Universitat Politècnica de València
Departamento de Sistemas Informáticos y Computación

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Análisis y detección de odio en mensajes de Twitter

Gretel Liz De la Peña Sarracén

Tutor:

Paolo Rosso

Trabajo Final de Máster desarrollado dentro del Máster en
Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Valencia, Septiembre 2019
Curso 2018/2019

Resumen

En la actualidad, la Web constituye un medio donde usuarios de todo el mundo interactúan entre sí, realizando actividades como el comercio digital, la búsqueda de información y la toma de decisiones. De esta forma sitios como las redes sociales han capturado el interés de usuarios y también de analistas. Si bien este fenómeno puede representar una ventaja para el desarrollo de las comunicaciones y la adquisición de información, en este contexto también se han detectado algunas manifestaciones negativas que pueden afectar a diferentes grupos de personas.

Los mensajes de odio son un ejemplo de dichos comportamientos negativos, que se publican con frecuencia en redes sociales de gran difusión como Twitter. Estos mensajes expresan odio hacia determinados grupos de personas en función de algún aspecto específico de su identidad, tal como su origen étnico, nacionalidad o religión. Se caracterizan generalmente por ser mensajes virales y por el anonimato de sus autores. Además, diferentes especialistas han identificado que incitan al odio contra el grupo de personas que constituye el objeto de odio de los mensajes, y que incluso, en muchas ocasiones pueden provocar acciones violentas contra dichas personas.

Debido a la repercusión que este tipo de publicaciones puede causar en muchas personas, diferentes esfuerzos se han comenzado a realizar. En este sentido, en los últimos años se han organizado varias tareas de evaluación relacionadas con la detección de mensajes de odio.

En este trabajo se realiza un análisis de un conjunto de estas tareas, enfocadas en mensajes publicados en Twitter. Se analizan en general las propuestas realizadas por diferentes equipos y en particular nuestras propuestas. Con el estudio de diferentes factores involucrados en las tareas se realiza un conjunto de experimentos. Con lo que se hace una comparación de las estrategias utilizadas y de otras ideas que proponemos. Como resultado se proporciona un resumen de aspectos importantes que pueden servir como guía en el diseño de una aproximación para la detección de mensajes de odio, o como punto de partida para próximos estudios.

Abstract

Nowdays, the Web constitutes a way where users around the world interact with each other, carrying out important activities such as digital commerce, search of information and decision making. Thus, sites like social networks have captured the interest of both users and analysts. This phenomenon may represent an advantage for the development of communications and the acquisition of information. However, some negative behaviour, that may affect different groups of people, have also been detected in this context.

Hate speech is an example of such negative behaviour, which is frequently published on popular social networks such as Twitter. It expresses hatred towards certain groups of people based on some specific aspect of their identity, such as their ethnicity, nationality or religion. It is generally characterized by being viral messages and by the anonymity of their authors. Specialists have identified that it incites hatred against people who are the object of hate in the messages, and that it can bring on violent actions against them in many occasions. Due to the impact this can cause on many people, different efforts have begun to develop. In this sense, several evaluation tasks related to the detection of hate speech have been organized in recent years.

In this work we carry out an analysis of a set of these tasks focused on messages published on Twitter. We analyze the proposed approaches made by different teams in general, and our proposals in particular. A set of experiments is performed with the study of the different factors involved in the tasks. In this way a comparison is made of the strategies used and other ideas that we propose. As a result, we provide a summary of some important aspects. It can be useful as a guide for future studies or in the design of an approach to the detection of hate speech.

Índice general

Índice de figuras

Índice de tablas

Abreviaturas

Bi-RNN	Bidirectional Recurrent Neural Networks
Bi-LSTM	Bidirectional Long Short Term Memory
BERT	Bidirectional Encoder Representations from Transformer
CNN	Convolutional Neural Network
DAN	Deep Averaging Network
DNN	Deep Neural Networks
GRU	Gated Recurrent Unit
HS	Hate Speech
IberEval	Evaluation of Human Language Technologies for Iberian Languages
IberLEF	Iberian Languages Evaluation Forum
LR	Logistic Regresion
LSTM	Long Short Term Memory
MLP	Multilayer Perceptron
PLN	Procesamiento del Lenguaje Natural
RNN	Recurrent Neural Network
SemEval	Semantic Evaluation
STD	Standar Deviation
SVM	Support Vector Machine
TF-IDF	Term Frequency – Inverse Document Frequency
USE	Universal Sentences Encoder
WE	Word Embeddings

Capítulo 1

Introducción

1.1. Descripción del problema, motivación y objetivos

La evolución de la Web ha provocado una revolución en el desarrollo de diversas aplicaciones en Internet. En consecuencia ha emergido un gran número de plataformas que permiten la publicación de contenido, tales como blogs, redes sociales y portales de alojamiento de fotos, audio, vídeos y comentarios. De esta forma, los usuarios son dotados con la posibilidad de interactuar, intercambiando información que les permite comentar acerca de la realidad que los rodea, o encontrar comentarios de otros usuarios que les ayude en la toma de decisiones.

Sin dudas, dentro de la lista de las herramientas en Internet, Twitter[1] se destaca como una de las más populares con una variedad de propósitos. Su éxito ha derivado en un sinfín de aplicaciones, muchas de ellas en el ámbito de la mensajería social. Con estas aplicaciones, los usuarios cuentan con herramientas de negocios, servicios de información, utilidades de marketing y medios para el intercambio de opiniones. Estas facilidades hacen que Twitter sea muy utilizada por los usuarios para obtener noticias de última hora.

Si bien estos avances representan ventajas para la búsqueda de información, muchos usuarios utilizan estos medios para difundir mensajes de odio contra diferentes grupos de personas. De esta manera, las oportunidades de interactividad de los usuarios en Internet, a pesar de representar un avance importante en la difusión de información y opiniones, incrementan la posibilidad de que sean propagadas publicaciones hirientes y denigrantes.

[1] Twitter

No existe una única definición de mensajes de odio, pero se pueden identificar algunos aspectos comunes que permiten caracterizar ese tipo de texto. En este sentido, de forma general se puede definir como mensaje de odio a cualquier expresión cuyo contenido es abusivo, insultante, intimidante, que hostiga o incita a la violencia, el odio o la discriminación. Este tipo de mensaje está dirigido contra personas en función de su raza, origen étnico, religión, sexo, edad, condición física, discapacidad, orientación sexual, convicción política, etc [33, 50]. Además se plantea que esos mensajes contribuyen a un clima general de intolerancia que provoca que los ataques sean más probables contra los grupos afectados[2].

Haciendo un análisis más detallado, el trabajo [34] realiza un estudio sobre los diferentes aspectos que distintas fuentes utilizan para definir al mensaje de odio. El análisis realizado muestra que todos los autores coinciden en que el mensaje de odio tiene objetivos específicos y se dirige a propiedades específicas de determinados grupos. Por otra parte, la mayoría de las definiciones señalan que el odio consiste en incitar a la violencia; y algunas otras definiciones establecen que el odio se manifiesta en las publicaciones al utilizar un lenguaje que ataque o disminuya a determinados grupos. Como resultado del estudio realizado, la autora define el mensaje de odio como el lenguaje que ataca o disminuye, que incita a la violencia o al odio contra determinados grupos, basado en aspectos específicas como religión, ascendencia, origen nacional o étnico, identidad de género u otras, y puede ocurrir con diferentes estilos lingüísticos, incluso en formas sutiles o cuando se usa el humor y el sarcasmo.

Una característica muy frecuente en los mensajes de odio es que se realizan de forma anónima y suelen ser virales. Estos aspectos hacen que este tipo de publicación sea potencialmente peligrosa. Por lo tanto, su identificación se convierte en una labor de gran importancia en diferentes campos de investigación, tales como la Sociología, la Psicología y el Procesamiento del Lenguaje Natural (PLN). De modo que algunas plataformas en línea han tomado medidas en contra de los mensajes de odio. Twitter por ejemplo, ha implementado una política relativa a las conductas de incitación al odio[3], que desaprueban la violencia, ataques o amenazas contra personas por motivo de su raza, origen étnico, nacionalidad, orientación sexual, género, identidad de género, afiliación religiosa, edad, discapacidad o enfermedad grave. Tampoco permite el uso de nombre de usuario, imágenes o símbolos

[2]Ilga Europe - Hate crime & hate speech
(https://www.ilga-europe.org/what-we-do/our-advocacy-work/hate-crime-hate-speech)

[3]Twitter - Hateful conduct policy
(https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy)

de incitación al odio en el encabezado de un perfil.

Muchos de los mecanismos utilizados para identificar mensajes de odio y llevar a cabo estas medidas se basan en informes de usuarios y moderadores de contenido. Sin embargo, a pesar de que esta constituye una vía que hace posible detectar estos mensajes, la gran cantidad de contenido que se publica diariamente en la actualidad, dificulta el análisis, haciéndolo impracticable. Lo que se complica aún más cuando se trata de publicaciones en diferentes variedades de los idiomas. Por lo tanto, construir sistemas capaces de detectar este tipo de mensajes de forma automática resulta una tarea de gran importancia.

Generalmente esta tarea se modela como un problema de clasificación supervisada, donde se debe predecir la presencia de alguna forma de odio. Esto no es trivial, ya que como se comentó no suelen utilizarse estructuras y/o expresiones lingüísticas específicas, y el lenguaje puede variar mucho según la gran diversidad de idiomas y sus variantes. Incluso, en muchas ocasiones el contenido de odio se expresa de forma sutil, aumentando así el reto de la detección automática de odio. Por lo que se hace necesario un análisis detallado de estrategias para el tratamiento automático de esta problemática.

En los últimos años la comunidad científica en el área del PLN ha tomado gran interés por este fenómeno. Lo que se ha visto reflejado con el desarrollado de una serie de tareas en el marco de diferentes eventos científicos. Dentro de ellas se puede mencionar la tarea en la detección de mensajes de odio HaSpeeDe[4] (*Hate Speech Detection*), propuesta como parte de EVALITA 2018; y las tareas HatEval[5] y OffensEval[6], en SemEval 2019. Otra tarea que ha sido desarrollada en relación con la problemática en cuestión es MEX-A3T[7] (*Autohorship and Aggressiveness Analysis*) en IberEval 2018 y más recientemente su segunda edición[8] en IberLEF 2019.

Como resultado de estas tareas se ha logrado que diferentes grupos de investigación se involucren en el tratamiento de la problemática. De esta forma, han sido propuestas aproximaciones basadas en diferentes estrategias, que van desde el uso de lexicones y técnicas tradicionales a modelos basados en algunas arquitecturas de aprendizaje profundo. Pese a que todas estas propuestas dan la posibilidad de analizar características y modelos que pueden ser adecuados para la detección de odio, en la práctica es complicado

[4]Haspeede-Evalita 2018 (www.di.unito.it/~tutreeb/haspeede-evalita18/index.html)
[5]HatEval-SemEval 2019 (https://competitions.codalab.org/competitions/19935)
[6]OffensEval-SemEval 2019 (https://competitions.codalab.org/competitions/20011)
[7]MEX-A3T-IberEval 2018 (https://mexa3t.wixsite.com/home)
[8]MEX-A3T-IberLEF 2019 (https://sites.google.com/view/mex-a3t/)

de realizar debido a la gran cantidad de eventos y participantes por cada uno de ellos. Donde en algunos casos algunos de los mejores modelos en una tarea no logran tan buenos resultados en otra.

En este trabajo se presenta un estudio de diferentes aproximaciones utilizadas en la detección de mensajes de odio con el propósito de brindar una visión general de las estrategias empleadas. El análisis se centra en las propuestas realizadas en eventos científicos recientes, donde se utilizan corpus conformados por tweets. Siendo el objetivo principal el análisis de importantes aspectos para la detección de tweets con mensajes de odio a partir de las experiencias acumuladas con la participación en las distintas tareas. De forma que se facilite una guía para el diseño de adecuadas aproximaciones.

Nuestra metodología está compuesta por tres etapas. Primeramente se evalúa la adecuación de nuestras propuestas con la participación en las distintas tareas. Luego se analizan los resultados alcanzados por dichas aproximaciones realizando una comparativa con otros sistemas presentados. Por último se realiza un conjunto de experimentos para obtener un resumen de los aspectos más importantes en la detección de mensajes de odio.

1.2. Estructura de la tesis

El resto de este trabajo está estructurado por cinco capítulos y un apéndice, que se introducen a continuación:

- **Capítulo 2** Estado de la cuestión.
  En este capítulo se describe el estado de la cuestión en la detección de mensajes de odio en texto. Primeramente se analizan las características de los mensajes de odio y otros tipos de mensajes estrechamente relacionados como los mensajes ofensivos y agresivos, la blasfemia y el ciberbullying. Luego se resumen las estrategias utilizadas por trabajos propuestos en el área. Por otra parte, se describen las características fundamentales de Twitter y de sus publicaciones. A continuación se presentan las particularidades de las aproximaciones en la detección de odio en tweets. Por último, se comentan brevemente las diferentes competiciones relacionadas con la tarea.

- **Capítulo 3** Marco teórico.
  En este capítulo se presentan los conceptos más importantes utilizados en el trabajo. Se describen las diferentes representaciones de texto que se han utilizado, así como las diferentes arquitecturas de los modelos empleados.

- **Capítulo 4** Detección de mensajes de odio en SemEval 2019.
  En este capítulo se detallan los aspectos relacionados con la participación en las tareas relacionadas con la detección de mensajes de odio en SemEval 2019. En cada caso se realiza una descripción de las características de la tarea, incluyendo el corpus utilizado. Luego se presenta un resumen de las estrategias enviadas a la competición. Por último, se comentan los resultados alcanzados, realizando una comparación con los modelos utilizados por otros sistemas. Posteriormente se dan unas conclusiones parciales como resumen de los resultados y experimentos realizados.

- **Capítulo 5** Análisis de agresividad en casos de estudios particulares.
  De forma similar al capítulo anterior, en este capítulo se detallan los aspectos relacionados con la participación en otras tareas relacionadas con la detección de mensajes de odio. Esta vez las tareas tienen la particularidad de centrarse en otras variedades de lenguajes.

- **Capítulo 6** Conclusiones y trabajo futuro.
  En el último capítulo se plantean las conclusiones extraídas como resultado del estudio realizado. Además, se propone una posible línea para el desarrollo de un trabajo futuro con el propósito de mejorar los resultados del estado de la cuestión.

- **Apéndice A** Publicaciones.
  En este apartado se muestran las publicaciones a las que ha dado lugar la investigación realizada en el marco de este trabajo final de máster y que se han descrito en la memoria.

Capítulo 2

Estado de la cuestión

En este capítulo se introducen los principales conceptos asociados al campo de investigación de la detección de mensajes de odio. Primeramente se analizan las características de este tipo de mensaje y de otros estrechamente relacionados, identificando las diferencias entre ellos. Luego se presentan diferentes estrategias utilizadas para el tratamiento de la tarea en cuestión; en especial, de las presentadas en diferentes competiciones relacionadas con el área. Además, se comentan las particularidades de las aproximaciones propuestas para el caso de mensajes publicados en Twitter. Para ello, primeramente se presentan algunas características de las publicaciones que pueden dificultar el análisis a nivel del procesamiento de textos; y de los aspectos específicos de esta red, que permiten la difusión de información. Por último, se presenta un resumen de diferentes competiciones que han sido realizadas en el área.

2.1. Detección de mensajes de odio

La propagación de mensajes de odio es un hecho que desafortunadamente ocurre frecuentemente mediante las redes sociales en Internet. Existe evidencia de que este tipo de mensajes produce violencia masiva, con lo que aumenta el daño y sufrimiento de los individuos afectados y con ello la probabilidad de suicidios.

Como se ha introducido, un mensaje de odio (*Hate Speech* (HS) en inglés) puede ser definido de forma resumida como un mensaje que, como indica el término, expresa o incita al odio hacia determinados grupos de personas en base a algún aspecto de su identidad. A menudo, el HS se considera en función del origen étnico, nacionalidad o religión, pero en teoría se puede

usar contra cualquier grupo. De hecho, a la hora de identificar HS el análisis no puede limitarse a las publicaciones dirigidas solo a categorías protegidas específicas. En la práctica pueden aparecer nuevos objetivos de odio de una categoría que no ha sido especificada, perjudicando a grupos de personas de forma posiblemente desapercibida. Tales mensajes también deberían detectarse como HS y ser tratados para evitar consecuencias negativas [34].

Identificar los HS constituye una tarea compleja con aspectos en los que el criterio de no todos los especialistas coincide. Algunos estudios describen una serie de propiedades usuales [56, 64, 48], de forma que puede detectarse contenido de odio en publicaciones que las cumplen tales como:

- Uso de insultos sexistas o raciales.
- Uso de términos despectivos con la intención de hacer daño.
- Comentarios hirientes sobre estereotipos bien conocidos.
- Respaldo de comentarios con HS.

Además, se exponen características a tener en cuenta para evitar la identificación errónea de odio en mensajes:

- La expresión de orgullo por pertenecer a un cierto grupo que tiende a discriminar no es considerada como HS, a menos que aparezca algún tipo de desprecio.
- La mención de una organización asociada con delitos de odio, no constituye un discurso de odio. Por ejemplo, el término *Ku Klux Klan* no es odioso en sí, ya que puede aparecer en comentarios sobre hechos o documentos históricos.
- Hablar de forma negativa de países y religiones no es considerado como HS, pero sí discriminar a grupos de personas en función de estas categorías.

Como se comentó anteriormente, estos son criterios que pueden servir de guía para la detección de HS. Sin embargo, muchos son subjetivos y pueden implicar que se determinen como mensajes de odio algunos que en realidad no lo son. Por ejemplo, en el segundo punto se plantea que se puede identificar como mensajes de odio a los que contienen términos despectivos; pero aún cuando la intención no sea hacer daño, el uso de ese tipo de términos puede provocar que el mensaje sea interpretado como odioso. Por lo que

el margen es muy estrecho, porque algunos usuarios utilizan un lenguaje cargado de esos términos pero la idea no es incitar al odio y la discriminación.

Por otra parte, el odio también puede identificarse en formas sutiles de discriminación. Donde muchas veces no se usan términos negativos, pero de forma implícita se expresa odio. Incluso, las bromas pueden contener contenido de odio y su repetición puede fortalecer las actitudes abusivas y provocar efectos negativos sobre determinadas personas [31, 41].

2.1.1. Conceptos relacionados con los mensajes de odio

Algunos autores definen a los mensajes de odio como aquellas publicaciones ofensivas en contra de un determinado grupo de personas [48]. Sin embargo, otros autores separan estos conceptos señalando diferencias entre ellos [23]. De modo que el HS puede verse como un caso particular de los mensajes ofensivos, los cuales pueden estar dirigidos también hacia una sola persona y abarcan además insultos, amenazas o lenguaje con malas palabras.

La agresividad es otro concepto relacionado estrechamente con el odio. La Real Academia Española (RAE) la define como la tendencia a actuar o a responder violentamente. Además, en [11] se dice que representa una forma de HS que de forma implícita o explícita necesariamente presenta, incita o sugiere actitudes y acciones violentas.

Con lo antes comentado, se puede resumir la relación entre estos conceptos de la forma que indica la Figura 2.1.

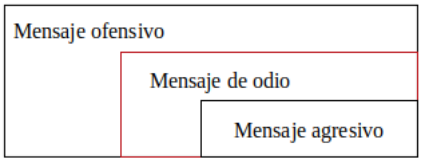

Figura 2.1: Relación entre algunos conceptos vinculados con HS.

Existen otros términos que también están estrechamente relacionados con la definición de HS, tales como el ciberbullying, el lenguaje tóxico, y la blasfemia. El ciberbullying es el ataque a través de vías electrónicas, que suele estar dirigido hacia una sola persona a diferencia del mensaje de odio que se dirige usualmente a un grupo de personas [42, 21, 13]. El lenguaje tóxico identifica a los mensajes groseros que pueden provocar que una persona termine una conversación [61], de forma que ellos contienen HS, pero no siempre se cumple lo contrario. Por otro lado, la blasfemia

se caracteriza por el uso de palabras obscenas [45], por lo que HS puede contener blasfemia pero no necesariamente lo contrario como antes.

El presente trabajo se centra en la detección de HS como una tarea de clasificación binaria, donde las clases están determinadas por la presencia o no de odio en un texto. Entendiendo por HS a cualquier mensaje que perjudique a un grupo de personas o a una persona en concreto. El término abusivo se utilizará como un concepto genérico para la referencia de cualquiera de los tipos de mensajes relacionados con HS.

2.1.2. Estrategias utilizadas en trabajos relacionados con la detección de mensajes de odio

Diferentes enfoques han sido empleados para el tratamiento computacional de la detección de mensajes de odio [35, 6]. Las aproximaciones propuestas van desde técnicas sencillas, basadas en métodos tradicionalmente utilizados en la minería de texto en general, hasta complejos modelos basados en el aprendizaje profundo y que constituyen el estado del arte en muchas otras tareas.

La estrategia más básica se corresponde con el procesamiento a nivel léxico del lenguaje natural. Se basa en el uso de recursos léxicos para el desarrollo de los métodos, de forma que la búsqueda de contenido de odio se determina según la ocurrencia de ciertos términos en el texto [60]. Diferentes tipos de lexicones se han utilizado, por un lado los que están conformados por un conjunto de palabras que suelen estar relacionadas con HS [23], por otro lado, los que contienen ciertas características que pueden indicar la presencia de HS según estudios realizados, tales como signos de puntuación y variaciones ortográficas [59]. Además, se ha analizado que determinados verbos pueden estar relacionado con el lenguaje utilizado en HS. En este sentido, en el trabajo [37] se genera un lexicón de verbos que pueden provocar violencia. Una de las ventajas de esta estrategia es posiblemente su facilidad de implementación. Por otra parte, permite identificar características que son adecuadas para la detección de HS y descartar otras que no lo son. Sin embargo, tiene como desventaja que falla en textos donde el odio es expresado de forma implícita, sin el uso de características obvias de HS, o con el uso de palabras enmascaradas con caracteres especiales como por ejemplo *m**rda*. Además, muchas veces el odio no se expresa solo en una palabra o característica, sino en todo el contexto.

Otra estrategia utilizada se basa en el diseño de métodos basados en plantillas, donde son identificadas palabras asociadas frecuentemente con expresiones de odio y sus contextos que posteriormente son utilizadas en

la detección de frases de odio. En [64] se utiliza un corpus de palabras de odio y para cada una de ellas se definen plantillas determinadas por diferentes tamaños de ventanas en su contexto. En [48] se propone la construcción de una estructura para el inglés con la que se pretende extraer los elementos de un mensaje de odio. Dicha estructura tiene la forma "I <intensity><userintent><hatetarget>", que se corresponde con textos como "I really hate immigrants". Donde, *intensity* captura los calificadores en caso de ser utilizados, en el ejemplo sería la palabra "really". El término *userintent* es el que expresa el odio en sí y *hatetarget* determina el objeto del odio. Con esta aproximación, el autor logra capturar mensajes de odio explícitos hacia algunos grupos de personas especificados, pero falla a la hora de detectar el odio en textos donde el odio no se expresa siguiendo la estructura propuesta.

Los métodos estadísticos tradicionales tales como Regresión Logística (LR, por las siglas en inglés), Árboles de Decisión y Máquinas de Vectores Soportes (SVM, por las siglas en inglés) [21, 17, 18, 9] también han sido ampliamente utilizados en la tarea. Estos métodos toman como entrada una representación del texto, obtenida a partir de un conjunto de características. De esta forma, los métodos aprenden determinados patrones estadísticos a partir de la representación que suele contener la información léxica. Dicha representación también puede estar enriquecida o simplemente conformada por otros tipos de información tales como características semánticas [65]. En este sentido han sido utilizados diferentes tipos de características como las que se obtienen con la técnica de bolsa de palabras [19] y de etiquetado gramatical [30]. Los n-gramas [43] han mostrado ser buenas características, aunque algunos trabajos señalan que el valor de n no debe ser pequeño debido a que palabras relacionadas pueden no estar muy cerca en la secuencia de palabras en el texto [21]. Generalmente estas características se extraen a nivel léxico, construyéndose un vector en el que se indica la presencia o no de las palabra del vocabulario definido. Un aspecto interesante es que varias investigaciones sugieren el uso de n-gramas de caracteres antes que de palabras, lo que puede deberse a que tratan el problema de la variabilidad ortográfica [46]. Además, se ha comentado que los sistemas pueden mejorar al combinarlos con otras características. En [50] se utiliza un modelo de regresión con diferentes características semánticas. Cuatro tipo de características son estudiadas: a nivel lingüístico, sintáctico, semántico y de n-gramas. Los mejores resultados reportados son obtenidos con la combinación de todas las características.

Muchos otros trabajos han utilizado métodos basados en el aprendizaje profundo. En este caso, el enfoque está más dirigido a la arquitectura de

los modelos que a las características lingüísticas en sí [36], aunque en algunos trabajos se combinan ambas estrategias. Las topologías más utilizadas son las redes convolucionales y las recurrentes, que se presentan con más detalle en el siguiente capítulo 3 (véase 3.2). Estos modelos han sido utilizados de forma independiente, pero también para la construcción de modelos más complejos a partir de la combinación de diferentes arquitecturas [53]. Además, han sido combinados con modelos estadísticos tradicionales [9, 28].

El análisis de sentimiento es una tarea que muchos identifican cercana a la detección de HS. De manera que es utilizada como una herramienta, considerando que generalmente los textos con sentimientos negativos contienen HS. En [62] se tiene en cuenta el número de características positivas, negativas y neutras para la detección de HS. Mientras que en [37] se detecta la polaridad negativa en textos como un paso previo a la detección de HS. Sin embargo, se debe tener en cuenta que no necesariamente un texto negativo contiene odio, por lo que esta estrategia puede sufrir de problemas de precisión en la detección de HS.

Otras aproximaciones se han desarrollado a partir de la suposición de que el HS está determinado por estereotipos bien conocidos y que para cada uno de ellos se suele utilizar un lenguaje particular. De forma que las propuestas se basan en buscar ciertas palabras o frases en los textos para identificar algún tipo de estereotipo, como por ejemplo latino o africano para detectar el odio hacia inmigrantes [64]. Esta estrategia falla con la variación del lenguaje y en la detección de odio hacia grupos de personas que no han sido considerados como posibles objetivos.

Algunas investigaciones sugieren que un aspecto que podría ser importante incluir en el análisis del HS es el contexto social en el que se crea un mensaje [59]. La mayoría de los trabajados propuestos se limitan al análisis del lenguaje utilizado, pero algunos trabajos sugieren incorporar métodos basados en el conocimiento sobre los usuarios. Tal es el caso de [52] y de [21], donde se utilizan patrones de los usuarios para poder identificar aquellos que tienen un comportamiento abusivo.

Por otra parte, en [8] se hace un análisis sobre la diferencia de resultados reportados en la detección de HS y los que se obtienen en la práctica. Señala que si bien se reportan muy buenos resultados en algunos conjuntos de datos específicos, en aplicaciones reales no se ha tenido mucho éxito. El estudio indica que el problema puede estar relacionado con el sobreajuste en cuanto a los usuarios, inducido por un sesgo significativo en relación con los usuarios en los conjuntos de datos etiquetados. Por lo que plantean que la detección de HS no se trata solo del lenguaje natural, sino también del análisis de los usuarios.

Muchos trabajos han sido publicados como resultado de las diferentes tareas de evaluación que se han organizado en los últimos años. Las diferentes aproximaciones serán presentados de forma resumida en capítulos posteriores.

2.2. Twitter

Actualmente, por el número de usuarios activos, Twitter constituye uno de los mayores exponentes de las redes de microblogging. Esta red se ha convertido en una importante herramienta de información en lo que respecta a comunicación oficial, principalmente por su capacidad de comunicar lo que está sucediendo en poco tiempo y de forma directa. Muchos personajes destacados como artistas, deportistas, empresarios, políticos y profesionales, utilizan su cuenta de Twitter como su canal de comunicación oficial.

Twitter permite publicar mensajes de texto plano con una longitud máxima de 280 caracteres. Por defecto, los mensajes son públicos, y cualquier usuario puede volver a publicar un mensaje de otro usuario, comentarlo y compartirlo.

2.2.1. Características de las publicaciones en Twitter

Entre las principales características o elementos que usualmente aparecen en un tweet se pueden señalar las siguientes:

- **Menciones**: Los mensajes pueden contener una o más referencias a otros usuarios de la red. Esta referencia se logra incluyendo el nombre del usuario en el texto, el cual aparece con el símbolo de @ (arroba).
- **Hashtags**: Se emplean para la representación de un tópico específico. El mensaje puede contener una o más de estas etiquetas, a través de las cuales se agrupan los tweets que hacen referencia a una de ellas en su texto. Se identifican por el símbolo inicial #.
- **URLs**: Permiten incluir referencias a enlaces en Internet.

Además de estas características presentes en la sintaxis de un tweet, es común encontrar otras características lingüísticas y emocionales. Los usuarios tienden a escribir palabras en forma de abreviaturas y a expresar sus emociones mediante conjuntos de símbolos o realizando énfasis en las palabras [16], como se presenta a continuación:

- **Emoticones**: Son expresiones faciales dibujadas empleando los símbolos del teclado. Estos por sí solos expresan un sentimiento sobre el estado de ánimo o una acción que desarrolla un individuo.
- **Lenguaje informal**: Es común encontrar en el texto de un tweet lenguaje informal [32]. Los términos informales constituyen abreviaturas y palabras con errores ortográficos intencionales, que se utilizan para expresar una palabra o frase con la menor cantidad de caracteres posibles. (Ej. Tqm!, que sería equivalente a escribir: Te quiero mucho!)
- **Palabras con énfasis**: Dos de los métodos más empleados por el usuario para hacer énfasis en una palabra son la escritura de todos los caracteres en mayúsculas, y la repetición continua de uno o más caracteres.

2.2.2. Particularidades en la detección de mensajes de odio en Twitter

Como se ha comentado, los tweets tienen rasgos particulares que por un lado pueden afectar la predicción si no se tienen en cuenta, pero que por otro lado pueden ser utilizados como características discriminatorias. Uno de estos rasgos es el cambio de caracteres alfabéticos por otro tipo de caracteres [50], tales como números y signos especiales. Las palabras *sh1t* y *@$$h*le* son ejemplos frecuentes en tweets con lenguaje abusivo. Este fenómeno puede implicar la pérdida de información al enmascararse palabras claves. Sin embargo, pueden beneficiarse los modelos que tengan en cuenta el uso de ese tipo de caracteres en algunas palabras del texto. En [49] se utiliza una distancia correspondiente al número mínimo de ediciones necesarias para transformar una cadena en otra, de forma que se pude determinar la similitud de una palabra con cambios ortográficos con respecto a palabras dentro de algún recurso lingüístico.

Por otra parte, los emoticones son términos que como también fue comentado anteriormente, contienen carga semántica y en muchos casos son utilizados para la clasificación de tweets según su polaridad. En el caso de la detección de HS esta idea también se ha aplicado [28]. En general, diferentes estrategias pueden ser utilizadas. En algunos casos pueden diseñarse modelos que utilicen los emoticones como características en la representación del texto, pero también pueden crearse y utilizarse diccionarios donde a cada emoticón se le asocie su significado, especificando el caso de los que pueden expresar odio.

Los hashtags son otros rasgos que pueden utilizarse como característica a tomar en consideración en los modelos. Muchos utilizan este recurso para compartir mensajes y hacerlos virales, pero muchas veces están conformados por una concatenación de palabras. Por lo tanto, procesarlos y usarlos como palabras clave podría ser una buena estrategia.

2.3. Tareas de evaluación relacionadas con la detección de mensajes de odio

En los últimos años diferentes tareas de evaluación relacionadas con el HS, han sido propuestas en el marco de varios eventos científicos. Tal es el caso de *Authorship and Aggressiveness Analysis* (MEX-A3T), *Identification of Offensive Language* en GermEval[1], *Automatic Misogyny Identification* (AMI)[2], HaSpeeDe, HatEval y OffensEval. En la Tabla 2.1 se resumen algunos aspectos importantes de las tareas en las que se centra el presente trabajo y que se detallan en capítulos siguientes.

[1]GermEval 2019 (https://projects.fzai.h-da.de/iggsa/)

[2]Automatic Misogyny Identification – IBEREVAL 2018 (https://amiibereval2018.wordpress.com/)

Tabla 2.1: Algunas competiciones relacionadas con la detección de HS.

Tarea	**Evento**	**Año**	**Idioma**	**Descripción**
MEX-A3T	IberEval	2018	Español	El objetivo de la tarea es detectar comentarios agresivos en Twitter. Un desafío añadido es que los textos están escritos en español mexicano. [5]
HaSpeeDe	Evalita	2018	Italiano	La tarea pretende identificar si un mensaje tomado de Facebook o Twitter contiene odio o no. Tres variantes son presentadas. Dos de ellas proponen crear el modelo con el corpus obtenido de la misma red social con que es generado el conjunto de prueba; la tercera propone crear el modelo utilizando solo un corpus de una red social diferente a la utilizada para construir el conjunto de prueba. [26]
OffensEval	SemEval	2019	Inglés	La tarea se enfoca en la identificación de mensajes ofensivos. Para ello se divide en tres subtareas: 1) identificar si el lenguaje de un mensaje es ofensivo o no, 2) categorizar el tipo de la ofensa en caso de estar presente, y 3) determinar el objeto de la ofensa. El corpus proporcionado está constituido por un conjunto de tweets [66]. [67]
HatEval	SemEval	2019	Español e Inglés	La tarea se centra en la detección de HS contra inmigrantes y mujeres en Twitter. Para ello se proponen tres subtareas: 1) detectar si un mensaje contiene odio o no, 2) determinar si el odio es contra un individuo o contra un grupo, e 3) identificar si además es agresivo. [11]
MEX-A3T	IberLEF	2019	Español	En su segunda edición, la tarea tiene como objetivo mejorar la detección de agresividad en Twitter e impulsar el tratamiento computacional del español mexicano. [7]

Capítulo 3

Marco teórico

En este capítulo se presentan diferentes técnicas y modelos que se han utilizado como base en el desarrollo del presente trabajo. Primeramente se describen las formas de representación de texto empleadas, prestando especial atención a modelos basados en redes neuronales. Posteriormente se comentan diferentes modelos basados en aprendizaje profundo.

3.1. Representación de textos

En forma general, los métodos utilizados para el procesamiento de textos toman como entrada una determinada representación en lugar del texto en sí. De manera que un paso importante es la transformación de un texto a un determinado espacio de representación. Para ello se han utilizado diferentes estrategias que varían en función de las diferentes metodologías.

La representación vectorial es una forma frecuentemente utilizada. Donde los textos son representados por vectores que contienen la información de la ocurrencia o frecuencia de ocurrencia en el texto de cada una de las palabras pertenecientes a un determinado vocabulario. Por tanto, el primer paso es la definición del vocabulario V como el conjunto de palabras relevantes. Posteriormente, cada texto T se representa como un vector vT con la información correspondiente a las palabras de V que T contiene.

En una de las aproximaciones más sencillas vT es de dimensión $|V|$, que se corresponde con la talla del vocabulario. La componente i-ésima de vT ($vT_i = t$) constituye una correspondencia $F(t,T)$ con la que se representa la palabra o término i-ésimo del vocabulario. Para esto se han utilizan diferentes estrategias. La más sencilla es la relación binaria, donde cada componente $F(t,T) \in \{0,1\}$ indica la presencia con 1 del término t o su ausencia

con 0 en T. Otra forma se basa en la frecuencia de aparición en T de t que denotaremos por $tf(t,T)$, de forma que $F(t,T) = tf(t,T)$. Además, existen variantes en las que los valores son normalizados tales como las expresadas en las ecuaciones (3.1) y (3.2), donde $|T|$ denota la cantidad de palabras en el texto T y el denominador de la segunda determina la frecuencia máxima de aparición de un término en T.

$$F(t,T) = \frac{tf(t,T)}{|T|} \tag{3.1}$$

$$F(t,T) = \frac{tf(t,T)}{max\{tf(t',T)|t' \in T\}} \tag{3.2}$$

Otra técnica ampliamente utilizada se conoce como TF-IDF, que introduce un factor de frecuencia inversa del documento (IDF). Dicho factor es una medida de cuánta información proporciona el término, es decir, si es común o no en un conjunto de documentos D. Se calcula como el logaritmo de la fracción entre el número total de documentos y el número de documentos que contienen al término t, tal como indica la ecuación (3.3). De modo que cada componente del vector $F(t,T) = TF\text{-}IDF(t,T,D)$ se determina ahora por la ecuación (3.4), donde los términos que tienen una alta frecuencia en el texto T y baja en el resto de los documentos tendrá un valor $F(t,T)$ elevado.

$$IDF(t,D) = log\frac{|D|}{|\{d \in D|t \in d\}|} \tag{3.3}$$

$$TF - IDF(t,T,D) = tf(t,T) \cdot IDF(t,T,D) \tag{3.4}$$

Un problema de este tipo de representación es que no tiene en cuenta el orden en la secuencia de las palabras. En muchas tareas, esa información es importante para capturar características semánticas. Una solución es el uso de n-gramas, los que consisten en secuencias de tamaño fijo n de elementos extraídos del texto. De esa forma se modela la concatenación de dichos elementos, que pueden ser palabras o caracteres.

Embeddings de palabras

Las codificaciones (embeddings) de palabras (*words embeddings* (WE) en inglés) son representaciones vectoriales, que como indica el nombre se obtienen a nivel de palabras. Por lo que en este caso un texto se representa como una secuencia de vectores o una matriz de dimensión CxN, donde

N es la dimensión establecida para los vectores (embeddings) que representan las palabras; y C es la longitud definida para los textos, cumpliéndose posiblemente $C = |T|$, siendo T nuevamente el texto y $|T|$ la cantidad de palabras que contiene.

Existen diferentes técnicas para obtener representaciones vectoriales de palabras. La más sencilla e intuitiva es conocida como *one-hot*, donde las palabras se representan con vectores binarios de gran dimensión. Además del problema que esto significa en cuanto al costo espacial, con este tipo de representación no se logra capturar información contextual de las palabras. Por tal razón suelen utilizarse otras técnicas con las que se obtienen embeddings como vectores de dimensión más reducida y que capturan información semántica. Con estos vectores se trata de mantener la similitud contextual que existe entre diferentes palabras, de forma que las que aparecen frecuentemente en contextos semejantes, obtienen representaciones cercanas en el espacio vectorial.

Un modelo muy utilizado es Word2vec, propuesto en el artículo [47]. En este artículo se presentan dos arquitecturas para obtener representaciones vectoriales de valores continuos: *Continuous Bag of Words* (CBOW) y *Continuous Skip-gram*, mostradas en la Figuras 3.1 y 3.2 respectivamente. En CBOW se predice una palabra $w(t)$ a partir de su contexto, establecido según un determinado tamaño de ventana. Por otro lado, con la segunda arquitectura, el modelo utiliza la palabra $w(t)$ para predecir la representación de las palabras en su contexto.

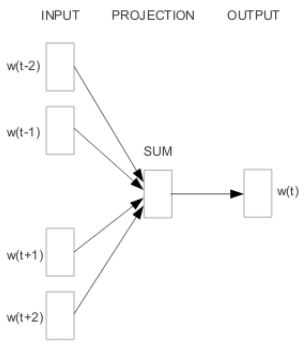

Figura 3.1: Modelo CBOW [47].

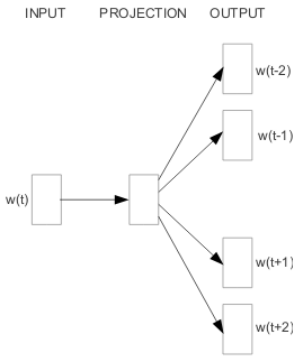

Figura 3.2: Modelo Skip-gram [47].

En [14] se propone FastText con otro enfoque basado en el modelo Skip-gram, en el que cada palabra se representa como un conjunto de n-gramas de caracteres. Cada n-grama de caracteres es asociado con un vector y la

representación de las palabras se obtiene por la suma de los vectores de sus n-gramas correspondientes. La ventaja es que permite obtener representaciones para las palabras que no aparecían en el entrenamiento.

Por otro lado, Glove [54] es también un modelo para obtener embeddings a nivel de palabras. A diferencia de Word2vec, que es un modelo predictivo, este es un modelo basado en conteo. Los vectores son generados a partir de la reducción de la dimensión en una matriz de conteos de co-ocurrencia. Cada fila de la matriz representa una palabra, mientras que cada columna representa un posible contexto, de forma que la matriz contiene la frecuencia de cada palabra en cada contexto. La cantidad de columnas de esta matriz es reducida mediante una factorización en la que se busca dimensiones inferiores que pueden explicar la mayor parte de la varianza en los datos de dimensiones altas.

Embeddings de oraciones

Como se ha comentado anteriormente, muchos esfuerzos se han realizado para obtener una representación vectorial de los textos, con lo que se facilita el cálculo de la distancia entre ellos en el espacio de representación y así, el diseño de modelos para el tratamiento de diferentes tares del PLN.

Si bien técnicas propuestas para obtener los embeddings de palabras constituyen pasos importantes en este sentido, la necesidad de capturar todo el contenido de un texto en el vector de representación del texto se hace necesario. Por tal razón, se han propuesto técnicas para obtener embeddings a nivel oración donde sea capturado todo su contexto [55].

Universal Sentence Encoder (USE) es un modelo propuesto por Google en el año 2018 para obtener embeddings de oraciones [20]. Dado un texto cualquiera, con este modelo se obtiene un vector de dimensión 512 como representación. Ha sido entrenado con una variedad de fuentes y de tareas, por lo que puede ser ajustado para modelar la semántica de una secuencia de palabras para un gran número de propósitos.

Este modelo tiene dos versiones disponibles en TF-Hub[1]. La primera versión se basa en el modelo Transformer [63] y la segunda versión se basa en un modelo *Deep Averaging Network* (DAN, siglas en inglés), que toma el promedio de los embeddings de las palabra como entrada a una red neuronal profunda (DNN) *Feed-forward*. A pesar de que la primera versión es más

[1]TF-HUB. Universal Sentence Encoder
(https://tfhub.dev/s?q=universal-sentence-encoder)

costosa computacionalmente, está diseñada para generar modelos con una mayor precisión.

Por otro lado, Transformer es un modelo constituido por un codificador (encoder) y un decodificador (decoder). El encoder, que es la parte del modelo en la que se basa USE, está conformada por una pila de 6 capas idénticas tal como muestra la Figura 3.3. Cada capa tiene un mecanismo de atención especial y una red neuronal Feed-Fordward, entre los que hay conexiones residuales seguidas de una normalización. Además, el modelo incorpora información sobre la posición de cada término en la secuencia (*positional encodings*) para controlar el orden ya que no hay recurrencia en el modelo. Para la variante del USE basada en Transformer se han publicado

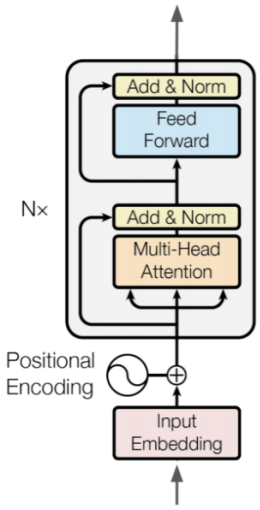

Figura 3.3: Codificador del modelo Transformer [63].

modelos pre-entrenados para el inglés y otros idiomas. Por lo que puede ser utilizado en el diseño de sistemas para idiomas como el español.

Bidirectional Encoder Representations from Transformers (BERT) también ha sido publicado por Google en el 2018 y se basa en la arquitectura del encoder de Transformer [29]. En general BERT es utilizado como un método para generar modelos de lenguaje, con el que no se limita el entrenamiento de dichos modelos al procesamiento de izquierda a derecha, a diferencia de otros métodos previos. Este es uno de sus rasgos clave, de forma que en lugar de predecir la siguiente palabra después de una secuencia de palabras, se enmascaran al azar algunas palabras en la oración para

predecirlas.

BERT ha sido diseñado para el entrenamiento no supervisado de representaciones bidireccionales profundas. Como fue comentado anteriormente, utiliza una estrategia de enmascaramiento aleatorio de algunos tokens del texto en la entrada al modelo, y el objetivo es predecir los tokens correctos basado solo en el contexto. También se utiliza otra estrategia basada en la tarea de predicción de la siguiente oración dada una primera oración, con lo que se pre-entrenan conjuntamente representaciones de pares de textos.

Con este método se obtienen modelos pre-entrenados de lenguajes que pueden ser ajustados a una gran variedad de tareas del PLN, tales como clasificación, reconocimiento de entidades y la tarea de preguntas y respuestas. Además, con los modelos es posible obtenerse embeddings de términos (tokens) y oraciones, por lo que BERT puede ser usado como una vía para generar representaciones de textos que contienen características significativas de los datos.

3.2. Modelos basados en aprendizaje profundo

La detección de HS suele tratarse como una tarea de aprendizaje supervisado. De forma que generalmente se diseña un sistema que aprende con un conjunto de textos etiquetados para identificar si un mensaje contiene odio o no. Como fue comentado en el capítulo anterior, diferentes algoritmos tradicionales han sido utilizados. Sin embargo, la potencialidad de los modelos basados en el aprendizaje profundo hacen que estos sean el punto de atención en muchas investigaciones. Para lograr una buena comprensión del presente trabajo, en este apartado se comentan algunas características importantes sobre los modelos que se han utilizado.

3.2.1. Redes neuronales recurrentes

Las redes recurrentes (RNN) son un tipo de red neuronal que permite el procesamiento de datos constituidos por secuencias de tamaño variable, a diferencia de redes más sencillas que toman como entrada un vector de tamaño fijo. Por lo que pueden ser convenientes en tareas de PLN, ya que el texto de entrada tiene una forma secuencial (secuencia de palabras).

De forma general, una RNN no solo tiene en cuenta un conjunto de pesos asociados a la entrada como otras redes neuronales tradicionales, sino que además contiene un estado oculto h para cada etapa de la secuencia que tiene en cuenta el contexto basado en el pasado, como muestra la Figura 3.4. Este estado captura la relación que los elementos de la secuencia pueden

tener entre sí y sigue cambiando en cada paso, de manera que cada entrada sufre una transición diferente. Por lo tanto, con una misma representación en la entrada en diferentes etapas de la secuencia se pueden obtener salidas diferentes en función de las etapas anteriores. Al final del procesamiento se obtiene una secuencia $H = h_1, h_2, ..., h_L$ de estados ocultos. Donde cada $h_i (i \in \{1, 2, .., L\})$ es el estado oculto que se obtiene de cada etapa, y L es la longitud de la secuencia.

Un aspecto importante es que al ser compartidos los parámetros de la red para cada elemento de la entrada, se elimina la restricción de tamaño fijo como fue comentado anteriormente, haciendo posible procesar secuencias de longitud variable. Además, este aspecto es fundamental para capturar la relación entre un elemento de la entrada y sus vecinos.

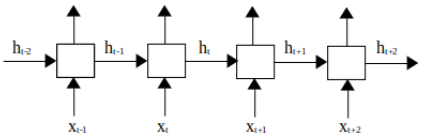

Figura 3.4: Arquitectura RNN.

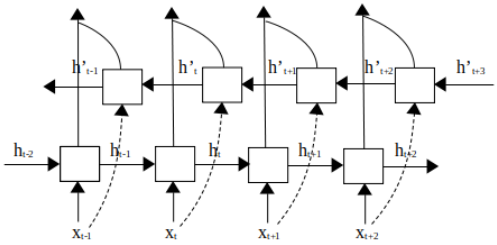

Figura 3.5: Arquitectura Bi-RNN.

Redes neuronales bidireccionales (Bi-RNN): Para obtener la salida en cada paso del procesamiento secuencial, esta variante de las RNN no solo se basa en el pasado, sino también en el futuro. En diferentes tareas esta estrategia es muy importante para el tratamiento de la ambigüedad, ya que permite tener en cuenta todo el contexto para cada elemento de la secuencia. En la Figura 3.5 se muestra una arquitectura con la que es posible comprender el funcionamiento general. En este caso la secuencia es analizada en los dos sentidos (de derecha a izquierda y de izquierda a derecha) y el estado oculto final en cada etapa t de la secuencia se obtiene por la unión de los dos estados correspondientes para cada uno de los dos procesamientos (h'_t hacia la derecha y h''_t hacia la izquierda). Frecuentemente la unión se realiza mediante una concatenación de los vectores, por lo que $h_t = [h'_t, h''_t]$.

GRU y LSTM

Si bien las RNN permiten el procesamiento de secuencias de tamaño variable, las primeras variantes de estas arquitecturas tienen algunos pro-

blemas en la práctica. Por un lado, toman en cuenta todo el pasado en cada etapa del procesamiento de la secuencia, pero en la realidad no siempre esto es deseable, a veces solo es necesario tener en cuenta el pasado reciente y en otros casos un pasado más lejano. De forma que es necesario un mecanismo que controle qué olvidar o qué información mantener en una etapa determinada del procesamiento. Además, otro problema importante está relacionado con el entrenamiento, donde el error que se propaga por la red para el ajuste de los pesos tiende a desvanecerse.

Como solución se han utilizado las unidades GRU (*Gated Recurrent Unit*) [22] y LSTM (*Long Short-Term Memory*) [40, 38], con la capacidad de mantener una memoria de las activaciones anteriores en una etapa dada de la secuencia. Esto permite recordar características durante el procesamiento y reducir la probabilidad del desvanecimiento del gradiente.

Las unidades LSTM contienen compuertas de entrada, de olvido y de salida. La compuerta de entrada regula la cantidad del estado oculto recién calculado para la entrada actual que desea dejar pasar, la de olvido controla cuánto de la memoria del estado anterior desea dejar pasar, y la de salida define cuánto del estado interno se usa para calcular la salida a las siguientes capas de la red. De forma que en el paso t el estado oculto h_t puede calcularse de la siguiente manera, lo que se muestra gráficamente en la Figura 3.6.

El vector u_t ($\tilde{\mathrm{C}}_t$ en la Figura 3.6) es un estado oculto auxiliar que depende de la entrada y el estado oculto anterior, mientras que C es la memoria interna (estado de la celda) de la unidad. Esta memoria tiene en cuenta la memoria anterior controlada por la compuerta de olvido, y el estado oculto u_t controlado por la compuerta de entrada. De esta forma se realiza una combinación de la memoria anterior y la entrada nueva, cada una de las cuales puede tener más o menos peso en dependencia de las compuertas. Si por ejemplo, la compuerta de olvido toma valor cero, la memoria anterior se ignora por completo. Por último, h_t es el estado oculto en la salida como fue comentado anteriormente, que depende de la memoria y la compuerta de salida [3]. A continuación se muestran las ecuaciones correspondientes:

$$
\begin{aligned}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) && \text{(compuerta de entrada)}\\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) && \text{(compuerta de olvido)}\\
o_t &= \sigma(W^{(o)}x_t + U^{(i)}h_{t-1} + b^{(o)}) && \text{(compuerta de salida)}\\
u_t &= tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)})\\
C_t &= i_t \otimes u_t + f_t \otimes C_{t-1}\\
h_t &= o_t \otimes tanh(C_t)
\end{aligned}
$$

Donde $W^{(*)}$, $U^{(*)}$ y $b^{(*)}$ representan respectivamente las conexiones a las entradas, las conexiones a los estados ocultos anteriores y los *bias*. Estos constituyen parámetros que deben aprenderse durante el entrenamiento. La función $\otimes$ representa la multiplicación entre elementos, mientras que σ es la función sigmoide, que al estar definida en el rango $[0,1]$ permite que las compuertas puedan definir la cantidad de información en un vector que deben dejar pasar.

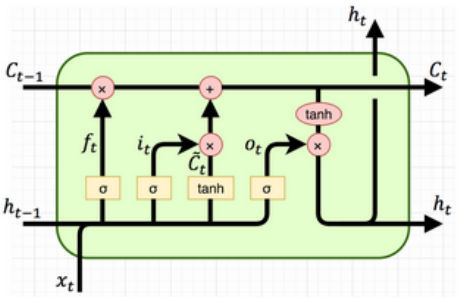

Figura 3.6: Arquitectura LSTM [3].

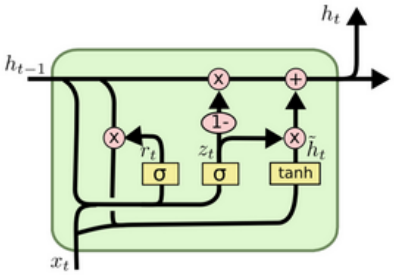

Figura 3.7: Arquitectura GRU [3].

Por otro lado, GRU es una unidad más sencilla con propiedades similares a las LSTM. Las compuertas de olvido y de entrada se combinan en una compuerta conocida como de actualización. Además, fusiona el estado de la celda (memoria) y el estado oculto. De forma que la arquitectura resultante es más simple que los modelos LSTM como se puede ver en la Figura 3.7. Tienen menos parámetros por lo que son más fácil de entrenar. Sin embargo el rendimiento es comparable al de las LSTM en el modelado de secuencias [3]. A continuación se muestran las ecuaciones correspondientes:

$$
\begin{aligned}
z_t &= \sigma(W^{(z)}x_t + U^{(z)}h_{t-1} + b^{(z)}) \\
r_t &= \sigma(W^{(r)}x_t + U^{(r)}h_{t-1} + b^{(r)}) \\
u_t &= tanh(W^{(h)}x_t + U^{(h)}(r_t \otimes h_{t-1}) + b^{(h)}) \\
h_t &= (1 - z_t) \otimes h_{t-1} + z_t \otimes u_t
\end{aligned}
$$

Donde r_t es una compuerta de reinicio y z_t la de actualización. En este caso, la compuerta de reinicio determina en qué medida combinar la memoria anterior con la nueva entrada, y la puerta de actualización define la cantidad de memoria anterior que debe mantenerse.

3.2.2. Modelos de atención

Diferentes tareas del PLN se caracterizan por tomar como entrada una secuencia de palabras y producir otra secuencia de elementos, que pueden ser también palabras como en el caso de la traducción automática. Para ello se han diseñado métodos que procesan la secuencia de la entrada en una primera parte conocida como el codificador (*encoder*). Donde se obtiene como resultado un vector de contexto c de tamaño fijo que constituye una representación interna del texto original, de forma que puede ser procesado en la segunda parte conocida como el decodificador (*decoder*) para obtener la secuencia de salida. Generalmente el *encoder* y el *decoder* están constituidos por RNN, y el vector c es obtenido como una función de los estados ocultos de cada uno de las etapas del *encoder*, esto es $c = q(h_1, h_2, ..., h_C)$. Donde C es la longitud de la secuencia y la función q suele tomarse como la asignación del último vector de la secuencia $c = h_C$. De esta forma, el contenido de todo el texto de la entrada se resume en un solo vector, como fue comentado anteriormente. Esto puede significar un problema para textos largos. Como una posible solución fueron propuestos los mecanismos de atención [10].

Con los mecanismos de atención, en lugar de utilizar toda la información del texto resumida en c, la idea es prestar atención solamente, o principalmente, a los elementos más significativos de la entrada. Para ello, en lugar de obtenerse un único vector c, se obtienen una secuencia de vectores de contexto c_i, $i = 1, ..., S$ con S la longitud de la secuencia de salida. Estos vectores son obtenidos como la suma pesada de los vectores obtenidos en el *encoder* ($\{h_1, h_2, ..., h_C\}$). Los pesos son calculados según la correspondencia entre los elementos en la entrada, de forma que la importancia de cada uno de dichos elementos se define según su peso correspondiente.

Existen diferentes estrategias para el cálculo de estos pesos. Una de ellas es mediante el uso de redes neuronales *Feed-forward*. En [44] se presentan diferentes tipos de atención. Por un lado, con la global se utilizan todos los vectores del *encoder* cuando se calcula cada vector del contexto. Mientras que con la local se localiza una posición en la secuencia de la entrada y solo se tienen en cuenta los elementos de contexto determinado por una ventana de un determinado tamaño.

3.2.3. Redes neuronales convolucionales

Las redes convolucionales (CNN) son un tipo de redes neuronales profundas que consisten en varias capas de filtros convolucionales de una o más dimensiones. La idea es identificar patrones simples dentro de los datos para

luego ser utilizados en la formación de patrones más complejos dentro de capas más altas. Tradicionalmente este tipo de arquitectura ha sido utilizada para el procesamiento de imágenes, donde los datos se representan con una estructura matricial y por tanto los filtros utilizados son 2D (3x3, 5x5, ...), definiendo el tamaño en ancho y largo. Sin embargo, en varias aproximaciones estas arquitecturas han sido extendidas para tareas de PLN.

En este caso, las CNN toman como entrada una secuencia 1D correspondiente a las representaciones de las palabras, dígase una secuencia de vectores. Por lo que en este caso los filtros son 1D (3, 5, ...), especificando la cantidad de elementos de la secuencia que se tiene en cuenta. Este tipo de red neuronal convolucional suele identificarse como 1D CNN.

Si bien 1D CNN puede ser adecuado para el análisis de las secuencias y por tanto puede aplicarse en tareas de PLN, un problema puede darse ya que en algunos casos con este tipo de patrones no puede ser capturada la relación entre palabras relacionadas pero lejanas en el texto.

Una técnica frecuentemente utilizada con las CNN es el max-pooling. El objetivo es reducir la dimensión de la representación en su entrada. La interpretación se relaciona con el análisis de características contenidas en las distintas regiones de dicha representación.

Redes neuronales convolucionales 1D

La 1D CNN puede verse como una red neuronal convolucional tradicional, donde una de las dimensiones es fijada al tamaño de la representación de los datos en la entrada como muestra la Figura 3.8. En el caso del PLN usualmente los datos son palabras. De esta forma, la entrada es una matriz

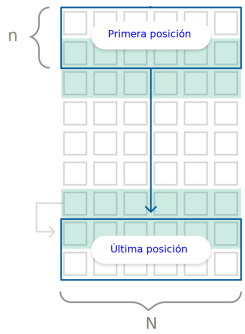

Figura 3.8: Arquitectura 1D CNN [2].

de los embeddings de las palabras por filas con dimensión CxN, y los filtros

son matrices de dimensión nxN. Donde N es la longitud de los embeddings, C es el tamaño de la secuencia correspondiente en teoría con la cantidad de elementos de la secuencia y n es la cantidad de términos de la entrada que analiza el filtro. El valor de n puede ser visto como el tamaño de los n-gramas que se toman en consideración en el modelo. Como resultado se obtiene un vector columna para cada filtro utilizado, cuya dimensión queda determinada por C, n y por el desplazamiento del filtro. La idea es utilizar esta arquitectura como extractor de características de n-gramas según los filtros utilizados, y que puedan ser filtrados los n-gramas menos relevantes con una operación de max-pooling.

Capítulo 4

Detección de mensajes de odio en SemEval-2019

SemEval es un taller internacional de evaluación semántica que se realiza cada año con el propósito de identificar problemas y soluciones en diferentes áreas de estudio, mediante la evaluación de sistemas relacionados con el análisis semántico computacional. Se entiende por análisis semántico computacional al análisis del significado del contenido textual mediante aproximaciones con implementaciones efectivas [12]. Lo cual supone una serie de dificultades debido a la naturaleza del lenguaje natural. SemEval surgió a partir del taller SensEval[1] que en sus inicios estaba enfocado a la tarea de desambiguación semántica de las palabras para diferentes idiomas, y que posteriormente incorporó otras tareas del PLN. Como continuación, en el año 2007 fue desarrollado SemEval abarcando un grupo más amplio de tareas e idiomas.

Áreas de estudio como el análisis de sentimiento fueron incorporados, motivando un importante grupo de investigaciones para obtener mecanismos con los que caracterizar textos según declaraciones subjetivas. Donde este tipo de declaraciones se corresponde con contenido textual que refleja los sentimientos de los autores o sus percepciones sobre entidades y eventos. Posteriormente, en el año 2013 se incluyó el análisis de textos publicados en Twitter, haciendo notar los nuevos retos del procesamiento de tweets.

El taller ha continuado evolucionando y ampliando las dimensiones del análisis semántico para dar tratamiento a importantes problemáticas como la que en cuestión se estudia en el presente trabajo. De tal forma que en este año, dos tareas relacionadas con la detección de mensajes abusivos han

[1]SensEval (http://web.eecs.umich.edu/~mihalcea/senseval/)

sido propuestas y desarrolladas en el marco de SemEval 2019[2]. HatEval por un lado, se ha enfocado particularmente en la detección de HS; mientras que OffensEval se ha dirigido al análisis de mensajes para detectar contenido ofensivo en general, que como se comentó en Capítulo 2 constituye un concepto más general que incluye a HS.

En este capítulo se describen ambas tareas, comentando las aproximaciones presentadas por los equipos participantes. Además, se detallan nuestros modelos propuestos y los experimentos realizados. Se analizan los resultados obtenidos y se comparan con los resultados generales. Por último se resumen las principales conclusiones adquiridas con la experiencia de la participación en las tareas, tomando como referencia los resultados generales alcanzados y los experimentos realizados con nuestros modelos.

4.1. HatEval: Detección multilingüe del discurso de odio contra inmigrantes y mujeres en Twitter

HatEval (*Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter*, según su nombre en inglés) fue una tarea propuesta en el marco de SemEval 2019 como se comentaba anteriormente. El objetivo de esta tarea es detectar contenidos de odio en textos publicados en Twitter, concentrando el análisis en el odio expresado contra dos grupos de personas: inmigrantes y mujeres. Otra característica de la tarea es su perspectiva multilingüe, de forma que se proporcionaron datos en inglés y español.

La tarea se divide en dos subtareas principales. La primera (tarea A) se centra en la detección de HS, mientras que el objetivo de la segunda (tarea B) es investigar aspectos que pueden caracterizar a los mensajes de odio para comprender cómo abordar la tarea. Para ello, esta segunda tarea se encaminó a identificar si el odio se dirige hacia un individuo o hacia un grupo de personas; además, determinar si es un mensaje agresivo o no.

La tarea A consiste básicamente en una clasificación binaria tal como se definió la detección de HS en capítulos anteriores. Al ser este el principal objetivo del presente trabajo, los experimentos comentados se corresponden solo con esta tarea, aunque en el siguiente apartado se describe el modelo utilizado para dar tratamiento a la otra subtarea relacionada con la detección de agresividad y se presentan los resultados.

[2]SemEval 2019 (http://alt.qcri.org/semeval2019/)

La tarea B se concentra en los tweets clasificados como HS, de los cuales se deben predecir dos aspectos:

- Objetivo del odio: Clasificación binaria para determinar si el objetivo del odio es una persona o si es un grupo de personas.
- Agresividad: Clasificación binaria para detectar si un mensaje además de contener odio es agresivo o si no lo es.

De esta forma, HatEval se puede resumir en tres subtareas de clasificación binaria donde las etiquetas para cada una de ellas son:

- Tarea A: hateful – non-hateful
- Tarea B.1: individual – generic
- Tarea B.2: aggressive – non-aggressive

Aproximaciones utilizadas por los participantes

Diferentes aproximaciones fueron utilizadas por los equipos participantes [11]. Una característica común en los sistemas propuestos para cada uno de los idiomas es que la mayoría se basan en el aprendizaje profundo utilizando diferentes modelos y estrategias de representación de textos. Sin embargo, en ambos casos también se propusieron métodos basados en modelos tradicionales de aprendizaje automático, alcanzándose los mejores resultados en la tarea A con SVMs. Además, otros métodos tradicionales fueron empleados tales como Regresión Logística (LR, por sus siglas en inglés) y el clasificador Naïve Bayes. Con ellos se utilizaron diferentes características como TF-IDF y n-gramas.

Para el corpus en inglés particularmente, el mejor sistema según los resultados reportados utiliza una SVM con las características obtenidas a partir de los embeddings de oraciones USE. El modelo utilizado en el sistema que obtuvo el segundo lugar no fue reportado. El tercer, cuarto y quinto lugar utilizaron modelos basados en redes neuronales. Específicamente emplearon las CNN y las RNN, arquitecturas también utilizadas por muchos otros sistemas. Varios tipos de embeddings fueron utilizados en las aproximaciones de aprendizaje profundo. Entre ellos FastText es empleado por el sistema en la tercera posición, mientras que el de la cuarta posición utiliza embeddings a nivel de oración siguiendo un modelo similar al de BERT.

Por otro lado, para el caso del corpus en español los dos mejores sistemas utilizan SVM con características basadas en bolsas de palabras y embeddings

obtenidos con FastText. Como fue comentado anteriormente, otros equipos propusieron modelos tradicionales y modelos basados en aprendizaje profundo, de los cuales cabe destacar la propuesta de un modelo generado a partir de un modelo pre-entrenado de BERT ajustado con un conjunto de tweets recopilados durante el mismo período de tiempo del conjunto de datos de entrenamiento de HatEval que alcanzó el tercer lugar.

El preprocesamiento de los textos, en general ha estado dirigido según las características específicas de los tweets. De forma que muchas aproximaciones realizan la eliminación de signos de puntuación, la normalización de palabras con errores como por ejemplo de las contracción, así como la sustitución de urls, números y menciones por determinadas etiquetas definidas.

4.1.1. Modelo propuesto

El modelo propuesto para la tarea A de HatEval [24] puede estructurarse en dos partes como se muestra en la Figura 4.1. La primera parte se ha diseñado con el objetivo de obtener características a partir de los datos. Estas características constituyen a su vez la entrada a un clasificador, que se corresponde con la segunda parte del modelo donde se predice la presencia o no de odio en el texto de entrada.

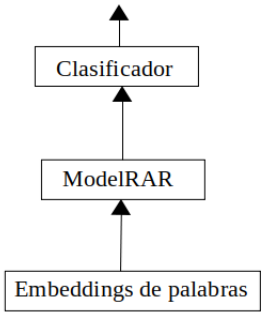

Figura 4.1: Modelo propuesto para la tarea A de HatEval.

Representación de los tweets

El método propuesto no realiza una etapa previa de preprocesamiento de los textos, debido a que se partió de la hipótesis de que los modelos basados en aprendizaje profundo podrían capturar importantes características a alto nivel. De esta forma, la representación para la entrada del modelo fue generada a partir de los tweets originales.

La representación de los textos se obtuvo a partir de embeddings de palabras. Este es el único punto donde difiere el modelo empleado para cada uno de los idiomas analizados en la tarea. Para los tweets en inglés se utilizaron vectores pre-entrenados con Glove a partir de dos billones de tweets. Mientras que para los tweets en español fueron utilizados vectores pre-entrenados de FastText.

Arquitectura para la extracción de características

Como se comentó anteriormente, el modelo propuesto está constituido por dos partes. La primera de ellas se ha definido como un extractor de características a alto nivel. Está conformada por dos arquitecturas RNN y una capa de atención como se muestra en la Figura 4.2. Debido a su estructura le hemos nombrado ModelRAR (Recurrencia-Atención-Recurrencia), término que se utilizará en todo el presente documento para referirnos a esta arquitectura.

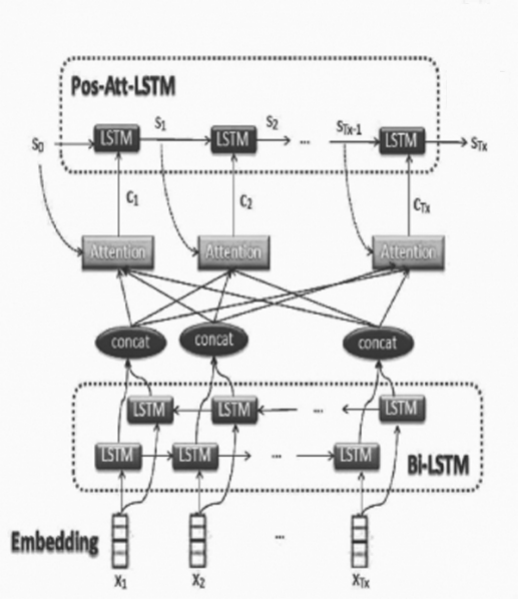

Figura 4.2: Arquitectura propuesta ModelRAR para la detección de HS [26].

La primera capa de la arquitectura consiste en una Bi-LSTM, que recibe la secuencia de vectores correspondientes a los embeddings de las secuencia de palabras en el texto y obtiene una secuencia H de estados ocultos. Como se comentó en el capítulo anterior, con una Bi-LSTM la secuencia de la

entrada es analizada en dos sentidos, de forma que la secuencia de estados ocultos que se obtiene es resultado de la unión de dos secuencias. En este caso de la concatenación de los estados correspondientes a cada elemento de la secuencia.

Posteriormente una capa de atención es añadida sobre la secuencia de estados H con el objetivo de determinar la contribución de cada elemento de la secuencia en la predicción. Para dicha capa de atención se tiene en cuenta además de la secuencia de vectores H el estado oculto de otra LSTM (Pos-Att-LSTM) agregada sobre esta capa. De forma que como se muestra en la Figura 4.3, la atención en la etapa t se determina sobre vectores que se obtienen como la concatenación de cada vector en H con el vector del estado oculto de la segunda LSTM en la etapa $t-1$.

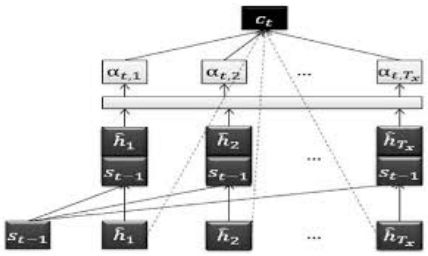

Figura 4.3: Capa de atención en ModelRAR [26].

Por último, como salida de ModelRAR se toma el vector correspondiente al último estado oculto de la segunda LSTM. Dicho vector se considera como la representación de las características a alto nivel extraídas de la entrada.

Clasificación para la tarea A

Para la detección de HS se utiliza finalmente un clasificador binario que toma como entrada el vector de características obtenido del paso anterior y se obtiene como resultado la etiqueta correspondiente a la predicción de HS o no. El clasificador consiste básicamente en una red neuronal *Fully-Connected*, donde la cantidad de capas de neuronas se ajusta a partir de experimentos que se describen más adelante. La última capa está conformada por dos neuronas con la función Softmax con lo que se determina la predicción final.

Clasificación para la tarea B

La tarea B está conformada por dos subtareas como fue comentado anteriormente. Por un lado, la predicción de objeto de odio en un mensaje detectado como HS, en cuanto a si era un individuo o un grupo de personas. Esta clasificación va más allá del objetivo del presente trabajo, sin embargo la propuesta realizada incluye el análisis de esta subtarea incorporando información correspondiente al etiquetado semántico de los textos. Una mejor explicación puede encontrarse en el artículo correspondiente [24].

Por otro lado, la segunda subtarea consiste en determinar si los textos detectados como HS, además contienen agresividad[3]. Para dar tratamiento a esta subtarea se construyó una lista de palabras y frases agresivas para cada uno de los idiomas involucrados en HatEval. Una característica es añadida al modelo anteriormente comentado como muestra la Figura 4.4. De forma que al vector que se obtiene como salida de ModelRAR se le concatena dicha característica antes de la etapa de clasificación.

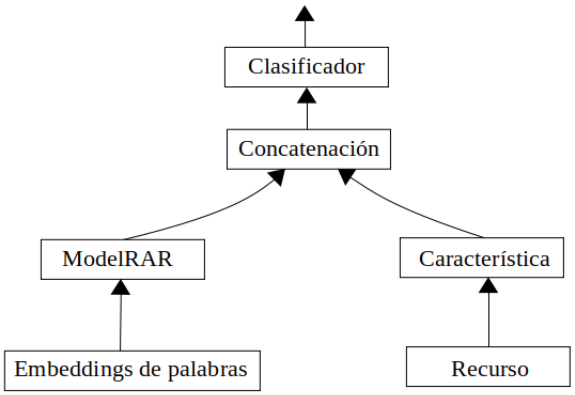

Figura 4.4: Modelo para la de detección de agresividad en HatEval.

Básicamente, la característica se corresponde con el número de ocurrencias en el texto de términos agresivos según el recurso construido. El vector resultante es la entrada al clasificador conformado por una red neuronal

[3]Estos son ejemplos tomados del corpus proporcionado:
Agresivo: @MirandaLanda22 Ya callate pinche puta y celebra la navidad hija de perra.
No agresivo: El tiempo no existe. La división espacio-temporal es una invención del hombre, por lo tanto,.. - Cállate, hijo(a) de puta.
Como puede verse, ambos tipos de textos pueden contener frases similares y por tanto, poder identificar la presencia o no de agresividad de forma automática es un reto.

Fully-Connected de una sola capa de neuronas y la clasificación se realiza con una última capa de dos neuronas con la función Softmax.

4.1.2. Experimentación

En este apartado se describen los experimentos realizados para HatEval. En primer lugar se presentan los datos propuestos para el desarrollo de la tarea. Luego se detallan los experimentos realizados y se realiza un análisis de los resultados obtenidos.

Corpus HatEval

Para el desarrollo de la tarea fue proporcionado un corpus para cada uno de los idiomas inglés y español. El corpus en inglés contiene 13000 tweet, mientras que el corpus en español contiene 6600. De forma que se recolectaron un total de 19600 tweets, de los cuales 9091 se corresponden al objetivo "inmigrantes" y los restantes 10509 a "mujeres". Las estadísticas generales de cada uno de los corpus son mostradas en la Tabla 4.1.

Tabla 4.1: Conjuntos de datos proporcionados en HatEval.

Corpus	**Entrenamiento**	**Prueba**
Inglés	10000	3000
Español	5000	1600

Como se puede ver, el corpus para el inglés es mucho más grande. Por lo que en este caso los sistemas tienen una mayor cantidad de datos para entrenar los modelos. Una característica importante a resaltar es que en cada caso, el número de textos con HS es inferior al de los textos que no contienen odio.

Medidas de evaluación

En la evaluación de la tarea A se tuvieron en cuenta las medidas que frecuentemente suelen utilizarse en tareas de clasificación: *Accuracy*, *Precision*, *Recall* y *Macro-averaged F1-score* (F1_score en ecuación (4.1)) [11]. De ellas, F1_score es la empleada para el ranking de los sistemas y la que se utiliza en este trabajo para los experimentos.

$$F1\_score : \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4.1}$$

Para la tarea B se utilizan dos medidas de evaluación: macro F1 y EMR. Con la primera se evalúa cada una de las tres subtareas propuestas de forma independiente y se reporta un promedio de los F1_scores. Por otro lado, la segunda medida utiliza conjuntamente las tres etiquetas predichas (la de la tarea A y las dos de la B) para medir el comportamiento de los sistemas. En nuestro caso solo tomamos el F1_score de la subtarea de agresividad por ser el punto de interés

Métodos de base

Como punto de referencia para la evaluación de los sistemas se utilizan dos métodos:

- MFC: Es una estrategia muy sencilla con la que se asigna la etiqueta más frecuente en el conjunto de entrenamiento a cada tweet en el conjunto de prueba.
- SVC: Utiliza una SVM con el núcleo lineal y la representación TF-IDF para los tweets.

Experimentos realizados

En la experimentación se analizó el modelo propuesto con los conjuntos de datos brindados para el entrenamiento con validación cruzada estratificada de 5 particiones. Los experimentos fueron diseñados apuntando a cuatro aspectos fundamentales.

El **primero** está enfocado en el estudio del impacto de la primera Bi-LSTM en ModelRAR. Como se ha comentado, dicha capa recibe la secuencia de vectores correspondiente a la secuencia de palabras del texto y genera una secuencia de vectores ocultos. Si bien, esta podría ser una buena aproximación con la que se capturan características, podría estar incorporando una gran cantidad de parámetros. Por lo tanto, podría afectar en la etapa de ajuste de parámetros. En función de este punto, se ha eliminado la primera capa Bi-LSTM, de forma que la entrada a la capa de atención, que ahora es la primera del modelo, en lugar de ser la secuencia de vectores correspondientes a los estados ocultos de la Bi-LSTM, es la secuencia de vectores correspondiente a los embeddings de las palabras del texto. En el análisis de los resultados más adelante se hace referencia a este modelo como Modelo_No-Bi-LSTM.

El **segundo** aspecto está dirigido a comprobar la importancia de la capa de atención en ModelRAR. Para ello todo el modelo se sustituye por

únicamente una capa Bi-LSTM. De esta forma, la entrada a la parte del clasificador del modelo en lugar de ser el vector correspondiente al último estado oculto del segundo LSTM, es el correspondiente al último estado oculto de la única Bi-LSTM en este caso. A este modelo se hace referencia como Modelo_No-Att.

El **tercer** aspecto tiene como propósito encontrar la cantidad de capas de neuronas en el clasificador del modelo. Para ello se realiza una serie de experimentos variando dicho parámetro y manteniendo la estructura ModelRAR. Básicamente las pruebas se han realizado con 0, 1, 2 y 3 capas de neuronas. Donde 0 se corresponde con la estructura que toma el vector de salida de ModelRAR y aplica una Softmax para obtener la predicción. En este caso se obtienen cuatro modelos según el número de capas añadidas en la parte del clasificador. La referencia a cada modelo es Modelo_Clf-i, donde i indica la cantidad de capas. El caso de $i = 2$ coincide con la cantidad de capas en el modelo propuesto.

Por último, el **cuarto** aspecto se relaciona con la detección de agresividad en mensajes clasificados como HS. En esencia, se ha estudiado el impacto de la característica correspondiente al recurso construido con las palabras y frases agresivas para cada uno de los idiomas. Para ello se ha evaluado el modelo sin tener en cuenta este recurso para comparar los resultados y determinar su relevancia. En este caso el análisis de los resultados se relaciona con la tarea B, y la referencia del modelo es Modelo_No-Recurso.

En cada una de las variantes los hiperparámetros para el entrenamiento coinciden según una serie de ajustes realizados previamente. Fijando el factor de aprendizaje (*learning rate*) a 0,001, el tamaño del *batch* en 100 y el número de épocas (*epoch*) en 75. En cuanto al optimizador se ha utilizado *Adam* con la función de pérdida de entropía cruzada, que se define como indica la ecuación (4.2). Donde y_i es la verdadera clasificación para el i-ésimo tweet y $\hat{y}_i$ la predicción realizada por el sistema.

$$L = -\sum_i y_i \cdot log\, \hat{y}_i \tag{4.2}$$

4.1.3. Análisis de los resultados

Las Tablas 4.2 y 4.3 muestran los resultados obtenidos para el corpus en español e inglés respectivamente, con la validación cruzada para cada uno de los experimentos anteriormente presentados. La métrica utilizada en el estudio es la macro F1, coincidiendo con la utilizada en HatEval. En cada caso se muestra la media de los resultados obtenidos con la validación

cruzada. Además, se muestra la desviación estándar (STD, por sus siglas en inglés) como medida de dispersión de los resultados, que puede dar una idea más clara en el análisis del rendimiento de cada variante.

Como puede observarse en los resultados experimentales para el caso del corpus en español, el rendimiento de todas las variantes es muy similar teniendo en cuenta la STD. Analizando el primer aspecto presentado en el apartado anterior, puede verse que la media de los resultados obtenidos para el modelo original es superior a la alcanzada con la variante Modelo_No-Bi-LSTM. Sin embargo, la STD muestra que los resultados son similares, indicando que la capa Bi-LSTM del modelo no implica una diferencia significativa en los resultados, aunque si puede representar una cierta mejoría. De forma similar, para el segundo experimento se puede observar que el resultado obtenido por el modelo original es también relativamente superior al obtenido con Modelo_No-Att, donde se ha eliminado la capa de atención. Esto indica que el mecanismo de atención no representa un aporte importante en el modelo para el caso del corpus en español.

Por otra parte, se puede observar que la cantidad de capas en la red neural *Fully-Connected* para la clasificación puede representar diferencias en los resultados obtenidos. Una vez más, con el modelo propuesto, que tiene dos capas de neuronas, se alcanza el mejor resultado en cuanto a la media del F1_score. Sin embargo, analizando la STD puede observarse que son resultados muy similares a los obtenidos con el modelo Modelo_Clf-1, que contiene solo una capa de neuronas. Por otro lado, resultados inferiores se obtienen con el modelo que no tiene ninguna capa y con el que tiene 3, siendo este último el de resultados más bajo. Lo que puede deberse a la gran cantidad de parámetros que supone aumentar este hiperparámetro, y que por tanto muestra que no es una buena alternativa incrementar la cantidad de capas en el clasificador.

Tabla 4.2: Resultados experimentales para español en HatEval.

Modelo	F1_score	Modelo	F1_score
Modelo propuesto	0.78 ± 0.017	Modelo_Clf-0	0.76 ± 0.009
Modelo_No-Bi-LSTM	0.76 ± 0.012	Modelo_Clf-1	0.77 ± 0.022
Modelo_No-Att	0.77 ± 0.015	Modelo_Clf-3	0.75 ± 0.016

Comportamientos similares a los observados en el corpus en español se han visto también en el corpus en inglés. Donde cabe señalar las diferencias obtenidas en cuanto al uso del mecanismo de atención. En este caso,

Modelo_No-Att obtuvo resultados notablemente inferiores a los obtenidos con el modelo original, indicando la importancia de la capa de atención en este caso, en el que cabe recordar que se más datos de entrenamiento fueron disponibles. Eso sugiere que la estructura de los textos en la entrada del modelo puede significar un factor importante de análisis a la hora de diseñar la estrategia de procesamiento, siendo la del mecanismo de atención en este caso muy importante para la estructura de los textos en inglés.

Por otra parte, cabe señalar que los resultados obtenidos al utilizar 3 capas de neuronas en la parte del modelo para la clasificación son también significativamente bajos respecto a los otros modelos. Lo que reafirma el análisis realizado con el corpus en español, donde se veía que no es una buena idea incrementar el número de capas de neuronas en el clasificador.

Tabla 4.3: Resultados experimentales para inglés en HatEval.

Modelo	F1_score	Modelo	F1_score
Modelo propuesto	0.78 ± 0.006	Modelo_Clf-0	0.76 ± 0.037
Modelo_No-Bi-LSTM	0.76 ± 0.029	Modelo_Clf-1	0.77 ± 0.032
Modelo_No-Att	0.68 ± 0.056	Modelo_Clf-3	0.64 ± 0.247

En la Tabla 4.4 se muestran los resultados obtenidos en la subtarea de agresividad de la tarea B para español e inglés. En el caso del español, los resultados sugieren que el rendimiento de los modelos puede mejoran al tener en cuenta una lista de palabras agresivas como característica. Sin embargo, en el caso del corpus en inglés los resultados son diferentes. De forma que los resultados obtenidos con el modelo que utiliza el recurso es significativamente inferior. Esto puede deberse a la calidad del recurso, el cual puede contener términos que introducen ruido en el modelo. Por lo que utilizar este tipo de estrategia puede ser una buena idea siempre y cuando el recurso utilizado sea adecuado para los datos en análisis.

Tabla 4.4: Resultados experimentales en la tarea de agresividad de HatEval.

Modelo	Español (F1_score)	Inglés (F1_score)
Modelo propuesto	0.78 ± 0.011	0.37 ± 0.029
Modelo_No-Recurso	0.77 ± 0.014	0.53 ± 0.021

4.1.4. Resultados con el conjunto de prueba

La Tabla 4.5 muestra los resultados obtenidos en la tarea A de HatEval. En forma general puede observarse que con el corpus en español se lograron mejores resultados respecto a los obtenidos con el corpus en inglés. Esto puede deberse a que es mayor la diferencia entre los datos en el conjunto de prueba y el conjunto de entrenamiento en inglés, que la diferencia en el corpus en español.

En español, el modelo propuesto alcanzó resultados bastante buenos respecto a la media de los resultados alcanzados por los sistemas propuestos. Además, a pesar de no acercarse al resultado del mejor sistema, si se logró superar a los dos métodos de base propuestos en la tarea. Por otro lado, los resultados alcanzados en inglés por el modelo propuesto no fueron buenos, quedando por debajo de la media y logrando superar solo a uno de los dos métodos de base. Lo que muestra que para los tweets en inglés el modelo no logra una buena generalización, debido quizás a ruido introducido en el conjunto de entrenamiento.

Tabla 4.5: Resultados en la tarea A de HatEval [11].

	Español (F1_score)	Inglés (F1_score)
Mínimo	0.4930	0.3500
Máxima (SVM)	0.7300	0.6510
Modelo propuesto	0.7180	0.3900
Media	0.6821	0.4484
Métodos de base		
SVC	0.7010	0.4510
MFC	0.3700	0.3670

En la Tabla 4.6 se muestran los resultados para el caso particular de la subtarea de agresividad en la tarea B. De igual forma a lo antes expuesto, en español se logran mejores resultados en general, y el modelo propuesto logra superar considerablemente a la media de los sistemas y a los dos métodos de base. Mientras que en inglés los resultados del modelo son inferiores y solo logra superar a uno de los métodos de base. En este caso, el comportamiento puede deberse al uso del recurso con los términos de agresividad, ya que como sugieren los resultados experimentales no es adecuado en el caso del inglés. Quizás, eliminando del modelo el uso de este recurso para el inglés,

los resultados obtenidos en la tarea hubiesen estado por encima de la media como en el caso del español. Eso puede ser evaluado en un trabajo futuro para un estudio más detallado.

Tabla 4.6: Resultados en la subtarea de agresividad de HatEval.

	Español (F1_score)	Inglés (F1_score)
Mínimo	0.5740	0.4190
Máximo (SVM)	0.7720	0.6320
Modelo propuesto	0.7400	0.5320
Media	0.7030	0.5478
Métodos de base		
SVC	0.7360	0.5780
MFC	0.4020	0.4210

4.2. OffensEval: Identificando y categorizando lenguaje ofensivo en las redes sociales

OffensEval (*Identifying and Categorizing Offensive Languagein Social Media*, según su nombre en inglés) fue otra tarea propuesta en el marco de SemEval 2019. Su objetivo es identificar mensajes de texto ofensivos publicados en Twitter. El idioma en el que se enfoca la tarea es el inglés, proponiendo para el análisis un corpus nuevo que más adelante en este capítulo se comenta con más detalles. La tarea se divide en tres subtareas, de forma que su propósito no se limita a la identificación de mensajes ofensivos, sino también a otras características de los mensajes como indica el objetivo específico de cada subtarea. Dichos objetivos se listan a continuación:

- Subtarea A: Identificación de lenguaje ofensivo.
- Subtarea B: Categorización de los tipos de ofensa.
- Subtarea C: Identificación del objetivo de insultos o amenazas.

Las subtareas B y C constituyen caracterizaciones de los mensajes identificados como ofensivos en la subtarea A. De forma que permiten analizar este tipo de mensaje a un nivel más profundo. En específico la subtarea B se dirige a determinar si el mensaje contiene o no insultos o amenazas. Por su

parte, la subtarea C se basa en los mensajes que además de ser identificados como ofensivos, son caracterizados como insultos o amenazas en B. Su propósito particular es determinar si el objetivo de la ofensa es un individuo, un grupo u otro tipo donde entran eventos, organizaciones, entre otros.

Como se comentó anteriormente en el Capítulo 2, el mensaje ofensivo es un concepto más general que incluye el mensaje de odio como una de sus variantes. De manera que realizar un análisis de las subtareas que caracterizan a los mensajes ofensivos en general va más allá de los objetivos del presente trabajo. Por lo que solo se tiene en cuenta la subtarea A, que está enfocada en identificar si un mensaje es ofensivo o no. En este sentido, los experimentos y análisis de resultados mostrados más adelante se corresponden solo con esta primera subtarea.

Aproximaciones utilizadas por los participantes

Diferentes aproximaciones fueron propuestas para esta tarea. Desde modelos basados en métodos tradicionales del aprendizaje supervisado como SVM y LR, hasta modelos basados en aprendizaje profundo, representados estos últimos por la mayoría de los sistemas [67].

Algunos sistemas utilizaron corpus adicionales de HS, aumentando el número de instancias para el entrenamiento de los modelos. En cuanto a la representación de los textos, diferentes tipos de embeddings fueron empleados. Por un lado, varios sistemas utilizaron embeddings a nivel de palabras como FastText y Glove. Por otro lado, otros sistemas utilizaron BERT, obteniendo en su mayoría los mejores resultados en esta tarea, a diferencia de HatEval donde los que utilizaron BERT no alcanzaron los mejores resultados.

En el preprocesamiento de los textos varios sistemas utilizaron técnicas en función de las características de los tweets. En este sentido se normalizaron tokens tales como hashtags, menciones, URLs y fechas. Además, algunos sistemas eliminaron palabras poco frecuentes y sustituyeron los emoticones por palabras o frases correspondientes.

4.2.1. Modelo propuesto

Para la subtarea A de OffensEval fueron propuestos 3 modelos [25]. Los dos primeros son modelos muy similares entre sí, diferenciándose solo en la arquitectura en las que están basados. El tercero es más complejo, incluyendo a los dos primeros como muestra la Figura 4.5. En cualquier caso, la aproximación se basa en modelos menos complejos que el utilizado en

HatEval.

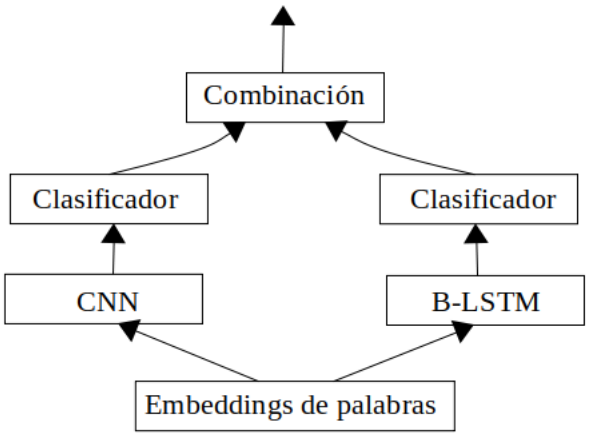

Figura 4.5: Modelo para la identificación de ofensas en OffensEval.

Cabe puntualizar, a pesar de que no se analiza el comportamiento del sistema para las subtareas B y C, que para estos casos se propuso una arquitectura similar que combina la información del etiquetado semántico como se explica en [25].

Representación de los tweets

La hipótesis con la que se partió en OffensEval para el preprocesamiento de los textos coincide con la de HatEval. De manera que los tweets no fueron preprocesados, sino que la representación se generó a partir de la secuencia original de palabras y términos.

Para la representación se utilizó un conjunto de vectores pre-entrenados con Glove a partir de dos billones de tweets, de igual forma que en el caso del inglés en HatEval.

Primer modelo propuesto para la subtarea A en OffensEval

El primer modelo propuesto para la identificación de mensajes ofensivos está conformado por una Bi-LSTM en la parte de la extracción de características y una red neuronal *Fully-Connected* como clasificador, que recibe como entrada el vector correspondiente al último estado oculto de la Bi-LSTM. Este último es un clasificador binario para lo que se ha empleado una capa de dos neuronas con función de activación Softmax.

Segundo modelo propuesto para la subtarea A en OffensEval

El segundo modelo propuesto es muy similar al primero. La diferencia radica en la arquitectura utilizada en el modelo. En este caso la arquitectura, en lugar de ser una Bi-LSTM es una 1D CNN.

Se utilizan 3 tipos diferentes de filtros para la convolución según su tamaño de 2, 3 y 4, lo que representa el análisis con bigramas, trigramas y 4-gramas respectivamente. En cada caso, la cantidad de filtros se ha fijado en 10. En la salida de la capa convolucional se realiza un max-pooling con tamaño y desplazamiento 2, reduciendo el tamaño de la salida a la mitad.

La salida de esta parte de convolución es básicamente un vector que constituye la representación de las características extraídas. Dicho vector es la entrada al clasificador igual que el caso del primer modelo.

Modelo general propuesto para la subtarea A en OffensEval

El modelo general básicamente es una combinación de los dos modelos presentados. De forma que la salida de la Softmax en ambos casos es promediada para obtener finalmente la clasificación binaria final, correspondiente a la presencia o no de ofensa en el texto de entrada.

4.2.2. Experimentación

En este apartado se describen los experimentos realizados para OffensEval. En primer lugar se describen los datos propuestos para el desarrollo de la tarea y se detallan los experimentos realizados. Luego se realiza un análisis de los resultados obtenidos.

Corpus OffensEval y medida de evaluación

Para OffensEval fue construido el corpus OLID (*Offensive Language Identification Dataset*, según su nombre en inglés), siguiendo un proceso de anotación descrito en [66]. Este corpus contiene 14100 tweets en inglés distribuidos según se indica en la Tabla 4.7.

Tabla 4.7: Conjunto de datos proporcionados en OffensEval.

Subtarea A	**Entrenamiento**	**Prueba**
Ofensivo	4400	240
No ofensivo	8840	620

Debido al claro desbalance en el corpus, donde la cantidad de tweets ofensivos es menor que la mitad de la cantidad de tweets no ofensivos, la medida utilizada para evaluar el comportamiento de los sistemas fue *Macro-averaged F1-score* (F1_score) como en la tarea analizada anteriormente.

Métodos de base

Dos sencillos métodos fueron utilizados como base para la evaluación de los sistemas:

- ALL OFF: Es una estrategia básica con la que se asigna la etiqueta correspondiente a la presencia de ofensa a cada tweet en el conjunto de prueba.
- ALL NOT: Es una estrategia muy similar a la anterior. La diferencia radica en que la etiqueta asignada se corresponde con la clasificación de mensaje no ofensivo.

Experimentos realizados

La experimentación se realizó de forma muy similar a como se hizo en HatEval. Donde se utilizó el conjunto de entrenamiento con validación cruzada estratificada de 5 particiones para medir el rendimiento de los sistemas.

En cada uno de los experimentos se utilizaron los mismos hiperparámetros, muy similares de igual forma a los utilizados en HateEval, ya que las evaluaciones fueron realizadas de forma paralela. De esta forma, el *learning rate* se ha fijado en 0.001 y como optimizador se ha utilizado Adam con la función de pérdida de entropía cruzada. La diferencia está en el tamaño del batch y el número de *epoch*, que en este caso son 32 y 100 respectivamente, lo que fue ajustado con una serie de experimentos previos.

Las direcciones en las que fueron realizados los experimentos pueden ser resumidas en tres.

La **primera** dirección se relaciona con el modelo que se basa en la arquitectura CNN. El parámetro estudiado se corresponde con el tamaño de los n-gramas tomados en consideración, y que se corresponden con el tamaño de los filtros utilizados en la 1D CNN. Básicamente, los tamaños estudiados son 2, 3, 4 y la combinación de ellos. Para cada una de las variantes se construyó un modelo del que se hace referencia más adelante en el apartado del análisis de los resultados como Modelo_CNN_i-gramas. Donde $i \in \{2, 3, 4\}$ indica el tamaño de los filtros utilizados en cada caso. El modelo original combina los 3 tamaños a la vez.

La **segunda** se corresponde también con el modelo basado en la CNN. En este caso se estudia la relevancia de la capa max-pooling añadida en la arquitectura propuesta. Un nuevo modelo se evaluó de esta forma, donde se eliminó la capa en cuestión. La referencia a este modelo en el siguiente apartado es Modelo_CNN-No-Maxpooling.

La **tercera** dirección se corresponde con uno de los aspectos estudiados en la experimentación de HatEval. Específicamente en el estudio de la cantidad de capas de neuronas en el clasificador del modelo. Esto es, la cantidad de capas en la la red neuronal *Fully-Connected* que toma como entrada la salida de la arquitectura CNN o Bi-LSTM según el modelo, y que obtiene la predicción. Con este experimento se pretende realizar una comparación con los resultados experimentales de la tarea anterior (HatEval) en este punto. Estos modelos son referenciados en el siguiente apartado como Modelo_arq_i, donde *arq* indica la arquitectura CNN o Bi-LSTM y la i la cantidad de capas. En el caso de $i = 2$ coincide con la cantidad de capas en los modelos propuestos.

Análisis de los resultados

La Tabla 4.8 muestra los resultados experimentales obtenidos para cada uno de los modelos anteriormente introducidos. En el caso del primero, correspondiente al tamaño de los filtros utilizados en el modelo basado en CNN, puede verse en la primera parte de la tabla que los mejores resultados son alcanzados con la combinación de los tres tipos de filtros (modelo propuesto CNN). De modo que al utilizarlos de manera independiente empeoran los resultados, siendo más crítico el empeoramiento cuando aumenta el tamaño del n-grama correspondiente. Es decir, que los mejores resultados son obtenidos con el modelo basado en bigramas, al considerar solo un tamaño de n-gramas, pero puede mejorarse al combinarlo con otros tamaños, tal como se hace en el modelo propuesto. En cuanto al uso de la técnica de max-pooling para el mismo modelo, puede verse que los resultados también empeoran al eliminar esta capa, siendo superior el resultado experimental del modelo original.

En la segunda parte de la tabla pueden observarse los resultados obtenidos al analizar la tercera dirección de los experimentos. Al igual que en los resultados experimentales en HatEval, puede comprobarse que al incrementar la cantidad de capas de neuronas en el clasificador, los resultados empeoran, lo cual puede deberse a que al aumentar la cantidad de parámetros en el modelo provoca un aumento del sobreajuste. En este sentido, el modelo con 3 capas de neuronas obtiene los peores resultados tanto para el

modelo basado en CNN, como para el basado en Bi-LSTM. Los resultados obtenidos para los otros tres modelos que contienen 0, 1 y 2 capas respectivamente son muy similares entre sí teniendo en cuenta la desviación estándar. Estos resultados sugieren que no puede tomarse un valor concluyente para este hiperparámetro, sino que debe realizarse un proceso de ajuste para cada conjunto de datos y modelo en cuestión.

Finalmente puede observarse que entre los tres modelos propuestos para la tarea, el basado en CNN obtiene un resultado superior aunque muy similar a los otros modelos propuestos. Mientras que el modelo construido a partir de la combinación de los modelos obtiene resultados relativamente inferiores, en contradicción con las hipótesis con la que se diseñó.

Tabla 4.8: Resultados experimentales para OffensEval.

Modelo	F1_score
Modelo propuesto CNN	0.71 ± 0.025
Modelo propuesto Bi-LSTM	0.70 ± 0.001
Modelo propuesto Combinación	0.68 ± 0.013
Modelo_CNN-2-gramas	0.67 ± 0.004
Modelo_CNN-3-gramas	0.65 ± 0.053
Modelo_CNN-4-gramas	0.60 ± 0.216
Modelo_CNN-No-Maxpooling	0.61 ± 0.074
Modelo_CNN-0	0.70 ± 0.001
Modelo_CNN-1	0.68 ± 0.056
Modelo_CNN-3	0.64 ± 0.035
Modelo_Bi-LSTM-0	0.68 ± 0.071
Modelo_Bi-LSTM-1	0.69 ± 0.029
Modelo_Bi-LSTM-3	0.55 ± 0.231

Resultados en el conjunto de pruebas

En la Tabla 4.9 se muestran los resultados alcanzados en la competición por los tres modelos propuestos. Al compararlos entre ellos puede verse el mismo comportamiento que en los resultados experimentales, donde el modelo basado en CNN obtuvo el mejor resultado y la combinación de este

con el modelo basado en Bi-LSTM obtuvo la peor.

Por otra parte cabe señalar que los resultados de los tres modelos propuestos lograron superar a los métodos de base de la tarea, y son además significativamente superiores al resultado obtenido por el peor sistema de la competición. Sin embargo, los resultados no estuvieron cerca del resultado alcanzado por el mejor sistema. Esto puede deberse a la falta de un preprocesamiento al igual que en el caso de HatEval. De forma que posiblemente se ha perdido importante información, pues los términos correspondientes a palabras mal escritas u otro tipo de fenómeno son excluidos del análisis, lo que puede ser estudiado con detalle en un trabajo futuro.

Tabla 4.9: Resultados en la subtarea A de OffensEval [67].

	F1_score
Mínimo	0.1710
Máximo (BERT)	0.8290
Modelo propuesto CNN	0.6600
Modelo propuesto Bi-LSTM	0.5984
Modelo propuesto Combinación	0.5925
Métodos de base	
ALL OFF	0.2182
ALL NOT	0.4189

4.3. Conclusiones

A lo largo de este capítulo se han analizado dos tareas de SemEval, relacionadas con la detección de mensajes de odio. Para cada una de ellas se han presentado sus características y se han resumido las principales aproximaciones propuestas por los participantes. En particular se han detallado nuestros modelos, estudiando una serie de aspectos que pueden influir en el rendimiento en la tarea.

A manera de resumen pueden identificarse algunas conclusiones parciales a tener en cuenta para el diseño de próximas aproximaciones. Primeramente es importante señalar que el ajuste de los hiperparámetros y el uso de recursos es dependiente de los datos con los que se trabaje. De forma que la configuración de un modelo en cuanto, por ejemplo a la profundidad de la

red neuronal utilizada o la arquitectura, puede variar según los estructura de los datos. Esto se puede comprobar con la capa de atención en el modelo propuesto para HatEval, donde para el español no significó un paso importante, sin embargo sí lo fue para el caso del inglés. Además, el uso de un recurso lingüístico representó una mejoría para la tarea de agresividad de HatEval en el caso del español, pero no para el inglés donde los resultados obtenidos fueron pobres.

Otro factor a tener en cuenta está relacionado con el preprocesamiento de los datos. En el caso de OffensEval, los modelos propuestos alcanzaron resultados inferiores a los obtenidos por otros sistemas muy similares, donde la diferencia detectada fue el paso de preprocesamiento que en nuestros modelos no se tuvo en cuenta. Por lo que en general, para las dos tareas se podrían haber alcanzado mejores resultados posiblemente al realizar algún preprocesamiento de los datos, lo que se plantea estudiar con detalle en un trabajo futuro.

En el próximo capítulo se continúa con el estudio de otras tareas con lo que se pretende comprobar estas conclusiones particulares y enriquecerlas.

Capítulo 5

Análisis de agresividad en casos de estudios particulares

La detección de mensajes de odio constituye un reto de gran importancia en la actualidad como se ha planteado a lo largo del presente trabajo. Una dificultad se agrega con la gran cantidad de idiomas y variedades de los mismos. Un mensaje puede expresar odio en función del lenguaje específico de cada país. De hecho, en dos países con el mismo idioma oficial puede variar la definición de HS según expresiones típicas del lenguaje local.

Dentro de las tareas propuestas en función de la detección de HS se pueden señalar dos que tienen como propósito analizar tweets para lenguajes específicos. Por una lado, HaSpeeDe se enfoca en el idioma italiano, mientras que MEX-A3T en el español mexicano particularmente.

En los siguientes apartados estas dos tareas son analizadas, puntualizando en las propuestas realizadas para darles tratamiento.

5.1. HaSpeeDe: Detección de odio en textos en italiano

HaSpeeDe (*Hate Speech Detection*, según su nombre en inglés) fue una tarea propuesta en Evalita 2018. El objetivo es identificar HS en italiano en textos publicados en dos sitios. De esta forma el análisis no se limita a Twitter, sino que además involucra a Facebook. Para ello fueron compartidos dos corpus que se describen más adelante.

La tarea se divide en tres subtareas, determinadas por la fuente de los textos utilizados para el entrenamiento y prueba de los sistemas.

- HaSpeeDe-FB: En esta subtarea son solo considerados los textos publicados en Facebook para entrenar y evaluar los sistemas.
- HaSpeeDe-TW: De forma análoga a la subtarea anterior, en esta tarea solo se tienen en cuenta los textos cuya fuente es Twitter.
- Cross-HaSpeeDe: El propósito de esta subtarea es evaluar el comportamiento de los sistemas entrenados con textos de una fuente determinada y evaluados con textos tomados de otra fuente distinta.
    - Cross-HaSpeeDe-FB: En esta subtarea los textos publicados en Facebook son utilizados para el entrenamiento de los sistemas, mientras que los publicados en Twitter son utilizados para la evaluación.
    - Cross-HaSpeeDe-TW: Esta subtarea es muy similar a la anterior. En este caso los tweets son utilizados para entrenar y las publicaciones de Facebook para evaluar.

Dado que los objetivos del presente trabajo se centran en el estudio de HS en tweets, solo será analizada la subtarea HaSpeeDe-TW. Los detalles para el resto de las subtareas pueden encontrarse en [26].

Corpus HaSpeeDe y medida de evaluación

El corpus proporcionado para el desarrollo de la tarea está conformado por un total de 4000 tweets en italiano distribuidos de la forma que indica la Tabla 5.1. En [58] se pueden encontrar todos los detalles del corpus, incluido el proceso de construcción y anotación.

Como puede verse es un corpus pequeño, donde el número de tweets con HS es muy inferior al de los tweets que no contienen odio. Estas características pueden influir en un mal comportamiento de los sistemas basados en aprendizaje profundo, lo que se comenta más adelante en el análisis de los resultados obtenidos.

Tabla 5.1: Conjunto de datos proporcionados en HaSpeeDe-TW.

HaSpeeDe-TW	**Entrenamiento**	**Prueba**
HS	972	324
No HS	2028	676

Para la evaluación de los sistemas se utiliza la medida *Macro-averaged F1-score* (F1_score) debido a que los datos no están balanceados.

Métodos de base

Un sencillo método de base fue utilizado como punto de referencia en la evaluación de los sistemas en esta tarea. Básicamente el método consiste en asignar la clase más frecuente en el entrenamiento a los textos en el conjunto de prueba.

Aproximaciones utilizadas por los participantes

Una gran parte de los sistemas propuestos en HaSpeeDe se basan en SVM [15]. De hecho los mejores resultados han sido alcanzados por estos sistemas. Esto puede deberse a la poca cantidad de textos proporcionados para el entrenamiento de los modelos, lo que puede afectar el ajuste de los parámetros de modelos basados en el aprendizaje profundo.

Las técnicas utilizadas para la representación de los textos han sido TF-IDF y n-gramas para los modelos basados en el aprendizaje automático tradicional como las SVM antes comentadas y otros modelos como árboles de decisión y LR.

Por otro lado, los embeddings a nivel de palabras basados en Word2vec y FastText han sido empleados en modelos basados en CNN y RNN.

5.2. MEX-A3T: Análisis de autoría y agresividad en Twitter para el caso del español mexicano

MEX-A3T (*Authorship and aggressiveness analysis in Mexican Spanish Tweets*, según su nombre en inglés) fue una tarea propuesta en el marco de IberEval 2018. El propósito fundamental de esta tarea radica en el análisis de textos escritos en el caso particular del español mexicano. Esto constituye un reto, ya que muchos términos y frases utilizados en México no son conocidos en el resto de los países hispanoparlantes.

Con este objetivo en el punto de atención, la tarea fue dividida en dos subtareas principales.

- Perfil del autor: El propósito de esta subtarea (*Author Profiling*, en inglés) es caracterizar a los usuarios identificando su lugar de residencia y su ocupación.
- Detección de agresividad: Con esta tarea (*Aggressiveness Detection*, en inglés) se pretende discriminar entre textos agresivos y no agresivos.

La segunda subtarea se corresponde con los objetivos del presente trabajo, y es por tanto la que se analiza en esta y otros apartados posteriores.

Corpus MEX-A3T y medida de evaluación

El corpus proporcionado para el desarrollo de MEX-A3T está conformado por un conjunto de tweets distribuidos de la forma que se indica en la Tabla 5.2. Los textos han sido tomados de usuarios mexicanos en un período de 3 meses, utilizando palabras clave groseras y hashtags controversiales relacionados con tópicos de política, sexismo, homofobia y discriminación. Más detalles sobre el proceso de recolección y anotación del corpus se encuentran en [5].

Tabla 5.2: Conjunto de datos proporcionados en MEX-A3T.

MEX-A3T	**Entrenamiento**	**Prueba**
Agresivo	2727	784
No agresivo	4973	2372

Como puede observarse, la clase mayormente representada es la correspondiente a los mensajes no agresivos. Tanto en el conjunto de entrenamiento como en el de prueba, siendo en este último el más significativo.

En la evaluación y comparación de los sistemas propuestos en la subtarea de detección de agresividad se tiene en cuenta el rendimiento en la clase de los tweets agresivos. En este sentido la medida utilizada para evaluar a los sistemas es la F1_score en la clase agresiva (F1_score_agg).

Métodos de base

Como métodos de base fueron utilizadas dos aproximaciones basadas en SVM. La primera utiliza una representación basada en bolsas de palabras, para lo cual se eliminan las palabras sin carga semántica (*stopwords*, en inglés) y los caracteres especiales. Por otro lado, la segunda aproximación utiliza una representación basada en 3-gramas, para lo cual se eliminan de igual forma las *stopwords* y los caracteres especiales.

Aproximaciones utilizadas por los participantes

Varios sistemas propusieron un preprocesamiento de los datos para limpiar los textos. Algunos llevaron todas las palabras a minúsculas y sustituyeron términos como números y menciones por símbolos especiales. Uno de

los sistemas además normaliza las diferentes formas de expresar un insulto por un determinado término. Con esta estrategia el sistema correspondiente no alcanzó buenos resultados [5].

Para la representación de los textos se utilizaron de forma general n-gramas de caracteres, palabras y etiquetas del análisis semántico con técnicas como TF-IDF y bolsas de palabras. Uno de los sistemas utilizó la técnica de análisis de componentes principales (PCA, por sus siglas en inglés) para la extracción de características, pero no obtuvo los mejores resultados.

En cuanto a la clasificación de los tweets en agresivos o no, se pueden señalar diferentes métodos como LR, SVM, árboles de decisión y modelos basados en aprendizaje profundo como CNN. Siendo el sistema que obtuvo el mejor resultado un ensembler. Dicho modelo combina la predicción de un modelo basado en un lexicón con palabras agresivas y otro modelo basado en la representación de textos con FastText [57].

5.3. Modelo propuesto para HaSpeeDe-TW y MEX-A3T

En forma cronológica, estas tareas fueron desarrolladas antes que las tareas de SemEval 2019 presentadas en el capítulo anterior. De forma que el modelo ModelRAR lo propusimos por primera vez para las dos tareas presentadas en este capítulo.

A diferencia de las aproximaciones comentadas con las tareas anteriores, las estrategias utilizadas en estos casos se basan únicamente en ModelRAR. La entrada de los modelos consiste en la secuencia de vectores correspondientes a los embeddings de palabras y la salida es un vector correspondiente al último estado oculto de la segunda LSTM del modelo. Dicho vector de salida se utiliza en la clasificación binaria de presencia o no de odio mediante una capa de dos neuronas con una función Softmax. El optimizador utilizado es Adam con la función de pérdida de entropía cruzada en ambas tareas.

Hasta aquí se puede ver que se ha utilizado la misma estrategia tanto en HaSpeeDe-TW como en MEX-A3T. Lo que permite establecer comparaciones y llegar a conclusiones más robustas sobre el modelo propuesto. La diferencia principal radica en la representación de los tweets como se comenta a continuación.

Además de esta aproximación, para MEX-A3T se propuso otra aproximación muy similar con la diferencia de que incorpora un recurso lingüístico. Este recurso se obtuvo a partir del estudio realizado por [39]. Donde los autores proponen una metodología para la detección de frases obscenas

y vulgares en tweets mexicanos. Si es cierto que los mensajes agresivos pueden contener también otros tipos de frases, este es un recurso que puede ayudar a identificar un gran número de tweets agresivos. De esta manera, se agregó una característica que define la presencia o no de palabras obscenas o vulgares en los tweets de acuerdo con un recurso construido en el trabajo citado.

En el caso de HaSpeeDe-TW no se contaba con un recurso similar, por lo que la estrategia que se siguió para evaluar la segunda variante en el corpus de esta tarea fue construir un recurso. Para ello se obtuvieron las palabras más frecuentes en los tweets etiquetados como HS en el conjunto de entrenamiento, sin tener en cuenta las stopwords. De forma que se construyó el recurso como una lista de palabras seleccionadas de los tweets de entrenamiento. En los siguientes apartados se hace referencia a esta aproximación como Modelo_Recurso para el análisis de resultados de las dos tareas.

Preprocesamiento y representación de los tweets

Como primer paso en la aproximación propuesta para ambas tareas los tweets son preprocesados. En este sentido se realiza una serie de normalizaciones de algunas características que pueden aparecer en este tipo de textos y que han sido comentados anteriormente en el apartado 2.2.1.

Primeramente son eliminadas todas las urls, y luego los emoticones son reconocidos y sustituidos por palabras correspondientes que expresan el sentimiento que ellos transmiten. Luego cada palabra es sustituida por su lema utilizando para ello la herramienta Freeling [51].

La representación de los tweets normalizados es obtenida a nivel de palabras. En el caso de HaSpeeDe-TW se han utilizado vectores pre-entrenados con FastText para el idioma italiano, mientras que para MEX-A3T se utiliza el algoritmo Word2vec para el español.

5.3.1. Resultados alcanzados en HaSpeeDe-TW

En la Tabla 5.3 se muestran los resultados experimentales obtenidos con el corpus de Twitter para HaSpeeDe. Como puede observarse los resultados alcanzados con las dos variantes utilizadas son similares, aunque en el caso de la estrategia que tiene en cuenta el recurso construido se logra incrementar en cierta medida el rendimiento.

En la Tabla 5.4 se muestran los resultados obtenidos con el conjunto de prueba. En este caso el mejor resultado se obtuvo con el modelo original donde no se utiliza el recurso. Esto puede deberse a que el conjunto de prueba

Tabla 5.3: Resultados experimentales en la tarea HaSpeeDe-TW.

Modelo	F1_score
Modelo propuesto	0.8530 ± 0.005
Modelo_Recurso	0.8770 ± 0.012

es muy diferente al de entrenamiento. De manera que palabras frecuentes en mensajes de odio en el entrenamiento también aparecen en gran medida en la validación, pero no así en el conjunto de prueba. Por lo que se puede estar introduciendo ruido en el aprendizaje del modelo al utilizar ciertas palabras para identificar odio cuando realmente no se debería. Esto comprueba la idea comentada en el capítulo anterior, donde la estrategia de utilización de este tipo de recurso dependía de los datos utilizados y del recurso en sí. Quizás, haciendo un análisis más detallado de los términos en el recurso pueda mejorar los resultados en conjuntos de textos que sean muy diferentes a los textos utilizados para la construcción de dicho recurso.

Por otra parte, puede verse que ambas aproximaciones propuestas logran superar al método de base de la tarea. Sin embargo no se obtienen resultados cercanos al sistema con mejor resultado de la competición.

Tabla 5.4: Resultados en la tarea HaSpeeDe-TW.

Modelo	F1_score
Mínimo	0.4033
Máximo (Bi-LSTM)	0.7993
Modelo propuesto	0.6638
Modelo_Recurso	0.6567
Método de base	0.4033

A pesar de que no se analizan en este trabajo los resultados con el corpus de Facebook, cabe señalar que los resultados alcanzados fueron mejores. Concretamente 0.7147 y 0.7144 para el modelo propuesto y para el modelo que utiliza el recurso respectivamente, mientras que el mejor resultado en la competición fue de 0.8288 y el peor de 0.2424. Quizás esto se deba a que el procesamiento de los tweets no es suficiente y al contener tantos términos especiales se afecta el entrenamiento de los modelos.

5.3.2. Resultados alcanzados en MEX-A3T

En la tarea MEX-A3T se obtuvieron en forma general resultados muy bajos por los sistemas propuestos. En la Tabla 5.5 se muestra un resumen de los resultados alcanzados. Como puede verse, el mejor resultado con la medida F1_score_agg, que es la utilizada para la comparación de los sistemas, es de 0.488 seguido de uno de nuestros modelos con 0.45. Este modelo es el que incluye la característica lingüística a partir del recurso comentado anteriormente. Con este resultado puede comprobarse la importancia del recurso en la aproximación. De manera que se logró subir desde un resultado que en F1_score_agg es el más bajo y que no supera a los dos métodos de base en la tarea, al segundo lugar de la lista de sistemas propuestos en general. Reafirma este resultado que el uso de recursos adecuados para los datos y la tarea en cuestión puede significar una gran diferencia en el rendimiento de los sistemas.

Tabla 5.5: Resultados en la tarea MEX-A3T.

Modelo	F1_score_agg	F1_score
Mínimo	0.3090	0.5830
Máximo (Ensembler)	0.4880	0.6200
Modelo propuesto	0.3090	0.5830
Modelo_Recurso	0.4500	0.6050
Métodos de base (SVM)		
Trigramas	0.4300	0.6080
Bolsas de palabras	0.3690	0.5760

5.4. Segunda edición de MEX-A3T

Este año 2019 se desarrolló la segunda edición de MEX-A3T en el marco de IberLEF 2019. El objetivo ha estado encaminado a impulsar el tratamiento computacional del español mexicano y a continuar mejorando en la detección de mensajes agresivos en Twitter [7]. Para ello se propuso el mismo entorno experimental que en la primera edición. De esta forma coincide el corpus para el entrenamiento y validación de los sistemas, así como la métrica para la evaluación y los dos métodos de base.

5.4.1. Modelos propuestos

Inicialmente se analizaron tres modelos diferentes entre sí [27], y diferentes además al modelo propuesto en la primera edición de la tarea. Con base es estos modelos se propusieron para la tarea tres aproximaciones. Esto nos permite estudiar el comportamiento de distintas estrategias para un mismo corpus. En adición, otras aproximaciones fueron estudiadas fuera de la competición con el corpus proporcionado a partir de las experiencias acumuladas. Más adelante se presenta un resumen del estudio realizado para lograr arribar a conclusiones más completas.

En cada uno de los modelos se realiza un preprocesamiento como paso inicial de normalización. Para ello se normalizan diferentes características, típicamente presentes en los tweets, y que posiblemente no tienen información semántica discriminatoria. De esta forma, los números se reemplazan por la etiqueta *_num*, las fechas por la etiqueta *_date* y todos los enlaces por la etiqueta *_url*. Además, las menciones del usuario, identificadas por el primer carácter @, son reemplazadas por *_user*. Los hashtags y emoticones no se procesaron para evitar perder información que estos puedan contener.

Primer modelo

Con la experiencia acumulada a partir de las tareas de evaluación anteriores y del análisis de los resultados obtenidos se diseñaron las nuevas aproximaciones. En primer lugar un preprocesamiento más profundo de los tweets, razón por la cual cada aproximación parte del paso de normalización comentado en el apartado anterior. El segundo aspecto se relaciona con el uso de n-gramas, debido a que una gran cantidad de sistemas utilizan esta técnica en sus aproximaciones. Otro aspecto también analizado se corresponde con la cantidad de parámetros a entrenar en los modelos basados en aprendizaje profundo, que al ser muy grande puede afectar el rendimiento de los modelos.

Para dar solución al tercer aspecto señalado se ha modificado ModelRAR para reducir la cantidad de parámetros utilizados. En este sentido se han sustituido las dos primeras capas correspondientes con la primera Bi-LSTM y la atención por una 1D CNN. El nuevo modelo, que identificamos con CNN-LSTM puede verse en la Figura 5.1.

La arquitectura de la capa 1D CNN está conformada por 150 filtros de tamaño 2. De esta forma se da tratamiento también al segundo aspecto comentado, ya que se analizan los bigramas de la secuencia de entrada. Como resultado de esta convolución se obtienen 150 vectores de tamaño

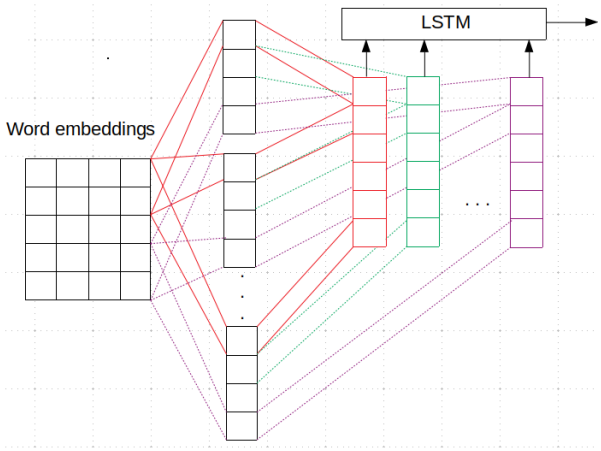

Figura 5.1: Modelo propuesto CNN-LSTM.

$N-1$, donde N es la longitud fijada de la secuencia, correspondiente con la cantidad de filas en la matriz conformada por la concatenación de los embeddings.

Esos 150 vectores son concatenados formando una matriz de dimensión $N-1$ x 150, interpretada como la representación de los bigramas de la entrada. Donde 150 es el tamaño de los nuevos vectores correspondientes a dichos bigramas. Esta representación es la entrada a su vez de una capa LSTM, cuyo último estado oculto es utilizado para la predicción, de forma análoga al otro modelo.

Haciendo un análisis de la cantidad de parámetros a entrenar en este nuevo modelo puede verse la reducción significativa respecto a ModelRAR, donde la cantidad asciende al orden de los millones. Mientras que en CNN-LSTM se queda en el orden de los miles. Puede determinarse fácilmente de la siguiente forma: $n*T*m+m$, donde n se corresponde con el tamaño de los filtros, siendo 2 en el modelo diseñado; T es la longitud de los embeddings en la entrada, siendo 200 el tamaño de los embeddings utilizados; y m es la cantidad filtros, 150 en el modelo. Por último, la adición de m se debe al vector de *bias* cuyo tamaño se determina por la cantidad de filtros. Por tanto, el máximo número de parámetros a entrenar que introduce esta capa es de $2*200*150+150 = 60150$.

Para la entrada del modelo los textos son representados con embeddings a nivel de palabras. Para esto, se utiliza el modelo Word2vec provisto por los organizadores de la tarea. Este ha sido entrenado con el corpus MEX-A3T, el que contiene 500000 tokens y el tamaño de los embeddings es 200.

Segundo modelo

El segundo modelo consisten en una estrategia más sencilla. En este caso se utiliza una red neuronal *Fully-Connected* cuya entrada es un vector que se corresponde con la representación del texto origen, y la última capa tiene dos neuronas con la función Softmax para la clasificación en agresivo o no. Las capas intermedias tienen 128, 64 y 32 neuronas respectivamente con la función de activación Relu.

La representación de los textos se realizó con TF-IDF y bigramas, obteniéndose de esta forma el vector al que se hace referencia anteriormente como entrada a la red. En los experimentos se realizaron pruebas con diferentes n-gramas, tal como se muestra más adelante en el apartado del análisis de los resultados, donde se hace referencia a este modelo como TFIDF-FC. Además, es agregada a la representación una característica relacionada con el recurso utilizado en la primera edición, indicando la cantidad de apariciones en el texto de las palabras listadas en el recurso.

Tercer modelo

Un tercer modelo muy similar al anterior fue propuesto, donde se varía la forma de representación de los textos. En este caso la representación vectorial de los textos se obtiene con USE multilingüe, que incluye al español. La idea es utilizar una forma de obtener embeddings a nivel de oración donde se tenga en cuenta el contexto de las palabras en la secuencia de entrada. Además, el número de parámetros disminuye considerablemente respecto al segundo modelo, ya que la representación para cualquier texto es un vector de dimensión 512, mientras que con el anterior se obtiene un vector de tamaño igual al tamaño del vocabulario definido.

Aproximaciones propuestas para la segunda edición de MEX-A3T

Partiendo de estos modelos se propusieron tres aproximaciones en función de los resultados experimentales que se comentan más adelante.

La primera aproximación consiste en una combinación de los tres modelos de la forma en que se muestra en la Figura 5.2. De forma que la predicción final para cada tweet se obtiene por mayoría de votos, dada la predicción de cada uno de los tres modelos.

Por otra parte, la segunda y tercera aproximación se corresponden con los modelos CNN-LSTM y TFIDF-FC respectivamente.

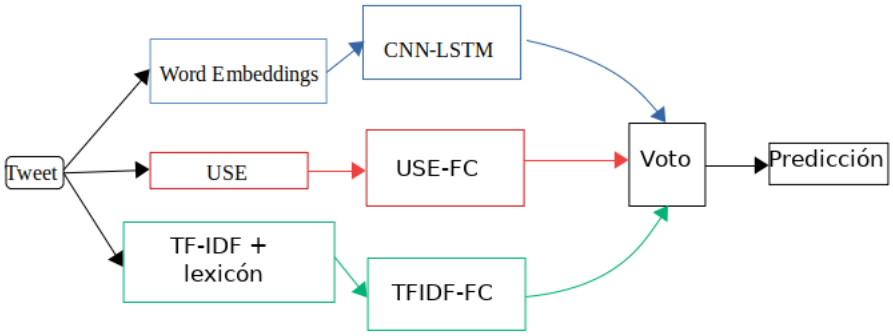

Figura 5.2: Arquitectura propuesta en la segunda edición de MEX-A3T.

5.4.2. Experimentación

En este apartado se describen los experimentos realizados para la segunda edición de MEX-A3T. En primer lugar se analizan los resultados experimentales obtenidos con los modelos propuestos y se presentan los resultados generales alcanzados en la competición. Luego se comentan diferentes experimentos realizados con el corpus de MEX-A3T fuera de la competición y se analizan los resultados.

Análisis de los resultados

En la Tabla 5.6 se muestran los resultados experimentales obtenidos en cuanto a las dos métricas comentadas para cada una de las aproximaciones. Puede verse una diferencia significativa en el modelo más sencillo, cuya representación está basada en la técnica tradicional TF-IDF. Esto puede deberse a que con la representación se logra dar mayor peso a términos frecuentes en textos agresivos, cuyo peso por otra parte es inferior en textos no agresivos.

En contraste, el resultado más bajo entre las tres aproximaciones propuestas se alcanza con el modelo similar al primero pero con la representación basada en USE. Una mejora en este sentido podría ser entrenar un modelo USE a partir de un gran conjunto de tweets mexicanos externos para ajustar los vectores a las características de los textos en análisis. Por falta de dicho conjunto de tweets este paso no fue realizado.

Por último cabe señalar que el modelo CNN-LSTM, si bien supera al último modelo comentado, no se acerca al rendimiento del modelo que utiliza como representación TF-IDF.

Tabla 5.6: Resultados experimentales en la segunda edición de MEX-A3T.

Modelo	F1_score_agg	F1_score
CNN-LSTM	0.67 ± 0.003	0.73 ± 0.011
TFIDF-FC	0.72 ± 0.005	0.79 ± 0.003
USE-FC	0.65 ± 0.021	0.74 ± 0.016
Combinación	0.70 ± 0.010	0.78 ± 0.006

Resultados en el conjunto de pruebas

En la Tabla 5.7 se muestra un resumen de los resultados alcanzados en la segunda edición de MEX-A3T. Como puede verse, el mejor resultado no logró superar al mejor resultado de la primera edición con 0.488. Sin embargo, en cuanto a dos de nuestras aproximaciones superaron a nuestro mejor resultado en la primera edición (0.45), obteniendo en este caso el tercer y cuarto puesto para TFIDF-FC y la combinación de los modelos respectivamente. En el caso del modelo USE-FC alcanzó la séptima posición. En todos los casos se logró superar los dos métodos de base de la tarea en cuanto a la métrica F1_score_agg, mientras que en la métrica F1_score el modelo CNN-LSTM solo superó a uno de los métodos de base.

Tabla 5.7: Resultados en la segunda edición de MEX-A3T.

Modelo	F1_score_agg	F1_score
Mínimo	0.2682	0.5311
Máximo (SVM y MLP)	0.4796	0.6464
Combinación	0.4635	0.6205
CNN-LSTM	0.4405	0.5920
TFIDF-FC	0.4749	0.6349
Métodos de base (SVM)		
Trigramas	0.4300	0.6080
Bolsas de palabras	0.3690	0.5760

5.4.3. Otros experimentos y resultados con el corpus de MEX-A3T

Otros experimentos fueron desarrollados utilizando el corpus de MEX-A3T, con el propósito de estudiar el impacto de diferentes aspectos en el rendimiento de distintas aproximaciones en la tarea. Cada uno de estos aspectos se comentan a continuación, presentando con ellos los diferentes experimentos realizados. Los resultados experimentales correspondientes se comentan en el siguiente apartado.

En varias tareas los mejores resultados fueron alcanzados con el uso de modelos tradicionales del aprendizaje automático. Por tal razón el **primer** aspecto estudiado se relaciona con este fenómeno. Para ello se han realizado experimentos con LR y SVM, que son los modelos tradicionales más frecuentemente utilizados. La representación de los tweets se ha realizado con TF-IDF y se ha variado el parámetro relacionado con el tamaño de los n-gramas. Concretamente se han analizado unigramas, bigramas y trigramas. Cada resultado es comparado con el resto de los modelos que se presentan en este apartado, haciendo hincapié en el segundo modelo propuesto para la tarea (TFIDF-FC), en el que se utiliza una red neuronal *Fully-Connected* con 3 capas densas de neuronas con igual representación de los textos. Es decir, los modelos relacionados con este primer aspecto son variaciones de TFIDF-FC, donde se cambia el modelo de clasificación. La referencia a estos modelos en el siguiente apartado son TFIDF-M, donde M representa al clasificador en cada caso entre LR, SVM y FC.

El **segundo** aspecto se relaciona con la representación de los textos. En el punto anterior se utilizaba una estrategia con la que se lograba una representación de un texto como un vector. Sin embargo, el interés por estudiar el impacto de otras estrategias basadas en el aprendizaje profundo crece, por un lado con la idea que subyace en cuanto al análisis del contexto de las palabras en el texto; por otro lado con la reducción significativa del espacio de representación. Además, crece el interés con los buenos resultados que han alcanzado en otras tareas. Por tal razón otro grupo de experimentos se han diseñado e implementado siguiendo esta idea. Para ello se han utilizado los mismo clasificadores presentados en el punto anterior: SVM, LR y FC (red neuronal *Fully-Connected*). Dos diferentes representaciones son utilizadas:

1. USE: De forma similar a la propuesta en el tercer modelo en la segunda edición de MEX-A3T, una representación estudiada es USE. Para esto se utiliza el modelo multilingüe basado en Transformer, con lo que se obtiene un vector de dimensión 512 para cada uno de los tweets en español.

2. BERT: Otra representación utilizada se basa en el modelo BERT multilingüe. Como muestra la Figura 5.3 con el modelo se puede obtener un vector correspondiente a cada token en la entrada. Suele utilizarse el vector correspondiente al token especial "CLS" como el vector de entrada a un clasificador adaptado a una tarea determinada. En nuestro caso, este vector es utilizado como la representación de los tweets y la entrada a cada uno de los clasificadores comentados.

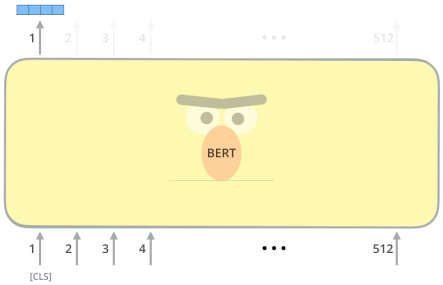

Figura 5.3: Representación con BERT. Tomado de [4] y modificado.

Un **tercer** aspecto se corresponde con el modelo CNN-LSTM propuesto. En este caso se analiza la primera parte del modelo, correspondiente a la red neuronal convolucional utilizada para obtener la representación empleada como entrada a la LSTM. De forma que se elimina la LSTM del modelo y los vectores resultantes de la CNN son concatenados, y el vector resultante es utilizado para la clasificación final con la función Softmax. La referencia a este modelo en el apartado del análisis de resultados es CNN, mientras que la del modelo original CNN-LSTM.

Otro punto de interés ha sido establecer una comparación entre los diferentes modelos y variantes estudiadas y propuestas en la primera y segunda edición de MEX-A3T. Por tal razón, el **cuarto** aspecto se relaciona con el estudio más detallado de ModelRAR en este corpus. Para ello se realizan una serie de experimentos que permiten analizar el rendimiento en la tarea de diferentes variantes, comparándolas con el modelo CNN-LSTM, propuesto en la segunda edición. La primera variante es básicamente el modelo eliminando la última capa LSTM (ModelRA), donde el vector correspondiente a la atención calculada para cada estado oculto de la Bi-LSTM es empleado para la clasificación con una Softmax. En la segunda variante se elimina la primera capa Bi-LSTM (ModelAR), de forma que la capa de atención recibe como entrada la secuencia de embeddings a nivel de palabras correspondiente a la entrada. De forma contraria a la segunda variante, la tercera

está conformada solo por la primera capa del modelo (ModelB), consistiendo básicamente en una Bi-LSTM donde la clasificación se realiza a partir de su último estado oculto como en los casos anteriores. Además, para comprobar cuán significativo es utilizar una Bi-LTSM en lugar de una LSTM (ModelL), una cuarta variante se ha estudiado muy similar a la tercera, donde se sustituye la arquitectura bidireccional por la que solo tiene en cuenta una dirección de procesamiento. Por último, la última variante es concretamente ModelRaR.

En cada uno de los modelos presentados basados en la representación de textos con embeddings a nivel de palabras se ha utilizado el modelo de Word2vec comentado al introducir el primer modelo propuesto en la segunda edición de MEX-A3T. Un **quinto** aspecto estudiado fue relacionado con la sustitución de este modelo de representación en cada uno de los modelos, utilizando FastText para ello, ya que la posibilidad de representar las palabras no vistas en el entrenamiento que brinda constituye una ventaja. La idea de estos experimentos es establecer comparaciones entre los resultados obtenidos con ambos modelos de representación. Para el uso de FastText se ha empleado un modelo proporcionado por los organizadores de MEX-A3T, que ha sido entrenado con un conjunto externo de tweets mexicanos. Contiene 1,247.3M tokens y el tamaño de los embeddings es de 100.

El **sexto** aspecto se relaciona con el uso del recurso conformado por el conjunto de palabras y frases abusivas. Los resultados obtenidos en la primera edición mostraron su importancia. En la segunda edición fue utilizado en cada uno de los modelos. Con este experimento la idea es poder llegar a una conclusión sobre el impacto que ha teniendo en cada uno de los modelos. Además, se pretende analizar la importancia del preprocesamiento, comparando los resultados obtenidos en cada caso con los que se conseguirían sin ese paso.

Por último, el **séptimo** aspecto se relaciona con la cantidad de datos utilizados para entrenar los modelos. En muchos casos, la cantidad de textos brindados para el entrenamiento de los modelos es muy pequeño y se hace difícil encontrar más textos etiquetados para la misma tarea, dificultando con ello el ajuste de la gran cantidad de parámetros que suelen tener los modelos basados en aprendizaje profundo. Por tal razón, una buena idea podría ser emplear alguna estrategia que permita aumentar la cantidad de datos de entrenamiento a partir de los datos proporcionados en el corpus original. En nuestro caso, una estrategia muy sencilla ha sido utilizada para ello. Concretamente se ha realizado mediante la traducción de los textos.

Un texto t del corpus se traduce a otro idioma y el texto obtenido es nuevamente traducido al idioma original, en este caso español, obteniéndose

un texto t'. En algunos casos t' coincide con t, pero solo se tienen en cuenta aquellos que constituyen textos diferentes. Los idiomas auxiliares utilizados fueron árabe, checo, hindi, inglés, francés, japonés, chino y alemán. Algunos idiomas son similares al español y otros muy diferentes como por ejemplo el chino. Con esto se logró un incremento considerable de los datos de entrenamiento de 7700 tweets a 55198 uniendo los textos nuevos obtenidos con cada uno de los idiomas. En la traducción fue utilizado el API oficial de Google Cloud [1] para la comunicación e integración del servicio con python. Señalar que fue necesario obtener credenciales para su uso, solicitando para ello un token de acceso desde el servidor de autorización de Google.

Análisis de resultados experimentales

Al igual que en los casos anteriores, para los experimentos desarrollados se utilizó validación cruzada estratificada con las medidas presentadas con la tarea MEX-A3T: F1_score_agg y F1_score, con la STD correspondiente en cada caso.

En la Tabla 5.8 se muestran los resultados experimentales obtenidos para el primer aspecto en análisis. En primer lugar puede verse que los mejores resultados en general son obtenidos por el clasificador SVM, coincidiendo con los resultados en varias competiciones relacionadas con HS como HatEval. Por otro lado, con los bigramas se tiene una cierta mejoría que puede indicar una característica a tener en cuenta, donde además se alcanzan resultados muy similares entre SVM y FC.

Tabla 5.8: Resultados experimentales con el corpus MEX-A3T. Análisis con diferentes clasificadores tradicionales y distintos tamaños de n-gramas.

Modelo	F1_score_agg		
	1-grama	2-grama	3-grama
TFIDF-LR	0.65 ± 0.031	0.67 ± 0.022	0.66 ± 0.038
TFIDF-SVM	0.70 ± 0.006	0.72 ± 0.001	0.72 ± 0.018
TFIDF-FC	0.69 ± 0.010	0.72 ± 0.005	0.70 ± 0.021
Modelo	F1_score		
TFIDF-LR	0.77 ± 0.006	0.76 ± 0.026	0.76 ± 0.012
TFIDF-SVM	0.78 ± 0.008	0.79 ± 0.003	0.78 ± 0.011
TFIDF-FC	0.76 ± 0.029	0.79 ± 0.014	0.77 ± 0.019

Los resultados con bigramas son también comparados con modelos que utilizan los mismos clasificadores pero diferentes representaciones de los textos, según fue señalado con el segundo aspecto. En la Tabla 5.9 se puede ver que en el caso de la representación con BERT los mejores resultados no son con SVM, sino con LR. Esto muestra una vez más que no existe un modelo definitivo con el que se garanticen los mejores resultados en la tarea, sino que se debe realizar un cuidadoso proceso de ajuste de hiperparámetros en el diseño de las aproximaciones.

Además, un factor que cabe señalar es que los resultados obtenidos con TF-IDF y bigramas son superiores a los obtenidos con otras formas de representación. Esto confirma lo comentado anteriormente referente a que los modelos de USE y BERT deberían ser ajustados a las características de los datos en particular, en este caso tweets mexicanos. Lo cual no se realizó debido a la falta de un corpus externo conformado por este tipo de textos.

Tabla 5.9: Resultados experimentales con el corpus MEX-A3T. Comparación entre diferentes estrategias de representación.

Modelo	F1_score_agg		
	TF-IDF (n=2)	USE	BERT
LR	0.67 ± 0.022	0.65 ± 0.013	0.65 ± 0.016
SVM	0.72 ± 0.001	0.67 ± 0.026	0.64 ± 0.032
FC	0.72 ± 0.005	0.65 ± 0.015	0.64 ± 0.029
Modelo	F1_score		
LR	0.76 ± 0.026	0.74 ± 0.018	0.74 ± 0.018
SVM	0.79 ± 0.003	0.75 ± 0.021	0.72 ± 0.016
FC	0.79 ± 0.014	0.74 ± 0.008	0.73 ± 0.002

En la Tabla 5.10 se muestran los primeros resultados experimentales de los modelos basados en aprendizaje profundo. Donde puede comprobarse que decrementan considerablemente con relación a los obtenidos con métodos tradicionales. Una posible causa es que el tamaño del corpus no es suficiente para el ajuste de la gran cantidad de parámetros de estos modelos.

En particular se muestra una comparación entre los resultados experimentales del modelo CNN-LSTM propuesto para la segunda edición de MEX-A3T y la versión que no incluye LSTM. Como puede observarse, en esta segunda variante los resultados empeoran, lo que sugiere que la ca-

pa LSTM es una buena estrategia en el diseño del modelo CNN-LSTM en particular.

Tabla 5.10: Resultados experimentales con el corpus MEX-A3T. Análisis de la segunda capa LSTM en el modelo CNN-LSTM.

Modelo	F1_score_agg	F1_score
CNN	0.56 ± 0.012	0.58 ± 0.025
CNN-LSTM	0.58 ± 0.016	0.66 ± 0.014

En la Tabla 5.11 se muestran los resultados obtenidos con cada una de las variantes de ModelRAR comentadas anteriormente. En primer lugar se puede ver que el resultado obtenido con la variante que no incluye la última capa LSTM tiene un comportamiento muy similar al del modelo original. Es diferente este fenómeno al caso de los experimentos anteriores donde para el modelo CNN-LSTM, los resultados empeoraban en cierta medida sin la última capa LSTM.

Tabla 5.11: Resultados experimentales con el corpus MEX-A3T. Estudio de la arquitectura de ModelRAR.

Modelo	F1_score_agg	F1_score
ModelRA	0.69 ± 0.003	0.75 ± 0.001
ModelAR	0.61 ± 0.014	0.71 ± 0.024
ModelB	0.56 ± 0.009	0.65 ± 0.030
ModelL	0.57 ± 0.002	0.67 ± 0.023
ModelRAR	0.69 ± 0.030	0.75 ±0.002

Por otra parte, la variante donde se elimina la primera capa Bi-LSTM obtiene resultados muy por debajo de los obtenidos con el modelo original. La diferencia es más significativa que en los experimentos realizados con el mismo modelo en HatEval. Este y el punto anterior muestran una vez más que el diseño y ajuste de los modelos debe realizarse de forma cuidadosa según su arquitectura y los datos con los que se trabaje.

En el caso de los modelos que eliminan la capa de atención y solo consisten en una red neuronal recurrente, los resultados decaen en un mayor grado. Esto muestra la importancia en el diseño del modelo original de las capas que se han eliminado en estas variantes en análisis. Además, a diferen-

cia de lo que podría esperarse, el modelo basado en una Bi-LSTM no logra superar al basado en una simple LSTM.

Los resultados experimentales de todos los modelos y variantes basados en aprendizaje profundo se muestran en la tabla 5.12. Con lo que se puede realizar una comparación de sus rendimientos según la técnica de representación basada en embeddings a nivel de palabras.

En primer lugar puede verse que los resultados mejoran significativamente en la mayoría de los casos al utilizar FastText en lugar de Word2vec. Esto puede deberse a que con FastText la codificación de las palabras se realiza a nivel de n-gramas de caracteres, de manera que palabras raras o no vistas en el entrenamiento del modelo tendrán una representación, a diferencia de Word2vec donde se obvian dichas palabras. Como se ha comentado los tweets suelen contener una gran cantidad de palabras raras, incrementándose con el lenguaje ofensivo donde muchos usuarios tienden a enmascarar las palabras de odio.

En segundo lugar cabe señalar que el modelo propuesto en la primera edición ModelRAR logra mejores resultados experimentales en comparación a los obtenidos con el propuesto en la segunda edición CNN-LSTM. Esto coincide con los resultados obtenidos en las competiciones (0.45 y 0.4405 respectivamente), donde ModelRAR supera, aunque no significativamente, a CNN-LSTM.

Tabla 5.12: Resultados experimentales con el corpus MEX-A3T. Comparación de los resultados en cuanto a las técnicas de representación basadas en embeddings a nivel de palabras.

Modelo	F1_score_agg		F1_score	
	Word2vec	FastText	Word2vec	FastText
ModelRA	0.69 ± 0.003	0.69 ± 0.014	0.75 ± 0.001	0.75 ± 0.012
ModelAR	0.61 ± 0.014	0.66 ± 0.015	0.71 ± 0.024	0.73 ± 0.008
ModelB	0.56 ± 0.009	0.66 ± 0.004	0.65 ± 0.030	0.74 ± 0.002
ModelL	0.57 ± 0.002	0.65 ± 0.016	0.67 ± 0.023	0.72 ± 0.016
ModelRAR	0.69 ± 0.030	0.69 ± 0.027	0.75 ± 0.002	0.76 ± 0.021
CNN	0.56 ± 0.012	0.63 ± 0.013	0.58 ± 0.025	0.66 ± 0.017
CNN-LSTM	0.58 ± 0.016	0.67 ± 0.009	0.66 ± 0.014	0.73 ± 0.008

En la Tabla 5.13 se muestran otros resultados experimentales para los

mismos modelos anteriores. En este caso el análisis se centra en el impacto del preprocesamiento incorporado en las aproximaciones y del uso del recurso, correspondiente al lexicón de las palabras y frases abusivas en el español mexicano.

En cuanto al preprocesamiento puede comprobarse una mejoría en cada uno de los modelos en la columna Lex-Preproc/Lex-No-Preproc de la Tabla 5.13, donde Lex-Preproc indica los resultados para cada modelo utilizando el lexicón y realizando el preprocesamiento. En el caso de Lex-No-Preproc es similar al anterior en el sentido de que se utiliza el lexicón, pero no se realiza el preprocesamiento. La columna No-Lex-Preproc se corresponde con los modelos que no incluyen el lexicón pero si realizan el preprocesamiento.

Tabla 5.13: Resultados experimentales con el corpus MEX-A3T. Estudio del impacto del lexicón (Lex) y el preprocesamiento.

Modelo	F1_score_agg	
	Lex-Preproc/Lex-No-Preproc	No-Lex-Preproc
Lex/No-Preproc	No-Lex	
ModelRA	0.69 ± 0.014 / 0.61 ± 0.009	0.63 ± 0.003
ModelAR	0.66 ± 0.015 / 0.58 ± 0.016	0.62 ± 0.010
ModelB	0.66 ± 0.004 / 0.56 ± 0.017	0.64 ± 0.012
ModelL	0.65 ± 0.016 / 0.56 ± 0.046	0.64 ± 0.027
ModelRAR	0.69 ± 0.027 / 0.63 ± 0.030	0.68 ± 0.008
CNN	0.63 ± 0.013 / 0.52 ± 0.022	0.61 ± 0.012
CNN-LSTM	0.67 ± 0.009 / 0.53 ± 0.040	0.64 ± 0.008
Modelo	F1_score	
ModelRA	0.75 ± 0.012 / 0.67 ± 0.009	0.73 ± 0.002
ModelAR	0.73 ± 0.008 / 0.66 ± 0.010	0.70 ± 0.005
ModelB	0.74 ± 0.002 / 0.64 ± 0.011	0.70 ± 0.008
ModelL	0.72 ± 0.016 / 0.62 ± 0.028	0.71 ± 0.021
ModelRAR	0.76 ± 0.021 / 0.67 ± 0.012	0.77 ± 0.016
CNN	0.66 ± 0.017 / 0.61 ± 0.009	0.66 ± 0.008
CNN-LSTM	0.73 ± 0.008 / 0.66 ± 0.018	0.69 ± 0.008

Los resultados mostrados comprueban la suposición realizada con los

resultados de las tareas en SemEval, donde se expuso que los resultados podrían mejorar al incorporar a las aproximaciones un preprocesamiento de los textos. Por la parte del uso del lexicón se puede ver que los resultados experimentales empeoran en cierta medida cuando no es utilizado. Esto coincide con los resultados obtenidos en la primera edición de la competición e indica la importancia de su uso para el corpus utilizado, independientemente del modelo.

En experimentos previos se ha visto que los modelos tradicionales alcanzan en general mejores resultados que los modelos basados en aprendizaje profundo. Una posible causa identificada es la insuficiencia de datos para el ajuste de la gran cantidad de parámetros en estos últimos modelos. Para ello fue desarrollado un último conjunto de experimentos como se comentó en el apartado anterior. En la Tabla 5.14 se muestra una comparación entre los resultados obtenidos con el corpus original y los obtenidos con el corpus ampliado de la forma presentada anteriormente.

Tabla 5.14: Resultados experimentales con el corpus MEX-A3T. Estudio del impacto del aumento del conjunto de entrenamiento.

Modelo	F1_score_agg		F1_score	
	Original	Ampliado	Original	Ampliado
ModelRA	0.69 ± 0.014	0.92 ± 0.003	0.75 ± 0.012	0.94 ± 0.005
ModelAR	0.66 ± 0.015	0.90 ± 0.003	0.73 ± 0.008	0.92 ± 0.002
ModelB	0.66 ± 0.004	0.91 ± 0.003	0.74 ± 0.002	0.93 ± 0.002
ModelL	0.65 ± 0.016	0.91 ± 0.002	0.72 ± 0.016	0.94 ± 0.003
ModelRAR	0.69 ± 0.027	0.92 ± 0.009	0.76 ± 0.021	0.94 ± 0.008
CNN	0.63 ± 0.013	0.92 ± 0.004	0.66 ± 0.017	0.94 ± 0.004
CNN-LSTM	0.67 ± 0.009	0.91 ± 0.010	0.73 ± 0.008	0.93 ± 0.002
TFIDF-LR	0.67 ± 0.022	0.79 ± 0.012	0.76 ± 0.026	0.84 ± 0.011
TFIDF-SVM	0.72 ± 0.001	0.82 ± 0.003	0.79 ± 0.003	0.85 ± 0.008
TFIDF-FC	0.72 ± 0.005	0.84 ± 0.005	0.79 ± 0.014	0.85 ± 0.010

Tal como se había supuesto, los resultados mejoran significativamente con el corpus aumentado. Para cada uno de los modelos se logran resultados por encima de 0.9, lo que no había sido alcanzado con ninguna de las aproximaciones anteriormente estudiadas. Si bien pudieran utilizarse otras técnicas para el aumento de los datos, la técnica utilizada ha mostrado ser

una buena alternativa que logra generar datos similares a los originales. En los nuevos textos generados se pueden observar diferencias en el orden de las palabras y/o en las palabras en sí con el cambio por sinónimos, aunque no se puede garantizar que se preserven los términos del español mexicano. Otras alternativas deberían ser estudiadas cuidadosamente para no incorporar ruido en el corpus y provocar un entrenamiento no adecuado.

Por último, un aspecto importante a tomar en cuenta es que con este aumento del conjunto de entrenamiento, los modelos basados en aprendizaje profundo logran superar a modelos basados en métodos tradicionales del aprendizaje automático.

5.5. Conclusiones

En este capítulo se ha presentado un análisis sobre tareas desarrolladas en relación con la detección de HS. Particularmente se han comentado los resultados obtenidos en las taresa de evaluación HaSpeeDe-TW y MEX-A3T y se ha realizado un conjunto más de experimentos.

Algunas ideas se han obtenido como resultado de la experimentación, las que pueden servir de guía o punto de partida para próximos estudios. Por una parte se ha comprobado que un adecuado preprocesamiento, en conjunto con técnicas que tengan en cuenta un tratamiento de las palabras raras en la representación, puede mejorar el rendimiento en la tarea. Por otra parte, todos los experimentos realizados señalan que para cada conjunto de datos debe realizarse un proceso particular y cuidadoso de diseño de la aproximación, incluyendo los recursos utilizados, arquitectura y representación de los datos. Por último, cabe señalar el impacto significativo del aumento de los datos para el entrenamiento de los modelos, con lo que se logró una importante mejoría en los resultados experimentales para cada uno de los modelos estudiados. Además, con esta técnica los modelos basados en aprendizaje profundo lograron superar a métodos tradicionales, a diferencia del comportamiento obtenido previamente.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo final de máster se presentó un estudio sobre la detección de mensajes de odio en Twitter. Para ello fueron analizadas diferentes tareas desarrolladas en los últimos dos años, cuyos objetivos estaban vinculados con la cuestión del presente trabajo. En forma general fueron comentadas las técnicas utilizadas por los equipos participantes, y en particular nuestras propuestas basadas en aprendizaje profundo en cada una de las tareas de evaluación.

La principal contribución de nuestro trabajo radica en el análisis realizado en cada una de las tareas, que dio paso a un conjunto de experimentos con los que se estudiaron diferentes aspectos en la detección de HS. Estas ideas pueden servir como punto de partida para próximos estudios o como guía para el diseño y desarrollo de aproximaciones. Además, diferentes arquitecturas y variantes fueron propuestas y estudiadas en la detección de mensajes de odio, y en específico de mensajes agresivos. Esto enriqueció la investigación, permitiendo comprobar varios aspectos interesantes. De todo el análisis realizado en general se pueden listar las siguientes conclusiones:

- El **preprocesamiento** es un paso importante en la detección de HS en tweets, que permite la normalización de las características particulares de este tipo de texto y facilita el entrenamiento de los modelos.
- El diseño de una aproximación para la detección de HS debe realizarse como un proceso cuidadoso en dependencia de los datos utilizados.

Si bien algunas estrategias representaron mejoras en cada uno de los modelos estudiados, algunas tuvieron un impacto en mayor medida en dependencia del corpus utilizado. En este caso se puede resaltar la **capa de atención** de ModelRAR, que supuso una mejora significativa para el corpus en inglés de la tarea de evaluación de HatEval, pero no para el corpus en español.

- El uso de **recursos lingüísticos** puede influir significativamente en la mejora o empeoramiento de los resultados obtenidos en dependencia del corpus utilizado y del recurso en sí. En este sentido se comprobó que para las tareas MEX-A3T y HatEval en español hubo una mejoría, mientras que para HatEval en inglés los resultados experimentales empeoraron considerablemente con un corpus construido de las misma forma que en el caso del español.

- En el diseño de una aproximación se debe prestar especial atención a la estrategia utilizada para la **representación de los textos**. Por un lado, se debe garantizar la representación de ciertas características típicas en los tweets como las palabras raras. Este punto pudo comprobarse con la mejoría de los resultados experimentales obtenidos con FastText, con lo que se representan las palabras basadas en los n-gramas de caracteres y permite la representación de dichas palabras raras que no fueron vistas en el entrenamiento.

  Por otro lado, pudo verse que en la representación de los textos a nivel de oración, la utilización de modelos pre-entrenados como BERT y USE, que constituyen el estado de la cuestión en muchas tareas, no alcanza buenos resultados. Por tal razón se debe realizar un proceso previo de ajuste de dichos modelos a las características particulares de los datos utilizados, empleando conjuntos de datos externos que sean similares. En los resultados de las tareas de SemEval, pudo verse que este tipo de modelos, como por ejemplo BERT en OffensEval, puede alcanzar muy buenos resultados. Sin embargo, no siempre es así, como en el caso de HatEval, donde modelos basados en SVM lo superaron.

- Los **métodos** tradicionales basados en aprendizaje automático, específicamente SVM, alcanzaron mejores resultados que los modelos basados en aprendizaje profundo, tanto en los resultados experimentales como en algunos resultados en las competiciones analizadas.

  En el estudio de este fenómeno se evaluó el **enriquecimiento del corpus** a partir de una técnica de traducción. Con esta estrategia los re-

sultados experimentales aumentaron considerablemente, observándose una superioridad en los modelos basados en aprendizaje profundo. La técnica de aumento, si bien puede parecer sencilla, logra los objetivos experimentales trazados.

Con el proceso de investigación desarrollado y las conclusiones anteriormente expuestas se logró el cumplimiento del objetivo propuesto en el presente trabajo.

6.2. Trabajo futuro

A partir de la investigación realizada y las conclusiones presentadas en el presente trabajo, se puede definir una metodología de trabajo para el diseño de modelos para la detección de mensajes de odio. En dichos modelos puede y debe agregarse otras características que permitan discriminar aun más los mensajes de odio para seguir mejorando los resultados.

En este sentido, como trabajo futuro se propone estudiar con un carácter exhaustivo el comportamiento de los usuarios en Twitter. De manera que las características de los autores de los tweets puedan ser incorporadas en los modelos para la identificación de mensajes de odio. El perfil de cada usuario puede aportar información sobre su lenguaje particular y relación con otros usuarios. Para el desarrollo de ese trabajo será necesario contar con conjuntos de tweets más amplios, en los que la información correspondiente a los usuarios sea proporcionada.

Bibliografía

[1] Google cloud translation. https://cloud.google.com/translate/?hl=es-419. Accessed: 2019-07-16. (Citado en la página 71).

[2] Keras conv1d: Working with 1d convolutional neural networks in keras. https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/. Accessed: 2019-07-16. (Citado en las páginas IX y 29).

[3] Lstm and gru – formula summary. https://isaacchanghau.github.io/post/lstm-gru-formula/. Accessed: 2019-07-16. (Citado en las páginas IX, IX, 26 y 27).

[4] Alammar, J. The illustrated bert, elmo, and co. (how nlp cracked transfer learning). http://jalammar.github.io/illustrated-bert/. Accessed: 2019-07-16. (Citado en las páginas IX y 69).

[5] Álvarez-Carmona, M. Á., Guzmán-Falcón, E., Montes-y Gómez, M., Escalante, H. J., Villasenor-Pineda, L., Reyes-Meza, V., and Rico-Sulayes, A. Overview of mex-a3t at ibereval 2018: Authorship and aggressiveness analysis in mexican spanish tweets. In *Notebook Papers of 3rd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IBEREVAL), Seville, Spain* (2018), vol. 6. (Citado en las páginas 17, 58 y 59).

[6] Anzovino, M., Fersini, E., and Rosso, P. Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems* (2018), Springer, pp. 57–64. (Citado en la página 11).

[7] Aragón, M. E., Álvarez-Carmona, M. Á., Montes-y Gómez, M., Escalante, H. J., Villasenor-Pineda, L., and Moctezuma,

D. Overview of mex-a3t at iberlef 2019: Authorship and aggressiveness analysis in mexican spanish tweets. In *Notebook Papers of 1st SEPLN Workshop on Iberian Languages Evaluation Forum (IberLEF), Bilbao, Spain* (2019). (Citado en las páginas 17 y 62).

[8] Arango, A., Pérez, J., and Poblete, B. Hate speech detection is not as easy as you may think: A closer look at model validation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2019), ACM, pp. 45–54. (Citado en la página 13).

[9] Badjatiya, P., Gupta, S., Gupta, M., and Varma, V. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion* (2017), International World Wide Web Conferences Steering Committee, pp. 759–760. (Citado en las páginas 12 y 13).

[10] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014). (Citado en la página 28).

[11] Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Pardo, F. M. R., Rosso, P., and Sanguinetti, M. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (2019), pp. 54–63. (Citado en las páginas x, 10, 17, 34, 39 y 44).

[12] Blackburn, P. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI Studies in Computational Linguistics. CSLI Publications, 2005. (Citado en la página 32).

[13] Bogdanova, D., Rosso, P., and Solorio, T. Exploring high-level features for detecting cyberpedophilia. *Computer speech & language 28*, 1 (2014), 108–120. (Citado en la página 10).

[14] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics 5* (2017), 135–146. (Citado en la página 21).

[15] Bosco, C., Felice, D., Poletto, F., Sanguinetti, M., and Maurizio, T. Overview of the evalita 2018 hate speech detection task. In

EVALITA 2018-Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (2018), vol. 2263, CEUR, pp. 1–9. (Citado en la página 57).

[16] Brody, S., and Diakopoulos, N. Cooooooooooollllllllll!!!!!!!!!!: Using word lengthening to detect sentiment in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2011), Association for Computational Linguistics, pp. 562–570. (Citado en la página 14).

[17] Burnap, P., and Williams, M. L. Hate speech, machine classification and statistical modelling of information flows on twitter: Interpretation and communication for policy decision making. (Citado en la página 12).

[18] Burnap, P., and Williams, M. L. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet 7*, 2 (2015), 223–242. (Citado en la página 12).

[19] Burnap, P., and Williams, M. L. Us and them: Identifying cyber hate on twitter across multiple protected characteristics. *EPJ Data Science 5*, 1 (2016), 11. (Citado en la página 12).

[20] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018). (Citado en la página 22).

[21] Chen, Y., Zhou, Y., Zhu, S., and Xu, H. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing* (2012), IEEE, pp. 71–80. (Citado en las páginas 10, 12 y 13).

[22] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural hetworks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014). (Citado en la página 26).

[23] Davidson, T., Warmsley, D., Macy, M., and Weber, I. Automated hate speech detection and the problem of offensive language. In *Eleventh International AAAI Conference on Web and Social Media* (2017). (Citado en las páginas 10 y 11).

[24] De la Peña, G. L. Gl at semeval-2019 task 5: Identifying hateful tweets with a deep learning approach. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (2019), pp. 416–419. (Citado en las páginas 35 y 38).

[25] De la Peña, G. L., and Rosso, P. Deepanalyzer at semeval-2019 task 6: A deep learning-based ensemble method for identifying offensive tweets. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (2019), pp. 582–586. (Citado en las páginas 46 y 47).

[26] De la Peña Sarracén, G. L., Pons, R. G., Muñiz-Cuza, C. E., and Rosso, P. Hate speech detection using attention-based lstm. In *EVALITA@ CLiC-it* (2018). (Citado en las páginas ix, ix, 17, 36, 37 y 56).

[27] De la Peña Sarracén, G. L., and Rosso, P. Aggressive analysis in twitter using a combination of models. In *In Proceedings of the First Workshop for Iberian Languages Evaluation Forum (IberLEF 2019), CEUR WS Proceedings* (2019). (Citado en la página 63).

[28] Del Vigna, F., Cimino, A., Dell'Orletta, F., Petrocchi, M., and Tesconi, M. Hate me, hate me not: Hate speech detection on facebook. (Citado en las páginas 13 y 15).

[29] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018). (Citado en la página 23).

[30] Dinakar, K., Reichart, R., and Lieberman, H. Modeling the detection of textual cyberbullying. In *Fifth International AAAI Conference on Weblogs and Social Media* (2011). (Citado en la página 12).

[31] Douglass, S., Mirpuri, S., English, D., and Yip, T. "they were just making jokes": Ethnic/racial teasing and discrimination among adolescents. *Cultural Diversity and Ethnic Minority Psychology 22*, 1 (2016), 69. (Citado en la página 10).

[32] Eisenstein, J. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2013), pp. 359–369. (Citado en la página 15).

[33] Erjavec, K., and Kovačič, M. P. "you don't understand, this is a new war!" analysis of hate speech in news web sites' comments. *Mass Communication and Society 15*, 6 (2012), 899–920. (Citado en la página 3).

[34] Fortuna, P. C. T. Automatic detection of hate speech in text: An overview of the topic and dataset annotation with hierarchical classes. master's dissertation, faculdade de engenharia da universidade do porto. (Citado en las páginas 3 y 9).

[35] Frenda, S., Ghanem, B., Montes-y Gómez, M., and Rosso, P. Online hate speech against women: Automatic identification of misogyny and sexism on twitter. *Journal of Intelligent & Fuzzy Systems 36*, 5 (2019), 4743–4752. (Citado en la página 11).

[36] Gambäck, B., and Sikdar, U. K. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online* (2017), pp. 85–90. (Citado en la página 13).

[37] Gitari, N. D., Zuping, Z., Damien, H., and Long, J. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering 10*, 4 (2015), 215–230. (Citado en las páginas 11 y 13).

[38] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems 28*, 10 (2016), 2222–2232. (Citado en la página 26).

[39] Guzmán, E., Beltrán, B., Tovar, M., Vázquez, A., and Martínez, R. Clasificación de frases obscenas o vulgares dentro de tweets. *Research in Computing Science 85* (2014), 65–74. (Citado en la página 59).

[40] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780. (Citado en la página 26).

[41] Kuipers, G., and Van der Ent, B. The seriousness of ethnic jokes: Ethnic humor and social change in the netherlands, 1995–2012. *Humor 29*, 4 (2016), 605–633. (Citado en la página 10).

[42] Kumar, A., Nayak, S., and Chandra, N. Empirical analysis of supervised machine learning techniques for cyberbullying detection. In

International Conference on Innovative Computing and Communications (2019), Springer, pp. 223–230. (Citado en la página 10).

[43] Liu, S., and Forss, T. Combining n-gram based similarity analysis with sentiment analysis in web content classification. In *KDIR* (2014), pp. 530–537. (Citado en la página 12).

[44] Luong, M.-T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015). (Citado en la página 28).

[45] Malmasi, S., and Zampieri, M. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence 30*, 2 (2018), 187–202. (Citado en la página 11).

[46] Mehdad, Y., and Tetreault, J. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (2016), pp. 299–303. (Citado en la página 12).

[47] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013). (Citado en las páginas ix, ix y 21).

[48] Mondal, M., Silva, L. A., and Benevenuto, F. A measurement study of hate speech in social media. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media* (2017), ACM, pp. 85–94. (Citado en las páginas 9, 10 y 12).

[49] Nandhini, B., and Sheeba, J. Cyberbullying detection and classification using information retrieval algorithm. In *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015)* (2015), ACM, p. 20. (Citado en la página 15).

[50] Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., and Chang, Y. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web* (2016), International World Wide Web Conferences Steering Committee, pp. 145–153. (Citado en las páginas 3, 12 y 15).

[51] Padró, L., and Stanilovsky, E. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC2012)* (2012). (Citado en la página 60).

[52] Papegnies, E., Labatut, V., Dufour, R., and Linares, G. Impact of content features for automatic online abuse detection. In *International Conference on Computational Linguistics and Intelligent Text Processing* (2017), Springer, pp. 404–419. (Citado en la página 13).

[53] Park, J. H., and Fung, P. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206* (2017). (Citado en la página 13).

[54] Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543. (Citado en la página 22).

[55] Perone, C. S., Silveira, R., and Paula, T. S. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259* (2018). (Citado en la página 22).

[56] Ross, B., Rist, M., Carbonell, G., Cabrera, B., Kurowsky, N., and Wojatzki, M. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118* (2017). (Citado en la página 9).

[57] Sánchez Gómez, C. N. Ingeotec at mex-a3t: Author profiling and aggressiveness analysis in twitter using μtc and evomsa. *OPENAIRE* (2018). (Citado en la página 59).

[58] Sanguinetti, M., Poletto, F., Bosco, C., Patti, V., and Stranisci, M. An italian twitter corpus of hate speech against immigrants. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)* (2018). (Citado en la página 56).

[59] Schmidt, A., and Wiegand, M. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media* (2017), pp. 1–10. (Citado en las páginas 11 y 13).

[60] Sood, S. O., Antin, J., and Churchill, E. Using crowdsourcing to improve profanity detection. In *2012 AAAI Spring Symposium Series* (2012). (Citado en la página 11).

[61] van Aken, B., Risch, J., Krestel, R., and Löser, A. Challenges for toxic comment classification: An in-depth error analysis. *arXiv preprint arXiv:1809.07572* (2018). (Citado en la página 10).

[62] Van Hee, C., Lefever, E., Verhoeven, B., Mennes, J., Desmet, B., De Pauw, G., Daelemans, W., and Hoste, V. Detection and fine-grained classification of cyberbullying events. In *Proceedings of the International Conference Recent Advances in Natural Language Processing* (2015), pp. 672–680. (Citado en la página 13).

[63] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008. (Citado en las páginas ix, 22 y 23).

[64] Warner, W., and Hirschberg, J. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media* (2012), Association for Computational Linguistics, pp. 19–26. (Citado en las páginas 9, 12 y 13).

[65] Waseem, Z., and Hovy, D. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop* (2016), pp. 88–93. (Citado en la página 12).

[66] Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:1902.09666* (2019). (Citado en las páginas 17 y 48).

[67] Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983* (2019). (Citado en las páginas x, 17, 46 y 52).

Apéndice A

Publicaciones

Las investigaciones presentadas en esta tesis han dado como resultado la publicación de los siguientes artículos:

- Muñiz Cuza, Carlos Enrique and De la Peña Sarracén, Gretel Liz and Rosso, Paolo. Attention Mechanism for Aggressive Detection. In Proceedings of the Iberian Languages Evaluation Forum co-located with 35th Conference of the Spanish Society for Natural Language Processing, (IberEval 2018), CEUR Workshop Proceedings. Vol. 2150. Pages 114-118. 2018.
- De la Peña Sarracén, Gretel Liz and Gil Pons, Reynaldo and Muñiz Cuza, Carlos Enrique and Rosso, Paolo. Hate Speech Detection Using Attention-based LSTM. In Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018). 2018.
- De la Peña, Gretel Liz. GL at SemEval-2019 Task 5: Identifying Hateful Tweets with a Deep Learning Approach. In Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019). Pages 416-419. 2019.
- De la Peña, Gretel Liz, and Rosso, Paolo. DeepAnalyzer at SemEval-2019 Task 6: A Deep Learning-based Ensemble Method for Identifying Offensive Tweets. In Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019). Pages 582-586. 2019.
- De la Peña Sarracén, Gretel Liz, and Paolo Rosso. Aggressive Analysis in Twitter using a Combination of Models. In Proceedings of the First

Workshop for Iberian Languages Evaluation Forum (IberLEF 2019), CEUR Workshop Proceedings. 2019.