UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Detección de Cyberbullying en Redes Sociales

TRABAJO FIN DE MÁSTER

Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Víctor Botti Cebriá
Tutor: Ana Maria García Fornes
Cotutor: Elena del Val Noguera

Curso 2019-2020

Resum

La detecció del cyberbullying és un tema a tindre en compte en les xarxes socials a causa de la importància que té detectar aquest tipus de comportaments per a poder actuar el més ràpid possible. L'objectiu principal d'aquest treball és detectar de manera automàtica quins usuaris de xarxes socials estan sent assetjats perquè, d'aquesta forma, es puga prendre les mesures necessàries en cada cas. Per a això es planteja el desenvolupament d'una eina que permeta detectar usuaris embolicats en una conversa relacionada amb el cyberbullying. Per a fer aquesta tasca es farà ús de diferents classificadors clàssics, com poden ser Random Forest, SVM, Naive Bayes o k-veïns, i també classificadors més moderns basats en xarxes neuronals. També s'analitzarà l'ús de les característiques que apareixen en els perfils d'usuari per a millorar aquesta classificació. Finalment, el classificador s'integrarà en la xarxa social PESEDIA per a facilitar a l'administrador de la xarxa la detecció d'escenaris d'assetjament i els usuaris involucrats.

Paraules clau: Cyberbullying, xarxes socials, aprenentatge automàtic, intel·ligència artificial

Resumen

La detección del cyberbullying es un tema a tener en cuenta en las redes sociales debido a la importancia que tiene detectar este tipo de comportamientos para poder actuar lo más rápido posible. El objetivo principal de este trabajo es detectar de forma automática qué usuarios de redes sociales están siendo acosados para que, de esta forma, se pueda tomar las medidas necesarias en cada caso. Para ello se plantea el desarrollo de una herramienta que permita detectar usuarios envueltos en una conversación relacionada con el cyberbullying. Para realizar esta tarea se hará uso de distintos clasificadores clásicos, como pueden ser Random Forest, SVM, Naive Bayes o k-vecinos, y también clasificadores más modernos basados en redes neuronales. También se analizará el uso de las características que aparecen en los perfiles de usuario para mejorar esta clasificación. Finalmente, el clasificador se integrará en la red social PESEDIA para facilitar al administrador de la red la detección de escenarios de acoso y los usuarios involucrados.

Palabras clave: Cyberbullying, redes sociales, aprendizaje automático, inteligencia artificial

Abstract

The detection of cyberbullying is an issue to take into account in social networks because of the importance of detecting this type of behavior in order to act as quickly as possible. The main objective of this work is to automatically detect which social network users are being harassed so that, in this way, the necessary measures can be taken in each case. To this end, the development of a tool that allows users involved in a conversation related to cyberbullying is proposed. To carry out this task, different classic classifiers will be used, such as Random Forest, SVM, Naive Bayes or k-neighbors, and also more modern classifiers based on neural networks. The use of the features that appear in the user profiles will also be analyzed to improve this classification. Finally, the classifier will be integrated into the social network PESEDIA to facilitate the network administrator the detection of harassment scenarios and the users involved.

Key words: Cyberbullying, social networks, machine learning, artificial intelligence

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

1.1 Motivación

El bullying es la exposición que puede llegar a sufrir un niño a daños físicos y psicológicos por parte de otros [9]. Dada esta definición, cabe destacar que existen distintos tipos de este [9], los cuales afectan a los niños de distintas maneras y se llevan a cabo de formas diferentes, estos tipos de bullying son:

- Físico: este tipo de bullying es aquel en el que se hiere el cuerpo de una persona o sus posesiones, como por ejemplo empujando, pegando, o rompiendo sus posesiones.
- Verbal: se hace uso de la escritura o el habla para intimidad a otras personas.
- Indirecto: Se daña la reputación o las relaciones de otra persona.
- Marginación social: se excluye a una persona a propósito, ya sea no dejándole participar en las actividades o conversaciones, como haciendo que el resto de personas no se relacionen con esta.

A parte de estos tipos de bullying, existe uno más acorde con la época en la que vivimos, donde las redes sociales se utilizan día a día haciendo uso de las nuevas tecnologías, el cyberbullying. Este tipo de bullying tiene ciertas propiedades que hacen que sea más nocivo para aquellas personas que lo reciben, a diferencia del bullying tradicional, este se hace online, por lo que se puede recibir a través de distintos medios, ya sea por comentarios en vídeos, en fotos publicadas o en cualquier otra conversación. Estos ataques pueden ser recibidos en cualquier momento del día, debido a que con las nuevas tecnologías estamos siempre conectados y esto hace que las víctimas no tengan ningún momento para descansar. Los comentarios recibidos permanecen siempre en la red, a menos que se solicite eliminarlo.

Actualmente, el tiempo que las personas estamos conectados en las redes sociales está aumentando rápidamente [39], por lo que a su vez, también están aumentando los casos de cyberbullying a través de estos medios [39, 3]. Según los datos [18], aproximadamente la mitad de los adolescentes han recibido bullying en las redes sociales, 1 de cada 3 han recibido amenazas, un 25 % han recibidos ataques repetitivamente y solamente 1 de cada 10 lo ha hablado con sus padres, por lo que no se van a dar cuenta de lo que esta pasando. La detección temprana de este tipo de comportamientos es muy importante, debido a que estos tienen una gran influencia sobre las víctimas [28], las cuales pueden llegar a tener dificultades para aprender, depresión u otros problemas psicológicos, relaciones deterioradas e incluso el suicidio.

Debido a esto, es necesario disponer de herramientas para detectar automáticamente estas situaciones de cyberbullying. Este trabajo, realizado dentro del proyexto del GTI-IA Agentes inteligentes para asesorar en privacidad en redes sociales (TIN2017-89156-R) del Ministerio de Ciencia, Innovación y Universidades, plantea un estudio de como las técnicas de IA pueden utilizarse para abordar este problema, y se ha desarrollado una herramienta que se ha integrado en la red social PESEDIA. Además este trabajo se presenta como una ampliación al ya desarrollado [6] presentado en el congreso CISIS 2020 [1], en el cual se implementa una herramienta para combatir la revelación de información sensible en redes sociales.

1.2 Objetivos

El objetivo principal de este trabajo es detectar qué usuarios están envueltos en conversaciones donde se esté realizando cyberbullying. Para ello se plantea el desarrollo de una herramienta la cual se va a entrenar haciendo uso de distintos clasificadores y técnicas para obtener los mejores resultados clasificando textos procedentes de redes sociales. Para ello se van a utilizar distintas librerías de clasificadores clásicos, como librerías de redes neuronales y características contenidas en los perfiles de las redes sociales.

Para alcanzar este objetivo, se han planteado una serie de subobjetivos que se describen a continuación:

- Analizar propuestas similares realizadas en otros trabajos.
- Obtener datasets clasificados para la detección de cyberbullying en redes sociales.
- Analizar información sobre los perfiles de los usuarios que puede ser usada para la detección de cyberbullying
- Evaluar el funcionamiento de distintos tipos de clasificadores que tengan en cuenta el contenido de los mensajes y el contenido de la información de los perfiles de los usuarios para la detección de cyberbullying en textos de redes sociales.
- Integrar el mejor modelo obtenido en la red social PESEDIA.

1.3 Tecnologías empleadas

1.3.1. Software

Para realizar el entrenamiento de los modelos clásicos y de las redes neuronales se ha hecho uso del lenguaje de programación Python [2] y el IDE Pycharm [3] desarrollado por JetBrains [4].

En concreto para el entrenamiento de los modelos clásicos se ha hecho uso de la librería Scikit-learn[5]. Esta librería es de código abierto y se utiliza en el ámbito del aprendizaje automático. Esta incluye algoritmos de clasificación, regresión y análisis de grupos. En concreto para este proyecto se van a hacer uso de los clasificadores que vienen incorporados a la librería, métodos de preprocesado y métodos para el análisis de los resultados.

[1] http://2020.cisisconference.eu
[2] https://www.python.org/
[3] https://www.jetbrains.com/es-es/pycharm/
[4] https://www.jetbrains.com/
[5] https://scikit-learn.org/stable/

Para el caso del entrenamiento de las redes neuronales se va a hacer uso de la librería Keras [6] la cual esta desarrollada sobre Tensorflow [7]. Keras es una librería escrita en Python la cual está pensada para facilitar el uso de las redes neuronales. Esta librería soporta todo tipo de redes, entre las que se incluyen las convolucionales y las recurrentes. Además permite lanzar las redes tanto en GPU como en CPU.

Para integrar los modelos generados con la red social PESEDIA se desarrollarán dos partes. La primera de ellas será cargar los modelos desde un servicio web en Python, para atender las peticiones de clasificar textos. La segunda parte será un script de JavaScript y PHP, el cual se encargará de hacer las peticiones al servicio web.

1.3.2. Hardware

Para la realización de este proyecto se ha hecho uso de dos ordenadores. El primero de ellos hace uso del SO Windows 10 actualizado a la última versión, este ordenador hace uso de un procesador AMD Ryzen 7 3800x de 8 núcleos, una tarjeta gráfica NVIDIA 2070 y 16GB de memoria RAM a 3200 Mhz.

Por otro lado el portátil usado hace uso del sistema operativo macOS Catalina, este utiliza un procesador Intel Core i7 de 6 núcleos, una tarjeta gráfica AMD Radeon Pro 560x y 16 Gb de memoria RAM a 2400 MHz.

1.4 Estructura

Teniendo en cuenta las secciones ya presentadas en este trabajo: motivación, objetivos y tecnologías empleadas, el resto del documento se estructura como se muestra a continuación:

- El capítulo 2 hace un recorrido por el estado del arte, en concreto en trabajos previos sobre el cyberbullying, datasets sobre acoso y una breve explicación de los clasificadores utilizados en el trabajo.
- El capítulo 3 explica el desarrollo realizado para entrenar los modelos, los distintos experimentos que se han realizado y el mejor modelo obtenido.
- El capítulo 4 detalla cómo se ha implementado el mejor modelo obtenido para la clasificación de cyberbullying como un servicio web para integrarlo en la red social PESEDIA.
- El capítulo 5 contiene las conclusiones, donde se realiza un pequeño resumen del trabajo realizado en el proyecto y de los trabajos futuros.

[6] https://keras.io/
[7] https://www.tensorflow.org

CAPÍTULO 2
Estado del arte

En este capítulo se va a hacer una revisión de los trabajos que se han realizado previamente para la detección del cyberbullying en las redes sociales. A continuación, se explican los datasets que se han encontrado sobre este tema y una pequeña explicación de la historia de la red social. Por último, se realiza una breve explicación del funcionamiento de los clasificadores que se van a utilizar para la realización de este trabajo.

2.1 Trabajos previos

Con el desarrollo de las nuevas tecnologías de la información y la comunicación, y la rápida propagación de las redes sociales entre los adolescentes, se ha conseguido un nuevo medio por el cual establecer comunicación. En estas redes sociales es común ver interacciones en las que se ve envuelto contenido ofensivo, siendo esto una de las principales formas de expresar las agresiones en situaciones de acoso online, como sería el cyberbullying.

La detección automática de cyberbullying puede ser usada en herramientas digitales para prevenir e intervenir en situaciones de acoso online. Los clasificadores deben de ser lo más precisos posibles para detectar de forma automática y precisa los casos, por ello se han realizado estudios y trabajos en esta tarea.

Rosa et al. [36] hacen un repaso de todos los trabajos que se han ido realizando para lograr obtener buenos clasificadores de detección de acoso. Los autores proporcionan datos sobre los trabajos, los clasificadores que se han utilizado y los resultados obtenidos por los equipos de investigación. También hace un repaso de los datasets que han sido utilizados en esta tarea. En concreto no existe un dataset específico sobre el que se haya trabajado [36]. Se han ido usando distintos datasets existentes o los grupos de investigación han recogido sus datos y los han etiquetado para crear su propio dataset. De esta forma se han usado múltiples datasets de distintas redes sociales, como pueden ser *MySpace*, *Twitter*, *Youtube* o *Instagram*, usando prácticamente siempre textos en inglés.

Zhao et al. [43, 42] han realizado dos trabajos de clasificación de cyberbullying. Para ello han hecho uso de distintas técnicas para entrenar los clasificadores. Uno de los clasificadores que proponen es un Embeddings-Enhanced Bag-Of-Words Model (EBoW)[43] donde hacen uso de una Bag of Words, Latent Semantic Analysis (LSA) y palabras características del cyberbullying. También hacen uso de otro tipo de clasificador, Semantic-enhanced Marginalized Stacked Denoising Auto-encoder (smSDA) [42] donde obtienen

unos mejores resultados que en su anterior trabajo [43]. Los datasets utilizados para estos trabajos han sido textos recogidos de Twitter [1] y de MySpace [2].

Bayzick et al. [4] se hace una pregunta inicial: ¿qué es el cyberbullying y qué tipos existen? Una vez responden estas preguntas crean ciertas reglas para la detección del cyberbullying, entre las que se incluyen insultos, pronombres, letras mayúsculas y palabrotas. Haciendo uso de esto se realiza la implementación de un programa para la clasificación de textos de la red social MySpace, donde buscan trazas de acoso en las publicaciones y en los perfiles de usuarios.

Reynolds et al. [33] crean su propio dataset recogiendo datos de la red social FormSpring y clasificándolos a mano. Una vez realizado este paso, hacen uso de la herramienta *Weka*, un paquete de software utilizado para el machine learning, el cual proporciona una gran cantidad de algoritmos. Después de hacer una evaluación de los distintos algoritmos proporcionados seleccionan el algoritmo J48, con el que obtienen sus mejores resultados.

Huang et al. [22] hacen uso del dataset CAW 2.0, el cual incorpora alrededor de 900000 tweets de 27135 usuarios distintos, de los cuales, para la detección del cyberbullying, solamente van a usar aquellos textos que incluyan '@' debido a que estos serán los mensajes que mencionen a otra persona de la red social. Una vez ya tienen los textos que quieren, hacen uso de la misma herramienta que en el artículo anterior, *Weka*, y al igual que antes prueban cual de los algoritmos de la herramienta funciona mejor para la clasificación de su dataset, obteniendo los mejores resultados con el algoritmo *Dagging*.

Rosa et al. [35] hacen uso de la técnica Fuzzy Fingerprint para la clasificación de textos de acoso. En este artículo se explican los pasos que han seguido para elegir este método, y las comparaciones con los clasificadores más usados, como pueden ser SVM o Naive Bayes. Para la realización de este trabajo hacen uso de un dataset de FormSpring, el cual está muy desbalanceado y aplican distintas técnicas para balancearlo y obtener unos mejores resultados.

Sugandhi et al. [38] diseñan un sistema para la detección de cyberbullying. Para ello se recogen textos de la red social Twitter, los cuales son clasificados manualmente con la herramienta Amazon Mechanical Turk[3]. A estos textos se les hace un preprocesado y son pasados al clasificador el cual indicará si es un texto de acoso o no. En paralelo a esto, se realiza un análisis de sentimiento sobre los textos, de forma que haciendo uso de esta medida, se clasifican los textos de bullying en tres categorías distintas (alto, medio o bajo) dependiendo del sentimiento que tenga la publicación. En el trabajo utilizan este mismo proceso con distintos clasificadores. Finalmente, el clasificador con el que obtienen los mejores resultados es SVM.

Zhang et al. [41] hacen uso de una red convolucional para la clasificación de textos. La idea que se propone es entrenar esta red neuronal con la pronunciación de las palabras. Lo que hacen para lograrlo es, primeramente, realizar un preprocesado de los textos del dataset eliminando el máximo ruido para dejar los textos lo más simples posibles, y a continuación aproximan el texto a su pronunciación, se puede ver en la Figura 2.7 un ejemplo de como se realiza este paso a continuación.

[1] http://research.cs.wisc.edu/bullying/data.html
[2] http://www.chatcoder.com/DataDownload
[3] https://www.mturk.com/

Word and phrases	Phonetic code	Effect on the data
"fuck, fuc, fuk"	*f'Vk*	Positive
"fuckk"	*f'Vkk*	Neutral
"shitt, shit"	*S'It*	Positive
"suck, suk, suc"	*s'Vk*	Positive
"dik, dic, dick"	*d'Ik*	Positive
"guesss, guess"	*g'Es*	Positive
"bitchh, bitch, bich"	*b'ItS*	Positive
"cool, cooll"	*k'u:l*	Positive
"cum, come"	*k'Vm*	Negative

Figura (2.1) Conversión de las palabras a su representación fonética.

Lo que hacen los autores es pasar del texto a la representación fonética. Para ello hacen uso de herramienta eSpeak [4] un sintetizador de voz de código abierto. Una vez han realizado este preprocesado, prueban con varios tipos de redes neuronales, obteniendo los mejores resultados con una red convolucional.

Hani et al. [30] proponen un enfoque de aprendizaje automático supervisado para detectar el cyberbullying. Para ello hacen uso de un dataset de Kaggle[5], al cual le realizan un preprocesado en el que se tokenizan los textos, se eliminan las "stop words" y haciendo uso de la API de Bing corrigen las faltas ortográficas de los textos. Una vez hecho esto, comparan dos tipos distintos de clasificador, por un lado SVM y por otro una red neuronal, todo esto haciendo uso de distintos n-gramas.

Hosseinmardi et al. [39] implementan un clasificador LinearSVM para la detección de acoso. Para entrenar este clasificador hacen uso de datos extraídos de Instagram. En concreto lo que hacen es recoger los comentarios que realiza la gente en las distintas imágenes publicadas por los usuarios y la imagen publicada. A continuación clasifican cada una de las imágenes como negativa o positiva según los comentarios que haya recibido la publicación. De esta forma, entrenan el clasificador proporcionándole las imágenes, algunos comentarios del post, la fecha de publicación y propiedades del usuario. Para la obtención de los resultados hacen distintas pruebas pasando solamente parte de los datos, como pasarle solamente la imagen. Los mejores resultados los obtienen utilizando todos los datos posibles que han recogido de las publicaciones.

Se puede ver que, aproximadamente, todos los artículos que se han realizado para la detección del acoso son muy parecidos. En general, los trabajos de los artículos revisados hacen una búsqueda de un dataset o recogen sus propios datos. A continuación deciden que técnica utilizar y entrenan distintos clasificadores para comparar cual de ellos les da los mejores resultados. Algunos de estos artículos han profundizado más y han implementado técnicas más innovadoras, como puede ser el último articulo de Zhang et al. [41] haciendo uso de la pronunciación, el artículo de Hosseinmardi et al. [39] utilizando datos de los usuarios o el de Sugandhi et al. [38] clasificando los textos usando también un análisis de sentimiento.

Para poder comparar los distintos artículos que se han revisado en el estado del arte, y los resultados que han obtenido, se ha elaborado una tabla resumen (ver Tabla 2.1). En la tabla se muestra para cada trabajo qué dataset han utilizado, qué características se han

[4] http://espeak.sourceforge.net/
[5] https://www.kaggle.com/

utilizado para clasificar y las medidas para evaluar los resultados (i.e, precision, recall, F1 y ROC). En la tabla se puede observar que la gran mayoria de los trabajos sólo tienen en cuenta el texto para clasificar.

	Dataset	Texto	Sentimiento	Perfil usuario	Accuracy	Precision	Recall	F1	ROC
Zhao (CBoW)	Twitter	X	-	-	-	76.8	79.4	78.0	-
Zhao (smSDA)	Twitter	X	-	-	84.9	-	-	71.9	-
	MySpace				89.7	-	-	77.6	-
Reynolds (J48)	FormSpring	X	-	-	81.7	-	-	-	-
Huang (Dagging)	Twitter	X	-	-	-	-	-	-	0.755
Rosa (Fuzzy)	FormSpring	X	-	-	-	81.1	81.0	81.0	
Sughandi (SVM)	Twitter	X	X	-	91.31	0.91	0.91	0.9	-
Zhanf (Pronunciation)	Twitter	X	-	-	98.9	99.1	97.0	97.8	-
	FormSpring	X			96.8	74.0	45.3	56.2	-
Hani	FormSpring	X	-	-	91.76	92.4	91.7	91.9	-
Hosseinmardi	Instagram	X	-	X	-	78.0	72.0	75.0	-

Tabla (2.1) Comparativa de los trabajos analizados en el estado del arte.

Estos resultados sirven para orientarnos, pero no se pueden comparar entre ellos ni con nuestra propuesta. Esto se debe a que cada uno de los trabajos que se han realizado sobre el tema han sido implementados haciendo uso de datasets distintos, de redes sociales distintas, y muchos de ellos están desbalanceados, por lo que cada uno de estos trabajos puede obtener buenos resultados por el dataset utilizado o por los métodos implementados. Debido a esto, los anteriores trabajos y los resultados obtenidos se van a utilizar como orientación.

2.2 Datasets

El cyberbullying está presente en todas las redes sociales. A continuación se va a hacer un pequeño recorrido por las redes sociales donde se han realizado estudios sobre cyberbulling así como una revisión de los datasets que se han encontrado de cada una de estas redes y que a día de hoy están accesibles para experimentación.

2.2.1. FormSpring

FormSpring fue una red social que se inicio en Noviembre de 2009 por los desarrolladores de Formstack [6]. La idea de crear esta red social surgió ya que los usuarios que utilizaban Formstack creaban mucho temas del estilo ”Pregúntame lo que sea” de forma que decidieron crear una red social para llevar a cabo este tipo de temas.

Esta red social fue una de las más utilizadas, llegando a obtener el 28 de junio de 2011 un total de 25 millones de usuarios, y fue nombrado en febrero de 2012 como una de las redes sociales más innovadoras por FastCompany [19].

Uno de los problemas que tuvo esta red social entre los adolescentes fue el cyberbullying debido a que las publicaciones se podían realizar de forma anónima, y esto dio pie a que una gran cantidad de gente lo utilizara como medio para acosar a otras personas. Esto llegó a cobrarse la vida de dos adolescentes lo cual produjo que se realizara un boicot a esta red social.

Debido a esto la red social aconseja a los usuarios que eviten las preguntas anónimas, bloqueen a usuarios o en casos más graves se pongan en contacto con la policía.

[6] www.formstack.com

Finalmente esta red social tras un cambio de nombre a Spring.me pasó a ser propiedad de Twoo[7], donde se puede seguir haciendo uso de esta red social.

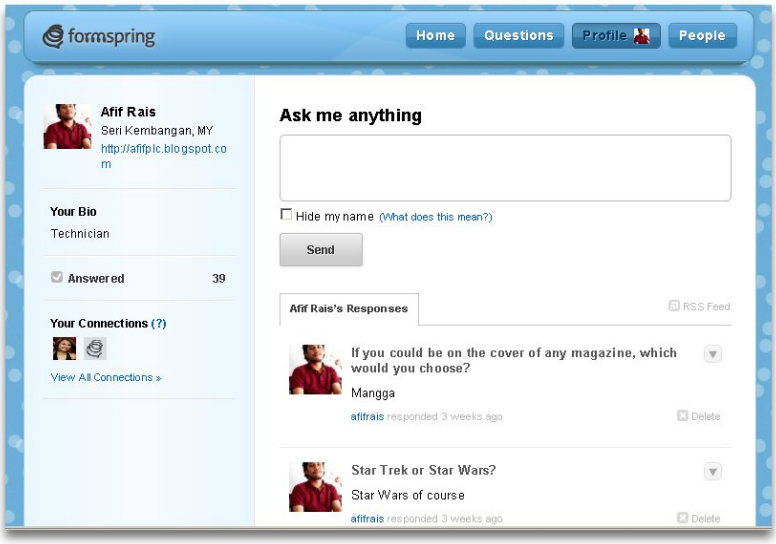

Figura (2.2) FormSpring.

Sobre Formspring se ha realizado un trabajo con su correspondiente aplicación llamada ChatCoder[10]. Esta aplicación es utilizada para analizar chats e identificar a las víctimas y acosadores en las redes sociales. Los autores de este trabajo han puesto en su página web los datos[8] que fueron utilizados para este trabajo. Estos datos fueron etiquetados haciendo uso de Amazon Mechanical Turk.

Los datos contienen la siguiente información del usuario:

- Bio: la biografía del perfil del usuario.
- Date: la fecha cuando se recupero el perfil.
- Location: localización dada del usuario.
- Userid: el identificador del usuario.

Y la siguiente información en cada una de las publicaciones:

- Text: la pregunta y la respuesta dada.
- Asker: el identificador del usuario que responde.

El etiquetado realizado por Amazon Mechanical Turk porporciona la siguiente información de cada una de las respuestas:

- Answer: YES o NO en caso de que haya o no cyberbullying
- Cyberbullyingwork: palabras que hayan hecho al etiquetador señalar el texto como cyberbullying.
- Severity: del 0 al 10 la gravedad del texto.

[7] https://www.twoo.com/
[8] https://chatcoder.com/Data/DataReleaseDec2011.rar

- Other: otros comentarios que haya querido dejar el etiquetador.
- Worktime: tiempo necesitado en segundos para clasificar el texto.
- Worker: el identificador del etiquetador.

Puede verse un ejemplo de uno de los datos del dataset a continuación:

```
<FORMSPRINGID>
  <BIO>Gema Loves Preston. :D</BIO>
  <DATE>20100731</DATE>
  <LOCATION>Jackson Michigan</LOCATION>
  <USERID>aguitarplayer94</USERID>
  <POST>
    <TEXT>Q: its funny how everyone hates you :DDDDD&lt;br&gt;A: uh huh not
        everyone prob the same fake person like u anonomouss;]</TEXT>
    <ASKER></ASKER>
    <LABELDATA>
      <ANSWER>Yes</ANSWER>
      <CYBERBULLYWORD>&quot;&quot;&quot;Everyone hates you&quot;&quot;&quot;</
          CYBERBULLYWORD>
      <SEVERITY>4</SEVERITY>
      <OTHER></OTHER>
      <WORKTIME>36</WORKTIME>
      <WORKERID>A111OICVQUW2B9</WORKERID>
    </LABELDATA>
    <LABELDATA>
      <ANSWER>Yes</ANSWER>
      <CYBERBULLYWORD>&quot;hates you, fake person like u&quot;</CYBERBULLYWORD
          >
      <SEVERITY>6</SEVERITY>
      <OTHER></OTHER>
      <WORKTIME>33</WORKTIME>
      <WORKERID>A1HDZE291YUBO5</WORKERID>
    </LABELDATA>
    <LABELDATA>
      <ANSWER>Yes</ANSWER>
      <CYBERBULLYWORD>hates you</CYBERBULLYWORD>
      <SEVERITY>3</SEVERITY>
      <OTHER></OTHER>
      <WORKTIME>7</WORKTIME>
      <WORKERID>A17AT6S84ZFYWG</WORKERID>
    </LABELDATA>
  </POST>
</FORMSPRINGID>
```

2.2.2. MySpace

Este red social fue creada por Thomas Anderson y Chris DeWolfe, lanzada en Agosto de 2003 con base en Beverly Hills, California. Esta red obtuvo en 2011 un total de 33.1 millones de visitantes solamente de EEUU.

Entre los años 2005 y 2008 MySpace fue la red social más visitada del mundo incluso sobrepasando a Google en el año 2006 en EEUU, aunque a partir de 2008 Facebook sobrepasó a MySpace y esta comenzó a tener una cantidad de usuarios mucho menor. Actualmente Meredith Corporation es la empresa que tiene los derechos de esta plataforma.

Figura (2.3) MySpace.

El dataset que se ha encontrado de esta red social ha sido creado por el mismo grupo que el de la anterior red social, los cuales desarrollaron la herramienta ChatEncoder. Al igual que el anterior dataset, este también se puede descargar fácilmente desde su página web[9]. Dispone de dos partes, una de ellas es un dataset sin etiquetar formado por datos de unos 120.000 usuarios y sus posts, y un dataset más pequeño el cual está etiquetado.

Cada uno de los posts de los usuarios en el dataset tienen la siguiente información:

- Username: nombre del usuario que hizo la publicación.
- Sex: sexo del usuario.
- Age: edad del usuario.
- City: ciudad del usuario.
- Province: información sobre el estado del usuario.
- Country: país del usuario.
- Date: fecha de publicación.
- Body: texto de la publicación.

Puede verse a continuación un ejemplo de uno de los posts que aparecen en el dataset:

```
−<post id="MS_1202_280568">
  −<user id="MS_3626836">
      <username>Helen is infinite</username>
      <sex>F</sex>
      <age>23</age>
      <city>Bloomington</city>
      <province>Indiana</province>
      <country>US</country>
   </user>
   <date>1095661620</date>
   <body>I am thinking about transferring to NCF. Anyone go there?</body>
</post>
```

[9] https://chatcoder.com/Data/BayzickBullyingData.rar

Se proporcionan distintos ficheros, donde cabe destacar "HumanConcensus.zip", donde aparecen las etiquetas que se le ha dado al post.

2.2.3. ASkfm

Esta red social fue fundada en Letonia para hacer competencia a FormSpring. Fue lanzada el 16 de Junio de 2010 y, al igual que FormSpring, se pueden hacer preguntas de forma anónima, dar me gusta a las respuestas y recibir preguntas. Esta red social ha llegado a superar a su competidor en tráfico generado en todo el mundo y tener a día de hoy alrededor de 70 millones de usuarios.

Al igual que se había comentado con FormSpring, AskFM tambien ha sido criticada por la prensa debido al cyberbullying que existe en la plataforma [17], debido a casos informados sobre pedofilia, mensajes ofensivos en modo anónimo y cuatro supuestos suicidios como resultado del acoso producido.

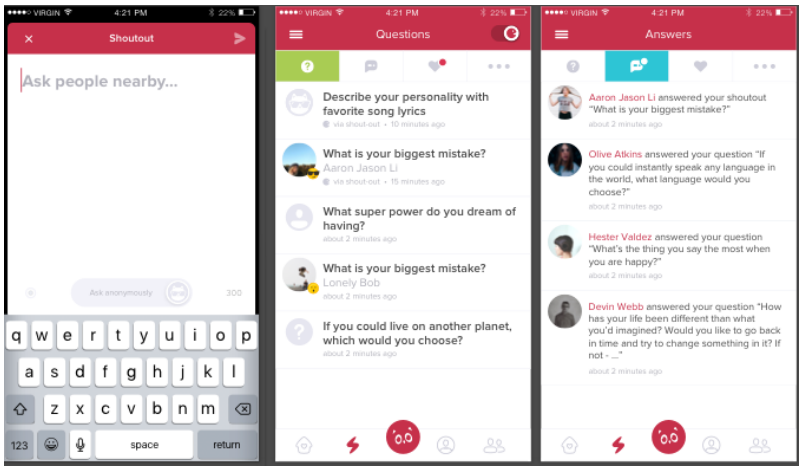

Figura (2.4) Interfaz de la red social ASKfm.

El dataset encontrado [24] sobre esta red se ha realizado haciendo búsquedas en el buscador de Google haciendo uso de términos concretos relacionados con el cyberbullying. De esta forma se encontraron los enlaces a publicaciones de la plataforma donde los usuarios recibían o producían estos ataques. Este dataset incluye información sobre 261.000 usuarios y aproximadamente 3 millones de preguntas y respuestas. El problema es que no se encuentra etiquetado. No se ha podido dar más información sobre este dataset debido a que no ha habido respuesta por parte de los creadores, pero se puede solicitar el acceso a este dataset a través de la página web[10] de los creadores.

2.2.4. Instagram

Instagram es una aplicación y red social creada por Kevin Systrom y Mike Krieger en octubre de 2010, llegando a tener más de 100 millones de usuarios activos en 2012. Solamente 2 años después de su lanzamiento, Facebook adquirió la compañía por un total 1000 millones de dolares. En enero de 2020 según datos de DataReportal [16], Instagram era una de las redes sociales más utilizadas, con un total de 1000 millones de usuarios activos.

Esta red social es actualmente una de las redes sociales más famosas entre los jóvenes, basada en la publicación de imágenes, a las cuales otros usuarios pueden publicar

[10] https://sites.google.com/site/cucybersafety/home/cyberbullying-detection-project/dataset

mensajes. También incorpora un chat directo entre usuarios. Instagram es considerada la red social donde más casos de cyberbullying se reportan [21].

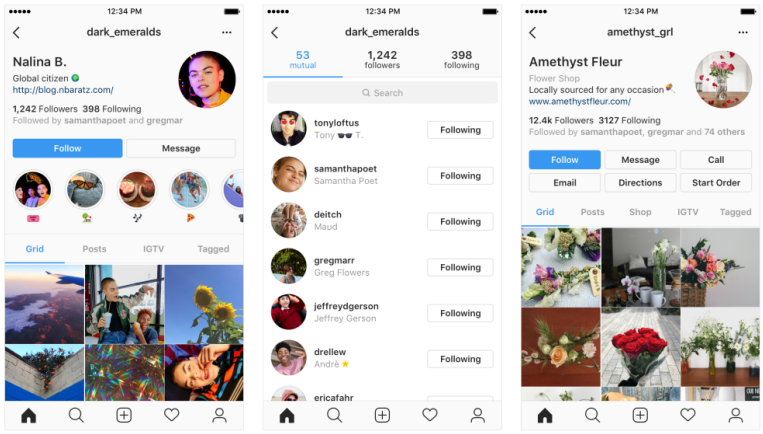

Figura (2.5) Interfaz de la red social Instagram.

El dataset encontrado de Instagram [24] incorpora aproximadamente 25.000 perfiles de usuarios donde cada uno de estos perfiles incluye las fotos que el usuario ha publicado y los últimos 150 comentarios que se han realizado a su publicación, el número de seguidores del usuario y los identificadores de usuarios que han compartido la publicación. Dispone de dos versiones, una etiquetada y otra sin etiquetar. Al igual que en el dataset de AskFm, no se ha obtenido respuesta por parte de los creadores, pero se puede solicitar el acceso a este dataset a través de la página web[11].

2.2.5. Vine

Vine fue una red social creada por Dom Hofmann y Rus Yusupov en junio del 2012 y comprada en octubre del mismo año por Twitter. Esta aplicación consistía en la creación de vídeos de una duración máxima de 6 segundos, donde la creación de vídeos divertidos hizo que creciera rápidamente, aunque finalmente, debido a que el número de usuarios de esta aplicación fue decayendo y que surgieron otras aplicaciones parecidas, fue cerrada finalmente el 17 de enero de 2017.

Figura (2.6) Interfaz de la red social Vine.

[11] https://sites.google.com/site/cucybersafety/home/cyberbullying-detection-project/dataset

El dataset [32] encontrado de esta red social incluye información de 60.000 usuarios, de los cuales se tiene información de todas sus publicaciones y de los comentarios que se han realizado a sus vídeos. Al igual que anteriormente, no se ha obtenido respuesta por parte de los creadores, aun así se puede solicitar el acceso a este dataset a través de la página web[12] de los creadores.

2.2.6. Twitter

Twitter es una red social creada por Jack Dorsey en 2006 donde los usuarios hacen publicaciones e interactúan entre ellos con mensajes cortos de hasta 140 caracteres. Esta red social es usada también por los medios y empresas para comunicar noticias y otras publicaciones. Esta red social incluye una gran cantidad de contenido, por lo que existen una gran variedad de datasets sobre esta red social.

Figura (2.7) Interfaz de la red social Twitter.

Varios autores han creado datasets para la detección automática de cyberbullying a partir de información extraída de twitter.

Rezvan et al. [34] crearon un dataset de unos 50.000 tweets recogidos entre 2016 y 2017, de los cuales aproximadamente la mitad se encuentran etiquetados en las siguientes categorías:

- Sexual: si el texto acosa haciendo uso de la sexualidad de la persona.
- Racial: si contiene acoso relacionado con la raza de la víctima.
- Appearance-related: si existe acoso sobre la apariencia.
- Intellectual: si hay acoso sobre lo que sabe o no la víctima.
- Político: si hay acoso sobre las ideologías de la víctima.

Además también crearon un léxico de 700.000 palabras que están consideradas como ofensivas y pueden ser utilizadas para el desarrollo de clasificadores.

[12] https://sites.google.com/site/cucybersafety/home/cyberbullying-detection-project/dataset

Xu et al. [40] crearon un dataset[13] abierto el cual incluye aproximadamente 7.000 tweets recogidos con la API de Twitter. En este dataset cada uno de los textos está etiquetado de forma que se puede saber quien es la víctima, el acosador o el defensor, por lo que es un dataset muy útil a la hora de poder detectar trazas de acoso en las redes sociales. Este dataset incluye los identificadores de los tweets, con los que se pueden recuperar los datos haciendo uso de la API de Twitter.

Founta et al. [11] recogieron unos 100.000 tweets, donde todos ellos fueron clasificados en 4 categorías:

- Abusive: en caso de que el texto contenga abuso hacia una persona.
- Hateful: en el caso de que en el texto haya odio.
- Spam: si el texto se trata de spam.
- Normal: si no incluye rastros de acoso.

El dataset no incluye los textos de las publicaciones, solamente los identificadores, por lo que para acceder a los datos se debe de utilizar la API de Twitter.

Estos dos últimos datasets serán utilizados para el desarrollo del clasificador. Estos datasets se han elegido debido por un lado, la red social sigue usándose a día de hoy, y por otro lado, contienen una gran cantidad de publicaciones las cuales siguen estando disponibles en la red.

Twitter actualmente está tomando mayores medidas contra todo lo relacionado con el acoso [20], por lo que muchos de los mensajes más antiguos que puedan haber sido recogidos por los datasets puede que hayan sido eliminados, por esto es aconsejable utilizar los datasets más recientes.

2.2.7. Resumen datasets

Para finalizar este apartado sobre los datasets encontrados, se ha creado una tabla en la que se muestran las muestras que dispone cada uno de los datasets, sus autores, la red social en la que se han recogido los datos y la forma de obtener el dataset.

Red social	Autor	Muestras	Accesibilidad
Formspring	Edwards et al. [10]	+19K	https://chatcoder.com
Myspace	Edwards et al. [10]	+128K	https://chatcoder.com
ASKfm	Kao et al. [24]	+3M	cucybersafety@gmail.com
Instagram	Kao et al. [24]	4M	cucybersafety@gmail.com
Vine	Rafiq et al. [32]	650K	cucybersafety@gmail.com
Twitter	Rezvan et al. [34]	50K	mohammadrezarezvan94@gmail.com
	Xu et al. [40]	8K	https://research.cs.wisc.edu/bullying/
	Founta et al. [11]	100K	a.m.founta@gmail.com

Tabla (2.2) Datasets encontrados sobre cyberbullying

Como se ha mencionado anteriormente, los datasets que se van a utilizar en este trabajo son los de Xu et al. y Founta et al. que aunque se puede ver que tienen una menor cantidad de muestras con respecto a los otros datasets, Twitter es una red social que todavía está en uso, no como algunas de las redes de otros datasets. Si comparamos los datasets elegidos con el que se ha descrito de Instagram, los obtenidos de Twitter tienen

[13] https://research.cs.wisc.edu/bullying/data.html

la ventaja de que se puede acceder fácilmente a toda la información de los perfiles de los usuarios, algo que en caso de Instagram se complica.

2.3 Conceptos previos

En este apartado se va hacer una pequeña explicación de cada uno de los clasificadores que se van a utilizar para la realización del proyecto.

2.3.1. Random Forest

Random Forest es un clasificador que pertenece a la familia de métodos de aprendizaje, para clasificación y regresión, basados en arboles, que incluyen un grupo de técnicas supervisadas no paramétricas que permiten segmentar el espacio de los predictores en regiones simples donde es más sencillo predecir la variable respuesta.

En concreto, este clasificador se basa en construir una combinación de arboles predictores (ver Figura 2.8). Cada árbol depende de un vector aleatorio y ha sido probado de forma independiente en el momento del entrenamiento, y como resultado da la clase (clasificación) o la predicción media (regresión) de los arboles individuales [13, 14].

El método de Random Forest es una modificación del proceso de bagging, el cual consigue mejores resultados debido a que decorrelaciona los árboles generados en el proceso. Este proceso de bagging se basa en que promediando un conjunto de modelos se consigue reducir la varianza, esto se cumple siempre que los modelos agregados no estén correlacionados. Si la correlación es alta, la reducción de la varianza será pequeña.

Random Forest y bagging hacen uso del mismo algoritmo, con la única diferencia de que en Random Forest antes de cada división se seleccionan de forma aleatoria m predictores. La diferencia en el resultado dependerá del valor m escogido. Si $m = p$, donde p es el número total del predictores, los resultados de Random Forest y bagging serán iguales. Un valor recomendado es $m = p\sqrt{p}$ [14]. Si los predictores están muy correlacionados, valores pequeños de m consiguen mejores resultados.

El primero de estos algoritmos fue desarrollado por Tin Kam Ho [13], utilizando el método del subespacio aleatorio [14], que, en la formulación de Ho, es una forma de aplicar el enfoque de discriminación estocástica a la clasificación propuesto por Egene Kleinberg [25, 26, 27].

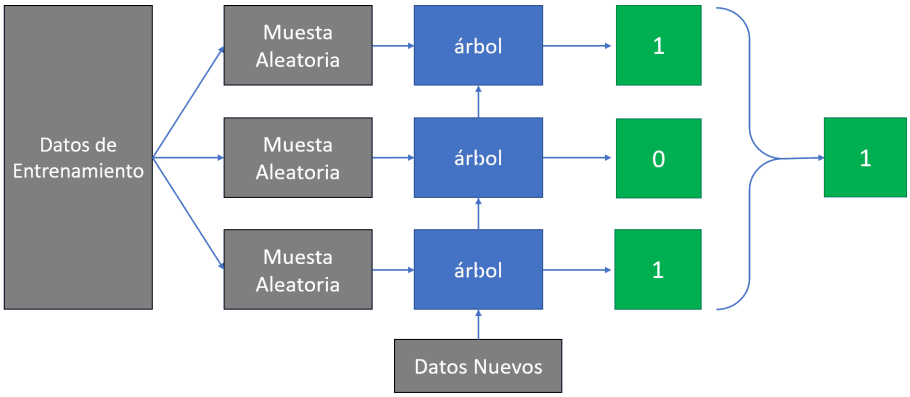

Figura (2.8) Funcionamiento de Random Forest

Las ventajas de estos tipos de clasificadores son: los árboles son fáciles de interpretar, aun cuando las relaciones entre predictores son complejas, los árboles pueden manejar

tanto predictores cuantitativos como cualitativos sin tener que crear variables extra, al ser no paramétrico no hace falta que se cumpla ninguna distribución específica, requieren una menor limpieza y preprocesado de los datos comparado con otros clasificadores, son muy útiles en la exploración de datos y son capaces de seleccionar predictores de forma automática. Como desventajas: cuando tratan con variables continuas pierden parte de su información al categorizarlas en el momento de la división de los nodos, por esta razón suelen ser modelos que consiguen unos mejores resultados en clasificación que en regresión.

2.3.2. Naive Bayes

Un clasificador Naive Bayes se trata de un método probabilístico basado en el teorema de Bayes, en el que se asume la hipótesis de independencia de las variables predictoras (de ahí el término Naive), la presencia o ausencia de una característica no esta relacionada con la presencia o ausencia de otra. Este método es una técnica de clasificación y predicción supervisada donde se predice la probabilidad de posibles resultados. De forma general la estimación de los parámetros de los modelos bayesianos utilizan el método de máxima verosimilitud. Se puede trabajar con el modelo naive bayes sin aceptar la probabilidad bayesiana.

El teorema de bayes está definido por la siguiente ecuación:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

- P(A) es la probabilidad a priori, es la forma de añadir conocimiento previo sobre los valores que puede tomar la hipótesis.
- P(B | A) es la verosimilitud de una hipótesis A dado B.
- P(B) es la verosimilitud marginal, la probabilidad de observar B promediados sobre todas las posibles hipótesis H
- P(A | B) es la probabilidad a posteriori.

El siguiente ejemplo, ver Figura 2.9 (tomado de la asignatura Sistemas Inteligentes) nos permite mostrar los conceptos y métodos utilizados en este tipo de clasificadores. :

Variables aleatorias de interés:

$Dolor:\quad D \in \{0,1\}$

$Caries:\quad C \in \{0,1\}$

$Hueco:\quad H \in \{0,1\}$

Representación:

$P(D = d, C = c, H = h)$

d	c	h	P
0	0	0	0,576
0	0	1	0,008
0	1	0	0,144
0	1	1	0,072
1	0	0	0,064
1	0	1	0,012
1	1	0	0,016
1	1	1	0,108
Suma:			1,000

Figura (2.9) Ejemplo del dentista Naive Bayes

Donde D representa tener dolor (0 = no dolor, 1 = dolor), C si se observan caries y H si hay algún hueco. En la Figura 2.9, la tabla de la derecha representa la información que ha obtenido el dentista examinando distintos pacientes, donde por ejemplo, el primer

registro de la tabla significa que el 57,6 % de los pacientes no padecían de dolor, no tenían caries ni tenían hueco. Otro caso sería que el 8 % de los pacientes no tenían dolor ni caries, pero si que tenían hueco.

Primero necesitamos las reglas de la suma (2.2) y el producto (2.3), las cuales pueden verse a continuación

$$P(x) = \sum_{y} P(x, y) \tag{2.2}$$

$$P(x, y) = P(x)P(y|x) \tag{2.3}$$

Podemos calcular primeramente la probabilidad de observar un hueco, la cual seria la siguiente:

$$P(h = 1) = \sum_{d=0,1} \sum_{d=0,1} P(d, c, h = 1) = 0{,}008 + 0{,}072 + 0{,}064 + 0{,}108 = 0{,}200 \tag{2.4}$$

O calcular la probabilidad de observar una caries tras haber observado un hueco:

$$P(c = 1|h = 1) = \frac{P(c = 1, h = 1)}{P(h = 1)} = \frac{0{,}072 + 0{,}108}{0{,}200} = 0{,}900 \tag{2.5}$$

A partir de la información de la tabla y aplicando Bayes, se puede calcular la probabilidad de que cuando se observe a un paciente con dolor tenga una caries. Queremos saber $P(c = 1|d = 1)$ sabiendo que $P(c = 1) = 0{,}34$, $P(d = 1) = 0{,}20$ y $P(d = 1|c = 1) = 0{,}36$. Aplicando el teorema de Bayes la probabilidad de caries sabiendo que hay dolor es la siguiente:

$$P(c = 1|d = 1) = P(c = 1)\frac{P(d = 1|c = 1)}{P(d = 1)} = 0{,}34 * \frac{0{,}36}{0{,}20} = 0{,}61 \tag{2.6}$$

Para obtener un clasificador basado en Naive Bayes lo que hace falta es añadir una regla de decisión, la cual consiste en elegir la hipótesis más probable o de mínimo riesgo de error. A continuación puede verse esta última regla:

$$\forall_y \epsilon Y : d * (y) = \arg \max_{d \epsilon D} P(d|y) \tag{2.7}$$

Las ventajas de hacer uso de este clasificador son las siguientes. La primera y más significativa, como se ha mencionado anteriormente, es que es un clasificador muy fácil de implementar, por lo que es una manera fácil y rápida de predecir clases para ciertos problemas de clasificación binarios y multiclase. Este clasificador funciona mejor que otros modelos de clasificación en aquellos casos donde haya independencia en los datos.

Aunque estos son unos clasificadores bastante buenos, también son conocidos por ser pobres estimadores, por lo que los resultados obtenidos por estos pueden no ser realistas. La presunción de independencia que se toma no refleja como son los datos en el mundo real.

2.3.3. Support Vector Machines

Las Support Vector Machines o en castellano máquinas de vectores de soporte, fueron desarrollados por Vladimir Vapnik [5, 7] en la década de los 90 en el campo de la ciencia

computacional. Si bien en sus inicios fueron pensados para ser usados como un método de clasificación binaria, su aplicación se ha extendido para los problemas de clasificación múltiple y regresión. Como resultado, se ha visto que se tratan de unos de los mejores clasificadores para una gran cantidad de problemas por lo que es considerado como uno de los métodos referentes en el ámbito del aprendizaje estadístico y el machine learning. Las SVM se basan en el Maximal Margin Classifier [37], que a su vez, se basa en el concepto de hiperplano.

El objetivo del algoritmo de SVM es encontrar un hiperplano en un espacio de dimensiones N (N - el número de características) que clasifique claramente los datos. Viendo la Figura 2.10, para separar las 2 clases de datos, hay muchos hiperplanos posibles que podrían ser elegidos. Nuestro objetivo es encontrar un plano que tenga el margen máximo, es decir, la distancia máxima entre los puntos de datos de ambas clases. Maximizar el margen proporciona un refuerzo para que los futuros datos puedan ser clasificados con más confianza.

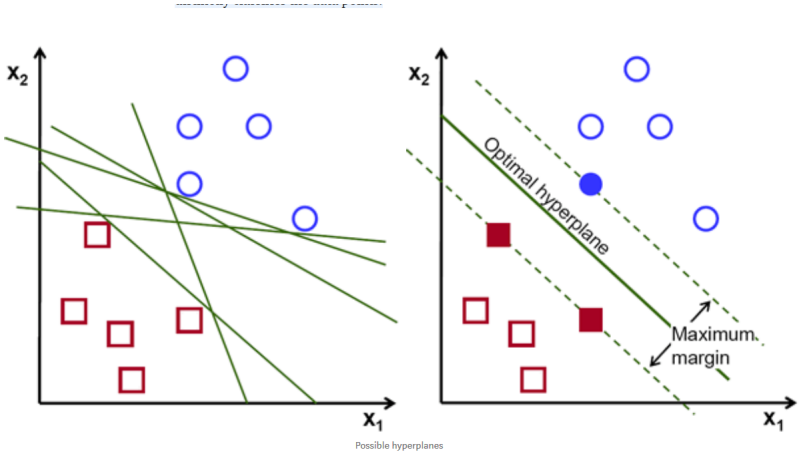

Figura (2.10) Clasificación de dos tipos de datos con SVM

Cómo se muestra en [23] cuando el límite de separación entre clases no es aproximadamente lineal una estrategia es expandir las dimensiones del espacio original.

El hecho de que los grupos no sean linealmente separables en el espacio original no significa que no lo sean en un espacio de mayores dimensiones. En la figura 2.11 se muestra como dos grupos, cuya separación en dos dimensiones no es lineal, sí lo es al añadir una tercera dimensión.

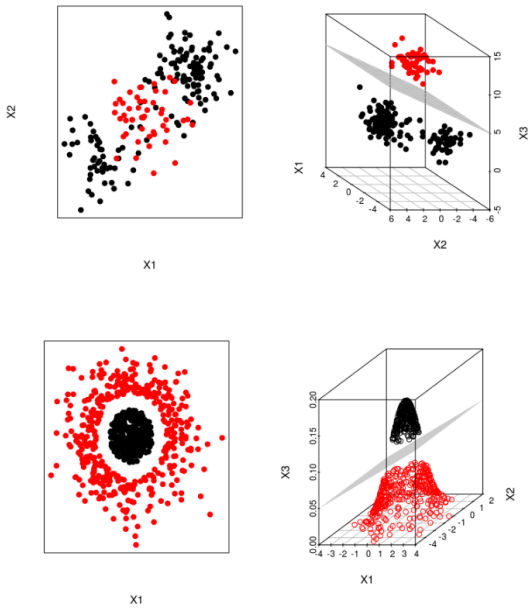

Figura (2.11) Ejemplo clasificación SVM

Las Máquinas Vector Soporte (SVM) son una extensión del Support Vector Classifier aumentando la dimensión de los datos. Lo que se consigue es que los límites de separación lineales generados en el espacio aumentado se conviertan en límites de separación no lineales cuando se proyectan en el espacio original.

Las ventajas que existen al hacer uso de este tipo de clasificadores son, principalmente, que es muy eficaz con muestras de entrenamiento de grandes dimensiones, incluso cuando el número de dimensiones es mayor al número de muestras. Hace uso de subconjuntos de puntos de entrenamiento (vectores de soporte) por lo que también es eficiente en memoria.

Como desventajas, si el número de características es mucho mayor que el de muestras, se debe de evitar el exceso de ajuste, y además, este clasificador no proporciona directamente las estimaciones de probabilidad, sino que se calculan realizando una validación cruzada.

2.3.4. K-Vecinos

Este método se trata de un algoritmo de clasificación supervisado basado en el reconocimiento de patrones criterios de vecindad [8]. Parte de la idea de que una nueva muestra será clasificada en la clase donde pertenezcan la mayor cantidad de vecinos más cercanos del conjunto de entrenamiento. Es un método de clasificación no paramétrico en el que se estima la probabilidad a posteriori de que un elemento x pertenezca a un clase C_i a partir de la información que nos da el conjunto de prototipos.

El método original se trata de "Nearest Neighbour" (NN) propuesto por Thomas Cover [8] o lo que seria lo mismo 1-Vecino, el cual solamente se fija en el vecino más cercano, por lo que cuando llega una nueva muestra se explora todos los datos almacenados de la fase de entrenamiento (los ejemplos de entrenamiento son vectores en un espacio multidimensional, cada ejemplo viene definido por n parámetros y la clase a que pertenece del conjunto de C clases considerado), se calcula la distancia a los datos haciendo uso de distintas fórmulas para obtener las distancia, por ejemplo la distancia Euclídea (ver fór-

mula 2.8), la distancia Manhattan (ver fórmula 2.9) o la distancia Minkowski (ver fórmula 2.10), y el nuevo dato se clasifica en la misma clase que el más cercano.

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} \tag{2.8}$$

$$d = \sum |x - y| \tag{2.9}$$

$$d(X, Y) = (\sum_{i=1}^{n} |x_i - y_i|^P)^{\frac{1}{P}}) \tag{2.10}$$

Este método NN no aprovecha de forma eficiente toda la información de los datos de entrenamiento por ello se utiliza normalmente k-vecinos, donde se comprobaran las clases a las que pertenecen los k vecinos más cercanos al dato a clasificar, y según donde estén los vecinos el clasificador decidirá si el dato pertenece a una clase u otra. Para poder entender mejor el funcionamiento de este clasificador se va a realizar un ejemplo.

En la Figura 2.12 hay un total de n=10 datos, donde 5 pertenecen a la clase C_A y 5 a la clase C_B. Si ahora añadimos un nuevo dato m, representado como la estrella roja, si elegimos que k = 3, al calcular las distancias se obtiene que sus vecinos son 1 elemento de la C_A y dos de la C_B, por lo que el dato m se clasificaría como C_B, pero en el caso en que se elija un valor k distinto, k = 6, una vez se han calculado los 6 vecinos más cercanos obtenemos que el dato m pertenece a la clase C_A

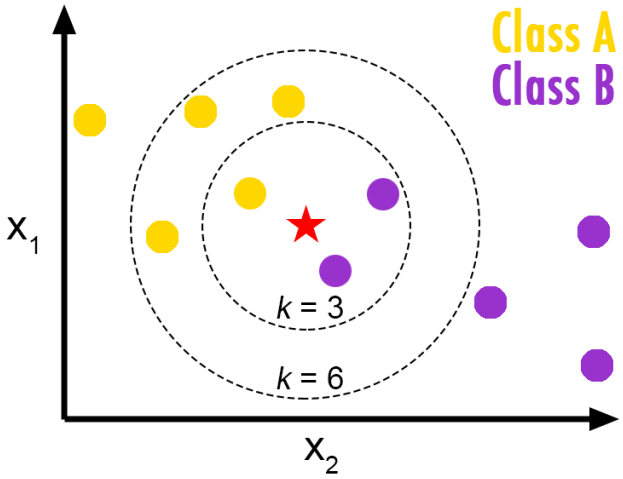

Figura (2.12) Clasificación de un dato con K-vecinos

Las ventajas de este clasificador son, primeramente, que no es paramétrico, por lo que no hace suposiciones, es simple y tiene una precisión bastante alta. Como desventajas de este cabe destacar que está basado en instancias por lo que no aprende explícitamente el modelo, es computacionalmente costoso y necesita una gran cantidad de memoria para almacenar los datos de entrenamiento. Otro inconveniente de este método es que se basa en la suposición de que los vecinos más cercanos son los que proporcionan la mejor clasificación, para ello utiliza todos los parámetros de los vectores de datos, el inconveniente de esta aproximación es que, en el caso en que el vector tenga muchos parámetros aquellos que son más relevante pierden peso entre los que son menos relevantes. Esto se puede corregir asignando pesos a las distancias de cada parámetro para que los más relevantes tengan mayor importancia (vecinos más cercanos con distancia ponderada), otra solución es identificar y eliminar los parámetros menos relevantes.

2.3.5. Redes Neuronales Artificiales

Una red neuronal artificial es un modelo matemático el cual está inspirado en el comportamiento biológico de las neuronas y en como están organizadas formando la estructura del cerebro. Existen distintos tipos de neuronas biológicas. En la Figura 2.13 se puede ver un esquema simplificado de una neurona, donde aparecen tres partes importantes:

- Soma: el cuerpo central donde se encuentra el núcleo celular.
- Axón: una prolongación del soma.
- Dendritas: un conjunto de ramificaciones terminales.

Aunque no se puede ver en la imagen, la conexión que existe entre neuronas se le llama sinapsis. Esta conexión no se realiza de una manera física, sino que es un espacio de alrededor de 2mm de separación entre neuronas por donde se establece una conexión unidireccional. La manera que tenemos los seres vivos de responder ante los estímulos del mundo exterior y el aprendizaje que realizamos del mismo esta relacionado con estas conexiones neuronales, y esto es lo que las redes neuronales artificiales quieren emular.

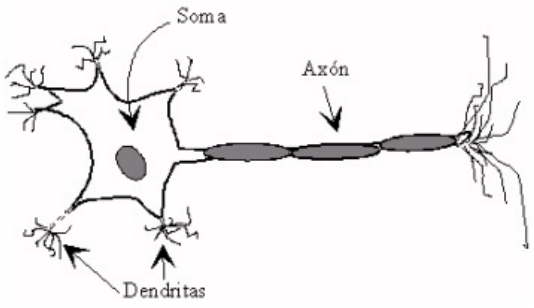

Figura (2.13) Representación de una neurona [15]

El primer modelo matemático de una red neuronal artificial fue creado en un trabajo conjunto entre Warren McCulloch y Walter Pits [29], se puede ver un resumen de su trabajo en la Figura 2.14.

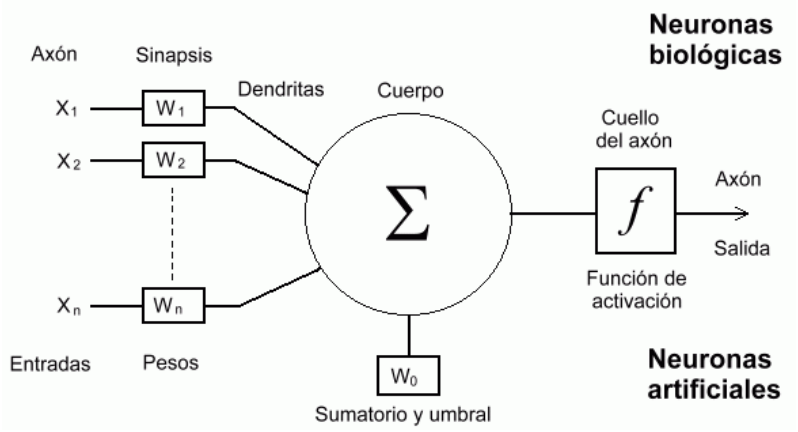

Figura (2.14) RNA McCulloch y Pits.

En la imagen puede verse la relación que existe entre cada una de las partes del modelo matemático desarrollado y una neurona biológica. En concreto, la figura muestra un modelo neuronal con n entradas, donde:

- Existe un conjunto de datos de entrada $x_1, ...x_n$.
- Los pesos sinápticos $w_1, ...w_n$ de cada entrada.
- Una función de agregación $\sum$.
- Una función de activación f.
- Una salida Y.

El conjunto de datos de entrada $x_1, ...x_n$ son los estímulos que recibe una neurona desde el mundo exterior, y la salida es la respuesta a esos estímulos. Estas neuronas son capaces de adaptarse al medio y aprender de el modificando los valores de los pesos sinápticos, también conocidos como parámetros libres, ya que pueden ser modificados y adaptados para distintas tareas.

En concreto en este trabajo se va a hacer uso de las redes neuronales LSTM (Long-Short Term Memory) [30] las cuales son una extensión de las redes neuronales recurrentes, por lo que se va a empezar explicando este tipo de redes.

Las redes neuronales recurrentes o Recurrent Neural Networks (RNN) fueron ideadas ya en los años 80, pero debido a los grandes costes computacionales no se han podido utilizar hasta los avances en la tecnología de los últimos años, donde ahora ya son más accesibles y se han popularizado.

Las redes neuronales que se han descrito anteriormente solo actúan en una dirección (unidireccionales), desde la capa de entrada hacia la capa de salida, por lo que no recuerdan valores previos. Una RNN es bastante parecida a estas pero incluye conexiones que apuntan hacia atrás, realizando una retroalimentación entre las neuronas.

La RNN más simple posible puede verse en la figura 2.15, donde se puede ver que la neurona tiene como datos de entrada X, y produce como salida Y, enviando esta salida como entrada a si mismo.

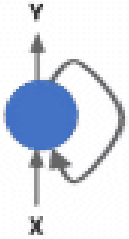

Figura (2.15) RNN simple

En cada instante de tiempo esta neurona recibe la entrada de la capa anterior además de su propia salida del instante de tiempo anterior, una forma de representar esto en una pequeña red algo más compleja que la anterior seria la que se muestra en la Figura 2.16.

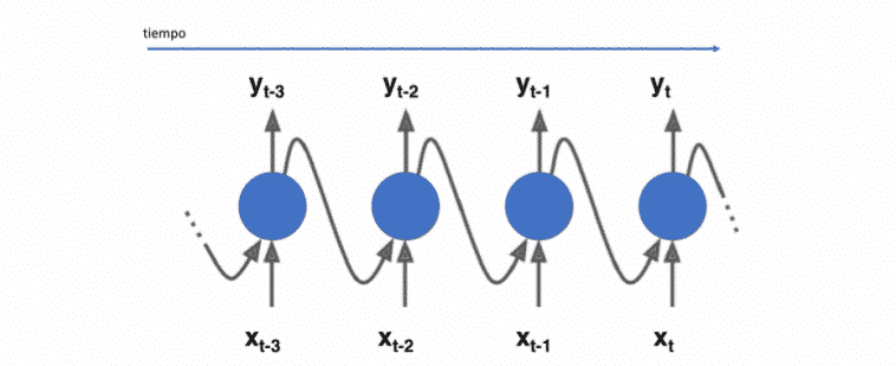

Figura (2.16) Red neuronal RNN

Una capa de neuronas recurrentes se puede implementar de forma que en cada instante de tiempo una neurona reciba dos datos de entrada. Por un lado los datos de la capa anterior y por otro lado la salida de la propia capa en el instante de tiempo anterior.

Como se ha mencionado al principio, las LSTM son una extensión de este tipo de redes neuronales recurrentes, donde básicamente lo que se hace es ampliar la memoria para aprender experiencias importantes que han pasado hace mucho tiempo. Las LSTM permiten a las RNN recordar las entradas durante un largo periodo de tiempo.

Esto se puede hacer debido a que las LSTM almacenan esta información en la memoria, la cual se trata de una "célula" la cual decide si almacenar o eliminar la información en función de la importancia de los datos que está recibiendo. Esta importancia se asigna en función de los pesos que se aprenden durante la ejecución de la RNN. De esta forma, la red va aprendiendo a lo largo del tiempo qué información es importante y cual no. Puede verse una representación de esta celda de memoria en la figura 2.17.

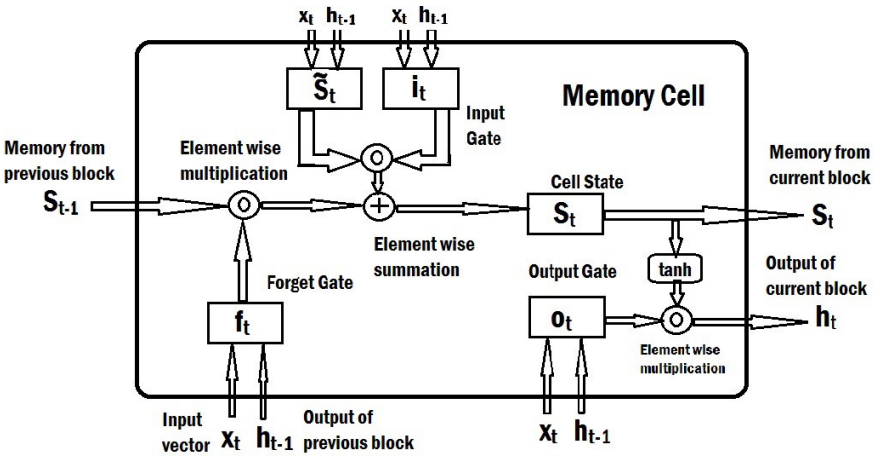

Figura (2.17) Memoria LSTM

2.3.6. Conclusiones sobre los clasificadores

Una vez visto este pequeño repaso a los clasificadores que se van a implementar en este trabajo, se ha podido ver que probablemente, debido al funcionamiento más complejo y preciso que tienen los clasificadores de tipo Support Vector Machines y LSTM, funcionen mejor a la hora de realizar la clasificación de textos. Además de por sus definiciones, también están las pruebas de los trabajos previos que se han detallado más arriba en el estado del arte, donde de los métodos que se van a implementar, los que más se han usado han sido SVM y LSTM.

En el capítulo siguiente se van a probar todos los clasificadores explicados en esta sección y se evaluarán empíricamente haciendo uso de distintas medidas para evaluar la clasificación.

CAPÍTULO 3
Desarrollo del detector de cyberbullying

En este capítulo se proponen y analizan distintas técnicas para el desarrollo del detector de cyberbullying. Para este análisis se han utilizado los datasets de Xu et al. [40] y Founta et al. [11] ampliándolos con información extraída de la red sobre los perfiles de los usuarios según la técnica a utilizar. En la sección 3.1 se van a entrenar los clasificadores haciendo uso solamente de los textos de las publicaciones que han sido etiquetados. Una vez realizada esta prueba, en la sección 3.2, se analizará si incluir información del perfil de los usuarios que publican los mensajes como características a tener en cuenta a la hora de clasificar mejora los resultados ofrecidos por los clasificadores. Finalmente, en la sección 3.3, se planteará realizar un clustering en base a las características del perfil de los usuarios que les hagan identificables. Este proceso sería útil para reducir la cantidad de textos a analizar y centrarse solo en aquellos textos generados por perfiles de usuarios que potencialmente pueden corresponder a acosadores.

3.1 Modelos basados en texto

Esta primera aproximación, consiste en buscar el mejor modelo para la detección de cyberbullying haciendo uso de distintos clasificadores y entrenándolos solamente con los textos de las publicaciones de los datasets.

Esta sección se desarrolla en cuatro apartados. En el primer apartado se explican los dos datasets que se han utilizado en este experimento. En el segundo apartado se describe el preprocesado que se ha realizado a los textos. En el tercer apartado se explica cómo se ha realizado la implementación de cada uno de los clasificadores. Por último se analizan los resultados obtenidos con los distintos clasificadores.

3.1.1. Corpus

Para realizar este experimento se va a probar con dos de los datasets mencionados en el estado del arte. El primero de ellos es el dataset creado por Xu et al. [40]. Este dataset, dispone de un número de textos bastante limitado, pero de entre todos los datasets encontrados es el que mejor etiquetado tiene para esta tarea ya que distingue muy bien todos los perfiles involucrados en un caso de cyberbullying (e.i, acosador, víctima, defensor, ...), por lo que en el caso de que se obtuviesen buenos resultados con los modelos entrenados con este dataset se podría identificar tanto al agresor como a la víctima, lo cual sería una gran ventaja para frenar a tiempo estos casos. Se puede ver un ejemplo de

la información contenida en el dataset en la Figura 3.1, donde **"id"** es el identificador del tweet, **"user_id"** es el identificador del usuario, **"x"** indica si hay trazas de bullying (y) o si no las hay (n), **"y"** indica el tipo de acoso, **"j"** el tipo de acoso que se ha realizado (físico, verbal, ...), **"a"** si se incluyen burlas en la publicación, **"label"** es el rol que toma la persona que publica el tweet (defender, reporter, victim, bully, accuser), **"c"** son las emociones encontradas en el texto.

Id	user_id	x	y	j	a	label	c
105730486382497793	322329899	y	self-disclosure	other	n	victim	none
107688644067856384	185389094	y	self-disclosure	other	n	defender	empathy
108654820042354688	361869056	n	NA	NA	NA	NA	NA
102206417217392640	226320672	n	NA	NA	NA	NA	NA
102779484725448704	297557032	n	NA	NA	NA	NA	NA
108676972149874688	157724561	n	NA	NA	NA	NA	NA
108901271301394432	91457415	n	NA	NA	NA	NA	NA
103911819474771969	197614459	n	NA	NA	NA	NA	NA
106590106873372672	62179998	n	NA	NA	NA	NA	NA

Figura (3.1) Dataset de cyberbullying descrito en [40] por Xu et al.

Para la realización de este primer experimento nos interesan la primera columna, la cual es el identificador del tweet y la séptima, la cual indica el rol que toma la persona que escribe el texto.

Por otra parte, el segundo dataset que se va a utilizar es el creado por Founta et al. [11]. Este dataset, aunque no tiene una gran cantidad de etiquetas y sólo identifica al agresor, tiene una gran cantidad de textos, por lo que probablemente sea mejor para realizar el entrenamiento de los modelos. Este dataset viene con varios ficheros. Uno de los ficheros viene con el identificador del tweet y su etiqueta correspondiente. El otro fichero viene con el texto y la etiqueta asignada a este (abusive, normal, hateful, spam). En el caso de este primer experimento se hará uso del último fichero cuyo formato se muestra en la Figura 3.2.

Text	Label	Id
RT @Papapishu: Man it would fucking rule if we had a party that was against perpetual warfare.	abusive	4
It is time to draw close to Him 🙏🏻 Father, I draw near to you now and always ❤️ https://t.co/MVRBBX2aqJ	normal	4
if you notice me start to act different or distant.. it's bc i peeped something you did or i notice a difference in how you act & ian fw it.	normal	5
Forget unfollowers, I believe in growing. 7 new followers in the last day! Stats via https://t.co/bunPHQNXhj	normal	3
RT @Vitiligoprince: Hate Being sexually Frustrated Like I wanna Fuck But ion wanna Just fuck anybody	abusive	4
Topped the group in TGP Disc Jam Season 2! Onto the Semi-Finals! @HighHorseGames https://t.co/N7LE8IX7Rm	normal	3
That daily baby aspirin for your #heart just might be preventing colon #cancer too. https://t.co/2TLawmHhEe	normal	5
I liked a @YouTube video from @mattshea https://t.co/niSeJrLKHq THE BLUE ARMY IS COMING! - Ancient Warfare 2	normal	2

Figura (3.2) Dataset de cyberbullying descrito en [11] por Founta et al.

De este dataset usaremos el texto que aparece en la primera columna, y la etiqueta que aparece en la segunda columna. La tercera columna que aparece se trata del identificador del etiquetador que revisó el texto para asignar una de las etiquetas.

En la tabla que se muestra a continuación (ver Tabla 3.1) se muestra un pequeño resumen las características de los datasets. El dataset de Xu et al. contiene un total de 7000 muestras muy desbalanceadas ya que la categoría "NA", la cual indica que el texto no contiene trazas de cyberbullying, es la que más tweets tiene y el resto de clases tienen una cantidad mucho menor. Por otro lado, el dataset de Founta et al. contiene un total de 100000 muestras las cuales están también están algo desbalanceadas, pero no de una forma tan notable como en el otro dataset.

Artículo	Nº Muestras	Etiquetas					
Xu et al. [40]	7321	Defender (82)	Reporter (394)	Victim (290)	Bully (171)	Accuser (154)	NA (3131)
Founta et al. [11]	99996	Hateful (4965)	Abusive (27150)	Normal (53851)	Spam (14030)	-	-

Tabla (3.1) Resumen de los datasets utilizados en los experimentos.

Se van a entrenar distintos clasificadores haciendo uso de estos dos datasets de forma separada. De esta manera se podrá ver con qué dataset y clasificador se obtienen los mejores resultados.

3.1.2. Preprocesado

El preprocesado del dataset de Xu et al. es algo diferente debido a que se tienen que obtener los datos de los tweets mediante la API de Twitter[1], En este dataset, como se ha visto en la anterior sección, solamente se proporcionan los identificadores de los textos. Para descargar los textos se ha hecho uso de la librería Tweepy[2] la cual proporciona herramientas para realizar de una forma sencilla consultas a la API de Twitter. Para ello debemos acceder como desarrollador en la página de Twitter, crear un proyecto y solicitar las claves para poder hacer uso de la API. Una vez disponemos de estas claves ya se puede utilizar la API a través de Tweepy. De esta forma se puede hacer una consulta proporcionando únicamente el identificador del tweet que queremos obtener. Debido a ciertas normas que cambiaron en los acuerdos de usuario de Twitter, muchos textos y cuentas que infringían las normas, principalmente casos de bullying, se eliminaron de la plataforma [31], por lo que no se han podido recuperar algunos tweets, quedando el dataset con 4369 muestras de entrenamiento.

Una vez se han realizado estás consultas con las que se han obtenido los textos asociados a los identificadores proporcionados en el dataset, se ha generado un nuevo fichero CSV con todos los datos. Además, se han pasado las etiquetas a números, de forma que sea más sencillo trabajar con los datos. Se puede ver el formato del fichero CSV en la Figura 3.5, donde se han guardado el identificador del tweet, el texto y la etiqueta codificada con un número.

id	Text	Label
107688644067856384	@bellathorne143 i herd that you got bullied when you was 6 years old and i feel bad,are you ok	4
102206417217392640	Bullying: O gesto mais idiota, estúpido e irracional que um ser humano pode praticar.	5
102779484725448704	The Bully at School Goes High Tech – Part 1: The Section of State and Local Government Law of the Am... http://t.co/DGWXoKd #slaw #law	5
108676972149874688	Esse @Felipemath fazendo bullying comigo! Haha	5
106590106873372672	AUISHUAHS eu e o @wallace_mancha tiramos o dia pra praticar bullying com a @helove_	5
109034091743154176	@Looweihao @haoyangg @sleepybed Hello, since when i got bully people? Never physically you know! :(Criminal is not sup	5
102533497637437441	For those keeping score at home- cousin who just had massive heart-attack is trying to bully the doctors into letting her out	3

Figura (3.3) Contenido del dataset creado por Xu et al. preprocesado.

Como el dataset de Founta et al. ya nos proporciona los textos de las publicaciones, ya tenemos los dos datasets en un formato con el que podemos trabajar. A los dos se les realizan los mismos pasos en el preprocesado, por lo que primeramente se leen los ficheros CSV haciendo uso de la librería Pandas[3]. Esta librería simplifica el trabajo con los datos que disponemos para el entrenamiento. Una vez tenemos todos los datos guardados en dataframes se realiza una limpieza de los datos haciendo uso de la función *clean_data*() que se ha implementado.

[1] https://developer.twitter.com/en/docs/twitter-api
[2] https://www.tweepy.org/
[3] https://pandas.pydata.org/

```
def clean_data(txt):
    try:
        text = np.unicode(txt, 'utf-8')
    except (TypeError, NameError):
        pass
    txt = unicodedata.normalize('NFD', txt)
    txt = txt.encode('ascii', 'ignore')
    txt = txt.decode("utf-8")
    txt = re.sub(" \d+", " ", txt)
    txt = re.sub(r'\b\w{1,3}\b', ' ', txt)
    txt = re.sub(r'https?:\/\/.*\/\w*', '', txt)
    txt = re.sub(r'https', '', txt)
    txt = re.sub("@[A-Za-z0-9_]+"," ",txt)
    txt = txt.lower()
    return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", txt).split())
```

Esta función sirve para limpiar los textos para que no haya ruido en el entrenamiento, de esta forma como se puede ver se eliminan direcciones http y https, @, palabras cortas y números, además de dejar el texto resultante en minúsculas.

A continuación, se separa el corpus en dos partes, una para entrenamiento y la otra para test. En esta parte se realiza un 10-fold Cross-validation, por lo que por cada clasificador y dataset se entrenan 10 modelos distintos cambiando las muestras de test y entrenamiento. Posteriormente se realiza una lematización de las palabras que se incluyen en los textos para dejarlas en la forma original de la palabra. Para ello se ha hecho uso de la herramienta de nltk WordNetLemmatizer. Dependiendo de si se van a utilizar los clasificadores clásicos o las redes neuronales, estos pasos pueden variar un poco debido a las librerías que se usan al implementar estos clasificadores. A continuación, se puede ver como se ha realizado esta parte del preprocesado en cada uno de los casos:

```
#En el caso de los clasificadores cl sicos se realizan los siguientes pasos

kfold = KFold(10)
for train, test in kfold.split(df_dataset):
    df_train = df_dataset.iloc[train]
    df_test = df_dataset.iloc[test]

    X_train, y_train = df_train.Text, df_train.Label
    X_test, y_test = df_test.Text, df_test.Label

    documents = [""]
    stemmer = WordNetLemmatizer()

    for sen in df_train.Text:

        # Lemmatization
        document = document.split()

        document = [stemmer.lemmatize(word) for word in document]
        document = ' '.join(document)

        documents.append(document)
```

```
#En el caso de las redes neuronales se realizan los siguientes pasos

kfold = KFold(10)
for train, test in kfold.split(input, target):
    df_train = input.iloc[train]
```

```
    df_test = input.iloc[test]

    y_train = target.iloc[train].values
    y_test = target.iloc[test].values

    sentences_train = df_train.values
    sentences_test = df_test.values

    y_train = np_utils.to_categorical(y_train)
    y_test = np_utils.to_categorical(y_test)

    documents = [""]
    stemmer = WordNetLemmatizer()

    for sen in sentences_train:

        # Lemmatization
        document = document.split()

        document = [stemmer.lemmatize(word) for word in document]
        document = ' '.join(document)

        documents.append(document)
```

Una vez se han lematizado los textos se vectorizan/tokenizan, donde dependiendo de si el clasificador usado es una red neuronal o un clasificador clásico se realiza de una manera u de otra. En el caso de los clasificadores clásicos se hace uso de la herramienta TfidfVectorizer, se entrena el vectorizer con las muestras de entrenamiento y una vez realizado este paso se transforman todos los textos, tanto de test como de entrenamiento, a vectores. Estos vectores serán la entrada de los clasificadores clásicos. Se puede ver como se realizó este paso a continuación:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=1500, sublinear_tf = True,
                             use_idf = True,
                             stop_words=stopwords.words('english'))
vectorizer.fit(documents)

X_train = vectorizer.transform(X_train).toarray()
X_test = vectorizer.transform(X_test).toarray()
```

En el caso de estar realizando el preprocesado para la red neuronal se hace uso de la herramienta tokenizer que viene incluida en la librería Keras y se realiza un padding para dejar todas las muestras con el mismo numero de datos en cada vector. Esto quedaría de la siguiente manera:

```
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(sentences_train)

x_train = tokenizer.texts_to_sequences(sentences_train)
x_test = tokenizer.texts_to_sequences(sentences_test)

maxlen = 100
x_train = pad_sequences(x_train, padding='post', maxlen=maxlen)
x_test = pad_sequences(x_test, padding='post', maxlen=maxlen)
```

Una vez realizado este preprocesado los datos ya están listos para utilizarlos como entrada de los clasificadores.

3.1.3. Clasificadores

La implementación realizada de los clasificadores es bastante diferente si comparamos la realizada en los clásicos con las redes neuronales. Por este motivo se va a dividir esta sección en dos partes. En una primera parte se describirá el desarrollo de los clasificadores clásicos y en una segunda parte se describirá el desarrollo de la red neuronal.

Clasificadores clásicos

Para realizar la implementación de los clasificadores clásicos de una forma sencilla, primero de todo se ha implementado el método *fit_predict_eval* al cual se le pasa el clasificador que se va a utilizar, los datos de entrenamiento y de test. Se puede ver este método a continuación.

```
def fit_predict_eval(classifier_loc, X_train_loc, y_train_loc, X_test_loc,
    y_test_loc):

    classifier_loc.fit(X_train_loc, y_train_loc) #Entrenamiento del modelo
    y_pred_loc = classifier_loc.predict(X_test_loc) #Fase de test

    from sklearn.metrics import classification_report, confusion_matrix,
        accuracy_score, f1_score, precision_score, recall_score

    #Calculo de las medidas de evaluaci n
    accuracy = round(accuracy_score(y_test_loc, y_pred_loc), 2)
    f1 = round(f1_score(y_test_loc, y_pred_loc, average='weighted'), 2)
    precision = round( precision_score(y_test_loc, y_pred_loc,
                                       average='weighted'), 2)
    recall = round(recall_score(y_test_loc, y_pred_loc, average='weighted'), 2)

    return accuracy, f1, precision, recall
```

En este método se realiza el entrenamiento del modelo y se realiza la fase de test, calculando el accuracy, precision, recall y F1 para su posterior análisis.

La implementación de los distintos clasificadores se ha realizado haciendo uso de la librería sklearn. El primero de los clasificadores que se ha implementado ha sido el de Random Forest (RF). Para este clasificador se ha hecho uso del "n_estimators=1000", donde "n_stimators" indica el número de árboles en el bosque. A continuación se muestra la implementación realizada:

```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators=1000)
accuracy, f1, precision, recall = fit_predict_eval(classifier, X_train,
                                                   y_train, X_test, y_test)
```

El siguiente clasificador implementado ha sido Naive Bayes (NB), para el cual se han usado los parámetros que vienen por defecto en la librería, por lo que su implementación ha quedado de la siguiente manera:

```
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()
accuracy, f1, precision, recall = fit_predict_eval(classifier, X_train,
                                                   y_train, X_test, y_test)
```

A continuación se ha realizado la implementación de los Support Vector Machines (SVM), en este caso se ha realizado una configuración diferente, haciendo uso de la herramienta BaggingClassifier la cual reduce la varianza de los estimadores introduciendo aleatoriedad. Esta herramienta es útil para reducir el sobre-entrenamiento del modelo y acelerar ligeramente la obtención de los resultados, ya que como se mencionará en la sección de los resultados, en algunos casos este clasificador se demoraba una gran cantidad de tiempo. De esta forma, la implementación de los Support Vector Machines ha sido la siguiente:

```
from sklearn.svm import SVC

n_estimators = 10
classifier = BaggingClassifier(SVC(kernel='linear'),
                               max_samples=1.0 / n_estimators,
                               n_estimators=n_estimators)

accuracy, f1, precision, recall = fit_predict_eval(classifier, X_train,
                                                   y_train, X_test, y_test)
```

Por último, el clasificador que se ha implementado ha sido K-Vecinos (KNN). Después de realizar diversos experimentos donde se han probado distintos números de vecinos para ver con cual se obtienen mejores resultados, se ha decidido utilizar 2 vecinos para decidir a qué clase pertenece el dato a clasificar. La implementación ha quedado de la siguiente forma:

```
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors=2)
accuracy, f1, precision, recall = fit_predict_eval(classifier, X_train,
                                                   y_train, X_test, y_test)
```

Redes neuronales (LSTM + WE)

La red neuronal implementada haciendo uso de Keras y Tensorflow se trata de una LSTM con WordEmbeddings. Se trata de una red neuronal secuencial, formada por varias capas. Puede verse un pequeño esquema en la Figura 3.4.

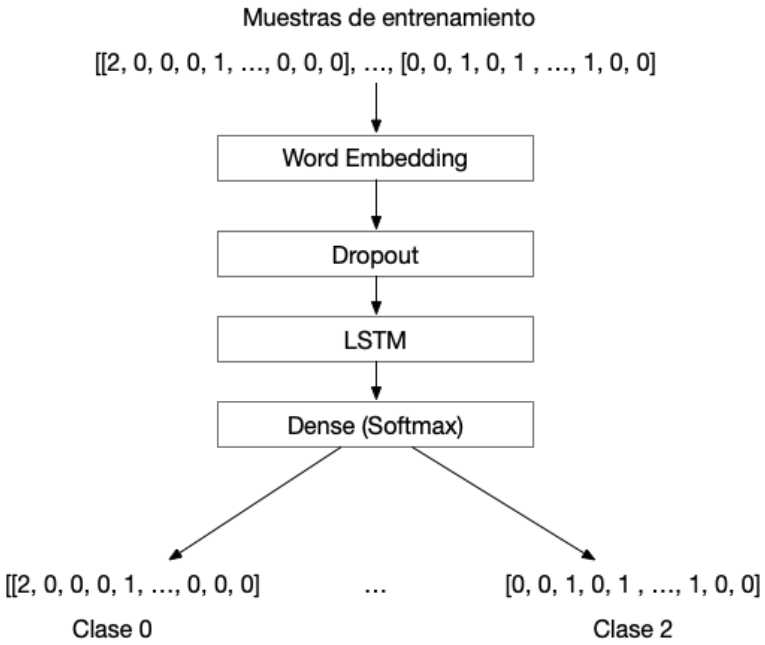

Figura (3.4) Esquema de la red neuronal LSTM+WE utilizada.

La primera de las capas es una capa de embedding, la cual va a tener como tamaño de entrada el tamaño del vocabulario y de salida 200. La segunda capa utilizada ha sido una capa de dropout a 0.5 para que no se sobre entrene la red. Como tercera capa se ha hecho uso de una LSTM bidireccional. Por último, se ha realizado una capa densa con activación softmax. Se puede ver la implementación de la red a continuación.

```
batch_size = 128
epochs = 100
embedding_dim = 200
vocab_size = len(tokenizer.word_index) + 1

model = Sequential()
model.add(Embedding(input_dim=vocab_size,
                    output_dim=embedding_dim))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(embedding_dim)))
model.add(Dense(4, activation='softmax'))

model.summary()
```

En el momento de compilar la red neuronal, se ha utilizado el optimizador Adam, al cual se le ha modificado el "learning rate" (lr) para evitar el sobre-entrenamiento de la red neuronal, y además como función de pérdida se ha utilizado "categorical crossentropy" debido a que se pretende clasificar un dataset multi-categoría.

```
adam=Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, amsgrad=False)

model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
```

```
                    verbose=1,
                    validation_data=(x_test, y_test))
```

Una vez se realiza el entrenamiento de la red, se generan los resultados haciendo uso de la librería sklearn, y se obtienen las medidas accuracy, F1, precision y recall.

```
y_pred = model.predict(x_test, batch_size=64, verbose=1)
y_pred_bool = np.argmax(y_pred, axis=1)

accuracy = round(accuracy_score(target_for_evaluate, y_pred_bool), 2)
f1 = round(f1_score(target_for_evaluate, y_pred_bool, average='weighted'), 2)
precision = round( precision_score(target_for_evaluate,
                        y_pred_bool, average='weighted'), 2)
recall = round(recall_score(target_for_evaluate,y_pred_bool,
                                    average='weighted'), 2)
```

3.1.4. Resultados

Como se ha realizado 10-fold cross-validation se han obtenido 10 resultados distintos por cada clasificador-dataset. Los resultados se van a mostrar haciendo la media de las 10 ejecuciones que se han hecho con cada uno de los clasificadores. Se han utilizado cuatro medidas.

La primera de ellas es ”precision”, esta medida trata de responder ¿Qué proporción de los datos marcados como positivos por el clasificador han sido correctos? Un modelo que no produce falsos positivos tiene una precision de 1.0. La precisión se calcula siguiendo la formula (ver Ecuación 3.1),

$$Precision = \frac{VP}{VP + FP} \tag{3.1}$$

donde ”VP” son los verdaderos positivos y ”FP” los falsos positivos.

La siguiente medida que se ha utilizado ha sido el ”recall” o exhaustividad, esta intenta responder a la pregunta ¿qué proporción de positivos verdaderos se han identificado correctamente? Un modelo que no produzca falsos negativos tendrá un recall de 1.0, la fórmula utilizada para el cálculo de la exhaustividad puede verse en la Ecuación 3.2,

$$Recall = \frac{VP}{VP + FN} \tag{3.2}$$

al igual que en la anterior, ”VP” son los verdaderos positivos y ”FN” son los falsos negativos.

La tercera de las medidas utilizadas ha sido el ”Accuracy” o acierto, la cual refleja en un porcentaje el total de muestras que se clasifican correctamente. La fórmula para calcular esta medida es la siguiente (ver Ecuación 3.3):

$$Accuracy = \frac{VP + VN}{FP + FN + VP + VN} \tag{3.3}$$

La última de las medidas utilizada ha sido F1, la cual es muy útil en los casos en que los datasets no estén completamente balanceados, donde en estos casos, funciona mejor que el acierto. Esta medida se calcula haciendo uso de la precisión y la exhaustividad (ver Ecuación 3.4):

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{3.4}$$

En los primeros resultados, los correspondientes al dataset de Xu et al., se puede observar que algunos clasificadores como Naive Bayes no obtienen buenos resultados. Esto puede ser debido a que los clasificadores no han podido entrenarse bien ya que el dataset tiene pocas muestras de entrenamiento y las clases están muy desbalanceadas. Puede verse en la Tabla 3.2 que aparece a continuación los resultados generales del experimento.

	Accuracy	**F1**	**Recall**	**Precision**
LSTM + WE	0.74	0.64	0.74	0.55
KNN	0.61	0.64	0.61	0.68
SVM	0.74	0.64	0.74	0.61
RF	0.75	0.67	0.74	0.662
NB	0.611	0.642	0.611	0.68

Tabla (3.2) Resultados de los experimentos utilizando los clasificadores clásicos y redes neuronales con el dataset de Xu et al. [40]

.

Se puede ver que los resultados que se han obtenido son muy parecidos entre todos los clasificadores, sin embargo, tres de ellos han destacado ligeramente, siendo estos: la red neuronal LSTM+WE, Support Vector Machines y Random Forest. Teniendo en cuenta estos resultados se espera que haciendo uso del dataset de Founta et al, se obtengan resultados similares o mejores ya que cuenta con un mayor número de muestras de entrenamiento.

En el caso de los resultados obtenido en el dataset de Founta et al. se ha tardado una gran cantidad de tiempo en obtenerlos. Concretamente los clasificadores Random Forest, Support Vector Machines y la red neuronal implementada ha tardado en realizar el 10-fold cross-validation aproximadamente entre 1-2 días cada uno. Esto ha hecho que este proceso de entrenamiento haya sido lento y con poco margen de mejora, ya que cada vez que se ha tenido que modificar algunos de los clasificadores con este dataset se necesitaban grandes cantidades de tiempo. Los resultados que se han obtenido finalmente han sido los siguientes:

	Accuracy	**F1**	**Recall**	**Precision**
LSTM + WE	0.85	0.81	0.80	0.83
KNN	0.55	0.57	0.55	0.63
SVM	0.78	0.74	0.78	0.63
RF	0.78	0.76	0.78	0.76
NB	0.24	0.257	0.24	0.65

Tabla (3.3) Resultados de los experimentos utilizando los clasificadores clásicos y redes neuronales con el dataset de Founta et al. [11].

Se puede ver que los clasificadores Naive Bayes y K-vecinos obtienen unos malos resultados, incluso peores que los obtenidos con el dataset de Xu et al. Sin embargo, los otros clasificadores (LSTM+WE, SVM y RF), si los comparamos, obtienen unos resultados bastante superiores, sobre todo si nos fijamos en el clasificador implementado con redes neuronales, el cual supera por bastante los resultados obtenidos anteriormente. Con estos resultados podemos concluir que el clasificador basado en redes neuronales es el que mejor funciona para este dataset de cyberbullying.

3.2 Modelos basados en texto y datos del usuario

Como uno de los objetivos es identificar a los usuarios que cometen este tipo de agresiones, se va a probar a implementar los clasificadores pasando, además de los datos del texto que publican, ciertas características que aparecen en su perfil de la red social como pueden ser: si el perfil tiene descripción, foto, si es una cuenta verificada, el número de mensajes generados por el usuario o la fecha de creación de la cuenta. Algunos trabajos previos ya mencionan el uso de estas características para la detección de perfiles de acoso [39]. En esta sección evaluaremos qué efecto tiene que los clasificadores los tengan en cuenta.

Al igual que en la anterior sección, se va a dividir esta parte en cuatro apartados. En el primero de ellos se detalla el dataset con el que se realizan los experimentos, en el segundo se detalla el preprocesado, en el tercero los clasificadores y en el cuarto los resultados obtenidos.

3.2.1. Corpus

Para este experimento se va a hacer uso de los dos datasets que se han utilizado en el anterior apartado, el de Xu et al [40] y el de Founta et al. [11]. La diferencia en este experimento, es que se necesita el identificador del usuario, por lo que por parte del dataset de Xu et al. (ver Figura 3.1) haremos uso de esta segunda columna. Por parte del dataset de Founta et al. se complica algo más, ya que el identificador del usuario no se proporciona explícitamente y se debe de obtener haciendo una petición a la API de Twitter.

3.2.2. Preprocesado

El preprocesado de las publicaciones es prácticamente igual que en el anterior apartado. En el caso del dataset de Xu et al. [40] se ha tenido que volver a hacer el mismo proceso mencionado anteriormente para recuperar los textos a partir del identificador, por lo que se utilizarán de un total de 4369 de este dataset. En el caso de Founta et al. [11] es necesario recuperar el identificador del usuario que hizo esa publicación. Para ello, haciendo uso de la API de Twitter y de la librería Tweepy, se recupera la información de la publicación a partir de su identificador. En los campos de esta respuesta por parte de la API se encuentra el identificador del usuario que realizó la publicación. Este identificador nos lo guardamos junto con el texto de la publicación y la etiqueta asignada para que desde este momento, se pueda realizar el mismo preproceso de los datos en ambos datasets. Al realizar este preproceso para obtener la información del usuario a partir del identificador, al igual que sucedió anteriormente con el otro dataset, muchas publicaciones han sido eliminadas de la red social, por lo que se han podido recuperar de esta forma un total de 58025 muestras del dataset de Founta et al..

A continuación, para los dos datasets, es necesario recuperar la información de los usuarios. Al igual que se ha realizado para recuperar los datos de los tweets haciendo uso de Tweepy, ahora se va a hacer algo similar para recuperar la información del perfil del usuario, obteniendo de esta forma la siguiente respuesta por parte de la API de Twitter:


```
{
    "id":177913822,
    "id_str":"177913822",
    "name":"Frederica",
```

```
    "screen_name":"Frederica__",
    "location":"",
    "profile_location":"None",
    "description":"    Instagram: Frederica__",
    "url":"None",
    "entities":{
       "description":{
          "urls":[

          ]
       }
    },
    "protected":False,
    "followers_count":933,
    "friends_count":435,
    "listed_count":5,
    "created_at":"Fri Aug 13 11:27:45 +0000 2010",
    "favourites_count":1057,
    "utc_offset":"None",
    "time_zone":"None",
    "geo_enabled":False,
    "verified":False,
    "statuses_count":86251,
    "lang":"None",
    "status":{
       "created_at":"Tue Jan 15 10:24:23 +0000 2019",
       "id":1085120505210855426,
       "id_str":"1085120505210855426",
       "text":"need bubbletea       ",
       "truncated":False,
       "entities":{
          "hashtags":[

          ],
          "symbols":[

          ],
          "user_mentions":[

          ],
          "urls":[

          ]
       },
       {...}

}
```


A partir de la información que se obtiene de cada uno de los usuarios se genera el fichero que se puede ver a continuación:

tweet_id	user_id	Text	Label	matrix
107688644067856384	185389094	@bellathorne143 i herd that you got bullied when you was 6 years old and i feel bad,are you ok	4	[1, 1, 0, 1, 0, 0]
102206417217392640	226320672	Bullying: O gesto mais idiota, estúpido e irracional que um ser humano pode praticar.	5	[0, 1, 1, 1, 1, 0]
102779484725448704	297557032	The Bully at School Goes High Tech – Part 1: The Section of State and Local Government Law of the Am... http://t.co/DGWXoKd #slaw #law	5	[1, 1, 1, 1, 1, 0]
108676972149874688	157724561	Esse @Felipemath fazendo bullying comigo! Haha	5	[1, 1, 0, 1, 1, 0]
106590106873372672	62179998	AUISHUAHS eu e o @wallace_mancha tiramos o dia pra praticar bullying com a @helove_	5	[1, 1, 1, 1, 1, 0]
109034091743154176	177913822	@Looweihao @haoyangg @sleepybed Hello, since when i got bully people? Never physically you l	5	[0, 1, 1, 1, 1, 0]
102533497637437441	70412906	For those keeping score at home- cousin who just had massive heart-attack is trying to bully the	3	[1, 1, 1, 1, 1, 0]
103679008855691264	11363462	BETTER ANTI-BULLYING AD SLOGANS: Hey, Bullies. Since it HAS to be you our us, we'll send fl	5	[1, 1, 1, 1, 1, 0]

Figura (3.5) Dataset de Xu modificado para incluir la información del usuario que generó el texto del tweet.

Se puede ver que se dispone del identificador tweet, identificador del usuario, el texto, la etiqueta y una matriz. Esta matriz contiene la información del usuario. El significado de las posiciones de la matriz son las siguientes:

- Campo 0: indica si el usuario tiene una descripción asociada a su perfil. En caso afirmativo el campo tomará el valor 1 y en caso contrario 0.
- Campo 1: indica si el usuario tiene una foto/imagen en su perfil. en caso afirmativo el campo tomará el valor 1 y en caso contrario 0.
- Campo 2: Este campo hace referencia al género del usuario. El género de usuario no aparece de forma explícita en su perfil. Para poder determinarlo se ha hecho uso de la librería Gender_guesser [4] que a partir del texto publicado infiere el género del usuario. Si no se puede determinar el género del usuario tomará el valor 0. En el caso de que se pueda identificar tomará el valor 1 en el caso de que sea masculino y 2 en el caso de que sea femenino.
- Campo 3: indica el número de seguidores. Se han considerado tres rangos. EL primer rango es [0, 15]. El segundo rango es [16, 100] y el tercer rango [100, >100]. Si el usuario tiene un número de seguidores dentro de estos rangos, el campo 3 tomará el valor de 0, 1 o 2 respectivamente
- Campo 4: indica si la fecha de creación de la cuenta tiene más de un año desde que se recogieron los datos. En caso afirmativo, el campo tomará el valor 1. En caso contrario 0.
- Campo 5: hace referencia a si el perfil del usuario está verificado. En caso afirmativo tomará el valor 1, en caso contrario tomará el valor 0.

Para poder añadir los datos del perfil del usuario, que están codificados en una matriz, a los datos de entrenamiento que teníamos de los anteriores experimentos se ha hecho uso de la herramienta de scipy "sparse.hastack". Esta herramienta permite unir dos matrices, como se puede ver a continuación.

```
#z_train y z_test son las matrices de los usuarios

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(sublinear_tf = True, use_idf = True, stop_words=
    stopwords.words('english'))
vectorizer.fit(documents)

X_train = vectorizer.transform(X_train)
X_train = scipy.sparse.hstack((X_train, np.asarray(z_train.values.tolist())))
```

[4] https://pypi.org/project/gender-guesser/

```
9
10 X_test = vectorizer.transform(X_test)
11 X_test = scipy.sparse.hstack((X_test, np.asarray(z_test.values.tolist())))
12
13 X_test = X_test.toarray()
14 X_train = X_train.toarray()
```

Una vez hecho este paso ya se puede pasar al entrenamiento de los clasificadores.

3.2.3. Clasificadores

En este experimento los clasificadores utilizados son exactamente los mismos que en la anterior sección, por lo que no se van a volver a explicar paso por paso como se han implementado.

3.2.4. Resultados

Los resultados obtenidos no son como se esperaban, ya que debería de haber aumentado la tasa de acierto, así como a su vez el resto de medidas, pero esto no ha sido así. Los resultados obtenidos han sido prácticamente los mismos que en los experimentos anteriores. Los resultados se pueden ver en la Tabla 3.4.

	Text + User data				**Text**			
	Accuracy	**F1**	**Recall**	**Precision**	**Accuracy**	**F1**	**Recall**	**Precision**
RF	0.743	0.67	0.743	0.65	0.75	0.67	0.74	0.66
SVM	0.73	0.68	0.73	0.67	0.74	0.64	0.74	0.61
KNN	0.56	0.615	0.56	0.69	0.61	0.64	0.61	0.68
NB	0.617	0.65	0.62	0.68	0.611	0.642	0.611	0.68
LSTM+WE	0.73	0.64	0.74	0.55	0.74	0.64	0.74	0.55

Tabla (3.4) Resultados obtenidos utilizando los textos más la información del perfil del usuario en los clasificadores clásicos y utilizando el dataset de Xu et al. [40].

Se puede ver en la Tabla 3.4 que los resultados obtenidos en este experimento son igual o inferiores a los que se habían obtenido en el experimento anterior. Al realizar el entrenamiento haciendo uso del dataset de Founta et al. sucede lo mismo que con el dataset de Xu et al. (ver Tabla 3.5), donde los resultados obtenidos haciendo uso de la información del perfil de usuario son iguales o menores que haciendo uso solamente de la información del texto.

	Text + User data				**Text**			
	Accuracy	**F1**	**Recall**	**Precision**	**Accuracy**	**F1**	**Recall**	**Precision**
RF	0.75	0.74	0.75	0.74	0.78	0.76	0.78	0.76
SVM	0.77	0.71	0.73	0.62	0.78	0.74	0.78	0.63
KNN	0.55	0.51	0.53	0.59	0.55	0.57	0.55	0.63
NB	0.20	0.24	0.22	0.63	0.24	0.26	0.24	0.65
LSTM+WE	0.81	0.78	0.76	0.80	0.85	0.81	0.80	0.83

Tabla (3.5) Resultados obtenidos utilizando los textos más la información del perfil del usuario en los clasificadores clásicos y utilizando el dataset de Founta et al. [11].

Teniendo en cuenta estos resultados podemos concluir que los datos que hemos añadido a la información de los textos no están aportando información que mejore el proceso de clasificación. Por esta razón se ha descartado la opción de realizar un clasificador basado en el texto y las características de los usuarios.

3.3 Clustering de los usuarios

Con el objetivo de reducir el volumen de información a clasificar en una red social, planteamos realizar un primer filtrado de los usuarios cuyos textos se iban a analizar teniendo en cuenta si su perfil tenia características que indicaran que potencialmente podía ser un perfil de un acosador. Para ello, se realizó un clustering de los usuarios en base a sus características para determinar si existían perfiles de usuarios claramente diferenciados.

3.3.1. Preprocesado

El preprocesado de los datasets han sido similares al que se ha realizado en el anterior experimento. Se han recuperado los datos de los usuarios, pero en este caso se han obtenido todos los campos de información del perfil del usuario y se han guardado en un fichero CSV (ver Figura 3.6).

label	location	name	description	gender	followers	year	friends	favs	verify
4	1	1	0	1	3	2010	0	1	0
5	0	1	1	1	610	2010	155	312	0
5	1	1	1	1	194	2011	315	0	0
5	1	1	0	1	178	2010	100	329	0
5	1	1	1	1	193	2009	404	30	0
5	0	1	1	1	963	2010	448	1062	0
3	1	1	1	1	592	2009	812	18526	0
5	1	1	1	1	715	2007	651	12859	0

Figura (3.6) Información extraída del perfil de los usuarios del dataset de Xu et al. [40].

Una vez hecho esto ya tenemos los datos necesarios para realizar el clustering de los usuarios.

3.3.2. Clustering

En este apartado se va a explicar el proceso seguido para realizar el clustering en el cual se hizo uso de todos los datos obtenidos de los usuarios. Para ello se ha utilizado la librería sklearn, la cual incorpora el algoritmo k-means [12] que es muy útil para realizar tareas de clustering.

Primero de todo se debe decidir cuántos clusters son necesarios para realizar este experimento. Para lograr esto se ha implementando en Python el método Elbow como heurística. Este método aplica el algoritmo K-means para un rango de valores de "K" y detecta el valor a partir del cual la reducción en la suma total de la varianza intra-cluster deja de ser significativa.

```
Nc = range(1, 20)
kmeans = [KMeans(n_clusters=i) for i in Nc]
score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]
plt.plot(Nc,score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
```

```
8 plt.show()
```

La gráfica resultante de aplicar el método Elbow se puede ver en la Figura 3.7.

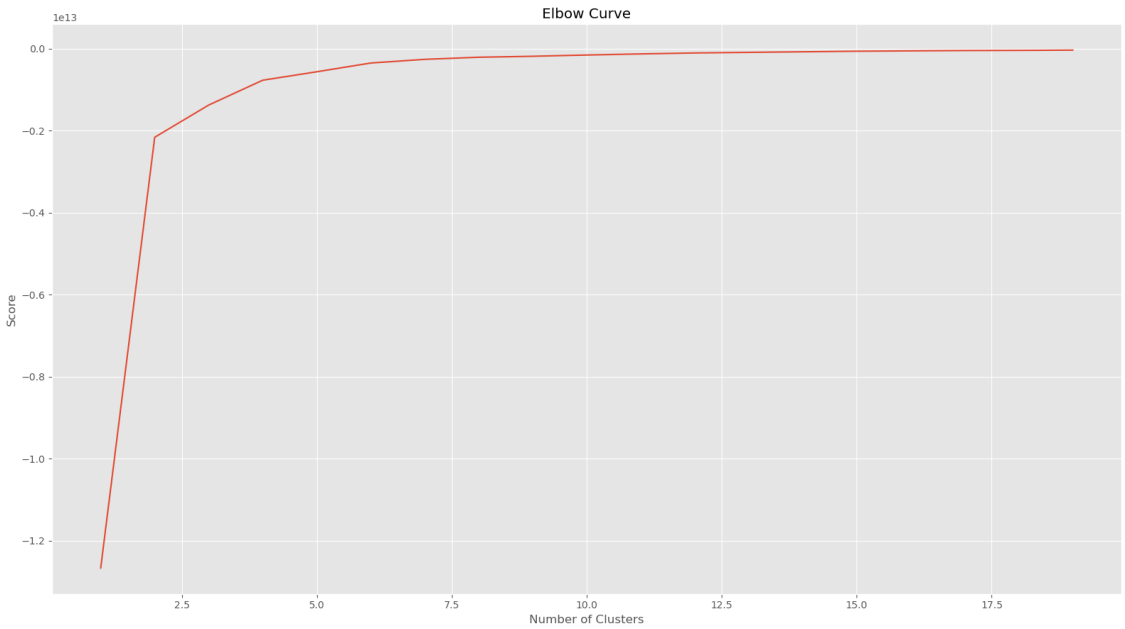

Figura (3.7) Curva Elbow de clusters para el dataset de Xu et al. [40].

Se puede ver que los mejores resultados se obtienen a partir de aproximadamente los 6 clusters, por lo que va a ser el número de clusters que se van a utilizar. Teniendo en cuenta este valor, ya se puede realizar el clustering. El resultado obtenido se muestra en la Tabla 3.7.

	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6
Bully	56	0	0	0	0	0
Victim	114	0	0	1	0	0
Accuser	67	0	0	1	0	0
Reporter	156	0	1	7	1	0
Defender	34	0	0	0	2	0
NA	1424	1	2	21	16	2

Tabla (3.6) Número de usuarios por cluster utilizando el algoritmo K-means con k=6. con el dataset de Xu et al. [40]

Observando los resultados del proceso de clustering vemos que la mayoría de los usuarios se encuentran en el cluster 1, por lo que las características de los perfiles no permiten en este caso hacer una distinción entre los usuarios con un perfil de acosador y/o víctima.

Para comprobar si la razón de este resultado era el número de características de los usuarios usadas se ha realizado un estudio mediante un script en Python en el que se realiza un PCA (Principal Component Analysis) para comprobar qué componentes son las más importantes. Una vez se ha ejecutado este script se han obtenido los siguientes resultados (ver Figura 3.8).

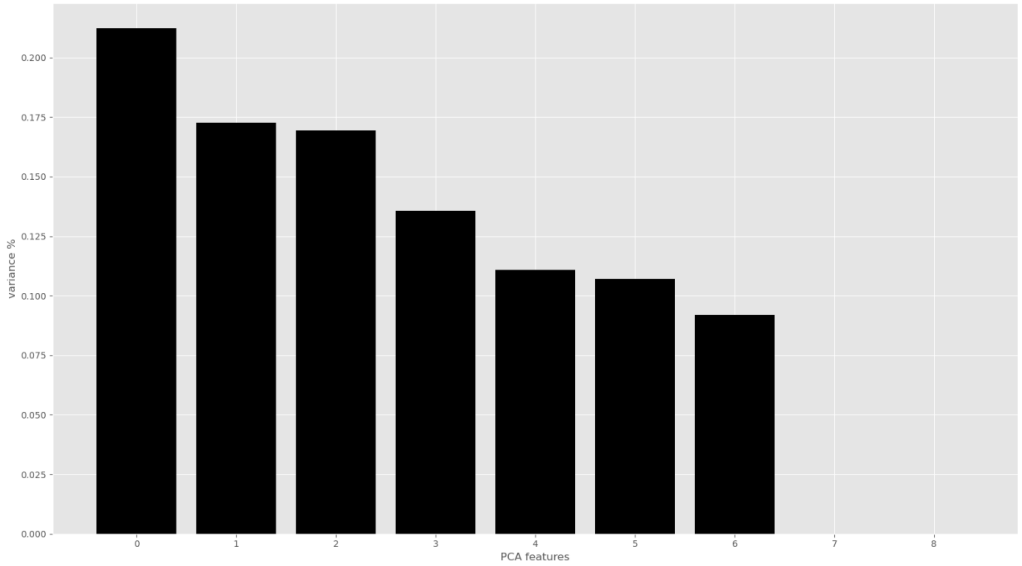

Figura (3.8) Importancia de las características de los usuarios para realizar el clustering del dataset Xu et al. [40]

Se puede ver que las componentes más influyentes para la realización del clustering son las 3 primeras, las cuales son las características de los usuarios "location", "name" y "description". Una vez tenemos esta información, se ha realizado el clustering teniendo en cuenta solamente estas 3 características de los usuarios (ver Tabla 3.7).

	Cluster1	**Cluster2**	**Cluster3**	**Cluster4**	**Cluster5**	**Cluster6**
Bully	11	34	9	2	0	0
Victim	26	69	13	7	0	0
Accuser	13	46	6	3	0	0
Reporter	26	120	10	12	0	0
Defender	6	27	3	0	2	0
NA	305	972	97	92	0	0

Tabla (3.7) Número de usuarios por cluster utilizando el algoritmo K-means con k=6 y realizando un PCA para reducir el número de características del dataset Xu et al. [40]

Aunque ahora los usuarios han quedado clasificados en un mayor número de clusters, no se pueden obtener unos resultados claros para diferenciar a los usuarios que puedan producir ataques de cyberbullying.

Ahora se va a hacer lo mismo con el dataset de Founta et al. [11], con el que se han podido recuperar una mayor cantidad de usuarios. Lo primero que se debe hacer es calcular la curva Elbow para comprobar cuantos clusters hacen falta. Puede verse la Figura 3.9.

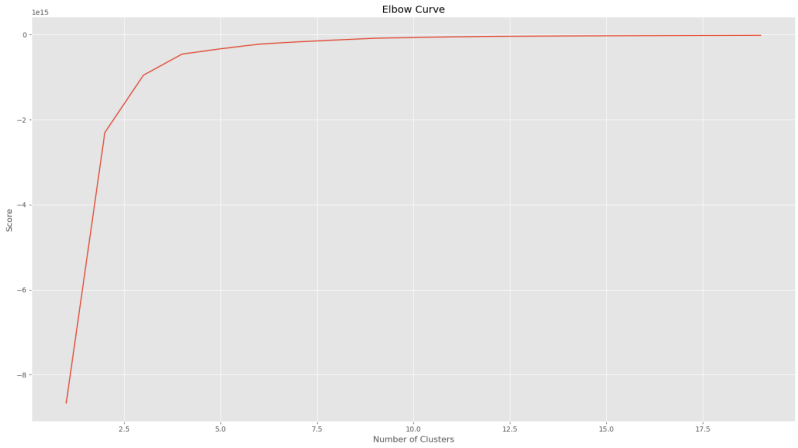

Figura (3.9) Curva Elbow de clusters para el dataset de Founta et al. [11]

Puede verse que la curva es muy parecida a la obtenida anteriormente, y que la puntuación máxima se obtiene alrededor de los 6 clusters, por lo que va a ser el número de ellos que se va a utilizar para realizar el clustering. Se puede ver los resultados en la Tabla 3.8.

	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6
Abusive	1001	0	1	0	0	0
Hateful	4586	0	0	0	0	3
Spam	3315	0	0	18	0	0
Normal	15342	1	4	29	2	6

Tabla (3.8) Número de usuarios por cluster utilizando el algoritmo K-means con k=6 con el dataset Founta et al. [11]

Se puede ver que la mayoria de los usuarios han sido añadidos al mismo cluster, por lo que no se puede obtener ninguna clasificación clara de los usuarios. Al igual que con el anterior dataset, se va a probar a realizar un PCA para comprobar cuales son las componentes más importantes en el caso de este dataset. Puede verse el resultado en la gráfica que aparece en la Figura 3.10

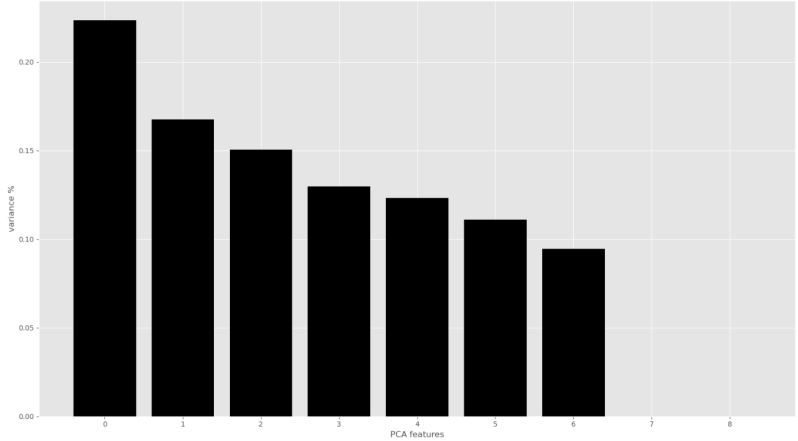

Figura (3.10) Importancia de las características de los usuarios para realizar el clustering del dataset Founta et al. [11]

Se puede ver que las componentes de los usuarios más importantes son las tres primeras, las cuales son: ”location”, ”name”, ”description”. Por lo que se va a hacer el clustering haciendo uso de estas 3 componenetes. Puede verse el resultado en la tabla 3.9

	Cluster1	**Cluster2**	**Cluster3**	**Cluster4**	**Cluster5**	**Cluster6**
Abusive	664	186	84	68	0	0
Hateful	2959	996	359	275	0	0
Spam	2176	591	401	165	0	0
Normal	10799	2649	1238	698	2	0

Tabla (3.9) Número de usuarios por cluster utilizando el algoritmo K-means con k=6, realizando un PCA para reducir el número de características del dataset Founta et al. [11]

Al igual que con el anterior dataset, no se pueden ver unos resultados claros para diferenciar a los usuarios con claridad. Por lo tanto, no se va a poder utilizar la información de los usuarios para identificar posibles perfiles de víctimas o de acosadores y reducir la cantidad de perfiles de la red social a analizar.

3.4 Conclusiones

Se han entrenado distintos modelos haciendo uso de dos datasets y diferentes técnicas, obteniendo finalmente un modelo que obtiene unos resultados por encima de los demás. En concreto, este modelo ha sido entrenado haciendo uso de las características del texto con redes neuronales LSTM. Se ha evaluado el efecto de hacer uso de los datos de los perfiles de los usuarios para aportar más información a la hora del entrenamiento, y para poder identificar determinados perfiles que redujesen la cantidad de textos a analizar, sin embargo los resultados obtenidos no han sido buenos por lo que finalmente se ha descartado esta opción.

Ahora que ya tenemos el modelo para detección de cyberbullying se va a integrar en la red social del grupo GTI-IA llamada PESEDIA.

CAPÍTULO 4
Detección de cyberbullying en PESEDIA

En este capítulo se va a explicar cómo se ha incluido el modelo entrenado como un plug-in para la red social PESEDIA, la cual es una red social para asesorar a los adolescentes en privacidad desarrollada por el grupo de investigación GTI-IA. Este capítulo consta de dos partes. Primero se describe la estructura de PESEDIA y del servicio web donde se va a implementar el modelo. En la segunda sección se va a explicar cómo se ha realizado la implementación del servicio web y del plug-in.

4.1 Herramienta de detección de cyberbullying

Como se ha dicho anteriormente, PESEDIA [1] es una red social desarrollada para asesorar a los adolescentes en un uso adecuado de las redes sociales [2, 1]. Para entender un poco más cómo funciona esta plataforma puede verse la Figura 4.1 donde se muestra la arquitectura del sistema.

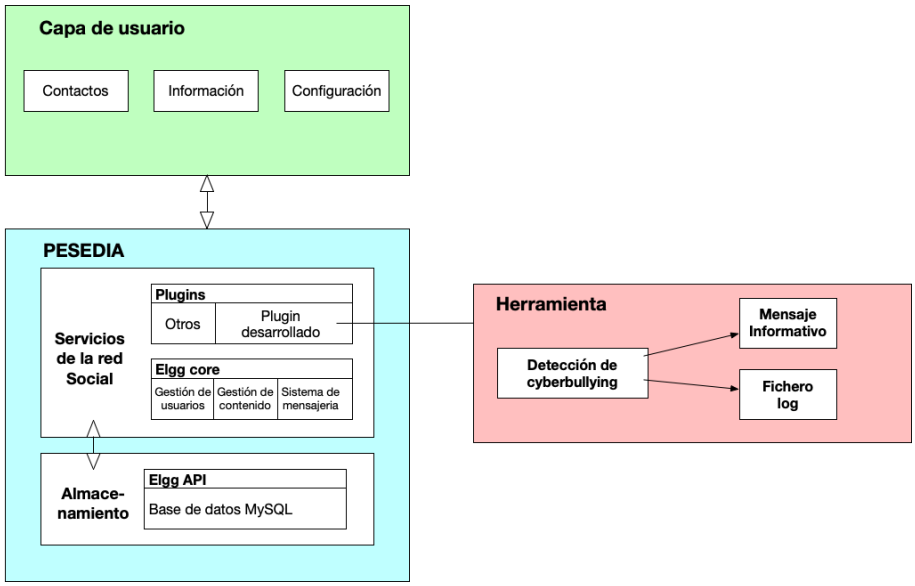

Figura (4.1) Visión general de la arquitectura de PESEDIA con la integración de la herramienta de detección de cyberbullying.

[1] https://pesedia.webs.upv.es

En la arquitectura de PESEDIA se identifican dos partes:

- **La capa de usuario**: es la encargada de permitir la interacción entre los usuarios y la red social.
- **El núcleo de la red social PESEDIA**: está formada por los servicios de la red social y por una capa de persistencia.

La capa de usuario es la que se encarga de mostrar toda la información asociada a los usuarios. Esta capa está dividida en tres partes: contactos (donde se encuentran los amigos y las relaciones del usuario), información (perfil, publicaciones, mensajes informativos de la red hacia los usuarios, etc.) y la configuración de la cuenta.

El núcleo de la red social PESEDIA está separado en dos partes principales. La primera son los servicios de la plataforma, donde se encuentran los plug-ins, para añadir funcionalidad a la red, aquí se encuentra el plug-in que se ha desarrollado y el core de la red. La segunda parte es una capa de persistencia donde se va almacenando toda la información generada en la red social (publicaciones, mensajes, etc.).

Se ha implementado como un servicio web y se ha integrado como una nueva funcionalidad dentro de la red social por medio de un plug-in. El plug-in es el responsable de obtener los mensajes de la red socíal y el servicio web es el encargado de procesar y analizar esos mensajes para determinar si hay indicios de bullying o no en una cuenta de usuario.

El funcionamiento del servicio web es el siguiente: primero llegan los mensajes, a los que se les hace un preprocesado para dejarlos limpios y en el formato necesario para ser analizados por el modelo entrenado con la red neuronal LSTM+WE. A continuación, una vez ya se han analizado todos los mensajes, se muestra por pantalla al usuario (solamente usuarios con perfil de administrador), cuántos mensajes del total de mensajes publicados por el usuario en su perfil han sido identificados como posibles mensajes de acoso. Esta información se muestra con un color representativo (verde, naranja o rojo) según la cantidad de mensajes detectados. Además, para dar más información al administrador, se genera un fichero de log en formato CSV. Este fichero tiene como nombre el identificador del perfil del usuario al que se le ha realizado el análisis, y en su interior los textos analizados y la etiqueta asignada que indica si el texto tiene contenido relacionado con el cyberbullying. Se puede ver este proceso en la Figura 4.2.

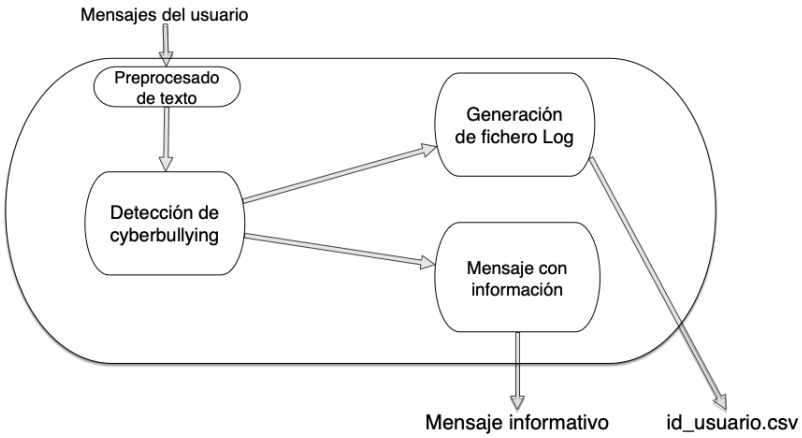

Figura (4.2) Diseño detallado de la herramienta para la detección de cyberbullying en PESEDIA.

Este fichero de log se genera para proporcionar información adicional al administrador. De esta forma, él puede comprobar de forma manual qué tipo de publicaciones se

han hecho en caso de que haya dudas sobre el comportamiento del usuario. Además, los ficheros de log revisados por el administrador podrán ser utilizados para ampliar el dataset con el que entrenar al clasificador.

4.2 Desarrollo del servicio web

La red social PESEDIA esta desarrollada en Elgg[2], una tecnología usada principalmente para el desarrollo de redes sociales.

Como se había mencionado anteriormente, se ha realizado la implementación un plug-in para poder conectar la plataforma con el servicio web. Para poder realizar una correcta implementación de los plug-ins en Elgg se debe de seguir una cierta estructura, para ello lo primero es generar los siguientes ficheros en la ruta raíz del plug-in:

- **start.php**, fichero en el que se realizan las llamadas para iniciar el plug-in e iniciará las funcionalidades.
- **manifest.xml**, fichero que sirve a modo de descripción de las funcionalidades del plug-in.

Los directorios que debe tener este tipo de plug-ins son los siguientes:

- **/actions**, en este directorio se incluyen las acciones que modifiquen las bases de datos.
- **/classes**, sitio donde se definen las clases para que puedan ser reconocidas por Elgg.
- **/languages**, ruta en la que se almacenan los diccionarios de palabras para poder utilizar el plug-in implementado en distintos idiomas.
- **/vendor**, ruta para librerías de terceros.
- **/views/default**, ruta del código que cambie o genere vistas en la red social.

Debido a que el plug-in a implementar deber leer de la base de datos, y hacer cambios en la interfaz del perfil de usuario (en la vista como administrador), se debe de crear, por un lado en la ruta "/views/default" el código para modificar la interfaz, y en "/action" el código para acceder a la base de datos.

Para el acceso a la base de datos se ha implementado un script en PHP. Este script hace una búsqueda en la base de datos de todas las publicaciones de un usuario a partir del identificador del usuario. Estas publicaciones se guardan en un fichero JSON cuyo nombre es el identificador de usuario. Por otro lado para la modificación de la interfaz se ha utilizado JavaScript. En esta parte se ha realizado una modificación del perfil de los usuarios, para que haya un botón el cual recupere el identificador del usuario y llame al script de PHP. Cuando el script php haya finalizado su ejecución, a través de la librería XMLHttpRequest, el código en JavaScript envía todos los mensajes del fichero JSON generado para ser analizados por el servicio web. Se puede ver un ejemplo de la interfaz de este plug-in en las Figuras 4.3 y 4.4.

[2] https://elgg.org

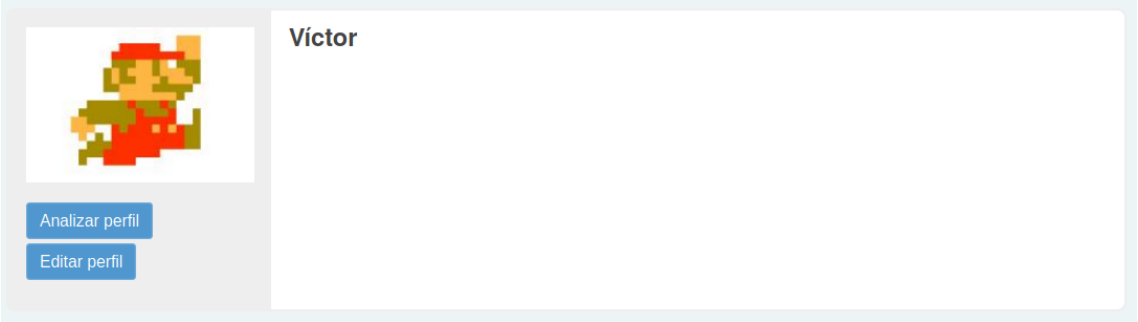

Figura (4.3) Interfaz del perfil del usuario antes del análisis de sus mensajes por parte del administrador.

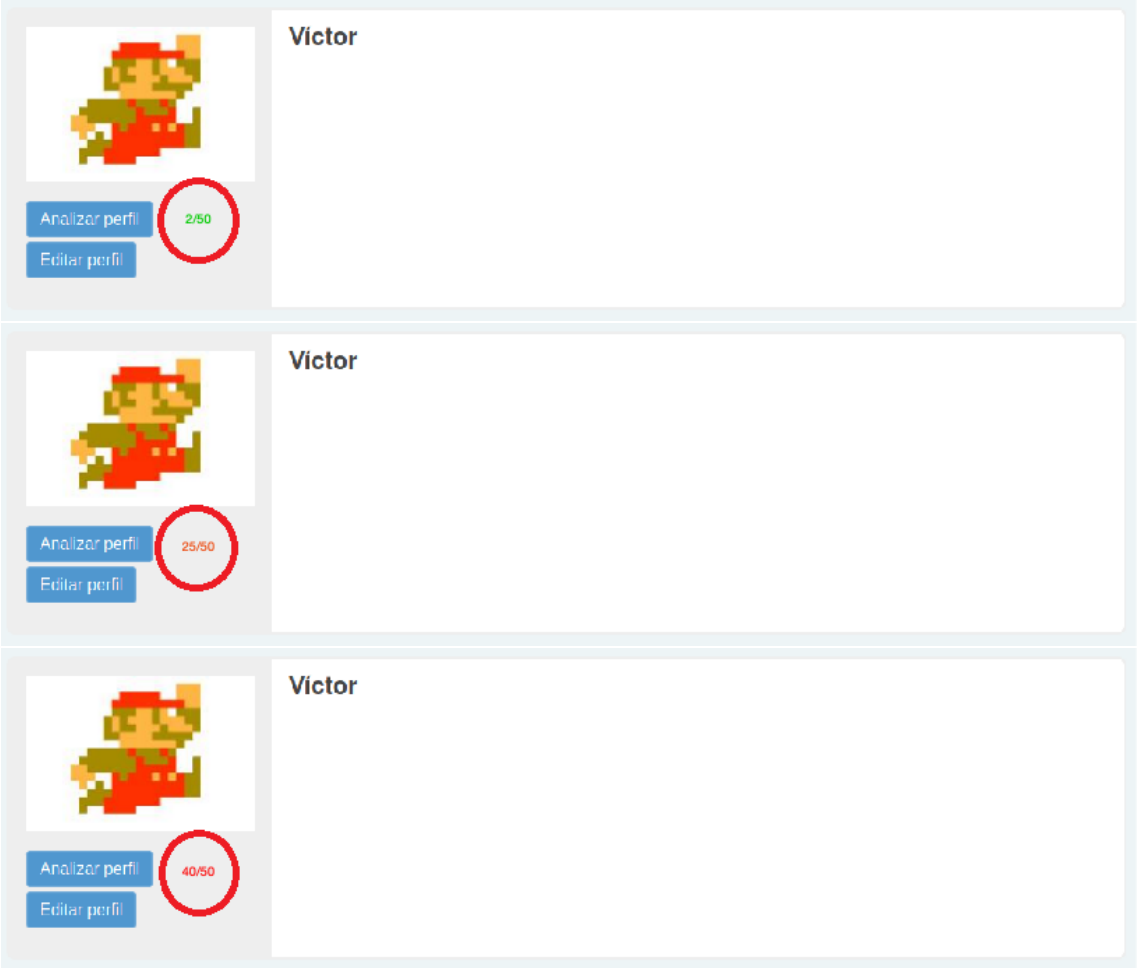

Figura (4.4) Interfaz del perfil del usuario después del análisis de sus mensajes. Según el número de mensajes con acoso detectados el resultado se muestra con color verde/naranja/rojo.

En la Figura 4.5 puede verse un ejemplo del fichero de log resultante al solicitar un análisis de los textos de un usuario. En este fichero se pueden ver las publicaciones realizadas y la etiqueta (en número) que le ha asignado el clasificador.

Texto	Etiqueta
Stop Bullying Ppl Old America	5
Georgetown got there asses kicked but china.. Bull got cleaned up with a chair	5
Three bully vans just went past Candace's Lewisham riots are starting again lorddddddd	5
totally. on one hand the retaliatory nature is pissy, on the other is great to see that bully-stick smackdown.	5
This is actually bullying , I want Wenger to watch this and fucking hold every piece of criticism that comes ! #RAGE	3

Figura (4.5) Fichero de log generado por el servicio web desarrollado

Como se ha mencionado, la parte del clasificador se ha implementado como un servicio web, haciendo uso de la librería de Python "http.server", la cual viene incorporada en este lenguaje de programación. Este servicio se ha configurado para que atienda todas las peticiones que entren en la dirección 0.0.0.0 y puerto 80, solamente procesando aquellas que sean de tipo POST. Una vez llegan todos los mensajes a procesar, limpiará los mensajes dejándolos en el formato adecuado, cargará el fichero en formato "h5" donde se guarda el modelo, se clasificarán los textos. El resultado de la clasificación se guardará en un fichero CSV con la etiqueta asociada. Además, el servicio responderá a la petición con el número de textos con acoso detectados. Esta respuesta es la que recibirá el plug-in desarrollado para mostrarla en la interfaz del perfil de usuario al administrador de la red.

CAPÍTULO 5
Conclusiones

En este TFM se ha desarrollado una herramienta para la detección de cyberbullying en redes sociales la cual se ha implementado en la red social PESEDIA. La herramienta en si consiste en tres fases, la primera de ellas es un preprocesado de los datos para dejarlos listos para poder ser clasificados. La segunda parte es la clasificación, la cual se hace mediante el modelo entrenado. La última es la generación de un mensaje para informar al administrador de la red y un fichero de log para poder comprobar los mensajes clasificados como acoso.

Inicialmente se realizó un trabajo de investigación para buscar trabajos que estuvieran relacionados con la detección del cyberbullying. Gracias a esta investigación se encontraron los clasificadores que suelen ser más utilizados para estas tareas, así como un conjunto de datasets generados por distintos grupos de investigación para combatir el acoso en las redes sociales.

El proceso de creación del modelo se ha realizado evaluando un conjunto de clasificadores clásicos y basados en redes neuronales con los diferentes datasets. De esta forma, se ha evaluado de forma empírica los modelos obtenidos por cada uno de ellos. Esta fase ha tomado una gran cantidad de tiempo, ya que algunos de los clasificadores, en concreto, Random Forest, Support Vector Machines y la red neuronal LST+WE han necesitado semanas para terminar de realizar el entrenamiento.

También se ha buscado mejorar los resultados obtenidos en la clasificación de textos haciendo uso de las características de usuarios. Esto, al igual que los anteriores experimentos también necesitó de mucho tiempo, aunque finalmente no mejoró los resultados. Por esta razón se decidió intentar realizar un filtrado de usuarios realizando un clustering para intentar encontrar que características tienen en común los distintos perfiles involucrados en escenarios de cyberbullying. Una vez realizado el clustering se pudo ver que los perfiles no podían ser separados en distintos clusters y por tanto no se podía realizar ese primer filtrado.

Finalmente el mejor modelo generado se ha integrado en un servicio web para ser utilizado en la red social PESEDIA, para la cual se ha tenido que implementar un plug-in para poder comunicar la plataforma con el servicio web.

5.1 Trabajos futuros

Para finalizar la realización de este trabajo se van a plantear posibles ampliaciones y trabajos futuros que se podrían realizar a partir del trabajo realizado.

Uno de los trabajos futuros sería realizar una implementación de una red neuronal BERT, una red neuronal basada en transformers la cual está siendo muy utilizada para el análisis de texto, por lo que podría obtener unos buenos resultados en la tarea planteada en este proyecto. Otro posible trabajo futuro sería recoger datos y etiquetarlos a mano, de esta forma se podría crear un dataset en español y con datos actuales, lo cual ayudaría a poder utilizar las características de los usuarios debido a que con el paso del tiempo muchas cuentas o publicaciones han sido eliminadas, ya sea por el propio usuario o por la plataforma, lo cual dificulta la recolección de datos y la obtención de unos buenos resultados.

Como una posible ampliación de la herramienta, se plantea realizar algo parecido a lo implementado con el fichero de log, pero que a la vez que da información al administrador, este también pueda confirmar si los textos han sido bien clasificados, y de esta forma poder reentrenar la herramienta con estos textos para ir mejorándola gradualmente con el tiempo.

Como experimentación futura, se podría probar con casos reales en PESEDIA en un escenario internacional. En la UPV, para este curso estaba planificado una acción de ese estilo con alumnos de intercambio llamada "Technovation Families" [1], la cual es un taller de jornadas abiertas impulsado por el American Space del Centro de lenguas de la UPV en la que se pretendía que alumnos internacionales hicieran uso de la red social PESEDIA. Debido a la situación extraordinaria de este año 2020, debido a la aparición del Covid-19 esta actividad tuvo que ser cancelada y post puesta para otro año, por lo que no se ha podido realizar esta experimentación con casos reales.

[1] https://cdl.upv.es/american-space/curiosity-machine-artificial-intelligence-family-challenge-2018

Bibliografía

[1] J. Alemany, E. del Val, J. Alberola, and A. García-Fornes. Enhancing the privacy risk awareness of teenagers in online social networks through soft-paternalism mechanisms. *International Journal of Human-Computer Studies*, 129:27–40, 2019.

[2] J. Alemany, E. del Val, and A. García-Fornes. Assisting users on the privacy decision-making process in an osn for educational purposes. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 379–383. Springer, 2020.

[3] E. L. Anderson, E. Steen, and V. Stavropoulos. Internet use and problematic internet use: A systematic review of longitudinal research trends in adolescence and emergent adulthood. *International Journal of Adolescence and Youth*, 22(4):430–454, 2017.

[4] J. Bayzick, A. Kontostathis, and L. Edwards. Detecting the presence of cyberbullying using computer software. 2011.

[5] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of machine learning research*, 2(Dec):125–137, 2001.

[6] V. Botti-Cebriá, E. del Val, and A. García-Fornes. Automatic detection of sensitive information in educative social networks. In Á. Herrero, C. Cambra, D. Urda, J. Sedano, H. Quintián, and E. Corchado, editors, *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020)*, pages 184–194, Cham, 2021. Springer International Publishing.

[7] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[8] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*, 13:21–27, 1967.

[9] U. I. de Valencia. Las diversas formas de bullying: físico, psicológico, verbal, sexual, social y ciberbullying.

[10] A. Edwards and A. Leatherman. Chatcoder: Toward the tracking and categorization of internet predators. 3, 01 2009.

[11] A.-M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, and N. Kourtellis. Large scale crowdsourcing and characterization of twitter abusive behavior. In *11th International Conference on Web and Social Media, ICWSM 2018*. AAAI Press, 2018.

[12] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[13] T. K. Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[14] T. K. Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[15] http://avellano.fis.usal.es/ lalonso/RNA/index.html. Redes neuronales.

[16] https://es.statista.com/estadisticas/600712/ranking-mundial-de-redes-sociales-por-numero-de usuarios/. Ranking de las principales redes sociales a nivel mundial según el número de usuarios mensuales activos en enero de 2020.

[17] https://www.dailymail.co.uk/news/article-2261588/Ask-fm-Pupils-parents-warned-social-networking-website-linked-teen abuse.html. Pupils and parents warned over social networking website linked to teen abuse.

[18] https://www.ditchthelabel.org/wp-content/uploads/2020/05/The Annual-Bullying-Survey-2017-2.pdf. The annual bullying survey 2017.

[19] https://www.fastcompany.com/most-innovative companies/2012formspring. Most innovative companies 2012.

[20] https://www.lavanguardia.com/tecnologia/20170201/413876302461/twitter-libertad-de-expresion-acoso-online-abuso bullying.html. Twitter anuncia medidas contra el acoso en su plataforma.

[21] https://www.theatlantic.com/technology/archive/2018/10/teens-face-relentless-bullying instagram/572164/. Teens are being bullied 'constantly' on instagram.

[22] Q. Huang, V. K. Singh, and P. K. Atrey. Cyber bullying detection using social and textual analysis. In *Proceedings of the 3rd International Workshop on Socially-Aware Multimedia*, pages 3–6, 2014.

[23] J. A. R. j.amatrodrigo@gmail.com. Máquinas de vector soporte (support vector machines, svms), 2017.

[24] H.-T. Kao, S. Yan, D. Huang, N. Bartley, H. Hosseinmardi, and E. Ferrara. Understanding cyberbullying on instagram and ask. fm via social role detection. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 183–188, 2019.

[25] E. Kleinberg. Stochastic discrimination. *Annals of Mathematics and Artificial intelligence*, 1(1-4):207–239, 1990.

[26] E. Kleinberg et al. An overtraining-resistant stochastic modeling method for pattern recognition. *The annals of statistics*, 24(6):2319–2349, 1996.

[27] E. M. Kleinberg. On the algorithmic implementation of stochastic discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):473–490, 2000.

[28] R. M. Kowalski and S. P. Limber. Psychological, physical, and academic correlates of cyberbullying and traditional bullying. *Journal of adolescent health*, 53(1):S13–S20, 2013.

[29] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[30] J. Mounir, M. Nashaat, M. Ahmed, Z. Emad, E. Amer, and A. Mohammed. Social media cyberbullying detection using machine learning. *International Journal of Advanced Computer Science and Applications*, 10:703–707, 01 2019.

[31] h.-c.-c.-b.-a.-a. Oscar Gutierrez. Twitter anuncia medidas para combatir el acoso en la red social.

[32] R. I. Rafiq, H. Hosseinmardi, S. A. Mattson, R. Han, Q. Lv, and S. Mishra. Analysis and detection of labeled cyberbullying instances in vine, a video-based social network. *Social network analysis and mining*, 6(1):88, 2016.

[33] K. Reynolds, A. Kontostathis, and L. Edwards. Using machine learning to detect cyberbullying. In *2011 10th International Conference on Machine learning and applications and workshops*, volume 2, pages 241–244. IEEE, 2011.

[34] M. Rezvan, S. Shekarpour, L. Balasuriya, K. Thirunarayan, V. L. Shalin, and A. Sheth. A quality type-aware annotated corpus and lexicon for harassment research. In *Proceedings of the 10th ACM Conference on Web Science*, WebSci '18, page 33–36, New York, NY, USA, 2018. Association for Computing Machinery.

[35] H. Rosa, J. P. Carvalho, P. Calado, B. Martins, R. Ribeiro, and L. Coheur. Using fuzzy fingerprints for cyberbullying detection in social networks. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7. IEEE, 2018.

[36] H. Rosa, N. Pereira, R. Ribeiro, P. C. Ferreira, J. P. Carvalho, S. Oliveira, L. Coheur, P. Paulino, A. V. Simão, and I. Trancoso. Automatic cyberbullying detection: A systematic review. *Computers in Human Behavior*, 93:333–345, 2019.

[37] R. E. Schapire, Y. Freund, P. Bartlett, W. S. Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

[38] R. Sugandhi, A. Pande, A. Agrawal, and H. Bhagat. Automatic monitoring and prevention of cyberbullying. *International Journal of Computer Applications*, 8:17–19, 2016.

[39] R. S. Tokunaga. Review: Following you home from school: A critical review and synthesis of research on cyberbullying victimization. *Comput. Hum. Behav.*, 26(3):277–287, May 2010.

[40] J.-M. Xu, K.-S. Jun, X. Zhu, and A. Bellmore. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, page 656–666, USA, 2012. Association for Computational Linguistics.

[41] X. Zhang, J. Tong, N. Vishwamitra, E. Whittaker, J. P. Mazer, R. Kowalski, H. Hu, F. Luo, J. Macbeth, and E. Dillon. Cyberbullying detection with a pronunciation based convolutional neural network. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 740–745. IEEE, 2016.

[42] R. Zhao and K. Mao. Cyberbullying detection based on semantic-enhanced marginalized denoising auto-encoder. *IEEE Transactions on Affective Computing*, 8(3):328–339, 2016.

[43] R. Zhao, A. Zhou, and K. Mao. Automatic detection of cyberbullying on social networks based on bullying features. In *Proceedings of the 17th international conference on distributed computing and networking*, pages 1–6, 2016.