Informe de Proyecto de Grado presentado al Tribunal Evaluador como requisito de graduación de la carrera Ingeniería en Computación

Año 2019

Identificación de Discurso de Odio en Redes Sociales

Autores:
Lucas Kunc
Manuel Saravia

Supervisores:
Mathias Etcheverry
Juan José Prada

3 de septiembre de 2020

Resumen

Este informe describe el desarrollo del trabajo realizado con el objetivo principal de identificar discurso de odio en redes sociales, a través de métodos de aprendizaje automático.

Para ello, se construye un corpus conformado por publicaciones de la red social *Twitter*, anotado según contengan discurso de odio o no. La anotación se realiza mediante *crowdsourcing*, a través de una aplicación web desarrollada a estos efectos, reportándose un acuerdo entre los anotadores de 0.537 según la alfa de Krippendorff. Luego, se comparan distintos modelos a partir del desempeño que presentan realizando la tarea de clasificación sobre este corpus. El mejor clasificador obtenido consiste en un modelo SVM, el cual logra un *f-score* de 0.846 sobre el conjunto construido.

El trabajo desarrollado muestra el enfoque utilizado para resolver la tarea de detección automática de discurso de odio, las principales dificultades encontradas y propuestas para intentar superarlas.

Palabras clave: clasificación de texto, discurso de odio, Twitter, anotación de corpus, aprendizaje automático, procesamiento de lenguaje natural.

Tabla de Contenido

1 | Introducción

¿Qué es la tolerancia? Según el diccionario de la Real Academia Española, se define como el "*Respeto a las ideas, creencias o prácticas de los demás cuando son diferentes o contrarias a las propias*". También implica el trato igualitario hacia cualquier individuo o colectividad, independientemente de su raza, etnia, religión, ideología, sexo, género, edad, condición física o mental, en general características que determinan una parte de su ser. Desafortunadamente, su opuesto, la intolerancia, aún tiene una presencia importante en la sociedad.

Un claro ejemplo en la edad contemporánea de las consecuencias a las que puede llevar la intolerancia fue el surgimiento del partido Nazi, el cual se gestó en la Alemania post Primera Guerra Mundial. Eran características de sus políticas sociales la detención de judíos, gitanos y homosexuales, con la póstuma reclusión en campos de concentración (los cuales llamaban, para evitar propaganda negativa, campos de trabajo). También impulsaban el trato diferencial de razas que consideraban inferiores, como los negros [41]. En el corriente año 2020, el asesinato de el ciudadano afroamericano George Floyd a manos de un policía blanco han provocado una verdadera revolución en Estados Unidos en contra del abuso diario que recibe la población negra en ese país por parte de la policía [13]. En otros ámbitos, la importancia que ha retomado el movimiento feminista en consecuencia de la misoginia que se encuentra arraigada a la sociedad, o las marchas que se llevan a cabo en el Día Internacional del Orgullo LGBT+ son muestras del rechazo hacia la marginalización que sufren día a día estos grupos de personas.

En el lenguaje, la intolerancia se manifiesta a través del fenómeno del **discurso de odio**, también conocido como *hate speech* por su denominación en inglés. Este refiere, como veremos más adelante, a cualquier tipo de comunicación en la escritura o el habla, que a través de ciertas características de una persona la denigren o discriminen.

En este trabajo, el objeto de estudio será el fenómeno de discurso de odio, al cual nos referiremos de ahora en adelante como DO, y cómo puede ser detectado automáticamente.

1.1. Motivación

Hoy en día, las redes sociales son una herramienta de comunicación que puede considerarse naturalizada, es decir, son parte de nuestro día a día y casi todos participamos en una o varias de ellas. Lamentablemente, sirve muchas veces de plataforma para que personas con malas intenciones causen dolor o angustia en otras.

Para resolver este problema, las redes sociales contienen reglas de convivencia entre los usuarios que apuntan a evitar la proliferación del DO. Aún así, no son siempre respetadas por lo que se debe recurrir a métodos que aseguren su acatamiento. En la actualidad, la solución más común consiste en filtros basados en palabras, los cuales pueden censurar erróneamente a usuarios que no emplean DO. Otros métodos como la revisión manual de publicaciones requieren un mayor esfuerzo para ser aplicados.

Nuestra principal motivación consiste en investigar opciones más sofisticadas, que tomen en cuenta la intención del usuario para detectar automáticamente si el mensaje que quiere transmitir contiene DO, a través de nuevas tecnologías. Por otro lado, la identificación de DO ha tomado importancia en el área del procesamiento de lenguaje natural y es de nuestro interés intentar aportar a su comunidad.

En el marco de este trabajo, tomamos como caso de estudio la red social ***Twitter***. En *Twitter*, los usuarios realizan publicaciones cortas conocidas como ***tweets***, sobre el tema que decidan y con el propósito que deseen. El siguiente es un claro ejemplo de como se manifiesta el discurso de odio en esta red social:

Se puede observar cómo el autor de esta publicación utiliza el término *sudaca* para referirse despectivamente a una persona de origen latinoamericano. Además, la palabra *tercermundista* en este caso es empleada para denigrar, aunque su significado fundamental no necesariamente apunta a generar una reacción negativa sobre el receptor.

1.2. Objetivo

El objetivo general del proyecto es identificar DO en redes sociales mediante técnicas de procesamiento de lenguaje natural. Sobre la base de esta problemática, se plantean siguientes objetivos específicos:

- Comprender el concepto de DO (en qué consiste y cómo se manifiesta, en particular en la red social *Twitter*).
- Estudiar el estado del arte sobre la identificación automática de DO, conociendo las contribuciones que se han realizado hasta el momento y las principales dificultades que deben ser afrontadas.
- Crear un conjunto de tweets anotados según contengan DO o no, que sirva como referencia para este y futuros trabajos.
- Experimentar con métodos de aprendizaje automático que logren identificar tweets con DO.

A partir estos objetivos, esperamos aplicar los conocimientos sobre aprendizaje automático y procesamiento de lenguaje natural, adquiridos en el transcurso de la carrera, para llevar a cabo la tarea con éxito.

1.3. Cronograma

A continuación, en la figura 1.1, se muestran las distintas etapas en las que consistió el proyecto y el tiempo aproximado dedicado a cada una de ellas.

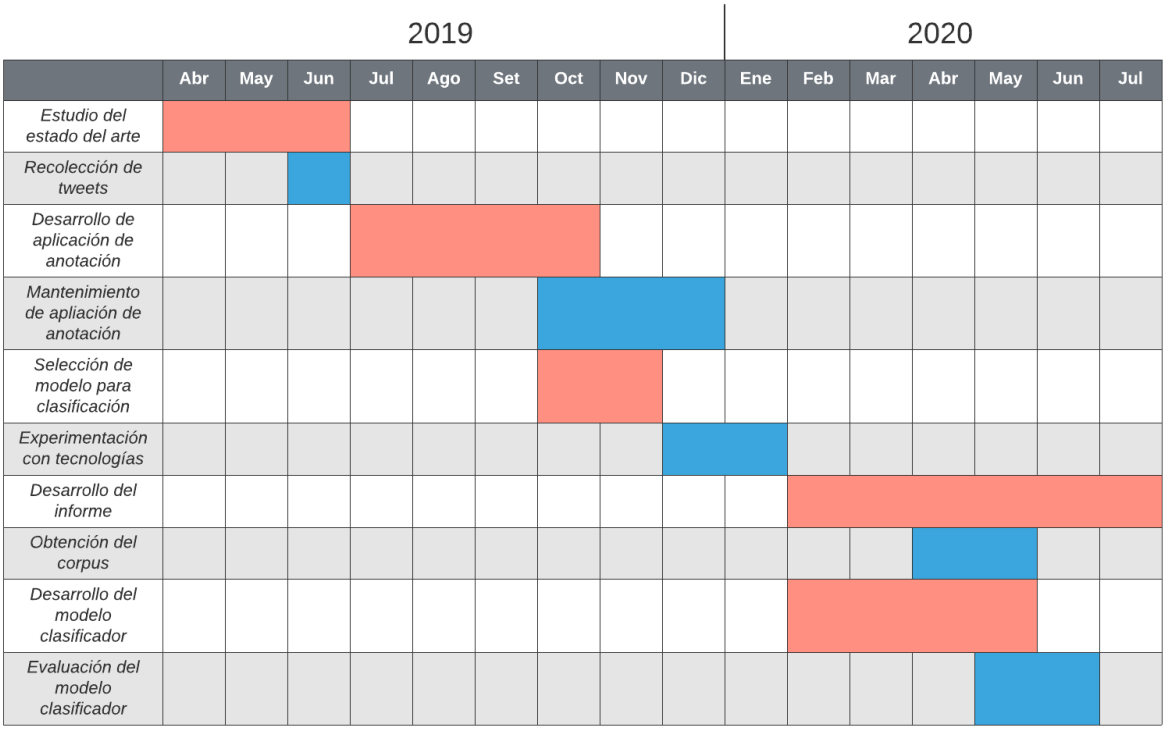


Figura 1.1: Diagrama de Gantt de las tareas desarrolladas en el transcurso del proyecto

1.4. Organización del documento

El documento se encuentra organizado de la siguiente manera: en el capítulo 2 se presentan conceptos base sobre el área del procesamiento de lenguaje natural y el aprendizaje automático en general, que serán manejados a lo largo del trabajo. Luego se dedica el capítulo 3 al análisis en profundidad de la problemática del DO y a presentar los principales trabajos que se han desarrollado para abordarla. En el capítulo 4 se detalla el proceso de construcción de un conjunto de datos para la detección de DO. Finalmente, el capítulo 5 presenta modelos de aprendizaje automático para la llevar a cabo la detección de DO, con el posterior análisis de los resultados en el capítulo 6 y las conclusiones que se derivan del trabajo realizado en el capítulo 7.

2 | Conceptos relacionados con el procesamiento de lenguaje natural

Para poder comenzar a hablar de detección de DO en el marco de este trabajo, es necesario comprender ciertos términos y conceptos fundamentales acerca de las técnicas utilizadas. Estas definiciones brindarán al lector los conocimientos básicos necesarios para entender las ideas presentadas en las secciones siguientes.

2.1. Procesamiento de lenguaje natural

El **procesamiento de lenguaje natural** (abreviado comúnmente **PLN**) es, según Daniel Jurafsky y James H. Martin [22] una disciplina que tiene como objetivo lograr que un sistema computacional resuelva tareas que involucran el lenguaje natural (i.e. el lenguaje utilizado por los seres humanos), como la comunicación persona-computadora, mejorar la comunicación entre las personas, en general, procesar de una manera útil y orientada a una tarea específica texto o habla. Existen muchas tareas asociadas al PLN, entre las más populares se encuentran la traducción automática, extracción de información, clasificación de texto, análisis semántico, generación automática de texto, reconocimiento de voz, entre otras.

2.2. Dataset y corpus

Todas las tareas de detección, cualquiera sea el objeto a detectar, se realizan sobre un conjunto de datos o *dataset*. Un ***dataset*** es una colección de datos preparados para ser procesados computacionalmente, usualmente acompañados por varias medidas estadísticas acerca de los datos contenidos (e.g. media, desviación estándar) u otros tipos de metadatos.

Para el caso de la detección en texto como la que se realiza en este trabajo, los *datasets* son construidos a partir de un **corpus**, es decir, una colección de textos que

poseen ciertas características en común, como pueden ser autor, tema, idioma en el que fueron escritos, etc. Se diferencian dos tipos de corpus de acuerdo a la información que contienen: anotado y no anotado. Un **corpus no anotado** esta formado puramente por texto, sin información adicional. Por otro lado, un **corpus anotado** contiene anotaciones manuales que proporcionan información adicional de carácter sintáctico o semántico a nivel de palabra, párrafo o texto. Un ejemplo en el contexto de detección de DO es un corpus anotado a nivel de párrafo con la etiqueta "ofensivo" si contiene lenguaje ofensivo o con la etiqueta "no ofensivo" en caso contrario. Los corpus anotados son utilizados tanto para entrenar modelos de aprendizaje automático (ver sección 2.4), como para la evaluación y comparación de diferentes enfoques para la problemática que se pretende resolver.

2.2.1. Medida de acuerdo entre anotadores

Cuando se desea anotar un conjunto de datos, es común que esta tarea sea realizada por más de un anotador para reducir el sesgo en la anotación. El **puntaje de acuerdo entre anotadores** (en inglés *inter-annotator agreement score*), refiere a las métricas que miden el acuerdo entre los anotadores. En base a este puntaje, se puede evaluar qué tanto confiar en las etiquetas asignadas al conjunto final. También se puede utilizar para determinar que tan difícil es la resolución de un problema. Mientras más desacuerdo existe entre los anotadores, más probable es la posibilidad de que el problema a resolver presente dificultades en su abordaje.

La elección de el método con el que se calcula la métrica depende del problema. La **alfa de Krippendorff** (Krippendorf, 2011 [25]) es uno de los más flexibles, permitiendo un número arbitrario de anotadores y categorías de tipo nominales, continuas, de intervalos, entre otras. Por otro lado el **kappa de Cohen** (Cohen, 1960 [9]) solo permite 2 anotadores, y el **kappa de Fleiss** (Fleiss, 1971 [17]) sólo funciona con categorías de tipo nominal. Las fórmulas que calculan estos valores son complejas, pero en general toman en cuenta la relación entre el desacuerdo real y el desacuerdo de una hipotética anotación al azar. La correspondiente a la alfa de Krippendorf es

$$\alpha = 1 - \frac{D_o}{D_e}$$

donde D_0 es el desacuerdo observado y D_e es el esperado si la anotación se realiza al azar, y ambos se definen como

$$D_o = \frac{1}{n} \sum_{c \in R} \sum_{k \in R} \delta(c,k) \sum_{u \in U} m_u \frac{n_{cku}}{P(m_u, 2)}$$

$$D_e = \frac{1}{P(n,2)} \sum_{c \in R} \sum_{k \in R} \delta(c,k) P_{ck}$$

Dónde:

- R es el conjunto de votos posibles
- u se conoce como una *unidad*, que refiere al conjunto de votos recibido por un ejemplo
- δ es una función de diferencia entre dos votos (para el caso de categorías nominales, vale 1 si los votos son diferentes y 0 en caso contrario)
- m_u es la cantidad de votos para una unidad u
- n es el número de total de votos emparejables en todas las unidades; un voto es emparejable si existen uno o más votos adicionales para el mismo ejemplo
- n_{cku} es el número de pares (c, k) de votos en la unidad u
- $P(i, j)$ es la cantidad de permutaciones de largo j de i elementos
- P_{ck} es la cantidad de veces que el par de votos (c, k) aparece en el conjunto de unidades

El valor resultante de esta medida se encuentra siempre en un rango entre -1 y 1, donde un valor igual a 1 indica acuerdo total, 0 indica que las anotaciones no son confiables en absoluto y un valor negativo indica un desacuerdo mayor al que podría ser esperado si los votos se asignan al azar.

2.2.2. Crowdsourcing

Si el conjunto tiene un tamaño considerable, se pueden utilizar recursos para minimizar el esfuerzo requerido, uno de ellos es realizar el ***crowdsourcing*** de la anotación. El término *crowdsourcing* proviene de la contracción entre las palabras en inglés *crowd* (multitud) y *outsourcing* (tercerizar el desarrollo de una tarea) (Schenk et al., 2009 [37]), por lo que podemos interpretar el significado como la tercerización de una tarea a la multitud. Cuando se usa en el ámbito de la internet, se refiere a lograr que los usuarios, en conjunto, resuelvan una tarea que requiere una alta disponibilidad de recursos para su resolución.

2.3. Preprocesamiento

El **preprocesamiento** es la etapa de preparación de un dataset para su posterior uso por un algoritmo de aprendizaje automático como los que se explicarán más adelante. Generalmente las colecciones de datos sin procesar contienen una gran cantidad de

ruido o información extra, irrelevante para la tarea que se desea realizar. Estas características pueden influir negativamente en el aprendizaje de los sistemas. Para el caso de datasets formados por texto, dependiendo de sus propiedades y de la tarea a realizar, el preprocesamiento puede incluir, entre otros, uno o más de los siguientes pasos:

- Pasar todas las letras a minúsculas
- Remover todos los números o convertirlos en palabras
- Remover signos de puntuación y otros símbolos
- Remover espacios
- Expandir abreviaciones
- Remover ***stopwords***, es decir, palabras que no aportan información desde el punto de vista de la tarea a realizar. Usualmente palabras comunes como "el", "la", "es", "de", etc.
- ***Tokenización*** del texto. Esto es, agruparlo en pequeñas piezas, o *tokens*, en lugar de representarlo como una secuencia de caracteres.
- Pasar el texto a una representación canónica, por ejemplo mediante **lematización** (asignar una palabra como la representante de todas sus formas flexionadas) o ***stemming*** (considerar las palabras con la misma raíz como el mismo *token*).

2.4. Aprendizaje automático

Hasta la década de los '80, los sistemas de PLN se basaban principalmente en conjuntos complejos de reglas creadas a mano específicamente para cada tarea a realizar. Si bien era posible llegar a buenos resultados para algunas tareas, éste enfoque implicaba altos costos de desarrollo, baja portabilidad de las soluciones, y aún así existían problemas como la traducción automática, subtitulado de imágenes y reconocimiento de voz, entre otros, para los que diseñar un sistema basado en reglas que funcionara de manera general y obtuviera buenos resultados era prácticamente inviable. A partir de entonces, gracias al incremento de poder computacional y los avances teóricos, se comenzaron a incorporar cada vez más los métodos estadísticos y los algoritmos de aprendizaje automático a los sistemas de PLN, convirtiéndose hoy en día en la forma de programación más utilizada para resolver esta clase de problemas. No obstante, existen tareas que los algoritmos basados en reglas resuelven de manera eficiente y por lo tanto éstos continúan siendo herramientas importantes para el área.

Aprendizaje automático es una rama de la inteligencia artificial cuyo objeto de

estudio son algoritmos computacionales que mejoran automáticamente a través de la experiencia (Mitchell, 1997 [30]). Estos algoritmos construyen un modelo matemático basado en datos de ejemplo, llamados "datos de entrenamiento", que les permite realizar predicciones o tomar decisiones sin haber sido programados explícitamente para ello. El hecho de basarse principalmente en datos para la toma de decisiones implica que estas técnicas se encuentren fuertemente relacionadas con la estadística computacional, por lo que generalmente los descubrimientos o resultados de una de estas áreas repercuten directamente sobre los de la otra.

A grandes rasgos, los algoritmos de aprendizaje automático se pueden agrupar en tres tipos: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzos. Existe también el concepto de aprendizaje semisupervisado, en el cual se combinan las ideas del aprendizaje supervisado y no supervisado que presentaremos a continuación. Si bien el aprendizaje por refuerzos es utilizado extensamente en otras áreas, su uso no es muy común para la tarea de identificación y clasificación de textos, y por lo tanto no lo detallaremos.

En el **aprendizaje supervisado** se le presenta al programa ejemplos con valores de entrada y sus respectivos valores de salida y el objetivo es aprender una función general cuya salida sea la correcta dada cualquier entrada. Un ejemplo en el contexto de DO es un programa que dado un conjunto de tweets anotados con etiquetas "tiene DO" y "no tiene DO", aprenda a decidir si un tweet cualquiera contiene discurso de odio.

Por otro lado, la idea del **aprendizaje no supervisado** es que el programa descubra estructuras o patrones únicamente a partir de los datos de entrada, sin el uso de información extra. Para el caso de DO, un ejemplo de este tipo de algoritmo sería un sistema que, dado un dataset de tweets que contienen DO, descubra si existen subcategorías para éstos, relacionadas al DO que contienen.

Estos dos enfoques resuelven problemas distintos y es usual que se combine el uso de ambos a la hora de diseñar soluciones a problemas de PLN.

Para determinar la efectividad de un modelo de aprendizaje automático en la resolución de una tarea, se utiliza una metodología que comúnmente se divide en 2 etapas: **entrenamiento y evaluación o *testing***. Para la realización de cada etapa se divide el conjunto de datos en 2 partes (una por cada etapa). La proporción del conjunto de datos destinada a cada etapa se determina *ad hoc*, pero en general se utiliza el 20 % para la fase de evaluación y el resto para la de entrenamiento. La fase de entrenamiento consiste, en líneas generales, en generar una predicción para cada ejemplo del conjunto de datos, compararla con el valor esperado y utilizar una métrica que determine el grado de error de la predicción realizada, para luego ajustar el modelo buscando reducir el error en pre-

dicciones posteriores. La métrica que mide el grado de error que comete el modelo en las predicciones se conoce como **función de pérdida**. Una etapa adicional del entrenamiento es la **validación**, la cual se puede utilizar para diversos fines. Uno de los más comunes es el ajuste de **hiperparámetros** (parámetros que controlan el proceso de aprendizaje y se mantienen fijos durante el entrenamiento), eligiendo la combinación de éstos que mejor desempeño logre en la fase de validación. También puede estar orientada a monitorear la evolución del modelo en el entrenamiento. Al finalizar la fase de entrenamiento se procede a realizar la fase de evaluación, en la cual se obtienen las métricas del modelo sobre el conjunto de evaluación para medir su desempeño sobre datos "no vistos". Este método se conoce como **método de retención o *hold-out***. También es usual que se elija para el modelo final el estado del modelo en la época que mejor desempeño logró sobre los datos de validación. Alternativamente al método de retención, existe la modalidad denominada **validación cruzada**, en la cual se crea una cantidad n de subconjuntos del conjunto de entrenamiento y con ellos se crean n particiones dejando uno de los subconjuntos para evaluación y dejando el resto para entrenar. El modelo se entrena y evalúa con cada partición y luego se reporta el promedio de las métricas obtenidas como medida de su desempeño. Se suele acudir a esta modalidad para obtener una mejor estimación del rendimiento del modelo. También puede utilizarse para identificar casos de **sobreajuste**, un fenómeno en el que el modelo logra un desempeño muy bueno sobre el conjunto de entrenamiento, pero malo en la fase de evaluación. En métodos como las redes neuronales (presentadas en la sección 2.7), es común dividir la fase de entrenamiento en épocas, en donde para cada una de ellas se lleva a cabo el proceso de entrenamiento sobre el conjunto de datos determinado para el mismo. La razón para esto es que el clasificador se puede sobreajustar a los últimos datos evaluados en el entrenamiento, y el entrenamiento en épocas ha demostrado reducir la probabilidad de que ocurran estos fenómenos. La técnica conocida como ***early stopping*** o **detención temprana** toma las métricas de la fase de validación para dejar de entrenar el modelo cuando no se nota una mejora en éstas durante una cantidad de épocas determinada.

Por lo general, se desea determinar si el modelo obtiene mejores resultados comparado con una metodología de menor complejidad. Es usual compararlo contra resultados obtenidos al utilizar un método más "*naive*", por ejemplo, en el contexto de la clasificación de datos, generar una predicción aleatoria con una probabilidad equitativa para cada resultado posible. Los resultados de este método se denominan **línea base** o *baseline*.

2.5. Regresión logística

El modelo de **regresión logística** es uno de los más utilizados en el área de aprendizaje supervisado. Los algoritmos basados en regresión logística calculan la probabilidad de que un ejemplo pertenezca a cierta clase o categoría para determinar si efectivamente pertenece o no a ella. Este cálculo se realiza asumiendo que la categoría y los atributos de la entrada se relacionan de acuerdo a una función logística. A modo de ejemplo, suponiendo que los elementos a clasificar se modelan como un par de atributos (x_1 y x_2) y la categoría a la que pertenecen (Y, con valores posibles 0 o 1), la probabilidad $P(Y = 1)$ de que un elemento pertenezca a la categoría 1 se rige por la siguiente ecuación:

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}}$$

donde β_0, β_1 y β_2 son los parámetros a aprender por el algoritmo.

La idea que captura este método es la de hallar un hiperplano que divida a los elementos del conjunto de entrenamiento, intentando maximizar la probabilidad de los elementos que pertenecen a una categoría y a al mismo tiempo de minimizarla para aquellos que pertenecen a la otra.

2.6. Support vector machine

Otros modelos de clasificación muy populares son las ***Support Vector Machines* (SVM)**. Al igual que los modelos de regresión logística, las SVM son modelos que permiten la clasificación de ejemplos nuevos en dos categorías determinadas. También comparten la idea principal de encontrar un hiperplano que separe a los datos del conjunto de entrenamiento. Sin embargo, la diferencia está en el método de exploración para hallar la solución. Como puede apreciarse en la figura 2.1, las SVM buscan el hiperplano que maximiza su distancia a ambos subconjuntos de datos. Otra diferencia entre los algoritmos de regresión logística y aquellos basados en SVM es que, la salida de un algoritmo de regresión logística es un número entre 0 y 1 que representa la probabilidad de que el ejemplo pertenezca a cierta clase, mientras la salida de una SVM, al no tratarse de un modelo probabilístico, es directamente 0 o 1, indicando la clase a la que el ejemplo pertenece.

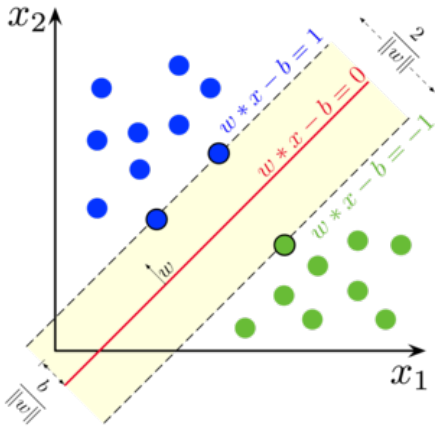


Figura 2.1: El hiperplano solución (rojo) hallado por una SVM divide a los elementos de las dos categorías (azul y verde) y maximiza su distancia a cada una de ellas.

2.7. Redes neuronales

Las **redes neuronales** son métodos de aprendizaje supervisado que se utilizan comúnmente para problemas que requieren la interpretación de datos complejos o ruidosos. La unidad básica de la red neuronal es la **neurona**. Ésta tiene como forma base el **perceptrón**, el cual recibe la secuencia de entrada, realiza una combinación lineal de la misma y emiten como salida 1 si el valor resultante es mayor que 0 o 0 si no lo es.
Las redes neuronales son agrupaciones de neuronas en **capas**. Cada capa consiste de un conjunto de neuronas, que pueden ser organizadas en dos o más dimensiones dependiendo de la forma que tengan los datos con los que se desea trabajar, las cuales reciben la salida de la capa anterior y realizan la combinación lineal de ésta. La capa que recibe los datos y la que emite la salida final se conocen como capas de **entrada** y de **salida** respectivamente, mientras que las posicionadas entre éstas dos son conocidas como **capas ocultas**. Para problemas de clasificación binaria, la capa de salida suele tener una única neurona que calcula la predicción final. Es común que a cada neurona de una capa oculta se le aplique una función antes de propagar la salida hacia la siguiente capa. Esta función se conoce como **función de activación**. Cuando la salida de cada capa se propaga hacia adelante, nos encontramos ante una red neuronal de tipo ***feed-forward***. Una agrupación de capas ocultas en serie se conoce como *perceptrón multicapa* (ilustrado en la figura 2.2). Al estudio y aplicación de modelos de redes neuronales con más de una capa oculta se lo conoce como **aprendizaje profundo**.

Otras arquitecturas de redes neuronales que se suelen encontrar en la literatura son las ***Convolutional Neural Networks*** (CNN) y las ***Recurrent Neural Networks*** (RNN).

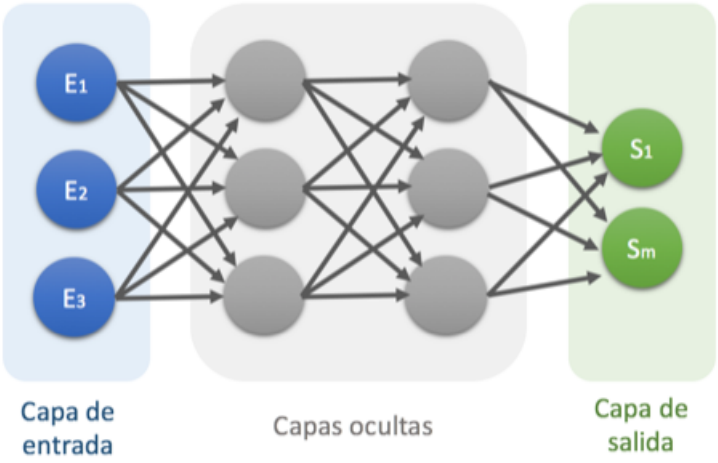


Figura 2.2: Perceptón multicapa, cada círculo corresponde a una neurona

Las CNNs surgen del campo de la visión artificial y constan de capas de **convolución** y **reducción** (o *pooling*) intercaladas, más una capa de entrada y una de salida. Las capas de convolución son capas bidimensionales que realizan una combinación lineal de un determinado filtro (básicamente una matriz de valores constantes) con la matriz de entrada en una operación denominada **convolución**. Las de reducción se encargan de tomar las capas convolucionadas y aplicar funciones estadísticas para obtener sus valores más representativos.

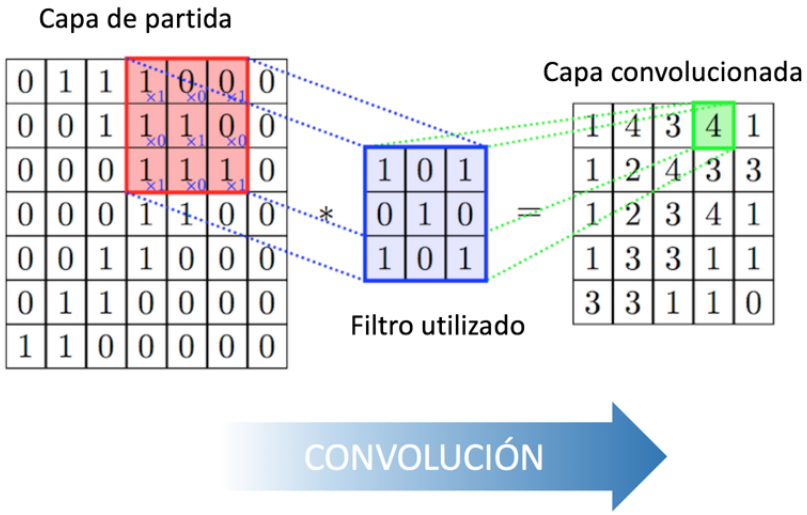


Figura 2.3: Ilustración de la operación que realiza una capa de convolución sobre la entrada que recibe

En cuanto a las RNNs, tienen la característica principal de que, además de propagar la salida hacia adelante, una o más capas utilizan su propia salida emitida para computar el valor o valores de salida posteriores, por lo que se dice que posee la capacidad de mantener un *estado*. Una arquitectura común para este tipo de redes se conoce como LSTM (*Long*

short-term memory), que agrega al concepto básico de las RNN la capacidad de regular la influencia que tienen en el estado la entrada, la salida y los valores de salida en etapas anteriores.

2.8. Word embeddings

El término ***word embeddings*** hace referencia a representaciones de palabras en vectores de valores reales. Tiene su origen en el trabajo de J. R. Firth y Zellig Harris sobre semántica distribucional. Cada uno de los valores del vector se determina en función de los contextos en los que ocurre la palabra a representar. Se busca que la relación entre los vectores sea acorde a la distribución de éstos contextos en el conjunto de datos. Si los datos que estamos manejando son, en lugar de palabras, unidades de texto mayores (como oraciones o párrafos), la representación vectorial suele denominarse *sentence embeddings*. A continuación definimos terminología y algunos de los métodos mencionados en este trabajo para la generación de *word* y *sentence embeddings*:

- Embeddings basados en **Bag-of-Words (BoW)**: El método *Bag-of-Words* genera representaciones de documentos en la que cada dimensión del vector generado corresponde a la cantidad de veces que ocurre una palabra de un determinado vocabulario en el documento que se desea representar. El vocabulario es generalmente obtenido de todas las palabras presentes en el conjunto de documentos. A partir de estas representaciones se pueden obtener embeddings a nivel de documento mediante técnicas como LSA (del inglés ***L**atent **S**emantic **A**nalysis*, análisis semántico latente) (Deerwester et al., 1990 [12]). Es común que, en lugar de considerar los términos por separado, se consideren agrupaciones de dos o más términos. A estas agrupaciones se le denomina *n-gramas*. Si al construir los vectores sólo se toma la cantidad de ocurrencias de cada término (es decir, la frecuencia) es probable que los más comunes tomen mayor importancia aunque no sean los más relevantes (como las preposiciones). La medida ***TF-IDF*** (*Term Frequency - Inverse Document Frequency*) puede utilizarse para evitar esto. En lugar de considerar la ocurrencia de los términos, se toma el producto de los dos valores que le dan su nombre. **Term Frequency** refiere en general a la frecuencia normalizada de un término t definida como, dado un documento d, la cantidad de veces que ocurre el término en el documento ($f(t,d)$) en relación a la palabra con mayor frecuencia en el documento:

$$tf(t,d) = \frac{f(t,d)}{max\{f(t',d) : t' \in d\}}$$

  La frecuencia inversa de documento (**Inverse Document Frequency**) se calcula, dado un término t y conjunto D de documentos, como el logaritmo del cociente

entre la cantidad de documentos en D y la cantidad de documentos que contienen el término t:

$$idf(t, D) = log \frac{|D|}{|\{d \in D : t \in d\}|}$$

TF-IDF es el producto de ambas métricas, es decir:

$$tf\text{-}idf(t, d, D) = tf(t, d) \times idf(t, D)$$

- **Continuous Bag-of-Words (CBoW)**: ***CBoW*** es un método para generar *word embeddings* en el que se entrena un modelo de redes neuronales simple para predecir una palabra en una secuencia, dadas las palabras que la rodean (es decir, su contexto). El clasificador toma las palabras del contexto codificadas con *embeddings* inicializados mediante alguna estrategia, realiza la predicción y ajusta los valores de los *embeddings* a partir de ésta. La cantidad de palabras que se consideran para el contexto es un parámetro que puede variar y se ajusta para lograr el mejor resultado en la tarea para la cual los vectores se utilizarán.

- **Skip-gram**: El método ***skip-gram*** es similar a *CBoW*, diferenciándose en que se predice la probabilidad que tiene cada palabra del vocabulario de pertenecer al contexto de una palabra determinada.

- **Vectores contextualizados**: Es una familia de métodos que utilizan redes neuronales recurrentes para generar tanto *word embeddings* como *sentence embeddings*. Provienen de una solución reciente al problema de traducción automática, mediante la cual se entrenan dos redes recurrentes: una que codifica a una representación vectorial el texto desde el idioma origen (llamada *encoder*) y otra que, a partir de esta representación, genera el texto en el idioma destino (llamada *decoder*). Esto permite utilizar el *encoder* para generar los *word embeddings* o *sentence embeddings* necesarios. Ejemplos de estos métodos son *ELMo* (Peters et al., 2018 [35]), *BERT* (Devlin et al., 2018 [14]) y *Universal Sentence Encoder* (Cer et al., 2018 [6]).

Otro método usualmente utilizado para construir *sentence embeddings* es generar primero los *word embeddings* de cada *token* en la oración y luego combinarlos, por ejemplo, linealmente o mediante otra estrategia.

2.9. Métricas de desempeño

Habiendo elegido un modelo para la tarea que se desea resolver, existen distintas métricas que permiten determinar cuán exitosamente funciona el modelo sobre los datos con los que se evalúa. A continuación definimos las más utilizadas.

Accuracy

También llamada **acierto**, es simplemente la proporción de predicciones acertadas en relación a la cantidad de predicciones realizadas:

$$\frac{predicciones\ acertadas}{predicciones\ realizadas}$$

Es usual para el análisis de modelos de aprendizaje automático, mostrar el resultado de su evaluación en términos de la **matriz de confusión**. En la tabla 2.1 se ilustran los valores que se calculan para conformar esta matriz

	Observación Positiva	**Observación Negativa**
Predicción Positiva	Verdaderos positivos (vp)	Falsos positivos (fp)
Predicción Negativa	Falsos negativos (fn)	Verdaderos negativos (vn)

Tabla 2.1: Construcción de una matriz de confusión

En base a los resultados de la matriz de confusión, se computan las métricas explicadas a continuación.

Precisión

Es la cantidad de predicciones correctas del valor positivo sobre el total de predicciones del valor positivo:

$$precision = \frac{vp}{vp + fp}$$

Con la precisión podemos observar que probabilidad de acierto tiene el clasificador cuando emite una predicción positiva.

Recall

También conocida como sensibilidad (aunque es más común encontrarla como *recall* en la literatura de cualquier idioma), es la cantidad de verdaderos positivos en relación a la cantidad de verdaderos positivos y falsos negativos:

$$recall = \frac{vp}{vp + fn}$$

El *recall* permite tener una idea de cuán exitoso es el clasificador prediciendo casos positivos.

F-Score

F-Score hace referencia comúnmente al F_1-$Score$, métrica que combina precisión y *recall*. Por si sola no tiene una interpretación relevante, pero es generalmente utilizada para comparar el desempeño general entre varios modelos predictivos. Se calcula como:

$$F\text{-}score = 2 \times \frac{precision \times recall}{precision + recall}$$

Area Under the Curve (AUC)

El *Area Under the Curve*, es decir, área bajo la curva, refiere en realidad al área bajo la curva característica operativa del receptor ($AUROC$). Ésta se obtiene de la gráfica donde el eje vertical contiene el *recall* y en el eje horizontal la tasa de ejemplo positivos clasificados erróneamente como tal, a medida que varía el umbral de decisión para el clasificador. Notar que, por esta razón, la métrica no puede ser utilizada con clasificadores en los que su salida no es una probabilidad. De esta manera se tiene una visión de como se ve afectada la capacidad del modelo para clasificar ejemplos correctamente cuando varía el umbral de decisión. Se espera que esta curva sea pronunciada, indicando que sin importar el valor elegido el modelo tiene un buen desempeño. Uno de los beneficios que otorga utilizar esta métrica es que se remueve el sesgo que puede inducir la elección del valor para el umbral.

3 | Estado del arte

En este capítulo se procederá a establecer el estado del arte en cuanto a la detección de DO. Antes de profundizar en este tema, es necesario comprender a qué nos referimos cuando hablamos de DO. Al igual que muchos conceptos referidos al lenguaje natural, no existe una definición de discurso de odio que se pueda considerar estándar, por lo general estos conceptos se adquieren de manera natural a medida que cada persona se desarrolla como ser humano social.

3.1. ¿Qué es el discurso de odio?

A continuación presentamos definiciones de DO en distintos ámbitos, sobre las que basamos gran parte de las decisiones tomadas durante el proyecto, que muchas veces implicaron desambiguar casos en los cuales no era claro si un texto presentaba DO.

John T. Nockleby plantea lo siguiente en la *Encyclopaedia of the American Constitution* [32] (cita original en inglés, traducción al español realizada por los autores de este trabajo):

«*Discurso de odio se considera usualmente a aquel que incluye expresiones de animosidad o menosprecio hacia un individuo o grupo basada en una característica compartida por un grupo como la raza, color de piel, origen nacional o étnico, género, invalidez, religión, u orientación sexual.*»

Elegimos comenzar citando esta "definición" ya que se puede encontrar en varios trabajos relacionados con el reconocimiento automatizado de DO. Es una definición con un trasfondo legal, acuñada por un autor con experiencia en el estudio del área.

También se añade frecuentemente a las condiciones de uso establecidas por la mayoría de las webs que contienen interacción entre sus usuarios (como las redes sociales). *Twitter* provee en sus reglas y políticas de conducta [38] una definición exhaustiva de qué se considera *hateful conduct* (conducta "odiosa") (cita original en inglés, traducción al español realizada por los autores):

«No deberás promover la violencia o atacar directamente o amenazar a otras personas en base a su raza, etnia, nacionalidad de origen, casta, orientación sexual, género, identificación de género, edad, discapacidad, o enfermedad grave. Tampoco permitimos cuentas cuyo propósito primario sea incitar daño hacia otros en base a esas categorías.»

El diccionario jurídico de la Real Academia Española [16] define como delito de provocación al odio lo siguiente:

«Delito que cometen quienes públicamente fomentan, promueven o incitan directa o indirectamente al odio, hostilidad, discriminación o violencia contra un grupo, una parte del mismo o una persona determinada por razón de su pertenencia a aquel, por motivos racistas, antisemitas u otros referentes a la ideología, religión o creencias, situación familiar, la pertenencia de sus miembros a una etnia, raza o nación, su origen nacional, su sexo, orientación o identidad sexual, por razones de género, enfermedad o discapacidad.»

Por último, el DO se puede encontrar definido en la mayoría de los documentos de carácter legislativo de cada país. En particular, el código penal uruguayo [8] define como "incitación al odio, desprecio o violencia hacia determinadas personas":

«El que públicamente o mediante cualquier medio apto para su difusión pública incitare al odio, al desprecio, o a cualquier forma de violencia moral o física contra una o más personas en razón del color de su piel, su raza, religión, origen nacional o étnico, orientación sexual o identidad sexual, será castigado con tres a dieciocho meses de prisión.»

Las cuatro definiciones son conceptualmente similares. En primer lugar, en todos los casos se plantea que la actitud percibida en el contenido del discurso debe ser de carácter violento, hostil o de menosprecio. También se puede decir que todas consideran que el DO puede estar dirigido hacia una persona o hacia un grupo de ellas. A su vez, especifican cuáles son las características atacadas por el discurso para que se considere de odio. La única diferencia notable entre estas definiciones son las características elegidas. Esto puede deberse a que el concepto de odio es muy amplio y es necesario tener una idea concreta de cuándo un discurso violento u hostil se considera DO.

Aunque no es lo usual, algunas publicaciones sobre detección de DO agregan una discusión sobre los aspectos que debe cumplir el texto analizado para ser considerado DO. Una de las primeras publicaciones sobre detección de DO (Warner et al., 2012 [39]) plantea que existen ciertos temas a discutir para entender bien el concepto. En primer lugar, la mención o incluso el alabo o promoción de personas u organizaciones asociadas al crimen no constituye en sí DO. Independientemente de si dicha persona u organización es autor frecuente de mensajes con DO, el hecho de mencionarla o promoverla sin que se trate de un ataque a otra persona o grupo de personas no cumple con las definiciones

de odio antes presentadas. Por otra parte los autores plantean que cuando se categoriza innecesariamente a un individuo como parte de un grupo (por ejemplo "negro ladrón"), hay una alta probabilidad de que se trate de DO. Por último, muestran que los epítetos raciales y algunos términos despectivos no siempre determinan si un texto contiene DO. El ejemplo que presentan es el siguiente:

«*Kike is a word often used when trying to offend a jew*»

El texto anterior explica que la palabra "kike" es usada para ofender a los judíos, pero el texto en sí no se trata de una agresión.

3.1.1. Discurso de odio y lenguaje ofensivo

Para estudios de carácter científico que requieran la participación de público inexperto, es necesario especificar aún más el concepto de DO. Los organizadores de la competencia SemEval 2019[1] (más adelante, en la sección 3.2.5 se detalla en qué consiste esta competencia) llevaron a cabo la anotación de un corpus de tweets con DO mediante *crowdsourcing* de las anotaciones. Para guiar a los participantes les proveyeron una serie de lineamientos[2] de modo que pudiesen reconocer cuando un tweet es DO (Basile et al., 2019 [2]). En particular plantean que el texto del tweet debe cumplir dos características:

- El tweet debe tener como objetivo principal el grupo al cual se dirige el DO y si es dirigido hacia un individuo, se debe asegurar la pertenencia del mismo a dicho grupo (el ataque no debe ser dirigido hacia sus características individuales o contener únicamente insultos genéricos).
- El mensaje debe propagar, incitar, promover o justificar odio o violencia hacia el objetivo, o debe apuntar a deshumanizar, lastimar o intimidar el objetivo.

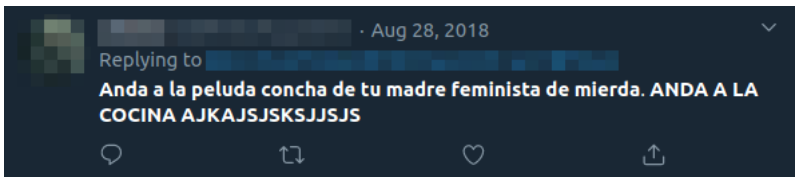
A modo de dejar aún más claro el concepto, agregan cuales pueden ser características del texto que no necesariamente implican DO, como lenguaje ofensivo o blasfemo, difamación, insultos hacia autoridades públicas o negación de hechos históricos.

Antes de proseguir, advertimos que a partir de aquí y hasta que termine esta sección se analizarán ejemplos con lenguaje que puede ser considerado agresivo o fuerte para el lector. Si lo desea, puede continuar hacia la sección 2.

[1] `https://aclweb.org/aclwiki/SemEval_Portal`

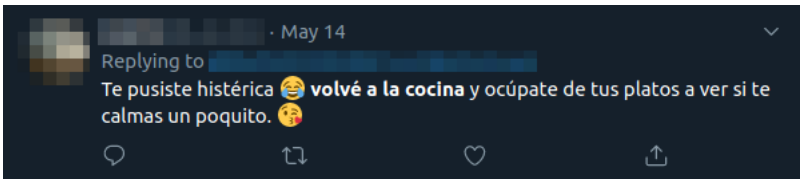
[2] `https://github.com/msang/hateval/blob/master/annotation_guidelines.md`

Tomemos el siguiente ejemplo:



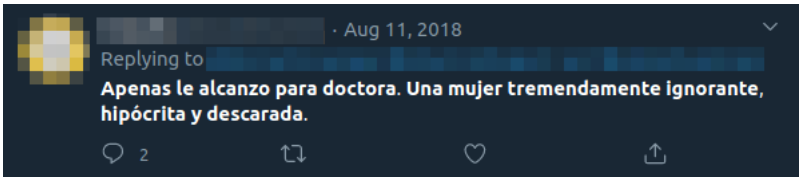
Es claro que el tweet contiene DO y contiene lenguaje ofensivo. Teniendo en cuenta las definiciones y los lineamientos citados anteriormente, se dirige hacia otra persona de género seguramente femenino de una manera despectiva, utilizando insultos y con ánimo de denigrarla, siendo el punto clave el uso de la frase "anda a la cocina". Esta expresión se utiliza muchas veces para reducir a la mujer al estereotipo de "ama de casa".

Veamos ahora un ejemplo más:



Este es un caso de tweet que no contiene lenguaje ofensivo pero que contiene DO. Sin embargo, aparece nuevamente el uso de la expresión "volve a la cocina", que implica como lo explicamos anteriormente una agresión hacia la mujer a la que le responde.

El siguiente es otro tweet interesante:



A simple vista puede parecer que el tweet contiene DO, ya que agrede a alguien (no presente en la conversación) explicitando que es mujer. Aunque no usa lenguaje grosero, la elección de palabras está orientada a incitar el odio a la persona que refiere, pero el objetivo del ataque no es su condición de mujer. Para entenderlo mejor podemos preguntarnos, ¿que pasaría si no fuera explícito que la persona a la que se dirige fuera mujer? ¿Tendría sentido el insulto? En este caso la respuesta es sí, pues las palabras "hipócrita", "ignorante" y "descarada" o "descarado" son insultos que no dependen del género de la persona a la cual son dirigidos. En los casos anteriores, el insulto cobra sentido sólo si el receptor es de género femenino.

Como último ejemplo, podemos citar un tweet que utiliza insultos fuertes pero no tiene DO:

El tweet es dirigido a una joven política que, en el momento en que se escribe este informe, es legisladora de la Ciudad de Buenos Aires. Podemos vernos tentados a decir que este tweet contiene discurso de odio, pero el autor en ningún momento hace uso de expresiones que denigren su condición de ser mujer, aún presentando una actitud violenta mediante insultos genéricos hacia ella.

Como se puede apreciar, el DO es difícil de identificar. En el transcurso del proyecto, se genera la discusión entre los participantes (alumnos y supervisores) sobre qué es exactamente DO. Es en esta instancia donde se logra apreciar que, a pesar de la noción por "instinto" que cada persona mantiene, no hay un contrato a nivel social que logre hacer de su identificación una tarea trivial. Todos nos vemos influenciados por el trasfondo social, nuestra moral y ética, círculos sociales en los cuales nos movemos, medios que consumimos, entre otros factores que moldean la percepción del DO.

En base a la investigación presentada, se decide acordar que entendemos por *discurso de odio* a cualquier expresión de agresión, animosidad, hostilidad y/o incitación al desprecio, violencia o discriminación hacia un individuo o grupo basado íntegramente en características que lo asocien con un grupo de personas que las comparten.

3.2. Trabajos previos

Habiendo comprendido qué es el DO y cómo se puede identificar en el lenguaje natural, y presentado los conceptos básicos que se manejan en la terminología del PLN y el aprendizaje automático, procedemos a establecer el estado del arte en el área de la clasificación de textos y en la detección de DO.

Si bien el lenguaje ofensivo como objeto de estudio desde el punto de vista del procesamiento de lenguaje natural y sistemas de información es un tema que se viene estudiando desde hace tiempo, el concepto de DO no había sido muy explorado hasta años recientes. A partir de entonces, gracias a los avances en el área de aprendizaje automático y la popularización de las redes sociales, entre otros aspectos, se han realizado una gran cantidad de publicaciones y aportes relacionados al tema. Es importante notar que la gran

mayoría de los trabajos previos tienen como lenguaje objetivo el inglés, y son escasas las publicaciones sobre investigación de DO para el lenguaje español. Aunque encontramos valioso analizar trabajos para cualquier idioma, es necesario tener precaución a la hora de intentar trasladar los resultados al idioma español, ya que las características del lenguaje en general influyen en el desempeño de las distintas técnicas utilizadas.

3.2.1. Primeros trabajos

Uno de los primeros trabajos relacionados a la detección de DO, que sirvió como base para muchos posteriores, es el realizado por Dawei Yin (Yin et al., 2009 [43]). Se centra en la detección de acoso en comunidades online, como foros o chats. Aunque los conceptos DO y acoso no son exactamente iguales, es claro que existe cierta relación entre ellos, y este trabajo brinda uno de los primeros resultados relevantes para el área. Los autores plantean la tarea como un problema de clasificación binario donde lo que se desea es saber si el mensaje a clasificar tiene como intención molestar o acosar a una o más personas. La solución propuesta es un modelo de clasificación supervisado que utiliza una SVM aplicada a *features* locales, sentimentales y contextuales extraídas de los mensajes. Todos los resultados obtenidos presentan una medida *F-score* menor a 0.5, siendo los mejores aquellos en los que se utilizan los 3 tipos de *features* en conjunto.

Como se mencionó previamente, Warner y Hirschberg en el año 2012 [39] llevan a cabo uno de los primeros trabajos de identificación de DO. Comienzan por caracterizar el concepto de DO, planteando varias definiciones para éste y presentando ejemplos típicos. Luego aplican ese conocimiento para escribir una guía de anotación de corpus y con la ayuda de tres anotadores construyen un *dataset* de 1000 párrafos en base a textos de 452 páginas web en inglés que previamente habían sido etiquetadas como ofensivas. Los párrafos podían ser anotados con cero o más de los siguientes tipos de odio:

- Hacia judíos
- Hacia afrodescendientes
- Hacia mujeres
- Hacia asiáticos
- Hacia musulmanes
- Hacia inmigrantes

Por último, utilizan el conjunto de datos generado para crear un modelo de aprendizaje automático capaz de detectar DO. Debido a la cantidad y distribución de los datos, los autores deciden centrarse únicamente en la detección de odio hacia judíos. Las *features*

son elegidas por medio de la estrategia basada en *templates* de Yarowsky [42], utilizando ventanas de palabras, *POS tags* (también conocidas como etiquetas gramaticales) y *Brown Clusters* (Brown et al., 1992 [4]). El clasificador elegido es nuevamente una SVM, al cual se le aplican distintas técnicas de ajuste de parámetros mediante validación cruzada. En cuanto a los resultados finales, a pesar de mejorar el acierto aproximadamente por un 3 % con respecto a la línea base, el *recall* del clasificador es bastante bajo y por lo tanto se concluye que existen patrones lingüísticos no detectados por el modelo.

3.2.2. *Word embeddings* y el contexto

El surgimiento de *word embeddings* generados mediante el uso de algoritmos de aprendizaje automático no supervisados, en especial los basados en redes neuronales (como los implementadoes en *word2vec* (Mikolov et al., 2013 [29]), constituyen una mejora considerable en el desarrollo de modelos de clasificación. En el 2015, Djuric et al. [15] utilizaron regresión logística para clasificar tweet con discurso de odio a partir de representaciones de tweet obtenidas con *paragraph2vec* (Mikolov et al., 2014 [26]). De manera similar a *word2vec*, *paragraph2vec* obtiene vectores para textos de largo variable mediante el entrenamiento de un modelo CBoW. Los autores entrenaron el modelo con un corpus de 951.736 comentarios extraídos de *Yahoo!* Finance, reportando los resultados del clasificador en términos del área bajo la curva para luego compararlos con los resultados obtenidos utilizando BoW y TF-IDF, obteniendo una mejora de un 1.5 % para en comparación con el primero y de un 15.5 % en relación con el segundo.

En el 2016, C. Nobata et al. [31] continúan la investigación basándose en todos los aportes anteriores para generar un detector de lenguaje abusivo (incluyendo DO, lenguaje despectivo y blasfemia) que supera los resultados de los clasificadores del estado del arte del momento. Los *dataset* utilizados constan de comentarios extraídos aleatoriamente de la sección de finanzas y noticias de la web *Yahoo!*. Éstos son anotados por un conjunto de empleados de la misma empresa *Yahoo!*, quienes recibieron entrenamiento previo para realizar la anotación. Además, los autores investigan el uso de *crowdsourcing* como herramienta alternativa para la anotación del corpus. Tras experimentar con la plataforma *Mechanical Turk* de *Amazon*, concluyen que el acuerdo entre anotadores es considerablemente más bajo utilizando *crowdsourcing* comparado con la anotación dentro de la empresa (0.843). El tamaño total del corpus de entrenamiento, sumando los 3 conjuntos de datos utilizados es de alrededor de 4 millones de comentarios. Las atributos con las que trabajan son n-gramas de caracteres, *features* lingüísticas (e.g. largo promedio de las palabras, cantidad de signos de pregunta/exclamación) que permiten distinguir el uso de palabras tanto abusivas como educadas, atributos sintácticos como POS-tags y relaciones de dependencia, y atributos semánticos, haciendo uso de técnicas de *word embeddings*

como las mencionadas anteriormente. El clasificador elegido es el modelo de regresión logística ofrecido por la librería de código abierto *Vowpal Wabbit*. Los resultados muestran que la combinación de todos los atributos superan en rendimiento al uso de cada atributo por separado.

3.2.3. El problema del lenguaje ofensivo

En el año 2017, Shervin Malmasi y Marcos Zampieri [27] introducen un nuevo desafío al problema de identificar DO: el uso de lenguaje ofensivo. A partir de lo propuesto por Davidson et al. [11], se propone crear una línea base para conocer qué tanto influye el lenguaje ofensivo en el resultado de la clasificación. El corpus utilizado para las pruebas realizadas (generado en Davidson et al. [11]), el cual contiene 14509 tweets en inglés de los cuales 2399 tienen DO, 4836 tienen lenguaje ofensivo y los restantes no contienen ni DO ni lenguaje ofensivo. Los autores entrenan 14 modelos diferentes en donde todos utilizan un modelo SVM, variando la manera en la que se computan los vectores para cada tweet usando 4 métodos: *skip-gram*, *Bag-of-Words* de n-gramas de palabras, *Bag-of-Words* de n-gramas de caracteres y una combinación de todos los anteriores. Reportan los resultados en términos del acierto de cada modelo, obteniendo el mejor resultado mediante la utilización de 4-gramas de palabras. Presentan los resultados en términos de una matriz de confusión donde se nota que, en efecto, el clasificador usualmente confunde tweets ofensivos pero sin DO con tweets que contienen DO.

3.2.4. Primeras aplicaciones de las redes neuronales

Los recientes avances en el aprendizaje profundo han realizado un gran aporte al avance en el área de la clasificación de texto. En particular, el uso de redes recurrentes ha demostrado ser uno de los modelos con mejores resultados. En el 2017, un grupo de investigadores del *International Institute of Islamic Thought* (Instituto Internacional del Pensamiento Islámico) publican uno de los primeros trabajos (Pinkesh Badjatiya et al., 2017 [1]) que utilizan redes neuronales para la detección automática de DO. El trabajo presenta primero un conjunto de modelos que sirven como línea base a partir de la que se comparan los modelos de clasificación.

La arquitectura propuesta para el modelo clasificador consta de una capa de *embeddings* inicializada utilizando 2 métodos diferentes, combinando cada uno de estos con 3 clasificadores basados en redes neuronales. No se detalla en profundidad la arquitectura de los clasificadores, pero se menciona que uno está formado por una CNN, otro por una LSTM y el último utiliza el modelo provisto por la librería *fastText* (Armand Joulin et al., 2016 [21]). La utilización de una capa de *embeddings* permite realizar un ajuste o *fine-*

tuning de éstos, obteniendo representaciones que pueden funcionar mejor para la tarea de clasificación específica.

Los autores entrenan los modelos sobre un conjunto de 16000 tweets en inglés (de los cuales 3383 son sexistas y 1972 racistas) presentando un *f-score* de 0.839 obtenido a partir de un modelo de CNN e inicializando la capa de *embeddings* con vectores pre-entrenados de GloVe (Jeffrey Pennington et al., 2016 [33]), una técnica que computa representaciones vectoriales de las palabras a partir de la probabilidad de que ocurran en el mismo contexto.

El trabajo de Biere [3] en el año 2018 continúa con la exploración de los métodos de aprendizaje profundos aplicados al PLN, creando un clasificador basado en una red neuronal convolucional para clasificar tweet en tres categorías mutuamente excluyentes: odio, lenguaje ofensivo y ninguna. Se trabaja con la definición de Nockleby [32] y se remarca la diferencia entre DO y lenguaje ofensivo. El *dataset* utilizado fue el "Hate Speech Identification Dataset" de la plataforma de anotación *Figure Eight*, que contiene 24.783 tweets en inglés, de los cuales 5 % contenían odio y 77 % contenían lenguaje ofensivo sin odio. Éste es dividido en las siguientes proporciones: 50 % entrenamiento, 30 % validación y 20 % test. En la fase de preprocesamiento se remueven caracteres no deseados, se divide el texto en *tokens*, y a las palabras se les aplica *stemming* y se pasan a minúsculas. El modelo utilizado es el mismo que Kim (2014 [24]) para clasificación de oraciones, que consiste de una capa convolucional no lineal, una capa de *max-pooling* y una de salida compuesta por una única unidad con función de activación *softmax*. La entrada esta compuesta por *embeddings skip-gram* de 300 dimensiones, pre-entrenados con el conjunto de datos de 3.000 millones de palabras de *Google News*[3], utilizando la implementación de *word2vec*. Durante la validación se observa que el modelo tiende a sobreajustarse a los datos. Una razón posible que plantea el autor es el pequeño tamaño del conjunto de datos. Los resultados obtenidos son buenos, logrando un *F-score* de 90 %, aunque una gran cantidad de tweet sin DO fueron clasificados de forma errónea como tweet con DO. El autor identifica el desbalance del conjunto de datos como uno de los factores que podría estar influyendo de manera negativa en los resultados, debido a que podría estar generando un sesgo a favor de los tweet con lenguaje ofensivo. Finalmente, se propone como trabajo futuro intentar diferenciar la entidad hacia la que se dirige el odio y explorar el uso de *ensemble methods* (i.e. combinar más de un modelo de clasificación) para esta tarea, ya que los resultados obtenidos utilizándolos en otras áreas son buenos.

[3] `https://github.com/mmihaltz/word2vec-GoogleNews-vectors`

3.2.5. Competencias de análisis semántico

Los avances en el área del PLN junto con el crecimiento de la comunidad asociada a ésta, ha dado lugar al surgimiento de competencias que concentran esfuerzos de equipos de investigadores provenientes de todas partes del mundo para resolver problemas conocidos del PLN. Una de las más populares es el *International Workshop on Semantic Evaluation*, mejor conocido como *SemEval*[4]. Habiendo abarcado 15 ediciones al momento de la realización de este trabajo, SemEval consiste en la resolución de problemas de PLN (como clasificación de texto, extracción de información, entre otros) orientados a un dominio en particular. La competencia es organizada en *tasks* o tareas, cada una correspondiente a un problema de PLN diferente. A su vez, cada tarea puede estar dividida en subtareas que proponen variantes en el problema principal a resolver. Para cada tarea, se hace público un conjunto de datos que se debe utilizar para resolver el problema planteado.

El trabajo realizado por Juan Manuel Pérez et al. [34] en el año 2019, presenta un conjunto de modelos que resuelve la subtarea principal de la tarea *hatEval* en español, orientada a la detección de DO hacia inmigrantes y mujeres. En la fase de preprocesamiento del tweet, se plantean dos metodologías: la primera es básica, se *tokeniza* el tweet, se reemplazan las menciones, URLs, e-mails y cadenas conformadas únicamente por letras repetidas por tokens especiales; la segunda, orientada al análisis de sentimiento, propone llevar todas las palabras a minúsculas, remover puntuación y *stopwords*, lematizar las palabras y agregar, cuando hay una palabra de negación que los precede, el prefijo `NOT_` a los 3 tokens posteriores o hasta que se encuentre un *token* que no corresponda a una palabra.

El mejor modelo, que consiste en un clasificador SVM que recibe una combinación de los vectores *Bag-of-Words* y *Bag-of-Characters* con pesos TF-IDF más vectores *skip-gram* construídos a partir de *fastText*, obtiene un *f-score* de 0.73 entrenado y evaluado sobre el conjunto de evaluación y entrenamiento propuesto para la tarea, otorgándoles el primer puesto en la competencia.

3.2.6. Detección de discurso de odio en la actualidad

Para culminar este capítulo, nos interesa mencionar uno de las últimos trabajos publicado unos meses después del comienzo de éste proyecto, que surge bajo la colaboración entre la Oficina Nacional de la Lucha contra los Delitos de Odio en España y las Universidades de Madrid y Cardiff. En este trabajo (Pereira-Kohatsu et al., 2019 [36]), se realizan distintas contribuciones, destacándose un conjunto de datos de 6000 tweet anotados según tienen discurso de odio o no, una detallada comparación de clasificadores construidos

[4] `https://semeval.github.io/`

combinando distintos modelos y métodos para representar textos y un sistema para la detección y monitorización de DO en *Twitter*, actualmente utilizado por el Ministerio del Interior español, denominado *Haternet*. Este sistema es un claro ejemplo de la utilidad de la detección de DO en la realidad actual.

Se puede apreciar cómo el área de la detección de DO ha tenido una evolución notable desde los primeros trabajos que tomaban en cuenta lenguaje ofensivo o abusivo en general, hasta los mas recientes, donde se plantea detectar DO tal como lo definimos en la sección 3.1. Gran parte de esta evolución deriva de los avances en el desarrollo de modelos utilizados en la clasificación de texto y los nuevos métodos de generación de representaciones vectoriales que surgen año tras año. También impulsa al estudio de esta área la relevancia que ha ganado el DO como fenómeno social en los años recientes, y la necesidad de regularlo en el ámbito de las redes sociales.

4 | Generación del corpus

Uno de los puntos ampliamente discutidos en el área del aprendizaje automático es cuáles son las características que debe cumplir un conjunto de datos para obtener resultados confiables al aplicar un método de aprendizaje automático que resuelva un problema utilizándolo. En general, depende del problema que queremos resolver. En el caso considerado en este trabajo pretendemos principalmente determinar si un texto tiene DO, lo que corresponde a un tipo de problemas se denomina, en el contexto del PLN, **clasificación de texto**.

Cuando nos enfrentamos a la construcción de modelos mediante métodos de aprendizaje automático, el tamaño que debe poseer el conjunto de datos es uno de los principales problemas. En general, se busca tener una gran cantidad de ejemplos diferentes para lograr una muestra que sea representativa de la realidad y tratando de minimizar tanto el ruido como el sesgo. El problema de cuántos datos son necesarios para entrenar un modelo de aprendizaje automático ha sido estudiado desde su surgimiento. Ian Goodfellow et al. [19] afirman que, para un problema de clasificación, si elegimos modelos de aprendizaje profundo un conjunto de datos con por lo menos 5000 ejemplos por categoría demuestra lograr resultados aceptables y llega a equiparar o mejorar resultados obtenidos por seres humanos si el tamaño supera los 10 millones de ejemplos. En el curso *Machine Learning Crash Course*[1] publicado por Google, se recomienda que el tamaño del conjunto sea un orden de magnitud mayor que la cantidad de parámetros del modelo que se quiere entrenar. La dificultad de obtener una gran cantidad de datos depende del problema a abordar. En primer lugar, el acceso a los datos de la realidad no siempre es sencillo, depende de las fuentes disponibles y si éstas tienen un acceso restringido o no. Otro factor importante es qué tipo de algoritmo pretendemos aplicar. Si es no supervisado, con sólo anotar los datos para evaluación es suficiente, pero para algoritmos supervisados se añade la dificultad de que se deben anotar todos los datos obtenidos. Otra característica que se considera importante es que la anotación sea lo más confiable posible, esto es, que las etiquetas se encuentren acertadas dadas las características que debe cumplir un ejemplo

[1] `https://developers.google.com/machine-learning/crash-course`

para pertenecer a la clase correspondiente a la etiqueta. Si la anotación de los ejemplos es realizada por dos o más anotadores, también es deseable evaluar su acuerdo mediante una método que mida el puntaje de acuerdo entre anotadores. Con este valor, se puede determinar que tan confiables son las métricas obtenidas por el modelo al entrenarlo con el conjunto de datos.

Si el modelo elegido corresponde a uno de aprendizaje supervisado, podemos resumir entonces que un tamaño adecuado y una alta confiabilidad de las anotaciones son características que siempre se buscan en un conjunto de datos anotados, pero lograr que ambas lleguen a un nivel alto no es trivial. A veces se debe evaluar la priorización de aquellas que tengan un impacto más positivo en el desempeño del modelo. Como característica adicional que a veces se busca en los conjuntos de datos anotados es el balance de clases, i.e. que cada clase contenga una cantidad de ejemplos lo más equitativa posible. Dependiendo del caso trabajado y de cuál sea el resultado que se quiere obtener, esto puede impactar de manera negativa en el modelo, por ejemplo, si se entrena un clasificador con un conjunto desbalanceado puede provocar que el desempeño sea malo. Aunque estos atributos sean valiosos para entrenar adecuadamente un modelo de aprendizaje automático, existen estrategias para contrarrestar el impacto negativo que se pueda tener si no se cumplen con ellos.

Los trabajos sobre detección de DO en los que ahondamos se centran principalmente en redes sociales, lo cual tiene sentido en cuanto a que la información de estas redes es muchas veces de fácil acceso y de gran disponibilidad. Como hemos establecido, en éste trabajo elegimos como caso de estudio la red social *Twitter*. Muchos usuarios de esta red la utilizan como blog personal, aunque también pueden encontrarse cuentas de medios que la aprovechan para divulgar noticias, artistas que divulgan su contenido, cuentas de instituciones que buscan aumentar la interacción con sus integrantes, entre otros tipos de cuentas. Nuestra elección se fundamenta en que es una de las redes sociales con uso más extenso en la actualidad, lo que implica una fuente diversa de textos. En ella encuentran su voz todo tipo de personas y comunidades, lo que añade una riqueza importante a los textos que se pueden encontrar ahí, y en nuestra experiencia es bastante común encontrar el fenómeno de los usuarios *troll*, denominación adjudicada a aquellos que se dedican a provocar o agredir con el único propósito de incitar una reacción negativa en el receptor. También agrega una particularidad al problema, que es el largo de los textos, puesto que *Twitter* tiene un límite de 280 caracteres para los mensajes que se pueden publicar en la plataforma (originalmente 140 caracteres). La plataforma cuenta con una API[2] web gratuita que permite la búsqueda y obtención de tweet de distintas maneras, provee velocidades de respuesta rápidas pero en su versión Standard restringe su acceso a un

[2] `https://developer.twitter.com/en/docs`

máximo de 180 consultas en un período de 15 minutos, requiriendo el acceso pago a otras versiones para levantar esta restricción.

A partir de esta premisa, comenzamos por buscar corpus disponibles en la web, a modo de establecer un "*estado del arte*" en cuanto a corpus disponibles y las características que poseen. El punto de partida para nuestra búsqueda fueron las competencias de evaluación de sistemas de análisis lingüístico computacional (como la ya citada *SemEval*). En particular, buscamos aquellas que proponían objetivos que se centraran en problemas relacionados con la identificación de DO. En ésta búsqueda encontramos que restringirla al idioma español añade dificultad al problema, pues la mayoría de los conjuntos de datos disponibles en la red corresponden al idioma inglés. Esto no es de extrañar, dado que es el idioma dominante tanto en el área de tecnologías de la información como en la internet en general. En el transcurso de esta investigación, encontramos un total de 2 corpus de publicaciones en *Twitter* en español.

A los efectos del trabajo abordado en esta investigación, nos propusimos como objetivo crear un corpus desde cero. Por un lado, vimos necesaria la experiencia de enfrentarnos a esta tarea; consideramos que lidiar con las dificultades y aprender de la experiencia podría ser una fuente importante de aprendizaje. También vimos valiosa la posibilidad de aportar a la comunidad del PLN. El corpus disponible para la tarea 5 del *SemEval* (*HatEval*) fue el *gold-standard* adoptado para construir nuestro corpus, éste cumple con dos características de las mencionadas anteriormente: balance y confiabilidad en las anotaciones, aunque según la literatura consultada no tiene un tamaño aceptable.

4.1. Obtención de tweets

Para la obtención de los tweets, nos inspiramos en la estrategia de recolección utilizada para la construcción del corpus de *hatEval* [2], la cual plantea obtenerlos de tres maneras:

- Buscar tweets dirigidos a cuentas de víctimas potenciales de DO.
- Recolectar los tweets de cuentas que profesan DO.
- Utilizar palabras clave o una combinación de ellas para buscar tweets que las contengan.

De estas estrategias, la que más aportó al conjunto obtenido fue la utilización de palabras clave. Las otras opciones fueron desestimadas pues consideramos que el esfuerzo necesario para llevarlas a cabo no hubieran aportado un gran valor en comparación con la búsqueda mediante palabras. En particular encontramos difícil mantener una lista de cuentas que incurren en el DO, ya que *Twitter* impone actualmente los términos y

condiciones por los que muchas de ellas eran vetadas poco tiempo después de que las encontrábamos.

La interacción con la API de *Twitter* se llevó a cabo con la librería para Python *Twitterscraper*[3]. Esta ofrece un *wrapper* alrededor de la API Standard, que permite evitar las restricciones que impone sobre la cantidad de consultas haciendo uso de servidores *proxy* disponibles gratuitamente en la web.

En la documentación de la API se pueden encontrar distintos operadores que permiten la construcción de búsquedas complejas. En el caso trabajado, se utilizaron los operadores OR (el cual busca tweets donde ocurren el subtérmino de la izquierda o el de la derecha o ambos, en orden aleatorio) y AND (el cual busca tweets donde ocurren tanto el subtérmino de la izquierda como el de la derecha, en orden aleatorio).

Además, a modo de establecer el alcance de la investigación, se eligió buscar tweets que contengan DO de tipo:

- misógino (hacia mujeres)
- homofóbico (hacia orientaciones sexuales diferentes la heterosexual)
- racista (hacia etnias, razas o colores de piel)
- ideológico (hacia corrientes de pensamiento)

Luego, para elegir las palabras que participaron en la búsqueda, generamos las siguientes listas:

- *hateful_words*: contiene términos de una o más palabras que seguramente implican DO hacia el grupo víctima (como *ramera* en el caso de DO misógino).
- *aggresive_words*: contiene términos de una o más palabras que no necesariamente implican DO pero son agresivos y se identifican con un grupo víctima de DO (como *puto* para el caso de DO homofóbico).
- *dependent_words*: contiene términos de una o más palabras representativos del grupo víctima que generalmente no son usadas en un contexto de DO (como *negro* para el DO racista).

El listado completo de palabras se encuentra en el anexo 10.1. Además se mantuvo una lista de insultos de alcance general. A partir de estas listas de palabras, se generaron los términos con las siguientes estrategias de combinación:

- Utilizando las palabras de la lista *hateful_words* individualmente.

[3] `https://github.com/taspinar/twitterscraper`

- Combinando palabras de la lista *dependent_words* con insultos.
- Combinando palabras de la lista *aggresive_words* con insultos.
- Combinando palabras de la lista *aggresive_words* con palabras de la lista *dependent_words*.
- Combinando palabras de la lista *aggresive_words* consigo mismas.

Adicionalmente, en cada caso excluimos combinaciones de términos repetidos.

La estrategia presentada está pensada para encontrar tweets que puedan contener combinaciones que impliquen discurso de odio en el contexto del tweet.

A partir de un intercambio realizado con Valerio Basile, uno de los organizadores de *SemEval*, surge la recomendación de que el rango de fechas elegido para la recolección no sea considerado ligeramente, puesto que si se elige un período corto un evento que ocurra durante éste puede influir en la variedad de los tweets obtenidos. Aprovechando que la librería utilizada permite elegir un rango de fechas al que se limita la búsqueda de tweets, elegimos aquellos publicados durante un período de un año, desde el 14 de Julio de 2018 hasta el 14 de Julio de 2019.

Con estos parámetros, se recolectaron alrededor de 60000 tweets, de los cuales, en un inicio, se tomaron 50000 para ingresar en la aplicación de anotación. Avanzada la publicación de ésta, intercambiamos 10000 de los que no se habían votado aún por tweets recolectados aleatoriamente (es decir, sin utilizar la estrategia original) para lograr obtener más tweets que no tuvieran DO en el conjunto final.

4.2. Aplicación de anotación

Con el objetivo de acelerar la generación del corpus, se decide abrir su anotación al público general. Para ello se desarrolla una aplicación web[4] que presenta los tweets extraídos a los usuarios y les permite decidir cuáles de ellos tienen DO y qué tipo de DO contienen. La página web se desarrolla tomando como base la creada para el proyecto de grado de clasificación de humor[5] de Santiago Castro y Matías Cubero, a la cual se le agregan distintas funcionalidades, adaptándola al contexto de odio y a los requerimientos del proyecto. En la figura 4.1 se muestra la página de inicio, como la ve el usuario al ingresar al dominio.

El flujo principal de la aplicación es el siguiente:

[4] `https://odioelodio.com`
[5] `https://clasificahumor.com/`

Figura 4.1: Página principal

1. Se le presenta un tweet al usuario y éste selecciona una de las opciones que determina si el tweet contiene discurso de odio y/o lenguaje ofensivo. Este paso se repite 3 veces con tweets distintos.

2. Si el usuario indicó que algún tweet contiene discurso de odio, se vuelve a mostrar el mismo tweet y el usuario selecciona una de las opciones que determinan el tipo de odio que contiene el tweet (ver figura 4.2).

3. Se regresa al paso 1.

En cualquier momento durante el paso 1 el usuario puede seleccionar la opción "saltear" para obtener un nuevo tweet en lugar del actual.

El uso de crowdsourcing agiliza el proceso de anotación pero también introduce ciertas dificultades que debemos afrontar. Discutiremos cuáles son estos problemas y qué medidas son tomadas para moderar su impacto.

En primer lugar, no todas las personas se encuentran familiarizadas con el término "discurso de odio" o con las definiciones particulares con las que trabajamos. Esto pone en riesgo la calidad de la anotación ya que no contamos con un método para asegurarnos de que los usuarios que anotan se encuentren bien informados. Para intentar evitar estos casos, se añade a la página un botón que despliega un modal con algunas de las definiciones de DO presentadas en la sección 3.1.

Figura 4.2: Anotación de tipo de odio

Otro problema que encontramos es que los preconceptos adquiridos por cada anotador, sumados a la ambigüedad introducida por la falta de contexto de los tweets, llevan a errores en la anotación y al desacuerdo entre los anotadores, aún cuando todos ellos conocen y entienden el concepto de DO correctamente. Como contramedida decidimos elaborar una breve guía de anotación con ejemplos, la cual también hicimos accesible desde la página (como se muestra en la figura 4.3), cliqueando en el botón de ayuda.

Una característica importante de la anotación abierta al público es que generalmente no es viable que todos los anotadores clasifiquen todos los tweets. Tampoco es deseable que cada tweet sea anotado por un solo usuario ya que, por los problemas anteriormente mencionados, la anotación puede no ser correcta. Se decide entonces elegir un número que determina la cantidad máxima de anotaciones (a las que llamamos votos) que un tweet puede recibir antes de considerarse completamente anotado. Para obtener la etiqueta final del tweet se cuenta la cantidad de votos pertenecientes a cada clase y se elige la clase con más votos. De esta forma queda definido un parámetro que nos permite regular cuánto esfuerzo de los usuarios es dedicado a una mejor calidad del corpus, en oposición a una mayor cantidad de tweets anotados. Originalmente este parámetro es fijado en 5 votos por tweet, pero debido a la caída de actividad de la página web a lo largo del tiempo, es necesario reducirlo primero a 3 y finalmente a 2 para lograr la cantidad esperada de tweets anotados. También se exploran distintas estrategias para la selección de los tweets que se les muestra a los usuarios. Inicialmente, los tweets se eligen aleatoriamente, priorizando aquellos que ya han sido anotados por algún usuario, de forma de completar rápidamente los votos de los que no han sido completados aún. Finalmente se decide que los tweets

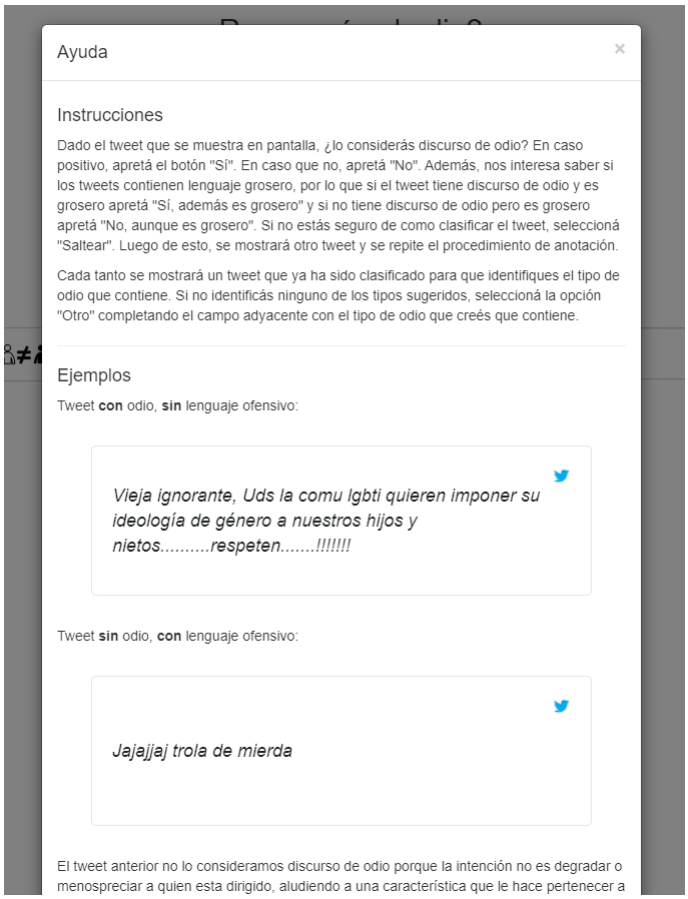


Figura 4.3: Guía de anotación

sean elegidos aleatoriamente, alternando entre los que ya han sido anotados y los que aún no han sido anotados por ningún usuario. Esto permite aumentar la cantidad de tweets votados, aunque también aumenta la cantidad de tweets que aún no cuentan con la cantidad máxima de votos.

4.3. Estadísticas y análisis de la votación

En esta sección se presentan las estadísticas de los votos obtenidos a partir de la publicación de la aplicación el 14 de mayo de 2019, hasta su extracción desde la base de datos el 10 de mayo delo 2020. También se presentan conclusiones que se pueden extraer a partir de su observación.

4.3.1. Cantidad de votos

En la tabla 4.1 se muestra un resumen de la cantidad de votos obtenidos para las clases *Contiene DO* y *Ofensivo*

	Ofensivo	No ofensivo
Odio	2114	4800
No odio	1684	6367
Total	14965	

Tabla 4.1: Resumen de los votos obtenidos para las clases *Contiene DO* y *Ofensivo*

Analizando una muestra de 50 tweets que fueron votados positivamente sobre si presentan DO pero negativamente en cuanto a si contienen lenguaje ofensivo, notamos que muchos claramente presentan este último, lo que nos lleva a concluir que posiblemente no se haya hecho suficiente énfasis en que también era importante votar los tweets según sean ofensivos o no.

En la tabla 4.2 se muestra la cantidad de votos por cada tipo de odio:

Ideológico	Racista	Homófobo	Misógino	Otro
1571	762	715	567	235
Total de votos: 3850				

Tabla 4.2: Resumen de los votos obtenidos para los distintos tipos de odio

Notar que la cantidad total de estos votos es menor que la observada en la tabla 4.1, puesto que existe la posibilidad de que un tweet con DO nunca sea votado según el tipo que contiene (por cómo se diseño el proceso de anotación). De los votados en la categoría *Otro*, en 150 no se especifica que tipo de odio tiene. En los que si se especifica, la mayoría fueron votados como xenófobos. Otras elecciones comunes fueron "clasista", "hacia gordos" o una combinación de distintos tipos. Analizando muestras de 50 tweets para las clases *Racista*, *Misógino* e *Ideológico*, vimos que en general la anotación es acertada. En cuanto a la clase *Homófobo*, observamos que muchos de los tweets que utilizan el insulto *puto* sin indicios de que la persona a la que es dirigido sea homosexual fueron votados como pertenecientes a esta clase. A raíz de este problema como el de la cantidad importante de tweets ofensivos que no fueron votados como tal, deducimos que no fue usual entre los usuarios de la aplicación acceder a la sección de ayuda. Esto pudo haber sido causado por la falta de hincapié en que los usuarios la accedan, o por el hecho de que el botón para acceder a ésta no se encuentra lo suficientemente a la vista o no es lo suficientemente llamativo. También es posible que muchos usuarios hayan accedido a esta sección, pero desistido de leerla con atención por su carga textual. Por otro lado, desde que se comienza a implementar este método para anotar el corpus, se busca que la aplicación sea accesible y que no abrume al usuario con un exceso de información, y es debido a esto que no se consideran estrategias que obliguen al usuario a leer la sección de ayuda en su totalidad

antes de comenzar con el proceso de votación. Considerando los resultados finales, una mejora que mitigue este suceso es dividir la sección de ayuda de manera que la guía de anotación se encuentre en una sección diferente y buscar al mismo tiempo que esta sea lo más simple posible, para poder hacer más énfasis en que sea leída antes de comenzar a votar.

Otra estadística considerada interesante para el análisis de la votación es la cantidad de tweets con votación *ambigua*. Se consideran de este tipo aquellos tweets cuya cantidad de votos para cada clase es igual o se diferencia únicamente por un voto. Del total de tweets se identifican 1746 pertencientes a esta clase. De estos tweets identificamos varios problemas que dieron lugar a la ambigüedad. En principio, encontramos que muchos tweets tienen lenguaje que se puede interpretar como DO pero que en algunas regiones de habla hispana no necesariamente lo implica, por ejemplo, el siguiente tweet:

«*Que sufriedera tan marica #ColSeBailaAQatar*»

Este tweet utiliza la palabra *marica* que, principalmente en Colombia y Venezuela, es usada para transmitir molestia u enojo, pero en otras regiones es comúnmente un insulto hacia una persona homosexual. En este caso, el usuario que publicó el tweet quiere expresar su sentir al sufrir viendo un partido de fútbol de la selección colombiana. Como mencionamos anteriormente, también influye no presentar información que contextualice los tweets, el siguiente ejemplo ilustra este problema:

«*Andas de más puto negro*»

El tweet fue votado 3 veces como conteniendo DO, en donde ambas veces fue identificado como racista, y 2 veces como no conteniéndolo. El tweet puede haber sido publicado con ánimo de denigrar a alguien por ser afrodescendiente y/o homosexual, sin embargo, si lo vemos en su contexto es un intercambio entre amigos y, aunque utilicen términos fuertes o que pueden ser considerados insultantes, no incurre en DO como lo definimos en esta investigación. Este problema fue de los más identificados por los usuarios que respondían a las publicaciones que realizábamos para promocionar la aplicación, pero desde un principio se desestima agregar contexto puesto que el objetivo es identificar DO sin utilizar más información que la proveniente del texto.

Otros tweets son ininteligibles aún teniendo contexto, como el siguiente:

«*Como vas a subir una foto del colo ré la concha d tu madre chino me kere matar*»

De este tweet no es posible ni siquiera conocer a que esta haciendo referencia. Además, no está publicado como respuesta a alguien, por lo que el contexto no serviría para desambiguarlo.

Se esperaba que los tweets ambiguos fueran salteados para poder identificarlos y removerlos, pero ninguno de estos tweets fue salteado una cantidad de veces que lo hiciese resaltar del resto. Para atacar estos casos, se eligió anotar manualmente parte de los tweets ambiguos.

4.3.2. Estadísticas de *Google Analytics*

Según las estadísticas recogidas por *Google Analytics*[6], un total de 894 usuarios ingresaron a la aplicación. En cambio, se registraron 800 anotadores en la base de datos que almacena los votos. Una posible explicación para esta diferencia es que los usuarios adicionales que reporta *Google Analytics* fueron *webcrawlers* (esto es, programas orientados a recoger automáticamente información de la red) o usuarios que ingresaron a la aplicación pero no votaron tweets. En la figura 4.4, se pude observar el número de usuarios que ingresaron cada día durante septiembre y octubre del 2019 (se elige este rango de fechas debido a que la gráfica completa no cabía en este espacio, la cual se puede encontrar en el anexo 10.4).

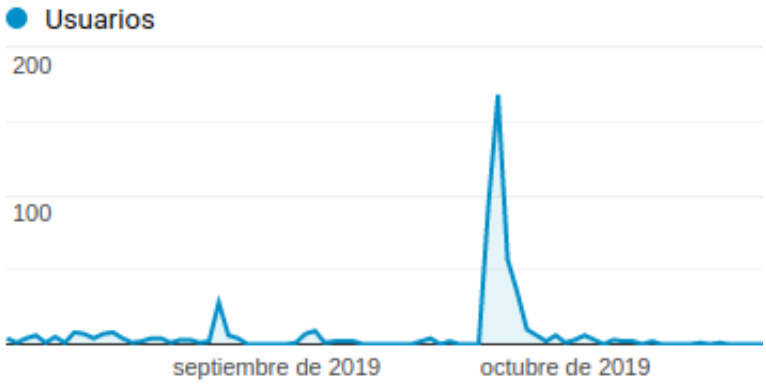


Figura 4.4: Usuarios que ingresaron a la aplicación cada día, durante setiembre y octubre del 2019

En la gráfica se puede notar los picos de tránsito por la aplicación. El primero, ubicado en septiembre de 2019, alcanza los 28 usuarios y corresponde con la colaboración de Valerio Basile en la difusión. El segundo pico, llegando a los 168 usuarios, surge de haber realizado publicaciones en *Facebook* y *Twitter*. Estos picos de actividad tienen una duración corta, en general no más de 10 días hasta que la actividad decae a cero. A su vez, *Google Analytics* reporta una duración media de la sesión de 1 minuto con 18 segundos. Para contextualizar esta métrica, se midió la cantidad de tweets en un período de 10 segundos que pueden clasificar los autores, resultando en un promedio de 1 tweet por período. De esta manera podemos deducir que, en promedio, los usuarios clasifican no más de 12 tweets por sesión. Esto refleja un problema en cuanto a la capacidad de la aplicación de mantener

[6] `https://analytics.google.com/`

al usuario activo por el mayor tiempo posible. Como hemos explicado anteriormente, la tarea de anotar tweets con DO es difícil incluso para alguien que tenga una clara definición de DO, lo que influye negativamente en la ventana de atención que puede mantener un usuario común hacia la tarea planteada. Por esto, una mejora que puede impactar positivamente a la anotación es agregar funcionalidades que la motiven. La aplicación ya cuenta con un contador de votos y un contador de tweets votados, esto se podría expandir en una *Leaderboard*, es decir, una tabla que muestre la cantidad de votos de cada usuario. De esta manera, se agregaría la sensación de competitividad entre estos, lo que puede motivarlos a anotar más tweets. Un sistema de logros sería otra opción que puede dar a los que utilizan la aplicación una manera de sentirse recompensados por utilizarla. También existen plataformas como las ya mencionadas *Amazon Mechanical Turk*[7] o *Appen*[8] que permiten la construcción de aplicaciones de anotación de datos para aprendizaje automático y proveen a los participantes de la anotación con remuneraciones monetarias, pero pueden llegar a tener un costo alto si se requiere una alta cantidad de participanetes.

Para cerrar esta sección, vemos interesante agregar como información anecdótica, la cantidad de usuarios por país reportada por *Google Analytics*, la cual se muestra en la tabla 4.5.

1.	Uruguay	**725** (80,92 %)
2.	United States	**49** (5,47 %)
3.	Argentina	**26** (2,90 %)
4.	Spain	**20** (2,23 %)
5.	Colombia	**11** (1,23 %)
6.	Mexico	**8** (0,89 %)
7.	India	**6** (0,67 %)
8.	Chile	**5** (0,56 %)
9.	Germany	**5** (0,56 %)
10.	United Kingdom	**5** (0,56 %)
11.	Netherlands	**5** (0,56 %)

Figura 4.5: Distribución de usuarios por país

No se muestra en la tabla, pero la sesión media para los usuarios de Chile, India, España y Reino Unido no supera el segundo, lo que coincide con la hipótesis de que existieron

[7]https://www.mturk.com/
[8]https://appen.com/

sesiones correspondientes a *webcrawlers*. Las posiciones que se encuentran desde la 12 en adelante no fueron incluidas por esta misma razón.

4.4. Conjunto de datos generado

En la tabla 4.3 se presenta la distribución de etiquetas para las clases *Odio* y *No odio* y cuantos de estos son ofensivos o no, así como a cuantos de estos no se les asignó etiqueta según fuesen ofensivos o no ya que tenían la misma cantidad de votos para ambas.

	Ofensivo	**No ofensivo**	**Sin asignar**	**Total**
Odio	1161	653	233	2047
No odio	1429	312	211	1952
Total	2590	965	444	3999

Tabla 4.3: Conteo de tweets con DO y lenguaje ofensivo del conjunto de datos final

Para asignar las etiquetas que establecen si el tweet tiene DO, primero se determina si el tweet presenta ambigüedad en cuanto a sus votos. Esto ocurre cuando la diferencia entre las cantidades para cada etiqueta posible es menor o igual a 1. En este caso, si se encuentra en el conjunto de aquellos que fueron desambiguados (es decir, anotados a mano por los autores de este trabajo como se describe en la sección 4.3.1), se asigna esa etiqueta. Si no es posible determinar la etiqueta para la clase, el tweet no es agregado al conjunto. Cuando el tweet tiene una cantidad de votos distinta y con una diferencia de dos o mayor para cada clase, se asigna la etiqueta con mayor cantidad de votos.

Para las etiquetas que determinan si es ofensivo o no, se procede de la misma manera descrita en el párrafo anterior, pero sin descartar los tweets que presenten ambigüedad, en cuyo caso se asigna la etiqueta "A" indicando que es ambiguo. Lo mismo se realiza para las etiquetas que indican el tipo de odio, pero tomando en cuenta los casos en los que no se tiene información, donde la etiqueta "N/A" es asignada.

La medida de acuerdo entre anotadores utilizada para medir el acuerdo en la clase *Contiene DO* es la alfa de Krippendorf, la cual se adapta a este caso ya que la cantidad de anotadores es variable y existe una cantidad considerable de información faltante, puesto que ingresaron mas de 100 usuarios a la aplicación y la cantidad máxima de votos para un tweet es 5. El valor de esta medida antes de corregir los tweets ambiguos es de 0.305. En este caso solo se consideran los votos realizados por los usuarios de la aplicación. Este valor indica un desacuerdo considerable, aunque menor que el esperado si los votos se asignan al azar. Luego de desambiguar 1200 tweets asignando las etiquetas manualmente, sin considerar los votos de los usuarios, la alfa de Krippendorf aumenta a 0.537. Según

Krippendorff en [18], este valor indica que la confiabilidad de los datos no es óptima y que para derivar conclusiones es necesaria evidencia adicional. Esto también coincide con las conclusiones presentadas en los capítulos 2 y 3: detectar cuando un texto contiene discurso de odio no es una tarea fácil, aún habiendo estudiado el tema, y se debe tener una clara guía para reconocerlo de la manera más objetiva posible.

En la tabla 4.4, se encuentran las etiquetas asignadas a los tweets que contienen DO según el tipo de éste. Las etiquetas fueron asignadas según la categoría más votada. Para todos los tweets una de las clases consiguió más votos que las restantes por lo que no fue necesario desambiguar ninguno de ellos. Por otro lado, hubo 313 tweets (de un total de 2047 tweets con DO) que no fueron votados según su tipo de odio puesto que no llegaron a esa etapa del proceso de anotación.

Ideológico	**Racista**	**Homófobo**	**Misógino**	**Otro**
739	416	302	223	54

Tabla 4.4: Distribución de etiquetas de tipo de odio en el conjunto de datos final

Para los votos según el tipo y para la clase *Es ofensivo* no se determinó la alfa de Krippendorff (explicada en la sección 2.2.1), debido a que no son utilizadas para clasificación si no que se toman en cuenta en el análisis en detalle de la clasificación.

5 | Arquitectura de los modelos

En este capítulo, presentamos los modelos que se utilizaron para los clasificadores y el procedimiento que genera la representación de los tweets. El proceso completo es un *pipeline* (ilustrado en la figura 5.1) que parte desde el tweet en su forma original, realizando su preprocesamiento, para luego obtener su representación vectorial que actuará como entrada para el clasificador que prediga si contiene DO.

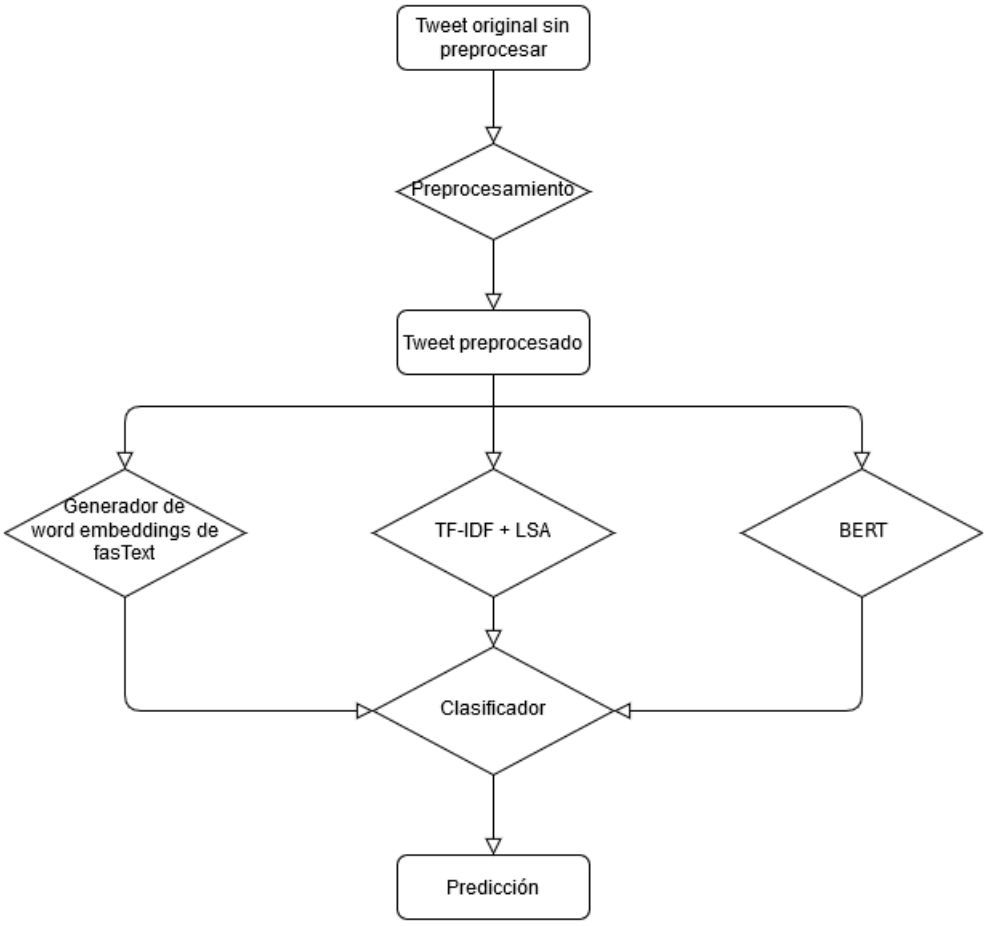


Figura 5.1: *Pipeline* del proceso de clasificación

5.1. Preprocesamiento

La primer etapa del *pipeline* de clasificación, que ocurre antes de comenzar el entrenamiento, consiste en una etapa de preprocesamiento sobre los datos que consiste en los siguientes pasos, en el orden indicado:

1. Todos los caracteres del texto se convierten a minúsculas.

2. Se utiliza el paquete tweet-preprocessor[1], que permite analizar y separar en *tokens* ciertos elementos que ocurren con frecuencia en los tweets. En nuestro caso se reemplazan todas las URLs, números, emoticones, hashtags y menciones por tokens especiales.

3. Se eliminan las *stopwords*.

4. El módulo Unidecode[2] se utiliza para convertir a representación ASCII aquellos caracteres Unicode que no hayan sido tokenizados antes de este paso.

5. Se ejecuta una búsqueda y reemplazo basada en una serie de expresiones regulares creadas para detectar tres o más signos de puntuación seguidos (incluyendo exclamación y pregunta) y secuencias de caracteres asociadas a risas (e.g. Jajajaj).

Adicionalmente, si al preprocesar el tweet todas sus palabras son vacías (resultando en un texto vacío), se elimina del conjunto.

A modo de ejemplo, en la tabla 5.1 se muestran los textos originales de dos tweets junto con el resultado de su preprocesamiento.

Antes de preprocesar	**Luego de preprocesar**
#ElProgreso | Confirman las condenas de 16 y 14 años a tres procesados de intentar matar a una mujer https://www.elprogreso.es/articulo/lugo/confirman-condenas-16-14-anos-acusados-intentar-matar-mujer-lugo/20190704181707138561 4.html ...	$HASHTAG$ | confirman condenas $NUMBER$ $NUMBER$ anos tres procesados intentar matar mujer URL $ELLIPSIS$
Uhh listo lo voy a matar, le dije q a vos no se te acerque. @USUARIO q te dije de acercarte a mi mujer??? Queres q te mate pelotudo???? Mira q estoy re loquito yo voy y te mato de una eee	uhh listo voy matar , dije q acerque . $MENTION$ q dije acercarte mujer $QUESTION$ queres q mate pelotudo $QUESTION$ mira q loquito voy mato eee

Tabla 5.1: Preprocesamiento de tweets

[1] `https://pypi.org/project/tweet-preprocessor`
[2] `https://pypi.org/project/Unidecode/`

5.2. Representaciones vectoriales

Finalizado el preprocesamiento, se debe lograr una representación de cada tweet que pueda servir como entrada del clasificador. Se generan tres tipos de representaciones:

- una que genera vectores para las palabras del tweet, que luego son procesadas por cada modelo de manera distinta para generar su representación (sección 5.2.1).
- una que genera un vector para el tweet mediante el método LSA (sección 5.2.2).
- una que genera el vector del tweet a través de un modelo BERT (Devlin et al., 2018 [14]) preentrenado (sección 5.2.3).

5.2.1. *fastText word embeddings*

La representación vectorial de las palabras que utiliza *fastText* (Armand Joulin et al., 2016 [21]) en la implementación de su clasificador fue elegida para la experimentación con los primeros modelos. Más adelante, en la sección 5.3.1, se describe este modelo con más detalle. Esta elección surge de observar que entrenar y evaluar el modelo propuesto por *fastText* obtiene buenos resultados considerando lo presentado en trabajos recientes (como el de Juan M. Pérez et al. [34]). Además, la librería oficial[3] que implementa el modelo facilita su entrenamiento y la obtención de los vectores generados.

El método para obtener la representación de cada palabra, presentado por Edouard Grave et al. [20], es una extensión del modelo *CBOW*, el cual fue comentado en la sección 2.8. En particular, reemplaza lo relacionado a *Bag of Words* por la técnica denominada en la publicación como *Bag of Character n-grams*, es decir, bolsa de n-gramas de caracteres. De esta manera, se entrenan representaciones de los n-gramas de caracteres y finalmente se obtiene el vector de la palabra sumando las representaciones. Además, siempre se incluye la palabra en su totalidad por lo que también se aprende la representación de esa palabra. Para ilustrar esto, supongamos que se quiere representar la palabra *hola*, primero se fija un largo máximo para los n-gramas de caracteres, por ejemplo 3. Con estos parámetros, los n-gramas obtenidos serán:

h
ho
hol
hola
o

[3] `https://pypi.org/project/fasttext/`

ol

ola

l

la

a

Un problema de esta técnica es que no es escalable. Por ejemplo, se pueden generar 31437 n-gramas distintos de largo 4 o menor para un alfabeto de 27 caracteres como el español y este crece cuanto mayor es el largo máximo considerado, lo que implica disponer de una alta capacidad de almacenamiento y cómputo. Para mitigar este problema, se generan *hashes* de los n-gramas. Un *hash* es el resultado de aplicar una función de *hashing*, que puede ser cualquier función que transforme elementos de un espacio a elementos de otro espacio distinto. En este caso, a partir de aplicar la función de *hashing* al n-grama, se obtiene un índice que indica cual vector lo representa. De esta manera se reduce la cantidad de representaciones diferentes que se deben aprender. A su vez, se logra obtener vectores para palabras desconocidas, ya que se usan sus n-gramas de caracteres y no la palabra en sí.

Para el español, se encuentran disponibles vectores de palabras de tamaño 300, obtenidos al realizar el proceso descrito anteriormente con n-gramas de caracteres y de palabras de tamaño 5, los cuales se encuentran disponibles en el sitio oficial de *fastText*[4]. Se utilizan dos corpus para entrenar los *embeddings*, uno de ellos provisto por la organización *Common Crawl*[5] y otro construido en base a textos de *Wikipedia*[6]. Estos vectores corresponden a realizar el proceso anteriormente descrito para la tarea de predicción de una palabra a partir de su contexto, de manera similar a *CBOW*, diferenciándose en que las representaciones de las palabras que forman parte del contexto se ponderan para que las que se encuentran próximas a la palabra central tomen una mayor importancia.

Para obtener los *word embeddings*, primero se entrena el modelo *fastText* con el corpus que se ocupará en el problema a resolver y, mediante métodos provistos por la librería de Python, se pueden obtener los vectores de cada palabra. Al entrenar el modelo, también estamos realizando un ajuste de las representaciones iniciales al dominio de nuestro problema. Esto se conoce como *fine-tuning*.

[4] `https://fasttext.cc/docs/en/crawl-vectors.html`

[5] `https://commoncrawl.org/`

[6] `https://www.wikipedia.org`

5.2.2. TF-IDF y Análisis Semántico Latente

La técnica conocida como **Análisis Semántico Latente** o **LSA** (Deerwester et al., 1990 [12]) por sus siglas en inglés, genera *embeddings* para una colección de documentos. El proceso de generación de éstos parte de una matriz término-documento, en la que cada fila corresponde a un documento que se desea representar y cada columna contiene un valor que representa el peso de cada palabra del vocabulario considerado en el documento. En el caso considerado para este trabajo, se calcula mediante la medida TF-IDF presentada en la sección 2.8.

A priori, se podrían utilizar las filas de la matriz como representaciones de los documentos, pero teniendo en cuenta que los textos tienen como máximo 280 caracteres, la cantidad de palabras en uno de ellos en relación con el total de palabras en el vocabulario es muy baja, lo que provoca que estos vectores sean dispersos, i.e. muchos de sus valores son 0. Por esto se pensó en utilizar técnicas que reduzcan su dimensionalidad. El método LSA parte de la matriz término-documento y aplica el algoritmo SVD (*Support Vector Decomposition*, en español Descomposición en Valores Singulares), logrando reducir la cantidad de dimensiones de la matriz y con ello la de las representaciones.

Este método parte de la matriz original y calcula su descomposición en valores singulares, la cual resulta en una factorización de la matriz con un rango determinado, menor que el de la original. Los valores resultantes de la factorización pueden verse como una combinación lineal de los valores originales. La descomposición en valores singulares garantiza conservar el ángulo entre los vectores del espacio original en los vectores resultantes, manteniendo la menor distancia posible entre estos y los vectores originales. De esta manera, se logra reducir la dimensión de los originales mediante la eliminación de información menos relevante.

5.2.3. BETO

El surgimiento de los modelos BERT (Devlin et al., 2018 [14]) para generación de *word embeddings* en 2018 permitió mejorar desde entonces el estado del arte en múltiples tareas de PLN. Por esta razón, se decide explorar el uso de BETO [5], un modelo BERT preentrenado exclusivamente con datos en idioma español. Si bien existe una versión multilenguaje de BERT, entrenada con corpus de varios lenguajes, los creadores de BETO presentan resultados de pruebas en las cuales se muestra una mejora de desempeño en casi todas las tareas, en comparación con la versión multilenguaje.

Para la implementación de este tipo de representaciones, se eliminó del preprocesamiento la remoción de palabras vacías o *stopwords* puesto que el modelo las considera para generar la representación del texto.

5.3. Modelos de clasificación

En esta sección se especifican los modelos evaluados sobre la tarea de detección de DO. Los modelos considerados son: el implementado por *fastText*[7], un modelo SVM y dos arquitecturas de redes neuronales: una construida en base a redes LSTM y otra a partir de redes convolucionales. En la tabla 5.2, se resumen cuales de las representaciones descritas en la sección anterior utiliza cada modelo.

	Vectores fastText	**BETO**	**EigenSent**	**LSA**
fastText	**X**			
Haternet	**X**	**X**		**X**
Convolucional	**X**	**X**		**X**
SVM	**X**	**X**	**X**	**X**

Tabla 5.2: Representaciones palabras/tweets utilizadas por cada modelo

5.3.1. Clasificador fastText

Como se mencionó en la sección 5.2.1, uno de los modelos de clasificación utilizados es el de *fastText* (figura 5.2). Las representaciones de palabras que emplea el modelo se obtienen tal como se describe en la sección 5.2.1. Luego de de generarlas, la siguiente capa (oculta) de la red obtiene la representación del tweet calculando la media de las representaciones de cada *token*. Finalmente, una capa de salida con función de activación *softmax* calcula la predicción final. Cabe destacar que *fastText* considera además los n-gramas de palabras que se encuentran en esta para añadir información sobre el orden parcial de las palabras.

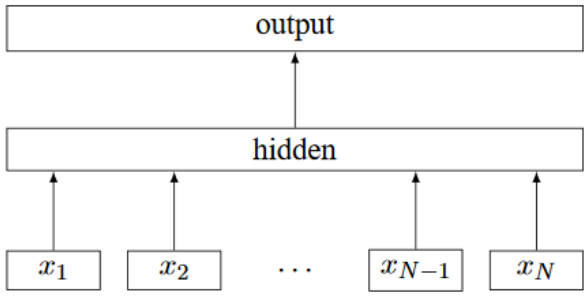


Figura 5.2: Arquitectura del modelo *fasttext*. Recibe los vectores $X_1, X_2, ..., X_N - 1, X_N$ que representan cada palabra, combinándolos en la capa oculta para luego generar la salida. Imagen extraída de [21]

[7]No confundir con el modelo que genera *word embeddings*

Este modelo, con sus hiperparámetros por defecto, fue considerado inicialmente como línea base debido a que, al no requerir implementar código adicional, no presenta una dificultad mayor entrenarlo y evaluarlo. Luego, se tomó la decisión de considerar una línea base más simple que permita comparar los modelos ante una solución trivial.

5.3.2. Redes neuronales

Debido al buen desempeño que han logrado las redes neuronales en tareas relacionadas, se decidió experimentar con su uso para la detección de DO. Existen varias librerías que facilitan el trabajo con modelos de aprendizaje automático y redes neuronales. Tras estudiar y discutir las distintas opciones se decidió utilizar la API Keras[8] con Tensorflow[9] como *backend*. Keras es una interfaz que simplifica la programación de redes neuronales, permitiendo al programador trabajar con conceptos como "capa" o "modelo", y con etapas claramente diferenciadas como entrenamiento (*fit*) y evaluación. Estas simplificaciones permiten experimentar rápidamente con distintas ideas y diseñar modelos en base a la combinación de módulos de uso común ya implementados. Por otro lado, Tensorflow es una de las plataformas más populares para el desarrollo de sistemas de redes neuronales, y además de funcionar como motor para la interfaz Keras, provee al programador de un ecosistema de herramientas, librerías y recursos que se mantiene actualizado. Keras se encuentra diseñado de manera que se puedan extender sus funcionalidades, en caso de que la abstracción de la interfaz no contemple en su totalidad la realidad del problema a resolver. Para ello, las utilidades de bajo nivel de Tensorflow son herramientas eficaces para desarrollar una solución más específica.

Se utilizaron entonces estas tecnologías para construir y evaluar dos arquitecturas que han tenido éxito en tareas de PLN relacionadas al análisis semántico y a la detección de DO: una propuesta para el proyecto *Haternet* [36] basada en redes recurrentes y otra en base a redes convolucionales (Chiruzzo et al., 2020 [7]). A continuación se detallan los componentes que forman parte de estas arquitecturas.

Arquitectura Haternet

La arquitectura propuesta por Pereira-Kohatsu et al. [36], se conforma por una capa LSTM que recibe los *word embeddings* que representan a las palabras del tweet y genera como salida su representación, seguida por capas ocultas con función de activación *ReLu* y distintas cantidades de neuronas y tasas de *dropout*. La organización de estas se encuentra ilustrada en la figura 5.3. El *dropout* tiene como finalidad regularizar la red. El término *regularización* hace referencia a técnicas que aumenten la capacidad de la red

[8] `https://keras.io/`

[9] `https://www.tensorflow.org/`

de generalizar, es decir, predecir adecuadamente datos no vistos a partir del conjunto de datos de entrenamiento. En este caso, el *dropout* lleva a 0 el resultado de cada neurona de la capa con la probabilidad que se le asigna.

La capa de salida se compone de una neurona con función de activación sigmoide. En las pruebas realizadas, el vector generado por la capa LSTM es concatenado a los vectores de LSA y BETO.

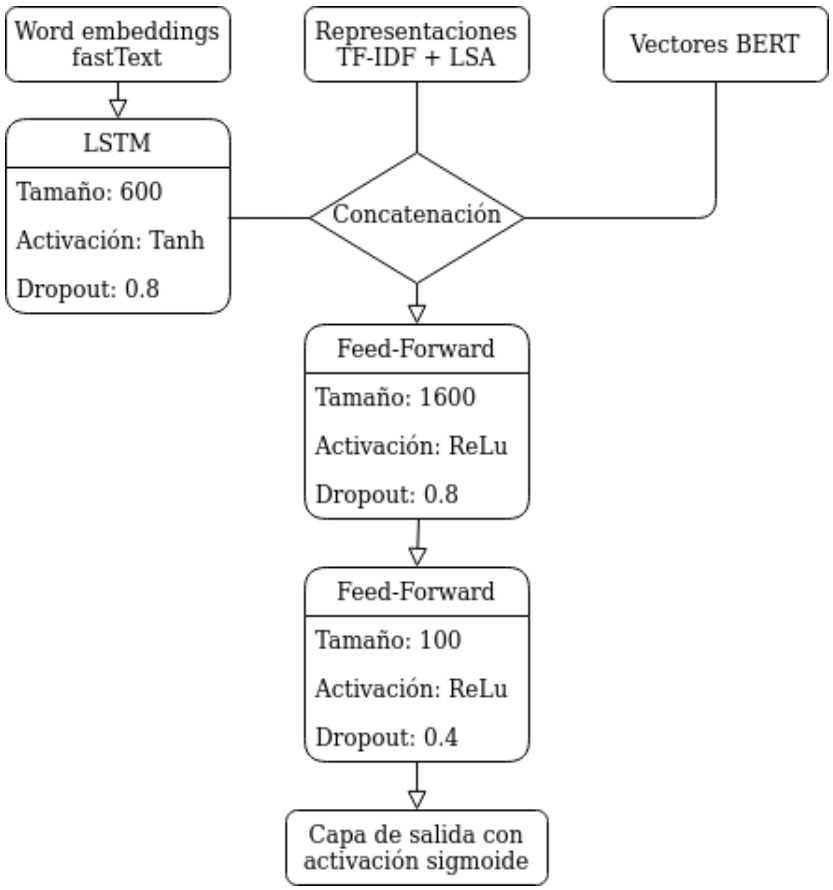


Figura 5.3: Arquitectura del modelo *Haternet*

Arquitectura convolucional

Se experimentó además con una arquitectura como la utilizada por Chiruzzo et al. en [7] para el TASS 2020, ilustrada en la figura 5.4. Ésta se compone, en primer lugar, de tres capas de convolución con kernel de distinto tamaño y con *ReLu* como función de activación. Estas capas reciben como entrada los vectores de las palabras que aparecen en cada tweet. Los resultados de las convoluciones son concatenados y procesados por una capa de max-pooling que se encarga de reducir la dimensionalidad de la salida eligiendo el valor máximo de cada filtro. La idea principal detrás de esta etapa es que las convoluciones aprendan atributos asociados a cada tweet a partir de los embeddings de las palabras que

contienen. Finalmente se emplea una capa densa con el objetivo de aprender la tarea de detección y una capa de dropout para regularizar.

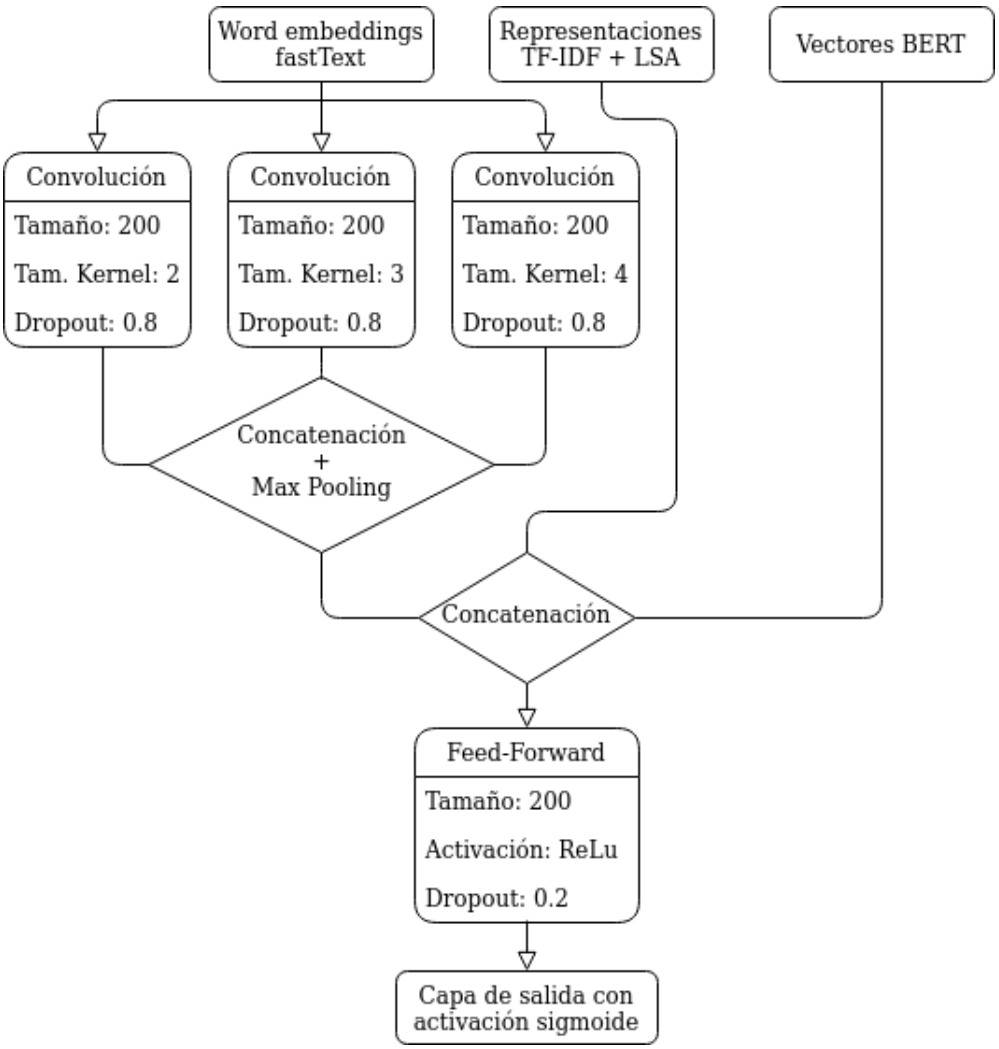


Figura 5.4: Arquitectura del modelo convolucional

5.3.3. SVM

El modelo SVM (Cortes et al., 1995 [10]) se encuentra explicado en la sección 2.6. La implementación utilizada es la disponible en la librería de Python Scikit-Learn[10], la cual permite configurar el modelo con distintos hiperparámetros como la función de *kernel*, el parámetro de regularización o la cantidad máxima de iteraciones que se realizan sobre el conjunto de datos.

En este caso se observa otro tipo de regularización que se basa en reducir la variabili-

[10] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC`

dad de los parámetros del modelo ante datos que pueden provocar un aumento o reducción considerable en estos. En el caso de la implementación de scikit-learn, el parámetro configura el grado de penalización sobre la función de pérdida. Esta penalización busca evitar que el ajuste de los parámetros se realice de manera controlada.

Antes de entrenar el modelo, para obtener el embedding del tweet a partir de los *word embeddings* de palabras se utiliza el método *EigenSent*, presentado por Kayal et al. [23]. Este se basa en un concepto del área de procesamiento de señales, que consiste en descomponerla en sus distintas frecuencias para poder analizar las propiedades que determinan la transición de un estado de la señal a otro. Los autores realizan un paralelismo con este concepto, en el que el estado de un texto es representado por el *word embedding* de una palabra en un punto del mismo. Proponen que si se logra obtener información sobre las propiedades que determinan la transición de una palabra a otra, esta información se puede utilizar para representar el texto del que forman parte en un espacio vectorial. Esta información se obtiene aplicando un algoritmo conocido como descomposición modal dinámica de alto orden, el cual, a grandes rasgos, obtiene a partir de la matriz formada por el valor del estado en distintos puntos del tiempo (los *word embeddings* de las palabras en el caso de un texto) los modos dinámicos, vectores que representan información sobre la transformación aplicada sobre un estado para pasar al siguiente. Estos vectores son concatenados para obtener el vector final que representa el tweet.

Además, las representaciones obtenidas mediante EigenSent se concatenan con los vectores de LSA y BETO.

5.4. Entrenamiento y evaluación

Considerando que el conjunto de datos obtenidos (presentado en la sección 4.4) es de tamaño pequeño en comparación con los obtenidos por Pereira-Kohatsu et al. [36] y Basile et al. [2], el método de entrenamiento y evaluación elegido fue una variante de validación cruzada que agrega estratificación a la selección de los ejemplos para cada *fold*. La estratificación es una estrategia de selección de ejemplos que asegura mantener la proporción de ejemplos cada clase en el conjunto original para cada partición generada. Para asegurar que se mantiene el balance en la cantidad de ejemplos para entrenamiento y evaluación en cada partición (80 % para entrenamiento y 20 % para evaluación), se fija $k = 5$ (la cantidad de *folds* generados). En el caso de los modelos de redes neuronales, el entrenamiento se realiza en épocas, donde para cada una de ellas se realiza el procedimiento de entrenamiento y validación. De esta manera, se puede monitorear el progreso del entrenamiento y agregar una condición de parada. En las pruebas realizadas, el entrenamiento se detiene cuando el resultado de la función de pérdida no percibe una mejora en 5 épocas.

Los hiperparámetros como el *learning rate*, la cantidad de ejemplos que se procesan antes de ajustar los parámetros se ajustaron de forma manual. Por otro lado, el modelo *fastText* fue entrenado con el método *hold-out*, separando el 20 % de los tweets en el conjunto final para evaluación del modelo.

El desempeño de cada modelo se mide en términos de *precision*, *recall*), el *f-score*, el AUC (sólo para los modelos de redes neuronales) y su acierto. Éstas métricas son reportadas para el modelo con mejor *f-score* del conjunto de modelos entrenados en cada partición y también se reporta el promedio de los valores para cada métrica.

En cuanto a las funciones de pérdida, se utiliza en el caso del modelo SVM la función *max-margin* (también conocida *hinge loss*) en su variación cuadrática, la cual se define como:

$$\ell(y) = \text{máx}(0, 1 - t \cdot y)^2$$

Donde ℓ es el valor resultante en base a la predicción y y t es el valor observado. Por otro lado, la función de pérdida para los modelos de redes neuronales es la *entropía cruzada binaria* también conocida como pérdida logarítmica (del inglés *log loss*), la cual calcula fundamentalmente la diferencia entre las distribuciones de probabilidad de la etiqueta real y la de las predicciones del modelo. Para casos de clasificación binaria, se define como:

$$\ell(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

Donde y es la etiqueta observada y p es la probabilidad predicha de que la etiqueta sea positiva.

6 | Análisis de los resultados

Hasta este punto, se han presentado las representaciones vectoriales escogidas para los tweets, los modelos elegidos para detectar odio a partir de la representación y las métricas para medir su desempeño en esta tarea. En las siguientes secciones se expondrán los resultados del entrenamiento y evaluación de los modelos sobre el conjunto de tweets conformado (descrito en la sección 4.4) y un análisis de clasificaciones puntuales que consideramos destacables.

6.1. Resultados de los modelos

Como mencionamos en la sección 5.4, presentamos los resultados de todos los modelos evaluados sobre el conjunto obtenido mediante el método de validación cruzada de tamaño 5, correspondientes al promedio obtenido por el modelo en cada partición. El modelo elegido como línea base corresponde al clasificador que considera todos los tweets como DO. En la tabla 6.1, se observan los valores obtenidos para cada modelo y la línea base. Todas las pruebas fueron realizadas con CPU Intel Core i7-3770K 3.50GHz y 8GB de memoria RAM.

	Accuracy	**Precision**	**Recall**	**AUC**	**F-score**
Línea base		0.512	1.000		0.677
fastText		0.655	0.663		0.648
SVM	**0.844**	0.849	0.844		**0.846**
Haternet	0.774	0.745	**0.923**	0.842	0.816
Convolucional	0.834	**0.864**	0.821	**0.906**	0.835

Tabla 6.1: Resultados obtenidos para cada modelo

El acierto y AUC para el modelo *fastText* no es reportado por la implementación en la librería que se distribuye, por lo que no se incluyen en la tabla. A su vez, el AUC para el modelo SVM no es calculable porque la salida del clasificador no es probabilística.
Lo primero que se púede destacar, es que todos los modelos, a excepción de fastText,

superaron el valor de *f-score* obtenido por la línea base. El modelo SVM obtuvo el mejor *f-score* y acierto. Esto puede observarse en el balance que logra entre *recall* y precisión. Aún así, los modelos Haternet y convolucional obtuvieron valores cercanos en esta métrica.
Es interesante observar que, a pesar de la baja precisión, el modelo *Haternet* presenta un *recall* alto en comparación con los otros modelos, sugiriendo que es más propenso a clasificar los ejemplos como positivos, arriesgando a equivocarse con los negativos. En cuanto al modelo convolucional, exhibe una buena precisión, clasificando una mayor parte de los ejemplos positivos correctamente.
El valor del AUC indica para los modelos *Haternet* y convolucional que en general, la probabilidad que producen como salida es alta cuando detectan un ejemplo como positivo y baja en caso contrario.
Añadimos además la matriz de confusión para cada modelo (figuras 6.4, 6.3 y 6.2).

		Valor verdadero	
		Positivo	**Negativo**
Valor predicho	**Positivo**	374	37
	Negativo	35	353

Tabla 6.2: Matriz de confusión para el modelo *SVM*

		Valor verdadero	
		Positivo	**Negativo**
Valor predicho	**Positivo**	405	72
	Negativo	5	318

Tabla 6.3: Matriz de confusión para el modelo *Haternet*

		Valor verdadero	
		Positivo	**Negativo**
Valor predicho	**Positivo**	398	47
	Negativo	12	342

Tabla 6.4: Matriz de confusión para el modelo convolucional

En el caso del modelo *Haternet*, fue el que logró mayor cantidad de ejemplos positivos clasificados correctamente, pero el que más clasificó ejemplos negativos erróneamente. Mientras que lo opuesto sucede con la arquitectura convolucional. El clasificador SVM logra el mejor balance entre clasificaciones acertadas y equivocadas, con una precisión menor que el los otros modelos.

6.2. Análisis de la clasificación

El propósito de esta sección es proveer una visión general sobre las predicciones realizadas por los clasificadores durante la evaluación. Para esto, se tomó para cada modelo los resultados del clasificador que mejor puntuación obtuvo en el proceso de validación cruzada, es decir, aquel con mejor *f-score* de los entrenados sobre cada partición formada al combinar los distintos *folds*.

En primer lugar presentamos algunos casos particulares donde los distintos modelos fallan al predecir la clase correcta. Tal como se describe en la sección 3.2.3, una de las dificultades principales al intentar detectar DO es diferenciarlo del lenguaje ofensivo. Los siguientes tweets fueron clasificados por todos los modelos como "con odio":

«tcómo puedes decir aquí se acaba la carrera de esta señora, quien eres tú, ella es maestra, y el título no se la dado tú padre, es q ver lo q has echo a esta mujer, mi dan ganas de decirte hijo de puta en tu cara, y te agarro de la mano como hicistes con ella sin vergüenza, H P.»

«@USUARIO @USUARIO #ChauDominguez, andate corrupto incapaz, inservible, inepto, idiota, mafioso, coimero, ladron, incompetente, lacra, acomodado, pelotudo, mal tipo, perverso, sos una verguenza para tus compatriotas pedazo de hdrmp»

«Es un puto meme, maldito orangutan de mierda, a mi las putas me llueven»

Ninguno de los tweets anteriores contiene DO según la definición de referencia para este trabajo, porque ninguno tiene como objetivo atacar a una o más personas por el hecho de pertenecer a un grupo en particular. Sin embargo, parece ser que los modelos en estos casos tienden a darle mayor importancia de la debida a la presencia de lenguaje ofensivo y fallan al reconocer la intención del mensaje.

Aún en algunos casos en los que los tweets no contienen lenguaje ofensivo, los modelos los clasificaron erróneamente como DO. Un ejemplo de esto es el siguiente:

«Me acuerdo de una vuelta que estábamos hablando en el grupo de la promo y uno tiró en un momento "callate feminista" que tiene que ver que sea feminista con que vos seas un rata que se queja de todo Raúl?»

Es probable que la razón por la cual este tweet es clasificado como DO sea que los modelos asocian muy fuertemente expresiones del tipo "callate feminista", que ocurren frecuentemente en mensajes con DO, con el hecho de que el tweet contenga DO. Nuevamente se priorizan expresiones de pocas palabras antes que el contexto y la intención general, llevando a una clasificación incorrecta.

Por otro lado, algunos tweets que contenían DO no fueron clasificados como tales por los modelos. Algunos de ellos son:

«*Jajajaja solo lo decía por que eres mujer y por eso siempre me quieres llevar la puta contra*»

«*JAJAJAJAJAJAJA que chino hdp (URL)*»

«*brother que vivo en tu puto pais gitano asqueroso*»

En los primeros dos ejemplos las expresiones de risa pueden estar influyendo de forma negativa en las predicciones, ya que las risas aparecen principalmente en ejemplos sin DO en el conjunto de datos. En cuanto al tercer tweet, en el conjunto de datos la palabra "gitano" solo aparece en tres ejemplos y de ellos solo dos contienen DO. Es probable que estos problemas se logren solucionar agregando más ejemplos con este tipo de expresiones al *dataset*.

Por último, presentamos dos ejemplos que tuvieron predicciones cercanas a 0,5 (siendo 1 odio y 0 no odio) en los modelos Haternet y Convolucional, y que creemos que muestran de forma clara algunos de los problemas de ambigüedad y falta de contexto que pueden ocurrir en esta tarea:

«*Te amo puto negro recoge algodón @USUARIO*»

«*No es lo mismo decir Ramona Cabrera que ramera cabrona.*»

Ambos tweets fueron anotados como "con odio" por los usuarios de la aplicación web. A pesar de ello, podemos observar características que pueden dificultar la tarea de detección. El primer tweet esta formado por insultos junto con una expresión de afecto que podrían resultar contradictorias para los modelos, mientras que el segundo tweet contiene un juego de palabras cuya intención (en caso de que la haya) no es del todo clara. Como se mencionó en la sección 4.4, existe una gran cantidad de tweets considerados ambigüos de acuerdo a los votos de los usuarios. Muchos de ellos comparten características con los tweets anteriores y resultan difíciles de clasificar tanto para humanos como para clasificadores automáticos.

El análisis realizado en esta sección trae a luz los problemas asociados a la hipótesis inicial de que solo es necesario considerar el texto del tweet para su clasificación. La mayoría de los ejemplos anteriores indican que el contexto es un factor importante. En algunos casos los modelos fallan al reconocer la intención implícita en el texto, pero en otros casos simplemente el contexto necesario no se encuentra en el tweet a clasificar si no en una imagen adjunta o en un mensaje anterior, por lo que los clasificadores no cuentan con los datos necesarios para realizar correctamente la tarea.

6.3. Resultados sobre corpus *Haternet*

Tomando en cuenta que el clasificador SVM obtuvo los mejores resultados sobre nuestro corpus, se eligió para comparar su desempeño con el corpus presentado por Pereira-Kohatsu et al. en [36]. La arquitectura de este es la misma presentada en la sección 5.3.2, diferenciándose en las representaciones que utiliza, las cuales consisten en embeddings para palabras, para *emojis*, de expresiones (palabras que indican emociones como *WTF* o *LOL*) y *BoW* con pesos TF-IDF.

El conjunto de datos contiene 6000 tweets, de los cuales 1567 contienen DO y el restante 4433 no lo contienen. Para el entrenamiento del clasificador SVM, se realizó un ajuste de hiperparámetros mediante una búsqueda aleatoria de 60 iteraciones y utilizando validación cruzada de tamaño 5 sobre el 80 % del conjunto de datos, evaluando el modelo correspondiente a la mejor combinación de hiperparámetros sobre el 20 % restante. Por otro lado, el entrenamiento del modelo *Haternet* (Pereira-Kohatsu et al., 2019 [36]) se realiza mediante una validación cruzada de tamaño 10 reportando el promedio de los resultados en cada iteración, y ajustando para cada una de las iteraciones los hiperparámetros del modelo con una validación cruzada de tamaño 3 sobre los datos de entrenamiento. La comparación se presenta en la figura 6.5.

Clasificador	**Precisión**	**Recall**	**F-Score**
Haternet	0.625	0.598	0.611
SVM-pgodio	0.617	0.524	0.567

Tabla 6.5: Comparativa de resultados del modelo SVM y los presentados en Pereira-Kohatsu et al. [36]

Como se puede apreciar, el *recall* es la métrica en donde más se encareció el clasificador SVM, aunque alcanzó una precisión muy cercana al clasificador *Haternet*. Esto muestra que aunque el clasificador SVM sea más simple en cuanto a cantidad de parámetros que el basado en redes neuronales, tiene potencial para obtener buenos resultados. Ya que los clasificadores SVM suelen presentar un mejor funcionamiento sobre espacios de menor dimensionalidad, utilizar un algoritmo de reducción sobre los vectores generados por *EigenSent* puede llevarlo a mejorar los valores de las métricas. Por esta misma razón, puede ser posible que realizando el mismo ajuste de hiperparámetros con los modelos basados en redes neuronales otorgue mejores resultados, pues suelen trabajar mejor en espacios de alta dimensionalidad. Por otra parte, generar representaciones no sólo para las palabras, si no para emojis y expresiones que expresan sentimientos similares puede añadir información valiosa a la representación vectorial.

Se evaluó además comparar resultados obtenidos sobre el corpus de *SemEval*, lo que fue desestimado ya que el corpus de evaluación solo se encontró en su versión sin anotaciones, lo que hace imposible la comparación con otros modelos de la competencia. Además, se consideró comparar resultados sobre el corpus en inglés presentado por Waseem et al. [40], pero el alcance del proyecto, sumado a las limitaciones de tiempo no permitieron realizar este experimento.

7 | Conclusiones y trabajo a futuro

Como conclusión inicial, podemos decir que se logró **comprender la problemática del DO**, destacándose que su detección no es una tarea para nada fácil tanto para el ser humano como para un clasificador automático. Un indicador de la dificultad que presenta la detección de DO reside en las diferentes convenciones que se encuentran en los trabajos sobre el área acerca de qué se considera DO, siendo algunas más estrictas que otras en cuanto a las características que debe cumplir un texto para contenerlo. A su vez, al momento de anotar un corpus para su detección mediante métodos supervisados, es necesario que los anotadores tengan claro como determinar cuándo un texto presenta DO. Por esto las guías deben ser claras y lo más específicas posibles para evitar anotaciones sesgadas, ya que es difícil ser imparcial a la hora de identificar DO. Además, se debe incentivar a que las guías sean leídas, procurando que estén escritas de una manera accesible para todos.

Uno de los productos generados en el marco del desarrollo de este trabajo, es una **aplicación web para el *crowdsourcing* de la anotación de tweets** según tengan DO o no. A partir de la experiencia adquirida durante su desarrollo y su subsecuente puesta en producción, se concluye que es necesario apuntar a motivar al usuario a que continúe con la anotación a lo largo del tiempo. En nuestro caso, si se deseara continuar con el uso de esta aplicación, sugerimos en primer lugar hacer más atractiva la interfaz de usuario, con el objetivo de lograr una mejor impresión en el primer acceso. Además, una mejor selección de los tweets a mostrar en el sitio puede captar un mayor interés en los usuarios una vez que acceden y comienzan el proceso de anotación. Otra mejora posible puede ser el uso de técnicas de refuerzo positivo para conservar el interés del usuario, de forma que reciba una recompensa por el esfuerzo dedicado.

Otro aporte del proyecto es la construcción de un **corpus anotado** con diferentes etiquetas a partir de los votos obtenidos, comparable en tamaño con los disponibles públicamente. El corpus está formado por 3999 tweets anotados por un total de 800 anotadores, resultando en una alfa de Krippendorff de 0.537. Como se observó, el acuerdo

entre anotadores no es del todo aceptable, por lo que se sugiere realizar una revisión de las etiquetas asignadas a cada ejemplo para elevar la calidad del conjunto de datos.

Por último, se utilizó el corpus construido para entrenar **modelos de aprendizaje automático** que presentan un desempeño superior a la línea base establecida, explorando distintos métodos y diferentes técnicas para generar representaciones a partir de su pre-procesamiento. El modelo que obtuvo mejor *f-score* fue el SVM (0.846), aunque la mejor precisión la tuvo el modelo convolucional (0.864) y el mejor recall el modelo Haternet (0.923).

Sobre el desarrollo de modelos de aprendizaje automático existe un sinfín de técnicas que pueden mejorar su desempeño. En el área de la clasificación de texto, la elección o generación de representaciones que logren confluir las características principales de los textos es tan o más importante como la elección del modelo para el clasificador. Además del uso de *word embeddings*, en muchos casos características semánticas o sobre el contexto (más allá del texto en sí), las cuales no fueron consideradas en el presente trabajo, pueden enriquecer las representaciones. En el caso particular del discurso de odio, el contexto es importante tanto para ser tomado en cuenta en estas representaciones, como para poder ser identificado adecuadamente por el ser humano.

Otro de los principales problemas que se debe tener en cuenta, es la poca cantidad de datos anotados que se encuentran disponibles, por lo que es necesario trabajar en la anotación de datos para generar conjuntos de gran calidad y cantidad. Técnicas para sobreponerse a estos obstáculos, como el sobremuestreo, podrían llegar a producir mejores resultados. También puede ser interesante evaluar los modelos mediante el entrenamiento con más de un corpus, de manera que la cantidad de datos sea mayor. Como sugerencia final para los trabajos futuros, el empleo de *transfer learning* (i.e. preentrenar un modelo para una tarea distinta a la que se quiere resolver) ha demostrado servir de gran ayuda en cuanto a la clasificación de texto se refiere. En este trabajo, se ha aplicado este concepto al realizar *fine-tuning* de los vectores pre-entrenados de *fastText*. Sin embargo, los vectores de BETO se utilizaron sin haberlos ajustado. Cuando se utilizan vectores pre-entrenados, ajustarlos para que representen mejor a los textos del dominio en el que se basa el problema suele mejorar los resultados, por lo que es recomendable hacerlo siempre que sea posible. Adicionalmente, entrenar los modelos previamente para una tarea distinta a la que se quiere resolver (por ejemplo, entrenándolo para clasificar lenguaje ofensivo y luego para clasificar DO) puede llegar a proveer mejores resultados.

Finalmente, creemos que, a pesar de que los resultados no son excelentes, se llegó al objetivo que se había propuesto, definiendo concretamente el problema y abriendo las puertas a nuevos trabajos sobre el tema, el cual aún conserva campo fértil para ser explorado, especialmente en el idioma español. Asimismo, se logró entender un problema sumamente vigente, que provoca consecuencias negativas en muchas personas, y aportamos nuestra contribución a un tema que se puede atacar a través de varias disciplinas. Esperamos que este trabajo actúe como inspiración para continuar con el abordaje de este asunto, el cual nos concierne a todos.

8 | Bibliografía

[1] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta y Vasudeva Varma: *Deep Learning for Hate Speech Detection in Tweets*. Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion, 2017. http://dx.doi.org/10.1145/3041021.3054223.

[2] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso y Manuela Sanguinetti: *SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter*. En *Proceedings of the 13th International Workshop on Semantic Evaluation*, páginas 54–63, Minneapolis, Minnesota, USA, Junio 2019. Association for Computational Linguistics. https://www.aclweb.org/anthology/S19-2007.

[3] Shanita Biere: *Hate Speech Detection Using Natural Language Processing Techniques*. 2018.

[4] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai y Robert L. Mercer: *Class-Based n-gram Models of Natural Language*. Computational Linguistics, 18(4):467–480, 1992. https://www.aclweb.org/anthology/J92-4003.

[5] José Cañete, Gabriel Chaperon, Rodrigo Fuentes y Jorge Pérez: *Spanish Pre-Trained BERT Model and Evaluation Data*. En *PML4DC at ICLR 2020 (todavía no publicado)*, 2020.

[6] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun Hsuan Sung, Brian Strope y Ray Kurzweil: *Universal Sentence Encoder*, 2018.

[7] Luis Chiruzzo, Mathias Etcheverry y Aiala Rosá: *Sentiment Analysis in Spanish Tweets: Some Experiments with Focus on Neutral Tweets*, 2020-03.

[8] *Código penal de la República Oriental del Uruguay, Ley 17.677, Artículo 1º*, 2003.

[9] Jacob Cohen: *A Coefficient of Agreement for Nominal Scales*. Educational and Psychological Measurement, 20(1):37–46, 1960.

[10] Corinna Cortes y Vladimir Vapnik: *Support-Vector Networks*. Mach. Learn., 20(3):273–297, Septiembre 1995, ISSN 0885-6125. https://doi.org/10.1023/A:1022627411411.

[11] Thomas Davidson, Dana Warmsley, Michael Macy y Ingmar Weber: *Automated Hate Speech Detection and the Problem of Offensive Language*, 2017.

[12] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer y Richard Harshman: *Indexing by latent semantic analysis*. JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, 41(6):391–407, 1990.

[13] Autor no determinado: *George Floyd: cómo fueron sus últimos 30 minutos de vida*. Diario La Nación, Junio 2020. https://www.lanacion.com.ar/el-mundo/george-floyd-que-paso-antes-su-arresto-nid2372161, visitado el 2020-06-07.

[14] Jacob Devlin, Ming Wei Chang, Kenton Lee y Kristina Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2018.

[15] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic y Narayan Bhamidipati: *Hate Speech Detection with Comment Embeddings*. En *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, página 29–30, New York, NY, USA, 2015. Association for Computing Machinery, ISBN 9781450334730. https://doi.org/10.1145/2740908.2742760.

[16] Real Academia Española: *Discurso de odio*. En *Diccionario de la lengua española*. Consejo General del Poder Judicial, 22a edición, 2001.

[17] J. L. Fleiss: *Measuring nominal scale agreement among many raters*. Psychological Bulletin, 76(5):378–382, 1971.

[18] Cristina M. Giannantonio: *Book Review: Krippendorff, K. (2004). Content Analysis: An Introduction to Its Methodology (2nd ed.). Thousand Oaks, CA: Sage*. Organizational Research Methods, 13(2):392–394, 2010. https://doi.org/10.1177/1094428108324513.

[19] Ian Goodfellow, Yoshua Bengio y Aaron Courville: *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[20] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin y Tomas Mikolov: *Learning Word Vectors for 157 Languages*. En *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[21] Armand Joulin, Edouard Grave, Piotr Bojanowski y Tomas Mikolov: *Bag of Tricks for Efficient Text Classification*, 2016.

[22] Daniel Jurafsky y James Martin: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volumen 2. Prentice Hall, 2da edición, 2000.

[23] Subhradeep Kayal y George Tsatsaronis: *EigenSent: Spectral sentence embeddings using higher-order Dynamic Mode Decomposition.* En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, páginas 4536–4546, Florence, Italy, Julio 2019. Association for Computational Linguistics. `https://www.aclweb.org/anthology/P19-1445`.

[24] Yoon Kim: *Convolutional Neural Networks for Sentence Classification.* En *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1746–1751, Doha, Qatar, Octubre 2014. Association for Computational Linguistics. `https://www.aclweb.org/anthology/D14-1181`.

[25] K. Krippendorff: *Computing Krippendorff's Alpha-Reliability.* Obtenido de `https://repository.upenn.edu/asc_papers/43`.

[26] Quoc Le y Tomas Mikolov: *Distributed Representations of Sentences and Documents.* En *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, página II–1188–II–1196. JMLR.org, 2014.

[27] Shervin Malmasi y Marcos Zampieri: *Detecting Hate Speech in Social Media.* En *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, páginas 467–472, Varna, Bulgaria, Septiembre 2017. INCOMA Ltd. `https://doi.org/10.26615/978-954-452-049-6_062`.

[28] Llew Mason, Jonathan Baxter, Peter Bartlett y Marcus Frean: *Boosting Algorithms as Gradient Descent.* En *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, página 512–518, Cambridge, MA, USA, 1999. MIT Press.

[29] Tomas Mikolov, Kai Chen, Greg Corrado y Jeffrey Dean: *Efficient Estimation of Word Representations in Vector Space*, 2013.

[30] Tomas M. Mitchell: *Machine Learning*, volumen 1. McGraw-Hill, 1997.

[31] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad y Yi Chang: *Abusive Language Detection in Online User Content.* En *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, página 145–153, Republic and

Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee, ISBN 9781450341431. https://doi.org/10.1145/2872427.2883062.

[32] John T. Nockleby: *Hate Speech*. En Leonard W. Levy y Kenneth L. Karst (editores): *Encyclopedia of the American Constitution*. Macmillan, New York, 2da edición, 2000.

[33] Jeffrey Pennington, Richard Socher y Christopher Manning: *GloVe: Global Vectors for Word Representation*. En *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1532–1543, Doha, Qatar, Octubre 2014. Association for Computational Linguistics. https://www.aclweb.org/anthology/D14-1162.

[34] Juan Manuel Pérez y Franco M. Luque: *Atalaya at SemEval 2019 Task 5: Robust Embeddings for Tweet Classification*. En *Proceedings of the 13th International Workshop on Semantic Evaluation*, páginas 64–69, Minneapolis, Minnesota, USA, Junio 2019. Association for Computational Linguistics. https://www.aclweb.org/anthology/S19-2008.

[35] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee y Luke Zettlemoyer: *Deep contextualized word representations*, 2018.

[36] Lara Quijano-Sanchez, Juan Carlos Pereira Kohatsu, Federico Liberatore y Miguel Camacho-Collados: *HaterNet a system for detecting and analyzing hate speech in Twitter*. Sensors (Basel), Octubre 2019. https://doi.org/10.3390/s19214654.

[37] Eric Schenk y Claude Guittard: *Crowdsourcing: What can be Outsourced to the Crowd, and Why ?* Workshop on Open Source Innovation, 2009.

[38] *Twitter help center, sección Rules and policies*. https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy, Accedido el 19 de enero de 2020.

[39] William Warner y Julia Hirschberg: *Detecting Hate Speech on the World Wide Web*. En *Proceedings of the Second Workshop on Language in Social Media*, páginas 19–26, Montréal, Canada, Junio 2012. Association for Computational Linguistics. https://www.aclweb.org/anthology/W12-2103.

[40] Zeerak Waseem y Dirk Hovy: *Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter*. En *Proceedings of the NAACL Student Research Workshop*, páginas 88–93, San Diego, California, Junio 2016. Association for Computational Linguistics. https://www.aclweb.org/anthology/N16-2013.

[41] Wikipedia: *Víctimas del Holocausto*. http://es.wikipedia.org/w/index.php?title=V%C3%ADctimas%20del%20Holocausto&oldid=126131743, 2020. Accedido del 12 de Junio del 2020].

[42] David Yarowsky: *DECISION LISTS FOR LEXICAL AMBIGUITY RESOLUTION: Application to Accent Restoration in Spanish and French.* En *32nd Annual Meeting of the Association for Computational Linguistics*, páginas 88–95, Las Cruces, New Mexico, USA, Junio 1994. Association for Computational Linguistics. `https://www.aclweb.org/anthology/P94-1013`.

[43] Dawei Yin, Zhenzhen Xue, Liangjie Hong, Brian Davison, April Edwards y Lynne Edwards: *Detection of harassment on Web 2.0.* En *Content Analysis in the WEB 2.0 (CAW2.0) Workshop at WWW2009*, Abril 2009.

9 | Glosario

- **Alfa de Krippendorff**: Medida de acuerdo entre anotadores para un conjunto de datos.
- **API**: *Application Progamming Interface*.
- ***Application Progamming Interface***: Interfaz utilizada para interactuar con un servicio u aplicación.
- **Aprendizaje no supervisado**: Algoritmos de aprendizaje automático donde se intenta descubrir estructuras o patrones únicamente a partir de los datos de entrada, sin el uso de información extra.
- **Aprendizaje supervisado**: Algoritmos de aprendizaje automático donde se le presenta al programa ejemplos con valores de entrada y sus respectivos valores de salida y el objetivo es aprender una función general cuya salida sea la correcta dada cualquier entrada.
- **Aprendizaje automático**: Área de la inteligencia artificial cuyo objeto de estudio son algoritmos que mejoran a través de la experiencia.
- **Atributo**: Propiedad o característica medible que se puede observar sobre los datos.
- **AUC**: *Area Under the Curve*, una métrica de desempeño.
- **Bag-of-Words**: Representación computacional de un texto a partir de la frecuencia con la que ocurren las palabras que contiene.
- **BERT**: Modelo basado en redes neuronales para la generación de representaciones vectoriales de textos.
- **BETO**: Modelo BERT preentrenado con un conjunto de corpus en español.
- **BoW**: Bag-of-Words.
- **Capa**: En el contexto de redes neuronales, agrupamiento de neuronas.

- **CBoW**: Continuous Bag-of-Words.
- **Clasificador**: Algoritmo que clasifica ejemplos según un criterio dado.
- **CNN**: Convolutional Neural Network (Red Neuronal Convolucional).
- **Continuous Bag-of-Words**: Método para generar *word embeddings* en el que se entrena un modelo de redes neuronales simple para predecir una palabra en una secuencia, dadas las palabras que la rodean (es decir, su contexto).
- **Corpus**: Conjunto de textos destinado a la investigación científica.
- **Corpus anotado**: Corpus que contiene anotaciones manuales que proporcionan información adicional sobre los textos.
- ***Crowdsourcing***: Tercerización de una tarea a la multitud, típicamente a través de internet.
- ***Dataset***: Conjunto de datos.
- **DO**: Discurso de odio.
- **EigenSent**: Método para generar *word embeddings*, con base en conceptos de procesamiento de señales.
- **Emoji**: Dibujo o signo que expresa una emoción o idea.
- **Ensemble methods**: Técnica que combina múltiples modelos de aprendizaje automático para generar un nuevo modelo.
- **F-score**: Métrica para la evaluación del desempeño general de un clasificador.
- **fastText**: Modelo de clasificación presentado en Armand Joulin et al. [21].
- **Feature**: Atributo.
- **Fine-tuning**: Refiere tanto a reentrenar un modelo ya entrenado para resolver una tarea distinta a la actual, como al ajuste de hiperparámetros de un modelo con el fin de que logre mejores resultados sobre el problema que se desea resolver.
- **Función de pérdida**: Función que un algoritmo de aprendizaje automático busca minimizar durante el entrenamiento.
- **Haternet**: Denominación del modelo clasificador presentado en Pereira-Kohatsu et al. [36].
- **HatEval**: Tarea de reconocimiento de DO en tweets, propuesta para la edición del año 2019 de *SemEval*.

- **Hiperparámetro**: Parámetro de un algoritmo de aprendizaje automático que controla el proceso de aprendizaje.
- **Hold-out**: Método de entrenamiento y *testing* de un modelo de aprendizaje automático que se basa en dividir el *dataset* en 2 partes, realizar el entrenamiento con una de ellas y la evaluación con la otra.
- **Latent Semantic Analysis**: Método que genera representaciones vectoriales para una colección de documentos.
- **LGBT**: Lesbianas, Gays, Bisexuales y Trans.
- **LGBT+**: Lesbianas, Gays, Bisexuales, Trans y demás individuos de género u orientación sexual no binario.
- **Línea base**: Modelo que se utiliza para establecer el nivel de desempeño mínimo a superar por otro modelo.
- **Long Short-Term Memory**: Arquitectura de red neuronal basada en redes neuronales recurrentes.
- **LSA**: *Latent Semantic Analysis*.
- **LSTM**: Long Short-Term Memory.
- **Max-pooling**: Proceso que reduce la dimensionalidad de la entrada seleccionando el valor máximo de cada elemento de ésta.
- **N-grama**: Agrupación de N elementos de una secuencia determinada.
- **Pipeline**: Proceso que se encuentra dividido en etapas con un orden claro y en el cual la salida de una etapa es la entrada de la siguiente.
- **PLN**: Procesamiento de lenguaje natural.
- **Preprocesamiento**: Etapa de preparación de un dataset para su posterior uso por un algoritmo de aprendizaje automático.
- **Procesamiento de lenguaje natural**: Disciplina que busca resolver computacionalmente tareas que involucran el lenguaje natural.
- **Regularización**: Refiere a los métodos utilizados para evitar el sobreajuste de modelos de aprendizaje automático.
- **Representación vectorial**: Representación de una cadena de caracteres en vectores de valores numéricos.
- **RNN**: *Recurrent Neural Network* (Red Neuronal Recurrente).

- **SemEval**: International Workshop on Semantic Evaluation (Taller Internacional de Evaluación Semántica).
- **Sentence embeddings**: Representación vectorial de oraciones.
- **SVM**: *Support Vector Machine*.
- **TASS**: Task of Sentiment Analysis at SEPLN (Tarea de Análisis de Sentimiento en SEPLN).
- **TF-IDF**: Term Frequency - Inverse Document Frequency.
- **Token**: Símbolo utilizado como unidad mínima a la hora de realizar una tarea sobre un texto.
- **Tokenizar**: Agrupar un texto en tokens.
- **Tweet**: Publicación en la red social *Twitter*. Puede tener un máximo de 280 caracteres.
- **Twitter**: Red social donde los usuarios publican *tweets*.
- **Validación cruzada**: Método de entrenamiento y evaluación de desempeño en el cual se divide el conjunto de datos en distintas particiones para el entrenamiento y evaluación, calculando las métricas de desempeño sobre cada una de ellas.
- **Word embedding**: Representación vectorial de palabras, construida a partir de la distribución de los contextos en los que aparece.

10 | Anexo

10.1. Listas de palabras para búsqueda de tweets con DO

10.1.1. Lista *aggresive_words*

- matar
- burro
- ignorante
- mediocre
- ridiculo
- tarado
- idiota
- mamon
- alcahuete
- payaso
- estupido
- imbecil
- imbesil
- criminal
- de mierda
- hdp

- sucio
- mamadera
- callate
- chupa pija
- maldito
- puto
- petero
- asqueroso
- chupame la verga
- chupame la pija

10.1.2. Listas para DO homofóbico

Lista *hateful_words*

- muerdealmohada
- marica
- maricon
- mariposon
- trolo
- brisco

Lista *aggressive_words*

- puto
- trola
- trolazo

Lista *dependent_words*

- transsexual
- lgbt
- lgbti

- torta

10.1.3. Listas para DO misógino

Lista *hateful_words*

- feminazi
- ramera
- buscona
- calientapollas
- calientapija
- argolluda

Lista *aggressive_words*

- perra
- zorra
- puta
- petera
- trola

Lista *dependent_words*

- feminista
- mujer

10.1.4. Listas para DO ideológico

Lista *hateful_words*

- fraude amplio
- facho
- putiprogre
- zurdito

Lista *aggressive_words*

- ladron
- chorro
- corrupto
- corruptos

Lista *dependent_words*

- chavista
- fascista
- comunista
- zurdo

10.1.5. Listas para DO racista

Lista *hateful_words*

- negrata
- espalda mojada

Lista *aggressive_words*

- ladron
- chorro
- patarrajada
- pata rajada
- esclavo

Lista *dependent_words*

- negro
- indio
- moro
- chino

- moreno
- blanquito

10.2. Ajuste de hiperparámetros del clasificador SVM

El ajuste de hiperparámetros del clasificador SVM sobre el corpus publicado por Pereira-Kohatsu et al. [36] se realiza mediante el algoritmo *Random Search*. Este algoritmo consiste en entrenar el modelo con una combinación aleatoria de hiperparámetros por un número específico de iteraciones o hasta cumplir con un criterio de parada determinado. Este algoritmo se ejecutó durante 60 iteraciones, con los siguientes conjuntos de valores para los parámetros que se permitían según la implementación de *Scikit-Learn*[1], librería utilizada para obtener el modelo:

- *Kernel*: $\{Lineal, Polinomial, RBF, Sigmoide\}$,
- Gamma: $\{\frac{1}{cantidad\_de\_atributos}, 0.5, 1, 1.5\}$,
- Penalización: $\{1.0, 0.5, 1.5, 2.0, 3.0\}$,
- Tolerancia: $\{0.01, 0.001, 0.0001, 0.00001, 1e{-}5\}$

Dónde cada parámetro se define cómo:

- **Kernel**: Función que se aplica a los datos de entrada, transformando el espacio sobre el cual se ajusta el hiperplano.
- **Gamma**: Parámetro del núcleo que controla el grado en que se ajusta el *kernel* a cada ejemplo del conjunto de datos.
- **Penalización**: Parámetro de penalización de la función de costo.
- **Tolerancia**: Valor utilizado para determinar el momento en que se debe detener el proceso de entrenamiento.

La combinación de hiperparámetros que obtiene el mejor desempeño consiste en un kernel lineal, con una penalización de 1.0 y una tolerancia de 1e−5 (el parámetro *gamma* no aplica cuando el núcleo es lineal).

[1] https://scikit-learn.org

10.3. Resúmenes de trabajos seleccionados

10.3.1. Malmasi y Zampieri

En el año 2017, Shervin Malmasi y Marcos Zampieri [27] introducen un nuevo desafío al problema de identificar DO: el uso de lenguaje ofensivo. A partir de lo propuesto por Davidson et al. [11], se propone crear una línea base para conocer qué tanto influye el lenguaje ofensivo en el resultado de la clasificación. El corpus utilizado para las pruebas realizadas (generado por los autores del trabajo en el que se inspira el descrito aquí), el cual contiene 14509 tweets en inglés de los cuales 2399 tienen DO, 4836 tienen lenguaje ofensivo y los restantes no contienen ni DO ni lenguaje ofensivo. Los autores entrenan 14 modelos diferentes en donde todos utilizan un modelo SVM, variando la manera en la que se computan los vectores para cada tweet usando 4 métodos: *skip-gram*, *Bag-of-Words* de n-gramas de palabras, *Bag-of-Words* de n-gramas de caracteres y una combinación de todos los anteriores. Reportan los resultados en términos del acierto de cada modelo, obteniendo el mejor resultado mediante la utilización de 4-gramas de palabras. Luego, presentan la matriz de confusión para cada clase (tal como se ve en la Figura 10.1). A raíz de esta matriz se puede notar que, en efecto, el clasificador usualmente confunde tweets ofensivos pero sin DO con tweets que contienen DO. El trabajo además introduce un método para generar una cota superior teórica mediante un clasificador oráculo, esto es, un meta-clasificador que obtiene un acierto cuando al menos uno de los modelos generados clasificó correctamente el tweet.

10.3.2. Primeras aplicaciones de las redes neuronales

Los recientes avances en el aprendizaje profundo han realizado un gran aporte al avance en el área de la clasificación de texto. En particular, el uso de redes recurrentes ha demostrado ser uno de los modelos con mejores resultados. En el 2017, un grupo de investigadores del *International Institute of Islamic Thought* (Instituto Internacional del Pensamiento Islámico) publican uno de los primeros trabajos (Pinkesh Badjatiya et al. 2017 [1]) que utilizan redes neuronales para la detección automática de DO. El trabajo presenta primero un conjunto de modelos que sirven como línea base a partir de la que se comparan los modelos de clasificación. La línea base surge de combinar representaciones vectoriales simples de oraciones (como BoW y TF-IDF) con modelos de regresión logística, SVM y árboles de decisión con potenciación de gradiente (*Gradient Boosted Decision Trees* o *GBDT*), un tipo de clasificador dentro de los *ensemble methods*, que surge de aplicar el método *Gradient Boosting* (Llew Mason et al. 1999 [28]) con árboles de decisión. La arquitectura propuesta para el modelo clasificador consta de una capa de *embeddings* inicializada utilizando 2 métodos diferentes, combinando cada uno de estos con 3 clasifi-

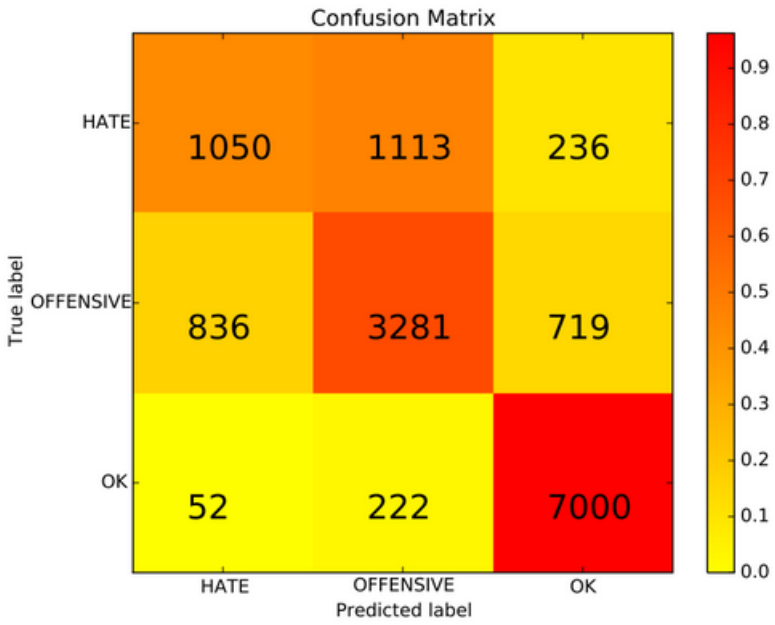


Figura 10.1: Matriz de confusión asociada al mejor modelo obtenido en [27]. El eje horizontal corresponde a la clasificación otorgada, mientras que el eje vertical corresponde a la etiqueta real. El mapa de calor corresponde a la proporción de tweet clasificados correctamente y también se muestra la cantidad actual de clasificaciones correctas

cadores basados en redes neuronales. No se detalla en profundidad la arquitectura de los clasificadores, pero se menciona que uno está formado por una CNN, otro por una LSTM y el último utiliza el modelo provisto por la librería *fastText* (Armand Joulin et al. 2016 [21]). La utilización de una capa de *embeddings* permite realizar un ajuste o *fine-tuning* de éstos, obteniendo representaciones que pueden funcionar mejor para la tarea de clasificación específica. En la siguiente tabla 10.1 se resumen los modelos presentados por los autores y los valores de las métricas reportadas, en base al entrenamiento y evaluación utilizando un conjunto de 16000 tweets en inglés construido por Zeerak Waseem y Dirk Hovy [40], de los cuales 3383 son sexistas y 1972 racistas:

Se puede observar que el modelo que presenta mejores resultados surge de la utilización de una arquitectura CNN e inicializando la capa de *embeddings* con vectores pre-entrenados de GloVe (Jeffrey Pennington et al. ([33]), una técnica que computa representaciones vectoriales de las palabras a partir de la probabilidad de que ocurran en el mismo contexto.

Al final del trabajo, se presentan resultados de una serie de clasificadores que combinan las representaciones obtenidas a partir del *fine-tuning* realizado al entrenar los modelos

Incialización de Embeddings	Modelo Clasificador	Precisión	Recall	F-Score
Aleatoria	CNN	0.813	0.816	0.814
GloVe	CNN	**0.839**	**0.840**	**0.839**
Aleatoria	FastText	0.824	0.827	0.825
GloVe	FastText	0.828	0.831	0.829
Aleatoria	LSTM	0.805	0.804	0.804
GloVe	LSTM	0.807	0.809	0.808

Tabla 10.1: Comparativa de modelos de clasificación generados por Pinkesh Badjatiya et al. en términos de Precisión, *recall* y *F-Score*

presentados anteriormente y un modelo GBDT. Combinando este modelo y los vectores del clasificador LSTM con inicialización de *embeddings* aleatoria supera al resto de los modelos con un resultado de 0.93 en precisión, *recall* y *F-Score*.

10.3.3. HatEval

Los avances en el área del PLN junto con el crecimiento de la comunidad asociada a ésta, ha dado lugar al surgimiento de competencias que concentran esfuerzos de equipos de investigadores provenientes de todas partes del mundo para resolver problemas conocidos del PLN. Una de las más populares es el *International Workshop on Semantic Evaluation*, mejor conocido como *SemEval*[2]. Habiendo abarcado 15 ediciones al momento de la realización de este trabajo, SemEval consiste en la resolución de problemas de PLN (como clasificación de texto, extracción de información, entre otros) orientados a un dominio en particular. La competencia es organizada en *tasks* o tareas, cada una correspondiente a un problema de PLN diferente. A su vez, cada tarea puede estar dividida en subtareas que proponen variantes en el problema principal a resolver. Para cada tarea, se hace público un conjunto de datos que se debe utilizar para resolver el problema planteado. Finalizada la competencia, se publica un documento que describe para cada tarea en qué consiste, el conjunto de datos correspondiente en términos de tamaño, características cualitativas como la proporción de clases y método de construcción, y se presentan los resultados destacados a modo de artículos dentro del documento. La última edición, correspondiente al año 2019, es de nuestro especial interés pues propone una tarea orientada a la detección de DO en *Twitter*, conocida como *hatEval*[3] (Valerio Basile et al. 2019 [2]). Para *hatEval* se proponen dos subtareas:

[2] `https://semeval.github.io/`

[3] `https://competitions.codalab.org/competitions/19935`

- La subtarea A consiste en clasificar tweets tanto en español como en inglés prediciendo si tienen DO (tweet con DO misógino o hacia inmigrantes específicamente).
- La subtarea B consiste en identificar, de los tweets con DO, si son agresivos y si son dirigidos hacia un individuo en particular o hacia un grupo.

Para el desarrollo de la tarea en tweets escritos en español, se distribuyeron un total de 6600 tweets en 3 *datasets*: uno de desarrollo, en el cual los participantes pudiesen desarrollar los primero modelos, uno de entrenamiento, donde se entrenarían los modelos presentados para la competencia y el último de evaluación, utilizado para obtener el puntaje final de cada modelo presentado. En la siguiente tabla se muestra la cantidad de tweets en cada clase para los conjunto de datos mencionados anteriormente:

Dataset	Total de Tweets	Tweets con DO
Desarrollo	500	222
Entrenamiento	4500	1838
Evaluación	1600	660
Total	**6600**	**2720**

Tabla 10.2: Cantidad de tweet totales y con DO de cada dataset distribuido para la tarea 5 de SemEval 2019

El conjunto de datos fue anotado mediante la utilización de la plataforma *Figure Eight* (que provee un servicio para *crowdsourcing* de anotación de conjuntos de datos, ahora denominada *Appen*[4]). El puntaje de acuerdo entre anotadores reportado para este conjunto es de 0.87, computado por la plataforma utilizando el puntaje de confianza (o *trust score*) de cada anotador, i.e. el porcentaje de acierto en un subconjunto de preguntas de test (una selección de preguntas que evalúan si el anotador es confiable).

10.3.4. Equipo Atalaya

El trabajo realizado por Juan Manuel Pérez et al. [34] en el año 2019, presenta un conjunto de modelos que resuelven ambas subtareas de *hatEval* en español, de los cuales uno de ellos alcanzó el primer puesto en la subtarea A y el cuarto puesto en la subtarea B. En la fase de preprocesamiento del tweets, se plantean dos metodologías: la primera es básica, se *tokeniza* el tweet, se reemplazan las menciones, URLs, e-mails y cadenas conformadas únicamente por letras repetidas por tokens especiales; la segunda, orientada al análisis de sentimiento, propone llevar todas las palabras a minúsculas, remover puntuación y *stopwords*, lematizar las palabras y agregar, cuando hay una palabra de negación que los precede, el prefijo `NOT_` a los 3 tokens posteriores o hasta que se encuentre un *token*

[4] `https://appen.com/`

que no corresponda a una palabra. Para conformar la representación del tweet, evaluaron utilizar las siguientes *features*:

- Vectores TF-IDF de unigramas y bigramas de palabras.

- Vectores TF-IDF de unigramas y bigramas de caracteres.

- Vectores *skip-gram* de tamaño 50 entrenados con *fastText* [21] con un corpus de 90 millones de tweet del cual no especifican el origen.

- Vectores contextualizados construidos utilizando la arquitectura ELMo (Matthew E. Peters et al. 2018 [35])

A partir de de estas *features* crean 3 clasificadores los cuales entrenan con el corpus de entrenamiento de *hatEval* [2] para las tareas A y B. Éstos consisten en:

- Un modelo que utiliza vectores ELMo consumidos por una capa LSTM bidireccional, seguido de una capa oculta de tamaño 64 y una del mismo tamaño con función de activación *softmax* para generar la predicción.

- Un modelo que surge de agregar al anterior una rama que recibe vectores TF-IDF que alimentan dos capas ocultas en serie de tamaño.

- Un modelo SVM que recibe una combinación de los vectores *Bag-of-Words*, *Bag-of-Characters* y *fastText*. Para generar la representación de los tweet a partir de los de vectores *fastText* de sus tokens, se calcula la media ponderada de los pesos en cada dimensión con un factor de ponderación $a = 0.8$.

Los modelos obtuvieron *F-scores* de 0.712, 0.721 y 0.730 respectivamente, siendo entrenados y evaluados sobre el conjunto de entrenamiento y de test de *hatEval*. Éstos resultados indican que para ciertas escenarios, clasificadores más simples como los que utilizan SVM pueden lograr mejores resultados que los basados en modelos más complejos como redes profundas.

10.4. Figuras

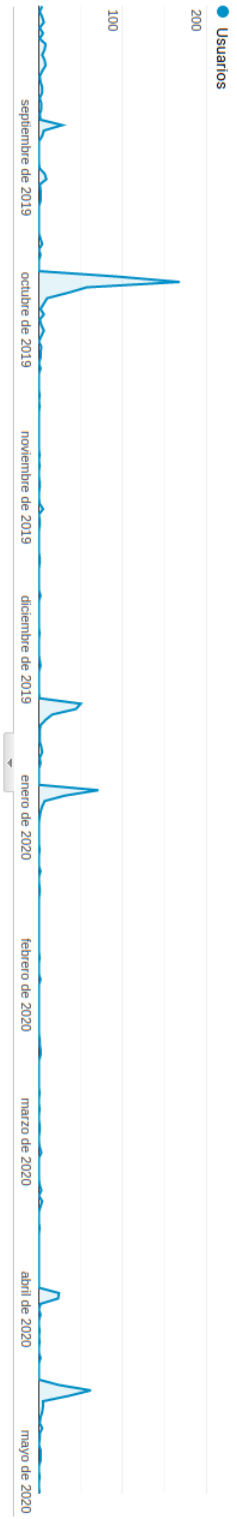


Figura 10.2: Accesos diarios a la aplicación de anotación desde el 4 de agosto de 2019 hasta el 9 de mayo de 2020