1 capa

Se emplea el texto sin truncar y sin eliminar palabras. La palabras son transformadas a vectores FastText aplicando previamente corrección automática de palabras. Se emplea un optimizador RMSprop con ratio de aprendizaje de 0,003. La capa convolucional se configura con un *kernel* de tamaño tres. La estructura de la red se muestra en al figura 6.3.

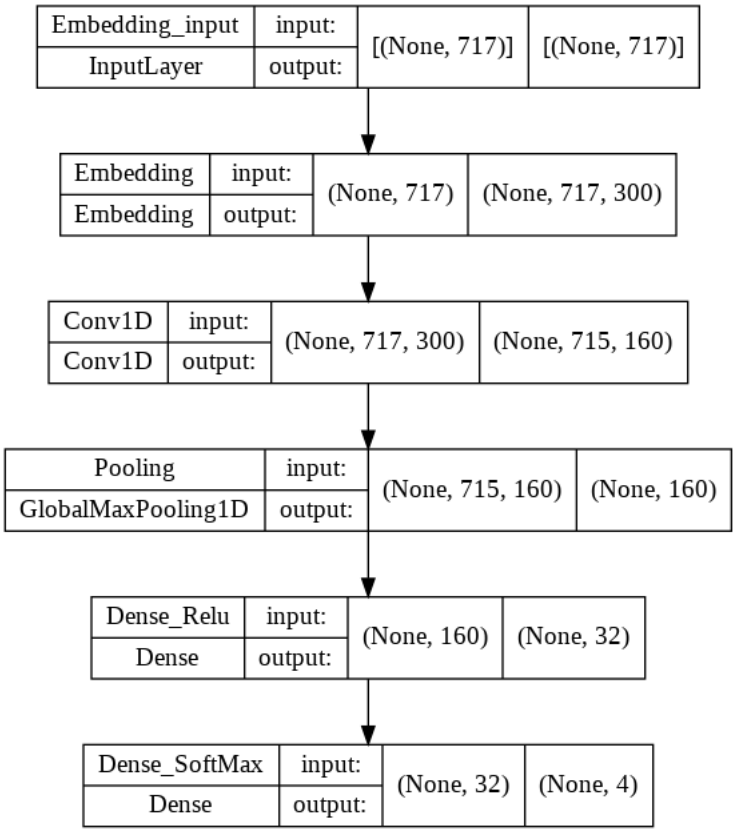

Figura 6.3: CNN 1 capa

Los resultados de la red CNN de una sola capa (tabla 6.19) mejora el resultado del modelo Bi-LSTM de 2 capas. Se pasando de un macro-f1 de 67,67 % a 68,13 % pero empeora los resultados de las etiqueta ofensivas, donde presenta problemas de sobreajuste.

	Entrenamiento		
	Precisión	Exhaustividad	f1
NO	0,9859	0,8684	0,9234
NOM	0,7336	0,8275	0,7777
OFG	0,5734	0,9764	0,7225
OFP	0,5607	0,9073	0,6931
macro	0,7134	0,8949	0,7792
micro	0,8715	0,8715	0,8715

	Pruebas		
	Precisión	Exhaustividad	f1
NO	0,9428	0,8832	0,9120
NOM	0,7377	0,7335	0,7356
OFG	0,3587	0,4454	**0,3974**
OFP	0,6136	0,7629	**0,6801**
macro	0,6632	0,7063	**0,6813**
micro	0,8403	0,8403	0,8403

Tabla 6.19: Resultado CNN 1 capa

6.3.2.12. CNN 3 capas

Se emplea el texto sin truncar y sin eliminar palabras. La palabras son transformadas a vectores FastText aplicando previamente corrección automática de palabras. Se emplea un optimizador RMS-prop con ratio de aprendizaje de 0,003. Las tres capa convolucionales se configuran con un tamaño de *kernel* de cuatro. La estructura de la red se muestra en al figura 6.4.

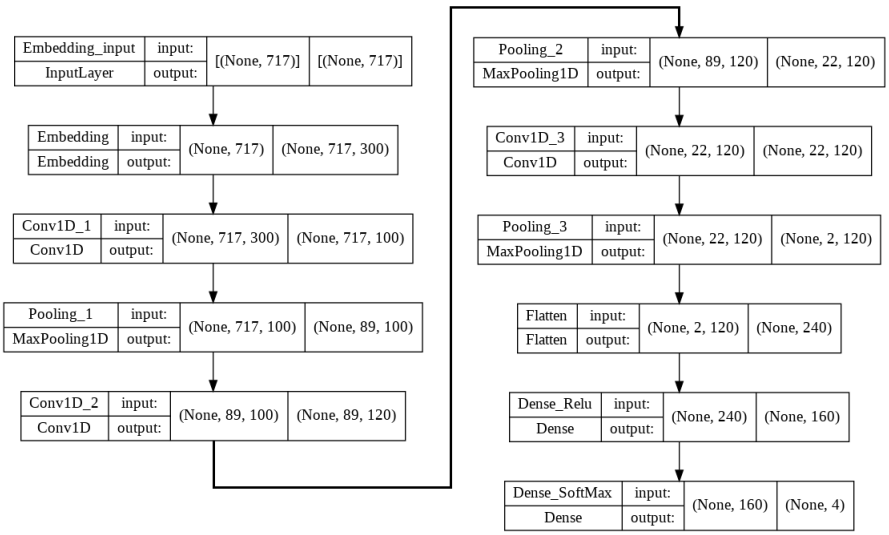

Figura 6.4: CNN 3 capas

La tabla 6.20 muestra los resultados del modelo. Los resultados obtenidos son inferiores al del resto de redes neuronales vistas hasta ahora. Al comparar los resultados de entrenamiento y prueba se aprecia que este empeoramiento se debe a que el modelo está sobreajustado.

Entrenamiento

	precision	recall	f1-score
NO	0.9924	0.9424	0.9668
NOM	0.7923	0.9951	0.8822
OFG	0.4116	1.00	0.5832
OFP	0.9160	0.9366	0.9262
macro	0.7781	0.9685	**0.8396**
micro	0.9463	0.9463	0.9463

Pruebas

	precision	recall	f1-score
NO	0.9347	0.8647	0.8984
NOM	0.6940	0.7728	0.7313
OFG	0.2574	0.3696	**0.3035**
OFP	0.6022	0.7243	**0.6577**
macro	0.6221	0.6829	**0.6477**
micro	0.8235	0.8235	0.8235

Tabla 6.20: Resultado CNN 3 capas

6.3.2.13. BETO

Se entrena versión del modelo BETO con mayúsculas y acentos, identificada con el nombre "*dccuchile/bert-base-spanish-wwm-cased*". Se emplea el texto sin truncar y sin eliminar palabras, este es preprocesado mediante la clase Tokenizer de BETO. El modelo se ajusta en un solo epoch para evitar sobreajuste.

El modelo preentrenado BETO ofrece resultados muy similares en entrenamiento y pruebas (tabla 6.21) lo que indica que apenas sufre sobreajuste. Las métricas de las etiquetas "OFG" y "OFP" mejoran considerablemente respecto a las obtenidas en los modelos aprendizaje automático o en redes neuronales visto hasta ahora.

Entrenamiento

	Precisión	Exhaustividad	f1
NO	0,9666	0,9482	0,9573
NOM	0,7307	0,8064	0,7667
OFG	0,5930	0,6462	0,6185
OFP	0,7857	0,8259	0,8053
macro	0,7690	0,8067	0,7869
micro	0,9189	0,9189	0,9189

Pruebas

	Precisión	Exhaustividad	f1
NO	0,9229	0,9565	0,9394
NOM	0,7766	0,7550	0,7656
OFG	0,5673	0,5592	**0,5632**
OFP	0,8068	0,6994	**0,7493**
macro	0,7684	0,7425	**0,7544**
micro	0,8855	0,8855	0,8855

Tabla 6.21: Resultado BETO

6.3.2.14. RoBERTuito

Se entrena versión del modelo BETO con mayúsculas y acentos, identificada con el nombre "*pysentimiento/robertuito-base-cased*". Se emplea el texto sin truncar y sin eliminar palabras, este es preprocesado mediante la clase Tokenizer de RoBERTuito. El modelo se ajusta en un solo *epoch* para evitar sobreajuste.

En los resultados obtenidos en la tabla 6.22 se muestra que el valor macro f1 de este modelo es del 80,11 %, es decir, 5 puntos por encima del modelo BETO. Esta mejora se debe principalmente al aumento de rendimiento en la etiqueta "OFG". que obtiene un valor de f1 de 67,21 % frente al 56,32 % del modelo anterior.

Entrenamiento

	Precisión	Exhaustividad	f1
NO	0,9695	0,9502	0,9598
NOM	0,7593	0,8097	0,7836
OFG	0,5221	0,7783	0,6250
OFP	0,8186	0,8493	0,8336
macro	0,7674	0,8469	0,8005
micro	0,9253	0,9253	0,9253

Pruebas

	Precisión	Exhaustividad	f1
NO	0,9320	0,9598	0,9457
NOM	0,8135	0,7700	0,7911
OFG	0,5920	0,7772	**0,6721**
OFP	0,8497	0,7479	0,7956
macro	0,7968	0,8137	**0,8011**
micro	0,9011	0,9011	0,9011

Tabla 6.22: Resultado RoBERTuito

6.3.2.15. Resumen de resultados

En las tablas 6.23 y 6.24 se muestras las métricas macro y micro obtenidas en la clasificación multietiqueta basado en textos. A estas tablas se han añadido los resultados del ganador de la tarea MeOffendES y también del baseline de esa tarea. En las tablas de resultado macro puede verse que

todos los modelos han superado el baseline y que los modelos basados en BERT superan al ganador de la tarea

Modelo	Precisión	Exhaustividad	f1
RoBERTuito	**0.7968**	**0.8137**	**0.8011**
BETO	0.7684	0.7425	0.7544
NLP-CIC	0.7679	0.7093	**0.7324**
CNN	0.6632	0.7063	0.6813
Bi-LSTM 2 capas	0.6295	0.7672	0.6767
Bi-LSTM	0.6408	0.7164	0.6717
SGD ajuste fino	0.6519	0.6706	**0.6569**
CNN 3 capas	0.6221	0.6829	0.6477
SGD sobremuestreo	0.6121	0.6366	0.6173
SVM	0.5948	0.6451	0.5947
SGD	0.6041	0.6167	0.5918
SVM sobremuestreo	0.5804	0.6365	0.5893
GradientBoosting	0.6166	0.5201	0.5509
AdaBoost	0.5233	0.6024	0.5427
RandomForest	0.5951	0.5012	0.5301
baseline-svm	0.6278	0.4831	**0.5236**

Tabla 6.23: Resumen de resultados macro

6.3.3. Clasificación binaria

Para realizar la clasificación binaria los comentarios no ofensivo y con lenguaje soez se han agrupado en la etiqueta, "NO". Por otro lado, los comentarios ofensivos a grupo y ofensivo hacia una persona se han agrupado bajo la etiqueta "OF".

Esta clasificación se ha realizado con 4 modelos distintos

- Modelo GradientBoosting basado en métrica: es el modelo que mejor resultado obtuvo en clasificación multietiqueta con métricas
- Modelo SGDClassifier basado en palabras: es el mejor modelo de aprendizaje automático sobre palabras
- Modelo CNN 1 capa: mejor modelo profundo no preentrenado
- Modelo RoBERTuito: mejor modelo basado en BERT
- Modelo SGDClassifier combinando métricas y palabras: al mejor modelo de aprendizaje se le añade información adicional en forma de métricas del lenguaje.

En la tarea MeOffendES 2021 no se realizó clasificación binaria con el conjunto de datos que aquí se utiliza, sin embargo, en el artículo del corpus OffendES (del Arco et al. (2021)) están disponibles los

Modelo	Precisión	Exhaustividad	f1
RoBERTuito	0.9011	0.9011	**0.9011**
BETO	0.8855	0.8855	0.8855
NLP-CIC	0.8815	0.8815	**0.8815**
SGD ajuste fino	0.8441	0.8441	**0.8441**
CNN	0.8403	0.8403	0.8403
Bi-LSTM	0.8374	0.8374	0.8374
SGD sobremuestreo	0.8385	0.8325	0.8346
baseline-svm	0.8285	0.8285	**0.8285**
Bi-LSTM 2 capas	0.8227	0.8227	0.8227
CNN 3 capas	0.8235	0.8235	0.8235
SGD	0.8356	0.8216	0.8233
SVM	0.8399	0.8060	0.8203
SVM sobremuestreo	0.8330	0.8028	0.8158
GradientBoosting	0.8135	0.8325	0.8147
RandomForest	0.8106	0.8304	0.8058
AdaBoost	0.7873	0.7149	0.7401

Tabla 6.24: Resumen de resultado micro

resultados de una clasificación binaria en la que ofrecen las métricas macro y ponderadas. Puesto que solo se dispone de esta referencia en esta sección se ofrecerán los resultados en esos dos conjuntos de métricas en lugar de macro y micro como se ha ido haciendo hasta ahora.

6.3.3.1. GradientBoosting basado en métricas

El modelo se entrena con los hiperparámetros:

- criterion : friedman_mse
- max_features: sqrt
- max_depth: 3
- n_estimators: 500
- learning_rate: 0,1

La exactitud en entrenamiento es del 91,87 % y en pruebas del 85,55 %. Los resultados por etiqueta y agregados se muestran en la tabla 6.25.A diferencia a lo que se vio en el modelo GradientBoosting multietiqueta (sección 6.3.1.2) aquí no se aprecia sobreajuste. Existe una diferencia significativa entre los resultados de las métricas de la etiqueta ofensiva y no ofensiva.

	Entrenamiento				Prueba		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9308	0,9787	0,9787	NO	0,8736	0,9613	0,9154
OF	0,7980	0,5360	0,5360	OF	0,7018	0,3956	0,5060
macro	0,8644	0.7573	0.7977	macro	0,7877	0,6785	0,7107
ponderado	0,9128	0,9187	0,9118	ponderado	0,8415	0,8555	0,8388

Tabla 6.25: Resultado GradientBoosting binario basado en métricas

6.3.3.2. SGDClassfier basado en palabras

En este modelo se probó el sobremuestreo en la etiqueta "OF" pero obtuvo peor resultado que el modelo sin sobremuestreo que se presenta aquí.

Se emplea un texto sin truncar y sin palabras vacías excepto pronombres personales. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. Los hiperparámetros aplicados son:

- alpha: 0,0001
- epsilon: 0,001
- loss: log
- penalty: l2
- tol: 0,01

La exactitud en entrenamiento es del 90,43 % y en pruebas del 86,32 %. Los resultados por etiqueta y agregados se muestran en la tabla 6.26. Los valores obtenidos en la etiqueta "OF" son similares a los que obtenidos por el modelo en su versión multietiqueta (sección 6.3.2.8)

	Entrenamiento				Pruebas		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9789	0,9088	0,9426	NO	0,9357	0,8931	0,9139
OF	0,6006	0,6006	0,7124	OF	0,6120	0,7334	**0,6672**
macro	0,7898	0,8921	0,8275	macro	0,7739	0,8132	**0,7906**
ponderado	0,9277	0,9043	0,9114	ponderado	0,8752	0,8632	0,8678

Tabla 6.26: Resultado SGD binario

6.3.3.3. SGDClassfier combinado métricas y palabras

Se emplea un texto sin truncar y sin palabras vacías excepto pronombres personales. El texto es vectorizado mediante TF-IDF con frecuencia mínima de palabra de 3 apariciones. A los datos vectorizados se añaden las métricas del lenguaje vistas en la sección6.3.1. Los hiperparámetros aplicados son:

- alpha: 0,0001
- epsilon: 0,001
- loss: log_loss
- penalty: l2
- tol: 0,0001

La exactitud en entrenamiento es del 90,66 % y en pruebas del 86,52 %. Los resultados por etiqueta y agregados se muestran en la tabla 6.27. Se aprecia una leve mejora en la etiqueta "OF", en este modelo es 0,62 puntos superior.

	Entrenamiento				**Pruebas**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9789	0,9116	0,9440	NO	0,9379	0,8934	0,9151
OF	0,6079	0,8749	0,7173	OF	0,6158	0,7429	**0,6734**
macro	0,7934	0,8932	0,8307	macro	0,7768	0,8181	**0,7942**
ponderado	0,9287	0,9066	0,9133	ponderado	0,8652	0,8652	0,8652

Tabla 6.27: Resultado SGD binario combinado

6.3.3.4. CNN 1 capa

Se emplea el texto sin truncar y sin eliminar palabras. La palabras son transformadas a vectores FastText aplicando previamente corrección automática de palabras. Se emplea un optimizador RMS-prop con ratio de aprendizaje de 0,003. La capa convolucional se configura con un *kernel* de tamaño tres. La estructura de la red se muestra en al figura 6.5.

La exactitud en entrenamiento es del 93,49 % y en pruebas del 88,14 %. Los resultados por etiqueta y agregados se muestran en la tabla 6.28. La métrica f1 de la etiqueta "OF" se sitúa en el 71,39 %, en los modelos anteriores el valor más alto alcanzado fue del 67,34 %.

	Entrenamiento				**Pruebas**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9970	0,9274	0,9610	NO	0,9495	0,9021	0,9252
OF	0,6797	0,9827	0,8036	OF	0,6502	0,7913	**0,7139**
macro	0,8384	0,9551	0,8823	macro	0,7998	0,8467	0,8195
ponderado	0,9541	0,93494	0,9397	ponderado	0,8935	0,8814	0,8857

Tabla 6.28: Resultado CNN binario

6.3.3.5. RoBERTuito basado en palabras

Se entrena versión del modelo BETO con mayúsculas y acentos, identificada con el nombre "*pysentimiento/robertuito-base-cased*". Se emplea el texto sin truncar, sin eliminar palabras y pre-

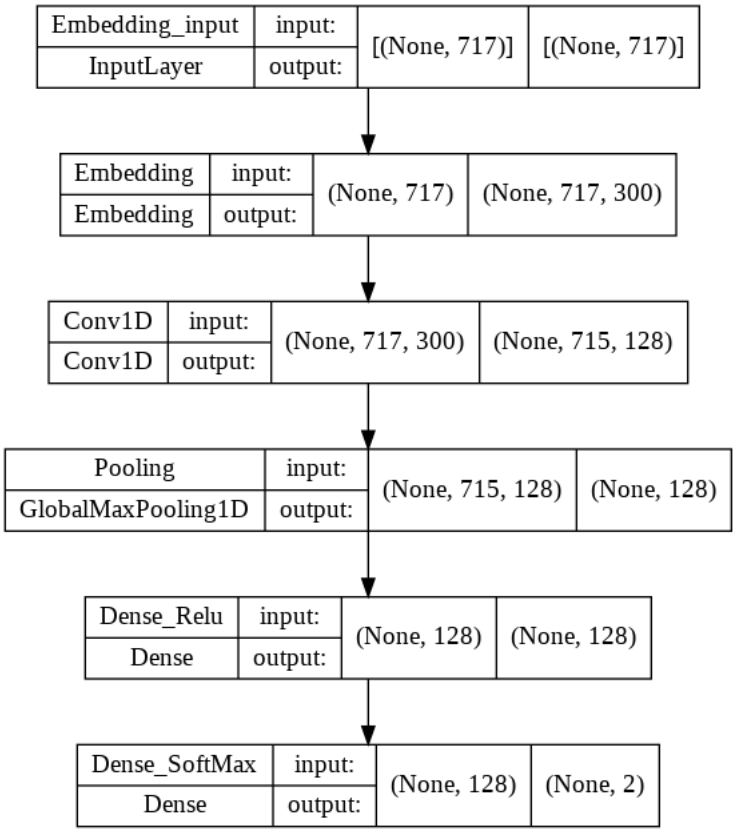

Figura 6.5: CNN Binario

procesado mediante la clase Tokenizer de RoBERTuito. El modelo se ajusta en un solo *epoch* para evitar sobreajuste.

La exactitud en entrenamiento es del 95,89 % y en pruebas del 92,90 %. Los resultados por etiqueta y agregados se muestran en la tabla 6.29. El valor de f1 tiene un incremento de 8,24 puntos respecto al modelo anterior, la exhaustividad baja 5 puntos pero la precisión aumenta 20.

	Entrenamiento				**Pruebas**		
	Precisión	Exhaustividad	f1		Precisión	Exhaustividad	f1
NO	0,9769	0,9755	0,9762	NO	0,9425	0,9720	0,9570
OF	0,8453	0,8528	0,8490	OF	**0,8591**	**0,7421**	**0,7963**
macro	0,9111	0,9142	0,9126	macro	0,9008	0,8570	0,8767
ponderado	0,9591	0,9589	0,9590	ponderado	0,9269	0,9290	0,9270

Tabla 6.29: Resultado RoBERTuito binario

6.3.3.6. Resumen de resultados

En las tablas 6.30 y 6.31 se muestran las métricas macro y ponderadas obtenidas en la clasificación binaria basado en textos. En la tabla se añaden como referencia los resultados de del Arco et al. (2021). Se aprecia como los resultados mejoran al aplicar modelos más complejos. Los modelos más

simples superan los resultados de referencia, hecho que se analizarán más adelante.

Modelo	Precisión	Exhaustividad	f1
RoBERTuito	0,9008	0,8570	0,8767
CNN	0,7998	0,8467	0,8195
SGDClassifier combinado	0,7768	0,8181	0,7942
SGDClassifier palabras	0,7739	0,8132	0,7906
OffendES	0,7042	0,7674	**0,7839**
GradientBoosting métricas	0,7877	0,6785	0,7107

Tabla 6.30: Resultados macro clasificación binaria

Modelo	Precisión	Exhaustividad	f1
RoBERTuito	0,9269	0,9290	0,9270
CNN	0,8935	0,8814	0,8857
SGDClassifier palabras	0,8752	0,8632	0,8678
SGDClassifier combinado	0,7768	0,8181	0,7942
GradientBoosting métricas	0,8415	0,8555	0,8388
OffendES	0,7042	0,7674	**0,7839**

Tabla 6.31: Resultados ponderado clasificación binaria

6.4. Experimentos sobre información textual

Estos experimentos tienen como objetivo encontrar, para cada tipo de modelo, la información textual que maximiza la métrica f1 . Los conjuntos de datos que se van a estudiar son los presentados en la sección 5.3, también se va proba el texto sin procesar y el texto procesado como se explicó en la sección 5.1.

6.4.1. Modelos de aprendizaje automático

Se toma como modelo para realizar las pruebas el modelo SGDClassifier. Se toma este modelo por ser el que mejor resultado obtuvo en los experimentos de modelos de clasificación basado en palabras (sección 6.3.2) y por tener un tiempo de ajuste corto.

En la tabla 6.32 se muestran los resultados obtenidos en los nueve conjuntos de datos con distinta información textual que se detallaron en la sección5.3. Los 3 primeros puestos los ocupan los conjuntos con mayor cantidad de información, sus resultados difieren en 0,09 puntos, mientras que con el cuarto existen 0,87 de diferencia.

Tipo de información	Descripción	Precisión	Exhaustividad	f1
comment_atp	Sin palabras vacías y con pronombres	0,6363	**0,6484**	**0,6271**
comment_c	Procesado inicial	**0,6407**	0,6458	0,6269
comment	Texto sin procesar	0,6387	0,6419	0,6262
comment_at	Sin palabras vacías	0,6264	0,6356	0,6175
comment_cat	Truncado solo con palabras léxicas	0,6222	0,6224	0,6081
comment_ls	Lematizado y truncado con pronombres	0,6133	0,6165	0,6025
comment_sp	Truncado con pronombres personales	0,6069	0,6107	0,5968
comment_l	Lematizado con pronombres	0,6088	0,6007	0,5930
comment_cs	Truncado solo con palabras léxicas	0,5992	0,5932	0,5868

Tabla 6.32: Resultados texto en modelos de aprendizaje automático

6.4.2. Redes neuronales

Para realizar las pruebas se toma como base un modelo Bi-LSTM de una sola capa sin ajustar hiperparámetros. Aquí se deben emplear los conjuntos de datos que no tienen palabras truncadas puesto que el método para asignar vectores espera palabras completas. Antes de asignar vectores preentrenados se realizan pruebas calculando los vectores durante el entrenamiento y se toma el conjunto de datos que obtiene mejor resultado para cargar vectores FastText preentrenados. Con estos vectores se entrenan dos modelos: el primero sin corregir palabras y el segundo corrigiendo palabras para aumentar la cobertura de vectores.

Los resultados de las pruebas se muestran en la tabla 6.33. En ella se aprecia como aplicar FastText mejora en 1,24 puntos los resultados respecto a realizar el cálculo de los vectores durante el entrenamiento. Además, utilizar la corrección automática palabras supone otros 1,69 puntos de mejora.

Preprocesado	Precisión	Exhaustividad	f1
FastText con correcciones	0,617	0,7018	**0,6509**
FastText sin correcciones	0,6024	0,7383	0,6340
comment_c	0,6165	0,6544	0,6216
comment_atp	0,5992	0,6355	0,6071
comment	0,5781	0,5781	0,5915

Tabla 6.33: Resultados texto en modelos de redes neuronales

6.4.3. Transformers

Dentro de los modelos BETO y RoBERTuito existen variaciones, BETO tiene disponible un modelo con mayúsculas y otro sin ellas pero ambos incluyen acentos. Por otro lado, RoBERTuito tiene disponible una tercera variante que no tiene mayúsculas ni acentos. Para cada preprocesado se aplicó la variante que mejor se adaptaba al preprocesado empleado.

En las tablas 6.34 y 6.35 se muestran los resultados junto con el modelo aplicado. En ambos modelos se obtienen mejores resultados a emplear los textos con el procesado inicial, que mantiene gran cantidad de información lingüística.

Preprocesado	Modelo	Precisión	Exhaustividad	f1
comment_c	bert-base-spanish-wwm-case	0,8223	0,5910	0,7422
comment	bert-base-spanish-wwm-case	0,8280	0,6847	0,7397
comment_atp	bert-base-spanish-wwm-uncase	0,7838	0,6440	0,6932

Tabla 6.34: Resultados texto en modelos BETO

Preprocesado	Modelo	Precisión	Exhaustividad	f1
comment_c	robertuito-base-case	0,8252	0,7399	0,7776
comment	robertuito-base-case	0,8200	0,7357	0,7730
comment_atp	robertuito-base-deacc	0,8104	0,7093	0,7515

Tabla 6.35: Resultados texto en modelos RoBERTuito

Capítulo 7

Análisis y discusión

En este capítulo se van a analizar los resultados obtenidos en los experimentos. Primero analizarán los resultados de los experimentos de clasificación tanto de clasificación multietiqueta como binaria. Luego se analizarán los resultados obtenidos en los experimentos realizados sobre las distintas formas de preprocesar los textos.

7.1. Clasificación

7.1.1. Clasificación multietiqueta

La experimentación con modelos de clasificación multietiqueta ha abarcado no solo 2 enfoques distintos, basado en métricas y en palabras, si no también varios tipos de modelos, comenzando con modelos de aprendizaje automático clásicos y terminado con los últimos avances en modelos para el procesamiento del lenguaje natural, como son los modelos BERT.

7.1.1.1. Modelos basados en características del lenguaje

Los modelos basados en características del lenguaje han obtenido un rendimiento bastante pobre, muy por debajo de los mismos modelo pero entrenados con los textos. En la tabla 7.1 se compara el mejor de estos modelos, el GradienteBooting, frente al mismo modelo entrenado con los textos. El modelo entrenado con métricas obtiene un f1 de 41,05 % y mientras que el otro obtiene 55,09 %, esto es una mejora de 14,14 puntos. Los resultados se han visto lastrados principalmente por la incapacidad de estos modelos para predecir la etiqueta "OFG" y por su poco rendimiento en la etiqueta "NOM".

Con estos resultados el rendimiento de los modelos basados en métricas del lenguaje queda muy lejos del primer clasificado de la tarea MeOffendES, con un f1 del 73,04 %. Ni siquiera se logra superar el baseline de la tarea con el 52,36 % en su métrica f1.

Estos modelos tienen margen de mejora si se incorporan nuevas métricas que aporten más información sobre somo diferenciar las etiquetas. Sin embargo, calcular nuevas métricas supondría dedicar más esfuerzo en tareas de experimentación y en la preparación de los datos. Considerando lo alejado que se encuentran los resultados obtenidos del baseline de la tareas, este tipo de modelos resultan ser poco viable. Sin embargo, como se vio en los modelos de clasificación binaria, al combinar los textos con características del lenguaje se obtiene un pequeña ganancia de rendimiento.

GradientBoosting con métricas

	Precisión	Exhaustividad	F1
NO	0,8109	0,9647	0,8811
NOM	0,5101	0,1443	**0,2250**
OFG	0,0454	0,0047	0,0085
OFP	0,6268	0,4551	0,5273
macro	0,4983	0,3922	**0,4105**
micro	0,7779	0,7779	0,7779

GradientBoosting con palabras

	Precisión	Exhaustividad	F1
NO	0,8600	0,9708	0,9120
NOM	0,7471	0,6121	**0,6729**
OFG	0,1372	0,0331	0,0534
OFP	0,7222	0,4645	0,5654
macro	0,6166	0,5201	**0,5509**
micro	0,8325	0,8325	0,8325

Tabla 7.1: Comparativa modelos binarios

7.1.1.2. Modelos basados en palabras

En la experimentación de este tipo de modelos se va a distinguir entre modelos de aprendizaje automático y modelos de aprendizaje profundo, haciendo esta distinción en el análisis se podrá comparar la mejora que suponen este último tipo de modelo.

Aprendizaje automático

En los modelos de aprendizaje automático se entrenaron 3 modelos basados en ensamblaje y 2 clasificadores lineales, en la tabla 6.24 se muestran los resultados de estos modelos. Como se ve en esta tabla todos ellos superan el baseline de la tarea y además se aprecia un salto de rendimiento entre los modelos de ensamblaje y los clasificadores lineales. El mejor modelo de ensamblaje (GradienteBoosting), tiene un resultado f1 de 55,09 % mientras que el peor clasificador lineal tiene un f1 de 59,18 %, y el mejor 59,47 % muy cerca del anterior. A pesar de la mejora de rendimiento respecto a los modelos entrenados con métricas, todos los modelos ven penalizados sus resultados por el poco rendimiento en la etiqueta "OFG", el valor más alto de f1 es 33,96 % el modelo SGD con ajuste fino.

Los dos clasificadores lineales superan el resultado obtenido por el cuarto participante de la tarea que obtuvo un f1 del 55,95 % con el modelo BERT (Garcıa and Bedmar (2021)). En el artículo citado no se indica sí se aplica el modelo BERT en inglés directamente sobre los textos en español o sí se tradujeron previamente los textos al inglés antes de aplicar el entrenamiento. No haber realizado la traducción de los textos explica el bajo resultado obtenidos comparados con los otros participantes que usan modelos BERT multilenguaje.

Tanto el clasificador SGD como SVM fueron entrenados con sobremuestreo de las etiquetas ofensiva, esto supuso una mejora en el clasificador SGD que pasó del 59,18 % al 61,73 %, en cambio SVM empeoró 1 punto. Por último, tras encontrar la mejor representación de los textos (vista en la

sección 7.2) y aplicando el sobremuestreo anterior se obtiene una mejor versión del SGD con un valor f1 de **65,69 %.** Con este valor no se supera al tercer clasificado puesto que existe una diferencia de 6,7 puntos, aún así el modelo SGD resulta muy interesante gracias a su rápida velocidad de entrenamiento con la que se puede desarrollar un modelo con buen rendimiento para usarse como baseline.

Aprendizaje profundo

En la categoría de aprendizaje profundo hay dos grupos de modelos diferenciados, por un lado están los modelos entrenados con vectores FastText preentrenados y cuyas arquitecturas han sido creadas en este trabajo, y por otro los modelos preentrenados basados en BERT.

En el primer grupo se encuentran las redes convolucionales CNN y las recurrentes Bi-LSTM. Se realizaron pruebas con distintos número de capas, se sometieron a búsquedas aleatorias de hiperparámetros y también se aplicó una corrección automática de palabras para aumentar la cobertura de los vectores preentrenados. De este proceso se determina que la mejor red se basa en una sola capa convolucional CNN, que obtiene un valor f1 del 68,13 %. Este resultado mejora en 2,44 puntos al mejor modelo de aprendizaje automático de este trabajo (clasificador lineal SGD) pero queda por detrás del tercer clasificado con un f1 de 72,39 %. Los resultados del modelo se ven perjudicados por el poco acierto en la etiqueta "OFG", cuyo métrica f1 se sitúa por debajo del 40 %.

El segundo grupo de modelos dentro de la categoría de aprendizaje profundo son los modelos preentrenados basados en BERT, se han realizados pruebas con dos **modelos en español**: BETO y RoBERTuito.

Con el modelo BETO se obtiene una métrica f1 que alcanza el 75,44 %**,** esto supone una mejora de 7,31 puntos respecto al modelo CNN y además con este resultado se **supera en 2,22 puntos al primer clasificado** de la tarea que obtuvo un f1 de 73.24 % con un modelo **multilenguaje** XML-RoBERTa (Garcıa and Bedmar (2021)). El participante no indica explícitamente que realice ningún tratamiento del desbalance de las etiquetas. En las pruebas realizadas en el modelo BETO (el más parecido al XML-RoBERTa), tratar el desbalance de las etiqueta supuso un 1 % de mejora en el valor de la métrica f1.

El salto de rendimiento de BETO respecto a los anteriores se debe principalmente a que se mejoran las métricas de las etiquetas menos representadas. Comparando con el modelo CNN, la métrica f1 de la etiqueta "OFG" pasa del 39.74 % al 56,32 %, un aumento del 16,58 puntos , mientras que en la etiqueta "OFP" se pasa del 68,01 % al 74,93 %, una mejora de 6,92 puntos.

El último modelo es el modelo RoBERTuito, al igual que BETO, es un modelo BERT pero entrenado con textos de Twitter, es decir, textos similares a los del conjunto de entrenamiento. El modelo obtiene un f1 del 80,11 % con el que supera al primer participante de la tarea (73.24 %) y al modelo BETO anterior (75,44 %). En esta mejora vuelve a ser clave el aumento de la métrica f1 de la etiqueta "OFG" que alcanza el 67.21 %, un incremento de 10.89 puntos.

7.1.2. Clasificación binaria

Para realizar la clasificación binaria se tomaron los mejores modelos de cada categoría (aprendizaje automático, profundo con FastText y BERT) y también enfoque (métricas y palabras). Aquí no se tienen como referencia los resultados de la tarea MeOffendES porque emplearon otro conjunto de datos para este tipo de clasificación. Los resultados se compararán con los presentados por del Arco et al. (2021). Estos fueron obtenidos sobre el mismo corpus, aunque con un número mayor de muestras ya que para la tarea MeOffendES se realizó un limpieza de instancias para aumentar la calidad del conjunto de entrenamiento.

El primer modelo evaluado es un modelo GradientBoosting entrenado solo con métricas del lenguaje su f1 es del 71,07 % y **queda por debajo del modelo de referencia** que tiene un valor del 78,39 %. El modelo se ve penalizado por su rendimiento en la etiqueta "OF" con un f1 del 50,60 % frente al 62,82 % de referencia.

El modelo lineal SGD entrenado con palabras y con ajuste fino obtiene un f1 del 79,06 %, resultado que **supera al obtenido por el modelo de referencia** (78.39 %). Empleando este modelo se probó a entrenar el modelo con una combinación de métricas y palabras, se obtuvo un resultado del 79,42 %, una mejora de solo 0,4 puntos.

Como referencia de modelos de aprendizaje profundo se tomaron el modelo CNN y RoBERTuito que como se vio en clasificación multietiqueta son los que tienen mejor resultado dentro de su categoría. El modelo CNN obtiene un f1 de 81,95 % que supera al modelo SGD y al de referencia en 3.13 puntos. La mejora se debe a un aumento considerable en el valor f1 de la etiqueta "OF" que pasa del 62,82 % de referencia al 71,39 %.

Por último, el modelo RoBERTuito alcanza el 87,67 %, mejorando el resultado del modelo de referencia en 9,28 puntos. La mejora en la etiqueta "OF" es de 16.18, se pasa del 62,82 % al 79, 63 %.

7.2. Información textual

En las pruebas realizadas en los modelos de aprendizaje automático, cuyo resultados se resumen en la tabla 6.32, se comprueba que los mejores conjunto de entrenamiento son:

1. Sin palabras vacías y con pronombres: que consiste en tomar solo palabras con caracteres latinos, limpiar acentos, pasar todo a minúsculas y eliminar las palabras vacías pero **manteniendo los pronombres personales**. Se obtiene un f1 de 62,71 %**.**

2. Procesado inicial: el texto procesado como se indicó en la sección5.1. Aquí no se eliminan palabras ni números, principalmente se trataba de recuperar palabras mal escrita y eliminar repeticiones. Se obtiene un f1 de 62,69 %

3. El texto sin procesar. Se obtiene un f1 de 62,62 %**.**

Los resultados son muy similares, solo existe una diferencia de 0.09 puntos entre el primero y el tercero. Esto es muy llamativo puesto que significa que el procesado implementado tiene poco impacto en el rendimiento del modelo. Aunque puede parecer que el texto sin procesar no sufre ningún cambio antes de entrenar los modelos, esto no es correcto puesto que el método empleado para vectorizar los textos aplica un procesado por defecto. Este consiste en tomar solo palabras de 2 o más caracteres y considerar cualquier símbolo no alfanumérico como un separador de palabras. Adicionalmente, en el proceso vectorización se conservaron solo las palabras que aparecen en 3 o más documentos. Con esto se quiere decir que todos los textos son procesado en mayor o menor medida.

Es interesante comparar el primer preprocesado con el cuarto que tiene f1 de 61,75 %, una diferencia de 0,96. La diferencia entre ambos es que el primero conserva los pronombres, de aquí se puede intuir que los pronombres tienen cierto peso en el modelo. En la tabla 7.2 se muestra el uso de los pronombre según la etiqueta del comentario. En esta tabla se ve como los pronombres que se refieren a la primera persona del singular o del plural aparecen con más frecuencia en comentarios no ofensivos. Esto tiene cierta lógica puesto que es menos probable que una persona se dirija comentarios ofensivos hacia su propia persona. También resalta el uso de los pronombres de la segunda persona del singular en los comentarios ofensivos, estos pronombres marcan una dirección concreta del comentario: del emisor al influncer u otro usuario. Se intuye que cuando se escribe un comentario dirigido a una segunda persona existe una probabilidad mayor de resultar ofensivo. Otro pronombre que destaca es el de tercera persona del plural, cuyo uso es más frecuente en los comentarios ofensivo que van dirigidos a grupos.

	NO	NOM	OFP	OFG
yo	0.0418	0.0546	0.0240	0.0303
me	**0.1154**	**0.1340**	0.0771	0.7713
mi	**0.0437**	**0.0566**	0.0179	0.0179
nos	0.0092	0.0103	0.0052	0.0041
tu	0.0561	0.0375	**0.1126**	0.0247
te	0.0873	0.0873	**0.1328**	0.0385
ti	0.0092	0.0047	0.0107	0.0013
le	0.0358	0.0327	0.0465	0.0482
ella	0.0125	0.0067	0.0204	0.0027
la	0.1727	0.2003	0.1916	0.2300
ellos	0.0050	0.0019	0.0025	0.0151
los	0.0683	0.0598	0.0527	**0.1363**
las	0.0430	0.0307	0.0376	0.0537
les	0.0091	0.0067	0.0057	0.0440

Tabla 7.2: Uso de los pronombres personales

En las pruebas realizadas sobre la red recurrente Bi-LSTM la diferencia entre usar el texto sin procesar y el texto con el procesado conservando la mayor parte de información textual es de 3,01 puntos. Una diferencia significativa comparada con los 0,09 puntos de diferencia vista en los mo-

delos de aprendizaje automático. Los primeros modelos Bi-LSTM entrenados debían aprender las representaciones de las palabras durante el entrenamiento así que emplear un conjunto de datos más ruidoso dificulta su aprendizaje y penaliza el rendimiento del modelo. El mejor resultado obtenido de esta manera fue de f1 del 62,16 %. En pruebas posteriores sus sustituyó el cálculo de vectores por vectores preentrenados FastText, primero sin aplicar correcciones de palabras y luego con ellas. El uso sin correcciones aumenta el f1 hasta el 63,40 %, una mejora de 1,24 puntos. Al corregir palabras este valor incrementa 1,69 puntos más, llegando así un valor total del 65,09 %. Esta mejora está ligada al aumento de la cobertura de vectores, antes de la corrección de palabras la cobertura era del 55,2 % y al aplicar la corrección aumenta hasta el 58,85 %.

Por último, en los modelos BERT se detectan diferencias significativas entre los distintos conjuntos de datos. El modelo BETO tiene un f1 de 74,22 % en el conjunto de datos con el procesado inicial (que conserva la mayor parte de información textual) y 69,32 % con el conjunto con menor información, esto es una pérdida de 5,01 puntos. En el caso de RoBERTuito la diferencia es menos acusada, 77,76 % con el procesado inicial y 75,15 % con el segundo. Las diferencias encontradas pueden deberse al hecho de que al eliminar las palabras vacía se está eliminando parte del contexto de otras palabras y esto penaliza el ajuste del modelo. Otro factor que penaliza a BETO es que el conjunto con menor información textual no tiene acentos mientras que cualquiera de las dos versiones de BETO sí los tiene, esto provoca que no se pueda dar cobertura a las palabras acentuadas.

Capítulo 8

Conclusiones y trabajos futuros

En este trabajo se ha visto la aplicación de diversos modelos de aprendizaje con enfoques distintos. El primer enfoque se basó en el uso de características lingüísticas extraídas de los textos (como longitud del texto o polaridad). Los resultados obtenidos por los modelos entrenados con este enfoque están muy por debajo que los obtenidos por los mismos modelos pero usando las representaciones vectoriales de los textos como datos de entrada. En la clasificación binaria la diferencia se acorta pero sigue siendo muy significativa. Aunque este enfoque no presenta buenos resultados por sí solo, al combinar algunas características lingüísticas con la información textual se obtuvo una ligera mejora, luego el uso de métricas no es una técnica descartable puesto que puede suponer una leve mejora en algunos modelos.

En el enfoque basado en el uso de los textos como entrada en los modelos se analizaron desde modelos de aprendizaje automático clásicos hasta los últimos avances en el campo del procesamiento del lenguaje natural, como son los modelos BERT. Dentro de los modelos clásicos, los modelos lineales destacaron frente a los basados en ensamblaje. Concretamente, el clasificador basado en el descenso estocástico del gradiente (SGDClassifier) ha resultado ser un modelo muy interesante. Es un modelo con una velocidad de ajuste muy rápida y que obtiene un resultado que puede servir de una buen baseline para este tipo de tareas.

Los modelos basados en redes CNN y Bi-LSTM quedan entre el modelo anterior y el ganador de la tarea que emplea un modelo BERT. La principal complejidad que se encuentra en la construcción de estos modelos es encontrar la mejor combinación de parámetros y capas. Por eso emplear modelos preentrenados pueden suponer una gran ventaja. En este trabajo se han empleado los modelos preentrenados BETO y RoBERTuito, que han supuesto un salto en el rendimiento de las predicciones.

A la hora de elegir un modelo hay que valorar otros aspectos a parte de su capacidad predictiva. Un aspecto fundamental son los recursos disponibles y el tiempo de respuesta. Los modelos BERT ofrecen muy buenos resultados pero requieren del uso de GPU de alta gama para su entrenamiento y las predicciones sobre el modelo ya ajustado tienen cierta latencia. Los modelos basados en CNN o Bi-LSTM pueden presentar problemas similares dependiendo de la complejidad de la arquitectura. Si

no se dispone de GPU o no se tienen unos requisitos de precisión muy elevado el modelo SGDClassifier puede resultar efectivo al menos en clasificación binaria.

El estudio de las distintas representaciones de los textos ha puesto de manifiesto que realizar un esfuerzo elevado en implementar un procesado intenso no siempre tiene que suponer una mejora proporcional en los resultados. Con el procesado que realizan por defecto las librerías empleadas puede ser suficiente para obtener desde el inicio buenos resultados. También se ha visto que algo tan frecuente como eliminar las palabras vacías puede empeorar el resultado por estar eliminando información útil para el problema concreto que se está tratando.

En el desarrollo realizado se han encontrado ciertas limitaciones que pueden tratarse como líneas de mejoras en trabajo futuros:

- Los comentarios ofensivos representan solo un 13 % del conjunto completo, esto limita la capacidad de aprendizaje de los modelos. Se propone aumentar el conjunto de datos añadiendo muestras de otras tareas de evaluación con temática similar (discurso de odio, racismo, etc.) en español. También se pueden incorporar conjuntos de datos en otros idiomas traduciéndolos al previamente al español.

- El corrector de palabras aplicado trabaja sobre un corpus formal que no contiene gran parte del vocabulario que se emplea dentro del ámbito de las redes sociales. Debido a esto el corrector modifica palabras que son correctas dentro de la red social, con la consecuente pérdida de información. Se propone como mejora ampliar el corrector de palabras incorporando palabras válidas dentro del ámbito donde se va a aplicar.

Bibliografía

Abraira, V. (2001). El índice kappa. *SEMERGEN - Medicina de Familia*, 27(5):247–249.

Álvarez-Carmona, M. Á., Guzmán-Falcón, E., Montes-y Gómez, M., Escalante, H. J., Villasenor-Pineda, L., Reyes-Meza, V., and Rico-Sulayes, A. (2018). Overview of mex-a3t at ibereval 2018: Authorship and aggressiveness analysis in mexican spanish tweets. In *Notebook papers of 3rd sepln workshop on evaluation of human language technologies for iberian languages (ibereval), seville, spain*, volume 6.

Aragón, M. E., Carmona, M. A. A., Montes-y Gómez, M., Escalante, H. J., Pineda, L. V., and Moctezuma, D. (2019). Overview of mex-a3t at iberlef 2019: Authorship and aggressiveness analysis in mexican spanish tweets. In *IberLEF@ SEPLN*, pages 478–494.

Aragón, M. E., Jarquín-Vásquez, H. J., Montes-y Gómez, M., Escalante, H. J., Pineda, L. V., Gómez-Adorno, H., Posadas-Durán, J. P., and Bel-Enguix, G. (2020). Overview of mex-a3t at iberlef 2020: Fake news and aggressiveness analysis in mexican spanish. In *IberLEF@ SEPLN*, pages 222–235.

Aragón, M. E. and López-Monroy, A. P. (2018). Author profiling and aggressiveness detection in spanish tweets: Mex-a3t 2018. In *IberEval@ SEPLN*, pages 134–139.

Canete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., and Pérez, J. (2020). Spanish pre-trained bert model and evaluation data. *Pml4dc at iclr*, 2020:2020.

Casavantes, M., López, R., and González-Gurrola, L. C. (2019). Uach at mex-a3t 2019: Preliminary results on detecting aggressive tweets by adding author information via an unsupervised strategy. In *IberLEF@ SEPLN*, pages 537–543.

Casavantes, M., López, R., and González-Gurrola, L. C. (2020). Uach at mex-a3t 2020: Detecting aggressive tweets by incorporating author and message context. In *IberLEF@ SEPLN*, pages 273–279.

Caselli, T., Basile, V., Mitrović, J., and Granitzer, M. (2020). Hatebert: Retraining bert for abusive language detection in english. *arXiv preprint arXiv:2010.12472*.

Chen, P. C., Huang, H.-H., and Chen, H.-H. (2020). Ntu_nlp at semeval-2020 task 12: Identifying offensive tweets using hierarchical multi-task learning approach. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2105–2110.

Chen, Y., Zhou, Y., Zhu, S., and Xu, H. (2012). Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80. IEEE.

De la Peña Sarracén, G. L. and Rosso, P. (2019). Aggressive analysis in twitter using a combination of models. In *IberLEF@ SEPLN*, pages 531–536.

De la Pena Sarracén, G. L. and Rosso, P. (2020). Prhlt-upv at semeval-2020 task 12: Bert for multilingual offensive language detection. In *Proceedings of the fourteenth workshop on semantic evaluation*, pages 1605–1614.

del Arco, F. M. P., Montejo-Raez, A., Urena-López, L. A., and Martín-Valdivia, M.-T. (2021). OffendES: A New Corpus in Spanish for Offensive Language Researc.

Djandji, M., Baly, F., Antoun, W., and Hajj, H. (2020). Multi-task learning using AraBert for offensive language detection. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 97–101, Marseille, France. European Language Resource Association.

Garcá-Dıaz, J., Jiménez-Zafra, S. M., and Valencia-Garcıa, R. (2021). Umuteam at meoffendes 2021: Ensemble learning for offensive language identification using linguistic features, fine-grained negation and transformers. In *Proceedings of the Iberian Languages Evaluation Forum (Iber-LEF 2021), CEUR Workshop Proceedings. CEUR-WS. org*.

Garcıa, M. N. and Bedmar, I. S. (2021). Detecting offensiveness in social network comments. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2021), CEUR Workshop Proceedings. CEURWS. org*.

Guzman-Silverio, M., Balderas-Paredes, Á., and López-Monroy, A. P. (2020). Transformers and data augmentation for aggressiveness detection in mexican spanish. In *IberLEF@ SEPLN*, pages 293–302.

Han, J., Wu, S., and Liu, X. (2019). jhan014 at SemEval-2019 task 6: Identifying and categorizing offensive language in social media. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 652–656, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Hassan, S., Samih, Y., Mubarak, H., Abdelali, A., Rashed, A., and Chowdhury, S. A. (2020). ALT submission for OSACT shared task on offensive language detection. In *Proceedings of the 4th*

Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, pages 61–65, Marseille, France. European Language Resource Association.

IberLEF (2021). Iberian languages evaluation forum 2021. http://ceur-ws.org/Vol-2943/.

Liu, P., Li, W., and Zou, L. (2019). Nuli at semeval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *SemEval@ NAACL-HLT*, pages 87–91.

Mahata, D., Zhang, H., Uppal, K., Kumar, Y., Shah, R., Shahid, S., Mehnaz, L., and Anand, S. (2019). Midas at semeval-2019 task 6: Identifying offensive posts and targeted offense from twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 683–690.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Nand, P., Perera, R., and Kasture, A. (2016). "how bullying is this message?": A psychometric thermometer for bullying. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 695–706, Osaka, Japan. The COLING 2016 Organizing Committee.

Nikolov, A. and Radivchev, V. (2019). Nikolov-radivchev at semeval-2019 task 6: Offensive tweet classification with bert and ensembles. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 691–695.

Oberstrass, A., Romberg, J., Stoll, A., and Conrad, S. (2019). Hhu at semeval-2019 task 6: Context does matter-tackling offensive language identification and categorization with elmo. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 628–634.

Pàmies, M., Öhman, E., Kajava, K., and Tiedemann, J. (2020). Lt@ helsinki at semeval-2020 task 12: Multilingual or language-specific bert? *arXiv preprint arXiv:2008.00805*.

Pham-Hong, B.-T. and Chokshi, S. (2020). Pgsg at semeval-2020 task 12: Bert-lstm with tweets' pretrained model and noisy student training method. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2111–2116.

Plaza-del Arco, F. M., Casavantes, M., Escalante, H. J., Martín-Valdivia, M. T., Montejo-Ráez, A., Montes, M., Jarquín-Vásquez, H., Villaseñor-Pineda, L., et al. (2021). Overview of meoffendes at iberlef 2021: Offensive language detection in spanish variants. *Procesamiento del Lenguaje Natural*, 67:183–194.

Poletto, F., Basile, V., Sanguinetti, M., Bosco, C., and Patti, V. (2021). Resources and benchmark corpora for hate speech detection: a systematic review. *Language Resources and Evaluation*, 55(2):477–523.

Pérez, J. M., Furman, D. A., Alemany, L. A., and Luque, F. (2021). Robertuito: a pre-trained language model for social media text in spanish.

Rozental, A. and Biton, D. (2019). Amobee at semeval-2019 tasks 5 and 6: Multiple choice cnn over contextual embedding. *arXiv preprint arXiv:1904.08292*.

Ruiter, D., Kleinbauer, T., España-Bonet, C., van Genabith, J., and Klakow, D. (2022). Exploiting social media content for self-supervised style transfer. *arXiv preprint arXiv:2205.08814*.

Sánchez-Gómez, C. (2018). Ingeotec at mex-a3t: Author profiling and aggressiveness analysis in twitter using μtc and evomsa. *OPENAIRE*.

Seganti, A., Sobol, H., Orlova, I., Kim, H., Staniszewski, J., Krumholc, T., and Koziel, K. (2019). Nlpr@ srpol at semeval-2019 task 6 and task 5: Linguistically enhanced deep learning offensive sentence classifier. *arXiv preprint arXiv:1904.05152*.

Tanase, M.-A., Zaharia, G.-E., Cercel, D.-C., and Dascalu, M. (2020). Detecting aggressiveness in mexican spanish social media content by fine-tuning transformer-based models. In *IberLEF@ SEPLN*, pages 236–245.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, S., Liu, J., Ouyang, X., and Sun, Y. (2020). Galileo at semeval-2020 task 12: Multi-lingual learning for offensive language identification using pre-trained language models. *arXiv preprint arXiv:2010.03542*.

Wiedemann, G., Yimam, S. M., and Biemann, C. (2020). Uhh-lt at semeval-2020 task 12: Fine-tuning of pre-trained transformer networks for offensive language detection. *arXiv preprint arXiv:2004.11493*.

Wiegand, M., Siegel, M., and Ruppenhofer, J. (2018). Overview of the GermEval 2018 Shared Task on the Identification ofOffensive Language.

y Jaime Collado-Montañez y Arturo Montejo-Ráez, R. L.-A. (2020). The text complexity library. *Procesamiento del Lenguaje Natural*, 65(0):127–130.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. (2019). Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.

Zampieri, M., Nakov, P., Rosenthal, S., Atanasova, P., Karadzhov, G., Mubarak, H., Derczynski, L., Pitenis, Z., and Çöltekin, c. (2020). SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of SemEval*.

Anexo A

TextComplexity

A.1. Lista completa de caractarísticas lingüisticas

- *charac*: número de caracteres
- *punt*: cantidad de signos de puntuación
- *words*: cantidad de palabras
- *rareW*: cantidad de palabras raras
- *lowFreqW*: cantidad de palabras de baja frecuencia
- *contentW*: cantidad de pabrabas de contenido (tiene significado propio como adjetivos, sustantivos y verbos)
- *distintContW*: cantidad de palabras distintas
- *sentences*: cantidad de frases
- *syllables*: cantidad de sílabas
- syllablesWord: número medio de sílabas por palabras
- *WordSentences*: número medio de palabras por frase
- *ILFW (Index of Low Frequency Words)* número de palabras de baja frecuencia por cada 100 *content words*
- *LC (Lexical Complexity)*: proporción de content words por frases
- *SSR (Spaulding Spanish Readability)* mide la complejidad de lectura
- *ASL (Average Sentence Length)* longitud media de las frases

- *CS (Complex Sentences)* el número de frases complejas por frase, a partir de un índice de frases complejas
- *SCI (Sentence Complexity Index)* medida de la complejidad de un texto destinado a lectores de un segundo idioma
- *ARI (Automated Readability Index*) dificultad de un texto a partir del número medio de caracteres (letras y números) por palabra y del número medio de palabras por frase.
- *MaxEmdedding:* tamaño del embedding más grande
- *MinEmdedding:* tamaño del embedding más pequeño
- *MeanEmdedding:* tamaño medio de los embeddings
- *Huerta*: adaptación del test de lectura de Flesch (medida que indica como de complejo es entender un pasaje en inglés)
- *IFSZ (Readability of FleschSzigrist)*: adaptación Flesch, se centra en medir el número de sílabas por palabras y el número de palabras por frase.
- *Polini (Comprehensibility of Gutíerrez de Polini):* mide el número medio de letras por palabras y el número de palabras por frase.
- *Mu:* se centra en medir el número de palabras, el número medio de letras por palabra y su variabilidad.
- *Min Age (Minimum age to understand)*: mide el número medio de sílabas por palabra y el número medio de palabras por frase para obtener la edad mínima necesaria para comprender un texto.
- *SOL*: mide la capacidad de lectura de un texto por medio del nivel de grado, que es el número de años de escolaridad requeridos para comprender el texto.
- *Crawford*: calcula los años de escolaridad necesarios para comprender un texto. Se mide el número de frases por cada cien palabras y el número de sílabas por cada cien palabras.