UNIVERSIDAD METROPOLITANA

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA EN SISTEMAS

SISTEMA DE DETECCIÓN DE LA CARGA OFENSIVA DE COMENTARIOS EN REDES SOCIALES.

Autores:

Maximiliano Casale

Andrés Medina

Tutor: Nicolas Araque

Caracas, febrero 2019

Derechos de autor

Quienes suscriben, en condición de autores del trabajo titulado “Sistema de detección de la carga ofensiva de comentarios en redes sociales.” declaramos que: Cedemos a título gratuito, y en forma pura y simple, ilimitada e irrevocable a la Universidad Metropolitana, los derechos de autor de contenido patrimonial que nos corresponden sobre el presente trabajo. Conforme a lo anterior, esta cesión patrimonial sólo comprenderá el derecho para la Universidad de comunicar públicamente la obra, divulgarla, publicarla o reproducirla en la oportunidad que ella así lo estime conveniente, así como, la de salvaguardar nuestros intereses y derechos que nos corresponden como autores de la obra antes señalada. La Universidad en todo momento deberá indicar que la autoría o creación del trabajo corresponde a nuestra persona, salvo los créditos que se deban hacer al tutor o a cualquier tercero que haya colaborado o fuere hecho posible la realización de la presente obra.

Autor: Maximiliano Casale
C.I.: V-25.217.545

Autor: Andrés Medina
C.I.: V-23.693.505

En la ciudad de Caracas, a los 22 días del mes de febrero del año 2019.

Aprobación

Considero que el Trabajo Final titulado

SISTEMA DE DETECCIÓN DE LA CARGA OFENSIVA DE COMENTARIOS EN REDES SOCIALES.

Elaborado por los ciudadanos

MAXIMILIANO CASALE

ANDRES MEDINA

Para optar al título de

INGENIERO DE SISTEMA

Reúne los requisitos exigidos por la Escuela de Ingeniería de Sistemas de la Universidad Metropolitana, y tiene méritos suficientes como para ser sometido a la presentación y evaluación exhaustiva por parte del jurado examinador que se designe.

En la ciudad de Caracas, a los 22 días del mes de febrero del año 2019.

_____

Tutor: Nicolas Araque

ACTA DE VEREDICTO

Nosotros, los abajo firmantes, constituidos como jurado examinador y reunidos en Caracas, el día ____ de _____ de 2019, con el propósito de evaluar el Trabajo de Grado titulado:

SISTEMA DE DETECCIÓN DE LA CARGA OFENSIVA DE COMENTARIOS EN REDES SOCIALES.

Presentado por los alumnos:
Maximiliano Casale
Carnet N°: 20161120297
Cédula de Identidad N°: 25.217.545

Andrés Medina
Carnet N°: 20141110086
Cédula de Identidad N°: 23.693.505
Para optar al título de: INGENIERO DE SISTEMAS

Emitimos el siguiente veredicto: Aprobado Reprobado

OBSERVACIONES:

_____

_____

Jurado Presidente Tutor Académico Jurado

_____ Directora de Escuela

Agradecimientos: Maximiliano

Gracias a todo el grupo que se tomó el tiempo y la dedicación de calificar los comentarios, sin ustedes esto no hubiese sido posible.

Gracias a mi esposa Isabella de Santis por su constante motivación y apoyo. Te amo mi vida.

Gracias a mi padre Cesar Casale y mi madre Laura Lara, por darme la oportunidad de hacer esta carrera y permitirme dedicarme a mi pasión. ¡Los amo!

Gracias a Abraham Chang, Gabriele Troncone, Luciano Pinedo, Gabriel de Lellis por su amistad, por todo el apoyo y los consejos durante toda la carrera.

Gracias a mis compañeros de trabajo en Aether y CXN por todos sus consejos en el desarrollo del código.

Gracias al profesor Rafael Matienzo por brindarnos información tan valiosa para la investigación y por su motivación.

Gracias al profesor Christian Guillen por todas las críticas constructivas sobre la redacción y el saber escribir, me enseñó que para escribir también hay que pensar de forma crítica.

Gracias a ambos por todas sus enseñanzas durante la carrera.

Gracias a mi compañero Andrés por acompañarme durante esta gran experiencia.

Gracias a nuestro tutor Nicolás por todos los conocimientos que nos aportó durante la investigación y el desarrollo de la red neuronal.

Agradecimientos: Andrés

El hecho de culminar esta etapa de mi vida, me obliga a recordar quienes me acompañaron en el camino, en primer lugar, a mis padres quienes siempre me alentaron a seguir adelante y no rendirme, también a mis amigos que siempre estuvieron para mí.

Gracias a mi padre Carlos Andrés Medina y a mi madre Nayarith Grau por apoyarme incondicionalmente en mis decisiones y estar presentes en todo momento. Los Amo.

Gracias a mis amigos, por estar conmigo durante la universidad y compartirme su visión sobre diferentes temas, los quiero.

Gracias a mi novia, por apoyarme en todo momento y acompañarme durante toda la carrera, Te Amo.

Gracias a mis compañeros de carrera, que me ayudaron en todo momento y con quienes compartí buenos momentos, los aprecio un montón.

Gracias a los profesores Rafael Matienzo, Christian Guillen, Nicolas Araque y Germán Biaggini, quienes fueron fundamentales en mi formación como ingeniero de sistema.

Gracias a mi compañero Maximiliano por ser mi compañero de tesis y estar presente durante esta gran etapa.

Gracias a nuestro tutor Nicolás, quien nos brindó su ayuda y disposición durante la investigación.

Contenido

Lista de tablas y figuras

Tablas

Figuras

Resumen

SISTEMA DE DETECCIÓN DE LA CARGA OFENSIVA DE COMENTARIOS DE REDES SOCIALES

Autores: Maximiliano Casale
Andrés Medina
Tutor: Nicolas Araque Caracas, febrero 2019

El presente trabajo de investigación tuvo como objetivo implementar un sistema de detección de la carga ofensiva de comentarios en redes sociales. Para ello se seleccionó la red social Twitter para extraer los comentarios. Dichos comentarios pasaron por un proceso de calificación donde cuarenta personas calificaron los comentarios como ofensivos o no ofensivos. Luego de esto, se elaboró una red neuronal recurrente con memoria de largo y corto plazo que detecta con aproximadamente un 80% de precisión la probabilidad de que un comentario sea calificado como ofensivo y se implementó en una aplicación web desarrollada utilizando los *frameworks Angular* y *Django.* Con esto se concluyó que las redes neuronales son una herramienta capacitada para aprender patrones en el texto y clasificarlos adecuadamente bajo un previo aprendizaje de cómo debería hacerse.

Palabras clave: Redes Neuronales, Long Short Term Memory, Django, Angular, Redes sociales, Ofensivo.

Introducción

Actualmente, las redes sociales se han vuelto parte del día a día de la mayoría de las personas. Esto ha logrado que los individuos más populares en las mismas generen un impacto importante en la sociedad, ya que generan grandes oleadas de comentarios a favor o en contra en cada una de sus publicaciones. Por tal razón, las empresas de redes sociales han visto la necesidad de generar un sistema que permita monitorear el comportamiento de los usuarios en los comentarios. Debido a que ha surgido una creciente tendencia de crear polémicas mediante el uso de insultos o material ofensivo, la calidad y satisfacción de los clientes que se entretienen haciendo debates en la sección de comentarios de cada publicación ha ido desmejorando.

Las redes neuronales son una herramienta que ayuda a solucionar la problemática mencionada anteriormente debido a que según Gurney (1997), son usadas típicamente en problemas que pueden ser planteados en términos de clasificación o predicción, como es el caso del reconocimiento de texto en lenguaje natural.

El objetivo de este trabajo fue implementar un sistema de detección de la carga ofensiva de comentarios y mensajes en español en redes sociales mediante el uso de redes neuronales con la intención de ofrecer una herramienta que permita identificar los comentarios ofensivos.

El trabajo fue realizado en cinco etapas. La primera etapa consistió en la revisión de la literatura relacionada a las redes neuronales. Seguido de esto se seleccionó la población y muestra de los datos necesarios para el entrenamiento de la red neuronal. A partir de esto, se procedió a extraer de la red social *Twitter* los comentarios que luego fueron clasificados por un grupo de personas designado. La cuarta etapa consistió en realizar el procesamiento de los datos a través de la red neuronal con el fin de que la misma aprendiera el comportamiento de dichos comentarios. Finalmente, se elaboró un software

donde se puede apreciar el comportamiento de la red neuronal cuando se le introducen comentarios nuevos.

El presente tomo está compuesto por los siguientes capítulos:

- Capítulo I, Tema de Estudio. Como introducción del tomo, contiene el planteamiento del problema, la presentación del objetivo general, los objetivos específicos y las limitaciones del trabajo.
- Capítulo II, Marco de referencia. En este se presenta el material teórico que sirve como base para este trabajo. Se desarrolla información sobre redes neuronales, sus tipos, características, parámetros y métodos de optimización y aprendizaje. También, se expone el contexto de las redes sociales sobre el que se fundamenta el trabajo y finalmente la teoría sobre la arquitectura de software utilizada.
- Capítulo III, Marco Metodológico. Detalla la metodología y las etapas con las cuales se realizó el trabajo.
- Capítulo IV, Resultados. En este capítulo se presentan los resultados obtenidos para cada objetivo específico.
- Capítulo V, Conclusiones y Recomendaciones. En el capítulo final se presentan las conclusiones obtenidas de la investigación realizada. Así mismo, se presentan recomendaciones para trabajos futuros.

Capítulo I. Tema de estudio

I.1 Planteamiento del problema

Las redes sociales se han vuelto parte del día a día de la mayoría de las personas, haciendo que los individuos más populares en las mismas generen un impacto importante en la sociedad, generando grandes oleadas de comentarios a favor o en contra de cada publicación. Por tal razón, las empresas de redes sociales han visto la necesidad de generar un sistema que permita monitorear el comportamiento de los usuarios en los comentarios debido a que ha surgido una creciente tendencia de crear polémicas mediante el uso de insultos o material ofensivo, desmejorando la calidad y satisfacción de los clientes que se entretienen haciendo debates en la sección de comentarios de cada publicación.

Aquellos usuarios que tienen una personalidad violenta o tóxica son un problema para las empresas de redes sociales. Según el psicólogo argentino Stamateas (2008) son aquellas personas “que pareciera encuentran placer en hacernos difícil la convivencia o nuestro trabajo.” (p.49). Estos usuarios ven la necesidad de resaltar en la sociedad mediante el uso de comentarios ofensivos. Debido a esto, algunas empresas de software han optado por la creación de sistemas de prevención y detección de comentarios tóxicos para mejorar la experiencia y la calidad de servicio que reciben los usuarios.

Dentro de la industria de desarrollo de software, Google está creando una herramienta de inteligencia artificial para tratar de combatir el acoso en línea y poder determinar si una conversación se está tornando agresiva. Sin embargo, está en una fase de desarrollo temprana y únicamente está disponible en el idioma inglés.

En el caso particular de Instagram, no existe un control en cuanto a la toxicidad de los comentarios escritos en español, por lo cual una publicación se puede llegar a minar de comentarios tóxicos, evitando que se pueda disfrutar de un servicio de calidad, disminuyendo los usuarios activos y por ende las ganancias por publicidad y venta de datos en la aplicación.

Deseando mejorar la calidad de servicio de las redes sociales y considerando lo anteriormente expuesto, en el presente trabajo de investigación se desarrolló una herramienta que permite detectar y clasificar los comentarios ofensivos en español, con el fin de identificar de forma automatizada aquellos mensajes agresivos y brindar a las empresas de redes sociales una alternativa para lidiar con los usuarios de personalidad violenta de habla hispana.

I.2 Objetivos del estudio

General:

- Implementar un sistema de detección de la carga ofensiva de comentarios y mensajes en español en redes sociales.

Específicos:

- Definir los criterios para evaluar la carga ofensiva de los comentarios.
- Calificar los datos extraídos de una red social seleccionada, a través de los criterios definidos.
- Diseñar el software basado en la Red Neuronal que detecta la carga ofensiva de comentarios en una red social
- Desarrollar el software bajo los *frameworks Django* y *Angular*

I.3 Limitaciones

En las empresas que manejan grandes volúmenes de datos se ha hecho necesario el uso de herramientas que automaticen la detección de patrones entre los registros. Para este requerimiento en específico se han desarrollado diferentes técnicas de *machine learning* cuya eficacia depende del escenario en que se pone a prueba y la arquitectura que se utilice. Regularmente, para casos de detección o clasificación de patrones se utilizan las denominadas redes neuronales cuya intención es análoga a simular el funcionamiento de las neuronas del cerebro humano, lo que computacionalmente corresponde a "una red interconectada paralelamente con elementos simples que siguen una interacción jerárquica" (Kohonen, 1989, p. 251).

Dentro del ámbito de redes neuronales existen diferentes modelos de arquitecturas de red, los cuales se destacan en diferentes categorías. Por ejemplo, para el análisis de cadenas de caracteres, el modelo *Long Short-Term Memory* (en adelante por sus siglas en inglés: LSTM) es una red neuronal recurrente con una estructura de celdas con compuertas de entrada y salida que permiten recordar y olvidar datos de contexto en series (Gers, Schmidhuber, Cummins, 1999) y es conocida por su capacidad de clasificación y compresión de texto en lenguaje natural y lenguaje sin contexto (Gers, Schmidhuber, s.f.).

Las redes neuronales recurrentes pueden contener ciclos entre los nodos, formando una especie de "memoria" que permite almacenar datos que tengan influencia en la respuesta de la red (Graves, s.f., p.18-19). El modelo LSTM ha demostrado exitosamente que puede resolver problemas de la vida real (como detectar patrones en texto sin contexto) que siguen siendo imposibles para cualquier otro modelo de red neuronal (Graves, s.f., p.32), motivo por el cual se utilizó para la investigación.

Para la introducción de la data en la red, investigaciones anteriores en el área han demostrado con éxito que el uso de N-gramas ha funcionado mucho mejor que especificaciones de la sintaxis del lenguaje o palabras y oraciones embebidas “debido a la mayor robustez que exhiben los N-gramas por las variaciones de ortografía que son muy comunes en discusiones en línea” (Dixon, Thain, Wulczyn, 2017). Por esta razón, para este trabajo de investigación, la red neuronal se entrenó usando registros de texto representados por secuencias de palabras de longitud N, llamadas N-gramas (Bruguier, Damavandi, Kumar, Shazeer, 2016), con la finalidad de detectar intenciones ofensivas en los mensajes o comentarios en español recuperados de la red social *Instagram*, mediante el modelo de red neuronal LSTM mencionado anteriormente.

Para el desarrollo de software en *machine learning*, una de las mejores alternativas es *Python* (Rebollo, M. 2018). Este lenguaje es conocido principalmente por sus librerías de acceso público y su fácil y rápida implementación (Python,2018). El beneficio más grande considerado es la velocidad de desarrollo gracias a las librerías que ya han sido desarrolladas e incluyen la mayoría de las funciones necesarias para llevar a cabo un software de inteligencia artificial.

Entre las librerías más populares están: *TensorFlow* (Abadi et al, 2015) (desarrollada por *Google* como librería de *machine learning*) y *Keras* (Keras, 2015) (desarrollada por François Challet como *frontend* para *deep learning*). Ambas librerías tienen las herramientas para realizar el modelo de arquitectura planteado y fueron utilizadas en conjunto durante la investigación.

Para la clasificación de la carga ofensiva de los comentarios, se tomó en consideración el lenguaje de los mismos. Definiendo como lenguaje "la manera de expresarse" (Real Academia Española, 2001). De la misma manera, una muestra de usuarios calificó los comentarios que consideraron ofensivos desde su perspectiva, tomando en cuenta las categorías del lenguaje soez (Blasfemia, Amenazas, Insultos, Dobles sentidos, interjección,

Disfemismos, Ironía, sarcasmo, tabúes, entre otros) en los mensajes. Dixon, Thain y Wulczyn (2017) recomiendan que la muestra de usuarios sea de alrededor de 40 personas, de forma que se pueda tener una base sólida de calificaciones por cada comentario. Debido a esto se utilizaron exactamente 40 personas cuyos correos electrónicos de contacto se encuentran en el Apéndice A. Estos comentarios calificados se utilizaron para entrenar la red neuronal de forma que esta sepa diferenciar entre comentarios ofensivos y no ofensivos y calcular la carga ofensiva de nuevos comentarios.

Para el desarrollo de la interfaz gráfica que los usuarios utilizaron para clasificar los distintos comentarios según su carga ofensiva, se utilizó el *framework Angular 7*, ya que éste es de código abierto, fácil implementación y altamente escalable (Angular, s.f.). Además, ofrece una tecnología capaz de mantener la aplicación en una sola página ofreciendo respuestas más rápidas a las interacciones del usuario. Para la integración con la red neuronal se utilizó *Django*, un *framework* web para *Python* compatible con *Angular 7*.

Este proyecto se elaboró durante el período de diciembre 2018 - febrero 2019 y su finalidad es principalmente desarrollar una red neuronal cuya arquitectura retorna como resultado un número entre 0 y 1, el cual corresponde a la probabilidad de que un comentario sea considerado como ofensivo y será clasificado según el resultado como “Poco probable de ser ofensivo”, “relativamente neutro” y “Bastante probable de ser ofensivo”.

Capítulo II. Marco de referencia

Este capítulo está compuesto de tres partes. Una primera parte donde se expone la teoría relacionada a las redes neuronales artificiales como herramienta de *machine learning*, sus propiedades y sus características fundamentales. En la segunda parte se expone sobre las redes sociales y el contexto utilizado para la investigación. Finalmente, en la tercera parte, se presenta la teoría sobre la arquitectura de software utilizada.

II.1 Redes Neuronales

Las redes neuronales se establecen a partir de nuestro entendimiento de las neuronas reales, según Burgos (2002) una neurona es una célula que tiene la capacidad de recibir y enviar estímulos mediante impulsos electroquímicos, estas generan una enorme red de neuronas a través de sinapsis. Cuando en una neurona se dispara un potencial de acción, este se va propagando por el axón y por medio de la sinapsis el potencial de acción puede transmitirse o inhibirse en otras neuronas.

II.1.1Red neuronal artificial

Las redes neuronales artificiales intentan imitar el comportamiento del cerebro. En 2009, Olsons y Hergenhan, Citado por Escobar (2014), sobre la definición de Red neuronal menciona:

> Es un modelo por computadora que simula la interconexión y la actividad de las células neuronales. La tarea básica de una simulación por computadora de una red neuronal es definir un conjunto de unidades neuronales artificiales o nodos, así como sus potenciales interconexiones y el peso de cada conexión. Después, a partir de una función de activación se determina si la activación de una neurona pre sináptica produce la activación de la neurona post sináptica. Posteriormente se añaden reglas bajo las cuales ocurren

> cambios en los pesos de las interconexiones bajo las cuales la red "aprende". Finalmente, la red se entrena y se observa cómo cambia a partir de la experiencia o se adapta al ambiente. (Pag,26)

La figura 1, ilustra un modelo simple de una red neuronal. Una red neuronal tiene una capa de entrada de datos, una de salida y al menos una capa intermedia denominada capa oculta.

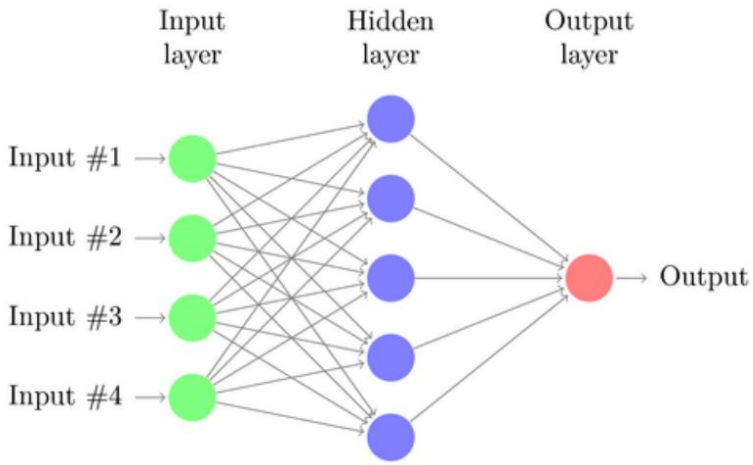

Figura 1. Topología de una red neuronal artificial

Fuente: Camacho (2016, pp. 16)

II.1.2 Neurona artificial o nodo

Las neuronas son la unidad principal de los modelos de redes neuronales, según Mcculloch y Pitts (1943) describen que en el primer modelo de red neuronal se buscó generar una unidad que compartiera características con las neuronas biológicas, con entradas excitatorias y entradas inhibitorias , y que tuviesen un estado binario que fuese activado por exceso de entradas excitatorias.

Las neuronas artificiales son la base fundamental de la red. Según Escobar (2014) son la unidad principal de toda red neuronal y simula el funcionamiento de una neurona. Además de esto, una unidad puede estar

compuesta de un conjunto de neuronas, estas neuronas reciben un valor de entrada y generan un valor de salida. Este valor de salida se conoce como valor de activación, son generados mediante funciones y simula el disparo de la neurona.

II.1.2.1 Funciones de activación

Una neurona biológica puede estar excitada o inactiva, teniendo así un estado de activación. Según Basualdo y Ruiz (2001) las neuronas artificiales buscan replicar dicho comportamiento teniendo diferentes estados de activación. Mediante el uso de funciones matemáticas se calcula el estado de actividad de la neurona, transformando el vector de pesos y el *bias* que recibe en un valor de activación. Este valor es un indicador del estado de la neurona ya que puede estar totalmente inactiva o activa dependiendo de su valor. Las diferentes funciones de activación se detallan con mayor profundidad en el apéndice B.

II.1.2.2 Pesos

Cada neurona de una red neuronal recibe valores de entradas, cada una de estas entradas posee su propio peso, Según Martínez (2007) cada peso va a influir en la importancia de dicha entrada dentro de la función de activación. Esto refleja que en una red neuronal todas las entradas poseen una importancia distinta y que unas tienen mayor efecto en el procesamiento de la neurona para producir la respuesta neuronal. Estos pesos son coeficientes que son modificados durante el entrenamiento de la red neuronal con la finalidad de generar una respuesta más certera.

II.1.2.3 *Sesgo*

El Sesgo o Bias según Hoff y Widrow (1960) es un parámetro que proporciona un grado de libertad adicional al modelo, priorizando una convergencia más rápida de la red.

II.1.2.4 Dropout

La capa de dropout (exclusión por su traducción) según Bonaccorso (2017) es una capa que se coloca para evitar el ajuste excesivo de la red al establecer aleatoriamente un número fijo de elementos de entrada en 0. Está capa se adopta durante la fase de entrenamiento y se desactiva durante la fase de prueba y validación. Las redes que poseen capas de dropout pueden tener velocidades de aprendizaje más altas, moviéndose en diferentes direcciones en la superficie de pérdida y excluir todas las áreas de superficie de error que no conducen a una optimización consistente.

II.1.3 Redes neuronales Unidireccionales

Las redes neuronales artificiales de topología unidireccional según Vásquez (2014) son el modelo más simple, están estructuradas por al menos 2 capas (entrada y salida), se caracterizan por ejecutar el procesamiento de datos en una sola dirección, ingresando la información por los nodos de entrada hacia los nodos de salida, sin generar bucles entre las neuronas.

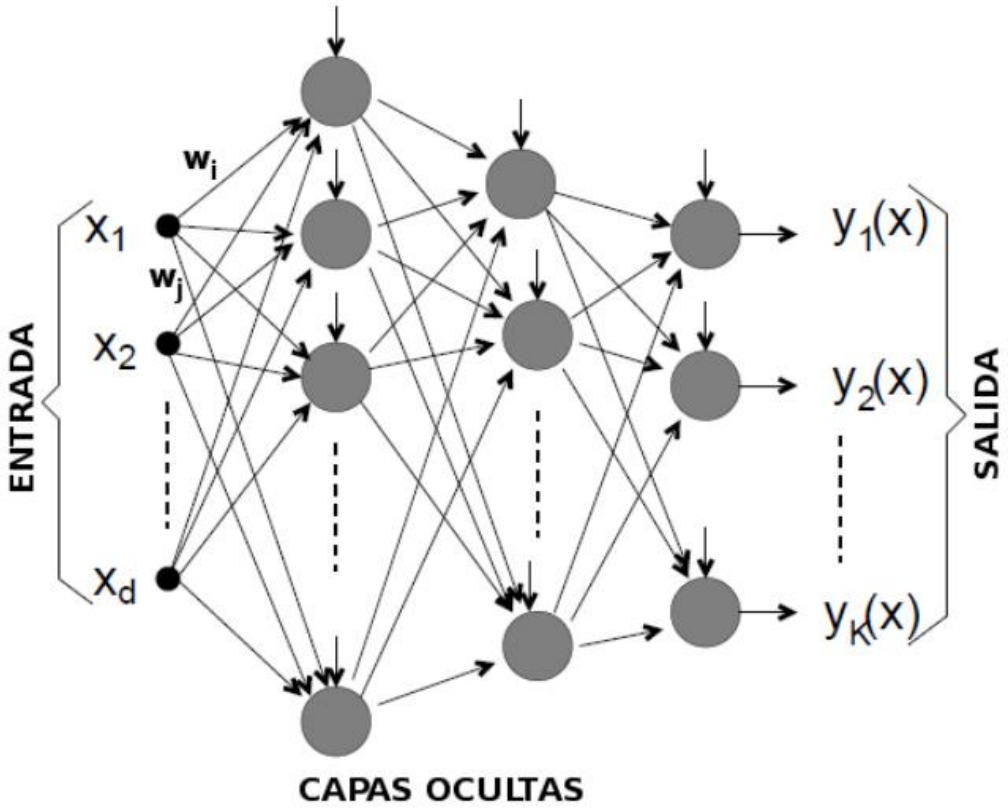

Figura 2. Diagrama de la estructura típica de una red neuronal artificial unidireccional

Autor: Vásquez (2014, pp. 6)

II.1.4 Redes neuronales retroalimentadas

Las *recurrent neural networks* (en adelante por su traducción del inglés: RNN*)* según García (2018) estas son similares en funcionamiento a las redes neuronales unidireccionales solo que añadiendo un nivel más de complejidad. Las RNN son redes neuronales dinámicas en oposición a las redes neuronales tradicional, esto significa que las salidas de la red dependen de la entrada de la misma, de entradas y salidas previas y también de estados ocultos de la red.

Según García (2018) para el aprendizaje de esta red se utiliza el algoritmo de propagación hacia atrás, buscando desdoblar la red a través de tiempo lo que implica que la sucesión de estados de la red estará en función

del tiempo, logrando obtener una red neuronal en la que cada capa corresponde a un paso temporal de la red original. Sin embargo, este tipo de red en su estado simple no es comúnmente usada debido a que posee limitaciones con la desaparición de gradiente.

En la figura 3 se muestra el esquema de una red neuronal retroalimentada, teniendo una capa de entrada, una capa oculta y una capa de salida.

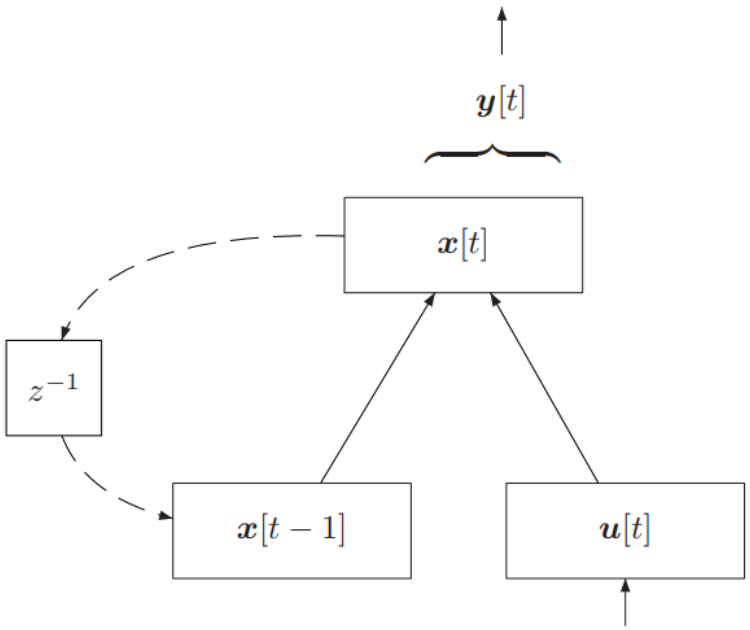

Figura 3. Esquema de la red totalmente recurrente de primer orden
Fuente: Williams y Zipser (1989, pp 270)

II.1.4.1 Limitaciones

Las redes neuronales retroalimentadas tienen limitaciones cuando se busca tener memorias largas o cuando se busca conectar relaciones de datos a distancias significativas en el tiempo, lo cual es necesario para generar modelos que comprendan el lenguaje y narrativa o correlacionar eventos en el mercado de valores. Cuando se aumentan las capas ocultas según Hochreiter

(1998) se genera un problema conocido como la desaparición del gradiente. Este se refiere a que mientras más capas se agreguen a la red y que esta se haga más profunda, más posibilidades se tendrá de que los gradientes de propagación hacia atrás se vayan acumulando y se reduzcan a cero.

En la figura 4 se puede observar la función de activación Sigmoide (en color azul) y los valores del gradiente (en color naranja), cuando los valores de la función sigmoide se mantienen entre 0 y 1, el gradiente tiene valores inferiores a 0.25, por lo cual al multiplicar el acumulado de gradiente utilizando propagación hacia atrás, el valor del gradiente va a tender a 0, generando una desaparición del mismo y por ende pérdida de información.

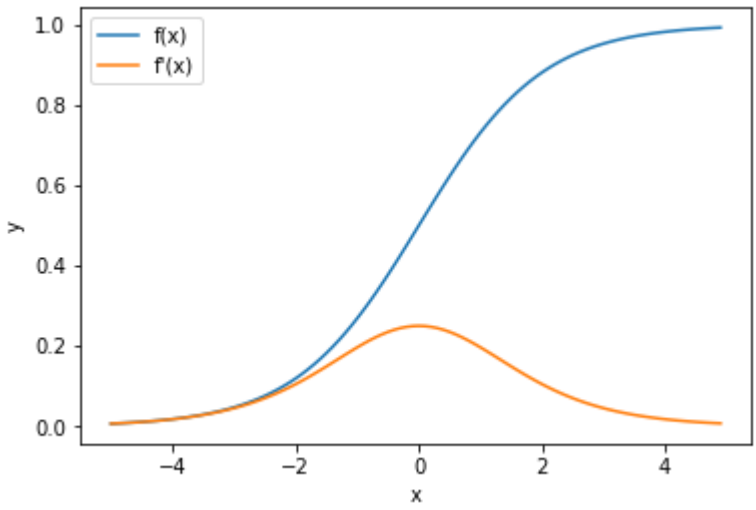

Figura 4. Función Sigmoide (azul) y gradiente de la función sigmoide (naranja).

Fuente: Adventures in Machine Learning (2017)

Finalmente, Hochreirte (1998), concluye su trabajo explicando que no existe una solución efectiva para la desaparición del gradiente en redes recurrentes en tiempos razonables y menciona que los modelos *Long Short-Term Memory* (en adelante por sus siglas en inglés: LSTM) para redes neuronales recurrentes son una solución más eficiente.

II.1.5 LTSM

Las LSTM son un tipo de red neuronal recurrente que según Camacho (2016) fueron diseñadas para modelar secuencias temporales de mayor precisión que las tradicionales, la diferencia con las redes neuronales recurrentes está en la celda LSTM que es una unidad lógica diseñada para reducir el problema de la desaparición del gradiente, por lo cual hace que sean útiles para tareas de memoria a largo plazo, siendo fundamental para las predicciones de secuencias de texto.

La estructura según Hochreiter y Schmidhuber (1997) se basa en un bloque lógico que contiene celdas de memoria, una compuerta de entrada y una compuerta de salida, estas compuertas son unidades multiplicativas con activación continua (en el intervalo unidad) y son compartidas por las celdas de un mismo bloque de memoria. Cada celda contiene una unidad lineal con conexión recurrente local llamada carrusel de error constante (de ahora llamada CEC) y la activación del CEC se conoce como el estado de la celda.

Para la tarea de predicción se emplean redes de tipo Long Short Term Memory (LSTM), siendo éstas un caso especial de redes neuronales tradicionales. Este tipo especial de redes neuronales son ampliamente utilizadas en problemas de predicción en series temporales debido a que su diseño permite recordar la información durante largos períodos y facilita la tarea de hacer estimaciones futuras empleando períodos de registros históricos. A diferencia de las redes neuronales tradicionales, en lugar de poseer neuronas de manera clásica, las redes LSTM tienen como neuronas bloques de memoria que están conectados a través de capas. Estos bloques de memoria facilitan la tarea de recordar valores para largos o cortos períodos de tiempo, por lo tanto, el valor almacenado no es reemplazado (al menos a

corto plazo) de forma iterativa en el tiempo, y el término de gradiente no tiende a desaparecer cuando se aplica la retropropagación (back-propagation, método de aprendizaje) durante el proceso de entrenamiento, tal y como acontece en el uso de las redes neuronales clásicas. (González et al, 2017, pág. 2)

Según Gers et al (2000) la limitación de la red LSTM inicialmente era que presentaba problemas cuando se procesaban secuencias de longitud arbitrariamente larga de forma continua, el modelo se vuelve inestable debido a que en algunas circunstancias el estado de los CEC crece indefinidamente, este problema se soluciona incorporando una tercera compuerta al bloque de memoria, llamada compuerta de olvido.

“La compuerta de olvido puede rebajar e incluso anular el estado interno de la celda, esto es, la activación del CEC, cuando su contenido caducó. Estas compuertas permiten que la red LSTM pueda procesar establemente secuencias de longitud arbitrariamente larga” (Gers et al, 2000, p)

En la figura 5 se muestra el esquema de una célula LSTM donde se detalla la compuerta de entrada, compuerta de salida, compuerta de olvido y las celdas de memoria o estado interno.

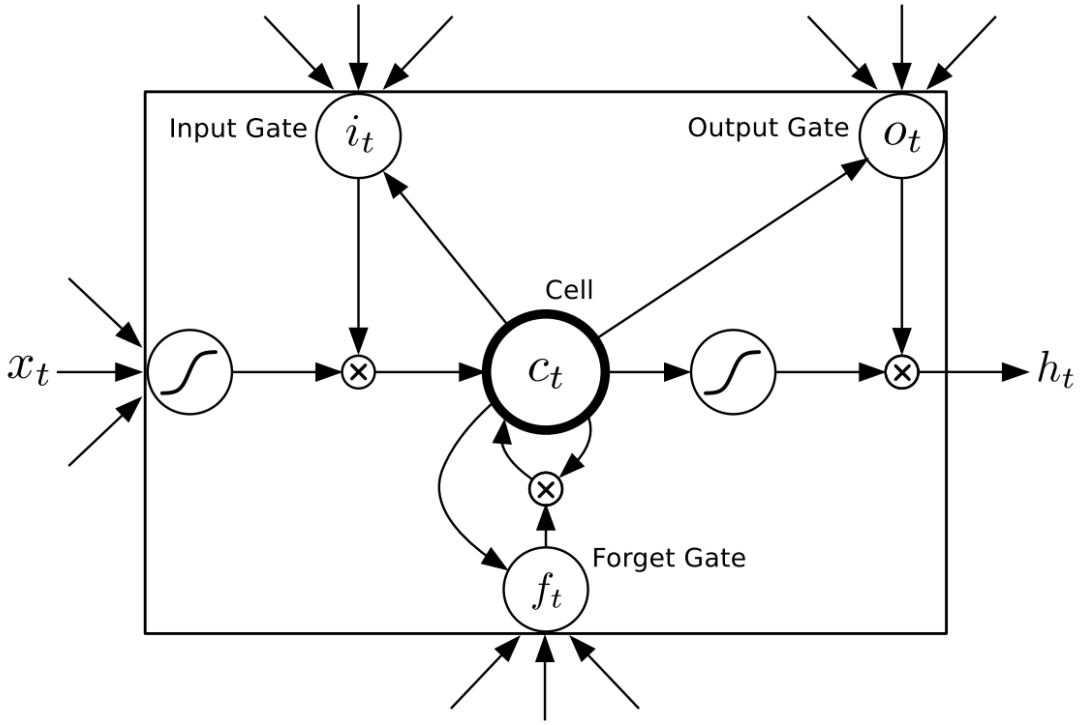

Figura 5. Esquema de una celda LSTM

Fuente: Camacho (2016, pp. 51).

II.1.5.1 LSTM Bidireccional

Las LSTM Bidireccionales son una manera de estructurar las redes neuronales, según Cui, Ke y Wang (2018) la idea de utilizar una red bidireccional es procesar los datos de secuencia en direcciones hacia adelante y hacia atrás con dos capas ocultas separadas (como se muestra en la figura 6), permitiendo que la capa de salida pueda obtener información de estados pasados y futuros simultáneamente, ya que ambas capas ocultas están conectadas a la capa de salida, esta estructura mejora sustancialmente los resultados en el campo del reconocimiento de textos, debido a que las LSTM tradicionales solo pueden preservar información de estados pasados.

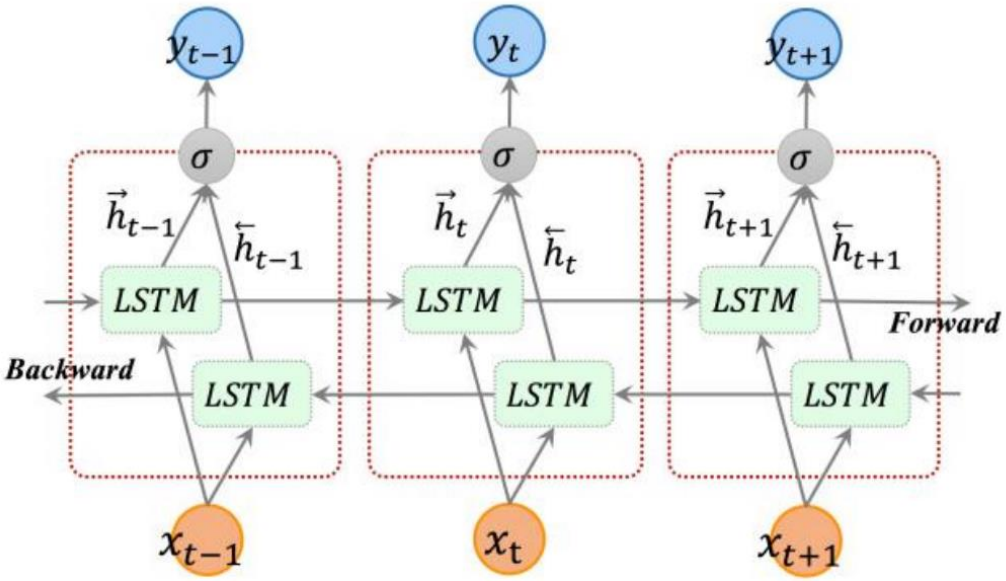

Figura 6: Arquitectura de la red LSTM bidireccional con tres capas consecutivas.

Fuente: Cui, Ke y Wang (2018, pp. 3)

II.1.6 Tokenizing

Para poder trabajar en una red neuronal con textos según Bonaccorso (2017) es necesario dividirlo en átomos, normalmente definidas como tokens (fichas por su traducción). Tal proceso es simple, sin embargo, se puede generar diferentes estrategias para resolver problemas particulares, mediante esto se permite vectorizar un texto en una secuencia de enteros (cada entero es el índice de un token en un diccionario). Ejemplo: si se tiene el texto “la UNIMET es grande, la UNIMET somos todos”, y se desea *tokenizar*, se buscará el índice de cada palabra, para generar el vector. Suponiendo que los índices son: 1=”la”, 2= “UNIMET”, 3= “grande”, 4= “es”, 5 = “somos”, 6= “todos”, el resultado de tokenizar será [1,2,4,3,1,2,5,6].

II.1.7 Embedding

La capa de Embedding (Embebido por su traducción) según Gal y Ghahramani (2016) es la primera capa del modelo y en esta se transforman los enteros positivos en vectores densos de tamaño fijo. Para el embebido de textos según Dahou, et al (2018) se representan las palabras individuales como vectores en un espacio vectorial de un tamaño seleccionado. Según Sánchez et al (2017) una de las aplicaciones más comunes del embebido es en el procesado de texto, asignándoles a cada palabra un vector de entero único, transformando los textos en vectores.

II.1.8 Pooling

La capa de Pooling (Agrupación por su traducción) según Scherer, Muller y Behnke (2010) sirve para reducir progresivamente el tamaño espacial de la representación de los datos dentro de la red para disminuir la cantidad de parámetro, acortando la cantidad de cálculos y controlar el sobreajuste. Esta capa se suele insertar entre capas de diferentes dimensiones. Adicionalmente, la capa de *Pooling* opera independientemente en cada sección de profundidad de la entrada y la redimensiona espacialmente.

II.1.9 Entrenamiento y optimización.

Dada una estructura de red neuronal, se procede al entrenamiento de la misma con el objetivo de que sea capaz de reproducir el comportamiento subyacente en los datos aportados. Según Vivas (2014) la estructura de una red no está completa si no podemos garantizar que funcione correctamente.

Al igual que nuestro cerebro, el comportamiento va mejorando a medida que el aprendizaje o entrenamiento sea el correcto. El entrenamiento de una red neuronal artificial es el proceso por el cual se ajustan sus pesos sinápticos, con la finalidad de obtener un resultado óptimo.

La clasificación de las redes neuronales se basa usualmente en sus algoritmos de entrenamiento y si son supervisadas o no supervisadas. Según Escobar (2014) las redes supervisadas son las más comunes y su entrenamiento se basa en generar diversos pesos asociados con diferentes valores de activación de las unidades de salida, compararlo con una salida esperada (error) y conforme a una regla se ajustan los pesos buscando disminuir el error. En contraparte, las redes no supervisadas presentan diversos patrones de entrenamiento y se observan los cambios a través de la experiencia sin compararlos con una salida esperada.

Para la optimización de la red neuronal, se emplean métodos de ajuste a los parámetros de la red (pesos de las conexiones y *bias* de las neuronas) teniendo en cuenta los valores resultantes de la función de costo. El método de ajuste comúnmente usado es el Descenso de gradiente en conjunto con la propagación hacia atrás.

II.1.9.1 Función de costo.

Según García (2018) Para todo tipo de datos de entradas y pesos de la red neuronal hay una función de costo o error que se encarga de medir el rendimiento de la red respecto al valor de salida esperado. El objetivo es encontrar la combinación de parámetros (pesos y bias) de tal manera que se minimice el error. Algunos algoritmos utilizados para encontrar el error mínimo son Backpropagation y el descenso por gradiente.

II.1.9.1.1 Binary Cross Entropy

La función de costo Binary Cross Entropy (entropía cruzada binaria por su traducción) según Botev, Kroese y Taimre (2006) es una técnica para la estimación y optimización de eventos. Está mide el rendimiento de un modelo de clasificación cuyo resultado es una probabilidad entre 0 y 1. La pérdida de

entropía cruzada aumenta a medida que la probabilidad predicha difiere de la etiqueta real.

II.1.9.2 Backpropagation.

La propagación hacia atrás (*Backpropagation*) es un método de cálculo del gradiente en redes neuronales. Según Castrillón, Perlaza, Van Schoonhoven, et al (2001) el método consta de dos fases propagación y adaptación. En la primera fase luego de aplicar una entrada a la red, está se va propagando a través de las capas hasta generar una señal de salida, esta se compara con la salida deseada y se estima el error. En la segunda fase, las salidas erróneas se propagan capa por capa en dirección contraria, pasando por cada una de las neuronas que hayan participado en el error ajustando sus pesos buscando los parámetros que minimizan la función de costo.

Según García (2018) este es el método de aprendizaje más usado, el principio se basa en tomar el valor de salida y compararlo con el deseado, posteriormente se mide el error asociado con cada neurona de la capa anterior y se ajustan los pesos de las neuronas para corregir el error, repitiendo con cada capa hasta alcanzar la capa de entrada. De esta manera se garantiza que se recorre la red neuronal hacia atrás de forma que se ajustan todos los pesos y sesgos de la red.

Según Gómez y González (1998) las funciones de activación de la neurona más utilizada en el algoritmo de retropropagación que cumplan con la condición de ser derivable y creciente, son las sigmoides (tales como función logística y la tangente hiperbólica).

En la Figura 7 se muestra la estructura paso a paso del algoritmo de la propagación hacia atrás, el primer paso (color azul) se muestra la propagación hacia la salida, en el segundo paso (color rojo) se propagan las señales de error y posteriormente en el paso 3 (color verde) se calculan los gradientes para finalmente actualizar los parámetros en el paso 4.

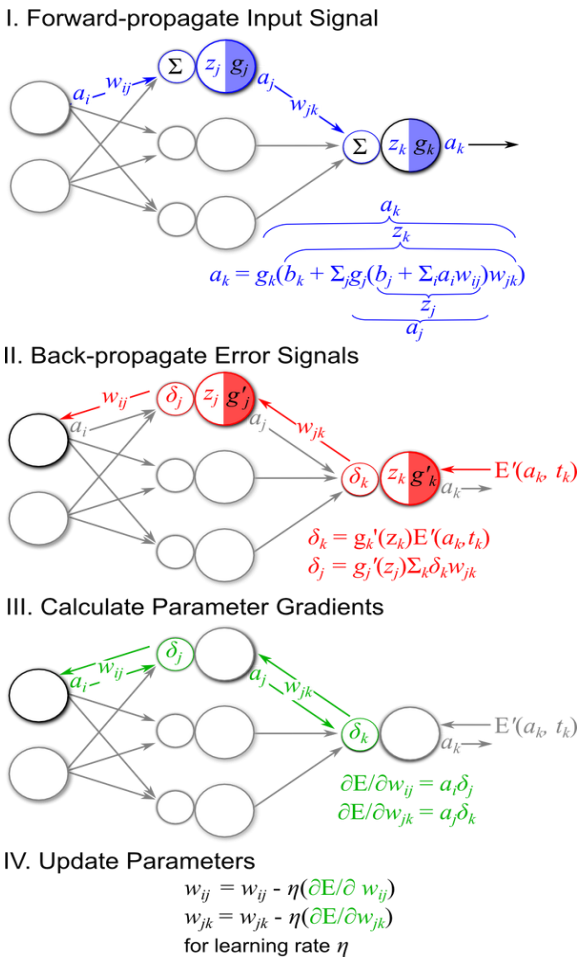

Figura 7. Estructura de una red neuronal Backpropagation

Fuente: Castrillón, Perlaza, Van Schoonhove y Owen (2001, pp. 143)

II.1.9.3 Descenso por gradiente.

La optimización de las redes neuronales es importante para reducir el error de la salida con la finalidad de tener mejores resultados. Según Escobar (2014) una de las formas más comunes de tratar de disminuir el error en las redes neuronales es usando el algoritmo de descenso de gradiente en conjunto con la propagación hacia atrás. El algoritmo de descenso de

gradiente consiste según Barrera (s.f.) en calcular la derivada y el gradiente de la función de error con respecto a cada uno de los parámetros del modelo. Está información del gradiente nos indicará la dirección hacia donde la función de error tendrá un mayor crecimiento. Como se desea minimizar el error, se toma la dirección negativa del gradiente para tener un decrecimiento del error. Posteriormente, la variación de los parámetros será proporcional al gradiente de la función de error y se repite el procedimiento hasta encontrar el mínimo del error. Este procedimiento se ejecuta inicialmente en las neuronas de salida y aplicando el método de propagación hacia atrás, se ejecutará este algoritmo en las otras neuronas que compartan el error para optimizar la red.

Tradicionalmente, el descenso por gradiente se realiza con toda la base de datos. Sin embargo, según García (2018) para grandes volúmenes de datos, en la actualidad se utiliza el descenso estocástico de gradiente para reducir el tiempo de computación. Este funciona eligiendo al azar un ejemplo para actualizar los pesos. Debido a que muchos ejemplos son similares se consigue un menor tiempo de computación sin sacrificar la precisión.

II.1.9.4 Método ADAM

El método ADAM es un algoritmo para la optimización de redes neuronales basada en gradientes de primer orden de la función de costo. Según Kingma y Ba (2015) es un método para la optimización estocástica que solo requiere gradientes de primer orden. Además, la implementación es computacionalmente eficiente y requiere pocos requisitos de memoria. Por consiguiente, es adecuado para problemas grandes en términos de datos y/o parámetros. El método calcula la dirección del descenso y el tamaño del paso usando momentum, lo que evita cambios bruscos en el paso. Esto lo hace estable para su uso con el descenso de gradiente estocástico donde las muestras pueden provocar grandes cambios en la magnitud del gradiente.

Adicionalmente, calcula un paso global en vez de utilizar un paso por cada variable y también es utilizado en problemas grandes en términos de datos y/o parámetros, como es el caso de las redes neuronales profundas.

II.1.9.4.1 Momentum

El Momentum según Bullinaria (2004) es un parámetro que se utiliza en el ajuste de pesos, es el resultado de la multiplicación dos componentes, el primero es el peso de un tiempo anterior y el segundo es un alfa que representa que tanto influye el peso anterior, esto ayuda a que los cambios en el vector de pesos se suavicen.

II.2 Redes Sociales

Las redes sociales se definen como “estructuras compuestas por personas conectadas por uno o varios tipos de relaciones (de amistad, de parentesco, de trabajo, ideológicas) con intereses comunes” (Islas y Ricaurte,2013, pp.1)

Según Bartolomé (2008) las redes sociales reflejan una serie de puntos representando individuos, unidos mediante líneas que representan relaciones. El carácter de una red social puede ser muy variado, las redes sociales mueven al mundo, uniendo a individuos lejanos físicamente y en gran número.

Según Boyd y Ellisson (2007) una red social es un servicio que permite a los individuos: tener un perfil dentro de un sistema delimitado, articular una lista de otros usuarios con cuales compartir una conexión y tener la posibilidad de ver y recorrer una lista de las conexiones realizadas por otros dentro del sistema. La naturaleza y nomenclatura de estas conexiones varían según el sitio.

II.2.1Twitter

La red social Twitter es usada según Islas y Ricaurte (2013) como herramienta para obtener y distribuir información de manera inmediata por una sociedad que está dispuesta a participar de manera pública o privada.

Según Islas y Ricaurte (2013) Twitter constituye un espacio en donde es posible seguir eventos en directo, se intercambian opiniones y aficiones, pero no necesariamente se propicia la interacción o el diálogo. Sin embargo, es un comportamiento subyacente a la publicación de opiniones por parte de los usuarios.

II.2.2 CyberBullying

Podemos definir CyberBullying, según Vandebosch y Van-Cleemput (2008) como el fenómeno que implica la intimidación, el acoso, la amenaza o insulto mediante medios electrónicos, con la intención de dañar a una persona.

El ciberbullying se divide en siete subtipos: mensajes de texto recibidos en el teléfono móvil; fotografías o videos realizados con las cámaras de los móviles, y posteriormente usados para amenazar a la víctima; llamadas acosadoras al teléfono móvil; mensajes de correo electrónico insultantes o amenazantes; salas de chat en las que se arremete contra uno de los participantes o se le excluye socialmente; el acoso mediante los programas de mensajería instantánea y páginas web donde se difama a la víctima, se descarga información personal a la red o se hacen concursos en los que se ridiculiza a los demás. (Smith et al ,2006)

II.3 Arquitectura de software

Según Bass, Clements y Kazman (2004), la arquitectura se refiere a cómo están jerarquizados los componentes de un software y la relación que existe entre ellos. Es también considerado como el diseño de alto nivel de la

estructura de un programa. Tomando en consideración lo anterior, en las últimas décadas se han establecido ciertos patrones estándares de arquitecturas de software en diferentes áreas, una de ellas es el área de aplicaciones web.

II.3.1 Arquitectura Cliente-Servidor

Está compuesto por dos módulos como su nombre lo indica, este patrón consiste de múltiples clientes que utilizan los servicios que provee un servidor centralizado de forma continua.

II.3.2 Arquitectura Modelo-Vista-Controlador

Este patrón de arquitectura conocido también como MVC según Beck y Cunningham (1987) está conformado por las tres partes que su nombre indica. El modelo, se encarga de manejar los datos de la aplicación y de persistir los datos; la vista, está encargada de mostrar la data a los usuarios; y el controlador, se encarga de manejar las interacciones entre la vista y el modelo. Este patrón permite estructurar el software en componentes que pueden ser reutilizados dentro del mismo.

II.3.2.1 Angular

Es un *framework* escrito en el lenguaje JavaScript que según Korva (2016) ayuda a los desarrolladores a crear aplicaciones escalables y permite la aplicación del patrón de arquitectura MVC. También, utiliza un patron de diseño orientado a componentes, el cual permite elaborar aplicaciones web dinámicas y ofrece herramientas que reducen la complejidad del desarrollo, así como el tiempo de elaboración del software.

II.3.2.2 Django

Al igual que Angular, Django es un *framework* de desarrollo web que según Holovaty y Kaplan-Moss (2008) ahorra tiempo y facilita el desarrollo.

Está escrito en el lenguaje Python, el cual mantiene un ecosistema de librerías que agilizan el desarrollo de software para *machine learning.* Además, Django posee la capacidad de funcionar como un servidor con el cual otras aplicaciones pueden comunicarse mediante el protocolo HTTP.

Capítulo III. Marco metodológico.

III.1 Tipo de estudio

El presente proyecto de investigación fue clasificado de acuerdo a su naturaleza y su alcance. Debido a su naturaleza, se deduce un enfoque cualitativo ya que Hernández, Fernández y Baptista (2014) exponen que las investigaciones cualitativas son aquellas cuyo objetivo principal es estudiar o evaluar las diferentes realidades de un tema en función de los actores que influyen en dicha realidad. En el caso propuesto, se estudió y aplicó conceptos para generar datos que ayudan a mejorar la experiencia de uso de los usuarios de redes sociales hispanoamericanos.

Por su alcance, la investigación es clasificada como descriptiva debido a que el estudio se enfocó en detectar la carga ofensiva que tienen los comentarios realizados en español en redes sociales. Esto permitió aplicar diversas soluciones a la problemática mencionada en el capítulo I.

III.2 Método de estudio

Durante la investigación, el método de estudio utilizado fue inductivo y deductivo. En las primeras etapas, el método de estudio fue inductivo debido a que, sobre una muestra representativa de una población de comentarios, se elaboró una evaluación sobre la carga ofensiva que pueden tener los casos particulares de la población completa.

Durante la etapa final se aplicó un método deductivo, ya que consistió en el desarrollo de una red neuronal que a partir de una serie de algoritmos y cálculos matemáticos puede hacer predicciones sobre la carga ofensiva de comentarios realizados en redes sociales.

III.3 Población y muestra

La población está definida por los comentarios, también denominados *Tweets,* que están únicamente en español dentro de la red social *Twitter.*

La muestra está compuesta por ocho mil de los comentarios en español tomados de *Twitter* de los cuales cuatro mil tienen contenido que puede ser considerado ofensivo. Por ejemplo: insultos generales, raciales o de género, ofensas sexuales, amenazas contra la integridad física, etc. El resto de los comentarios son frases u oraciones comunes en donde no existen palabras groseras en general.

III.4 Variables y operacionalización

La variable de mayor interés para la investigación es la probabilidad de que un comentario sea percibido como ofensivo o tóxico por un usuario común de redes sociales. Para poder obtener el valor de la probabilidad se tomaron en cuenta una variable y una constante para cada comentario:

- Total de calificaciones realizadas: debido a que cada comentario fue calificado por 40 usuarios, este valor es constante e igual a 40.
- Total de calificaciones como ofensivo: esta variable entera está comprendida entre 0 y 40 debido a que cada usuario que califica, dependiendo de su perspectiva del comentario, puede calificarlo como ofensivo o no ofensivo. Este valor representa la cantidad de usuarios que consideraron el comentario como ofensivo.

Utilizando estos dos valores se procedió a realizar el cálculo de la probabilidad de que el comentario sea calificado como ofensivo, dividiendo la cantidad de calificaciones negativas entre el total de calificaciones.

III.5 Instrumentos de recolección de la información

Para la recolección de la muestra se realizó un módulo de extracción utilizando la API de Twitter. Este módulo se encarga de buscar comentarios en *Twitter* dado una palabra u oración y pasan por un proceso de limpieza de la data donde se eliminan las menciones de usuarios para mantener el anonimato de los mismos. Luego, permite seleccionar los comentarios que se desean y los envía a la base de datos para su almacenamiento.

Aunado a esto, se desarrolló también un módulo de calificación de datos donde los usuarios seleccionados realizaron las respectivas calificaciones según su perspectiva de cada comentario.

III.6 Diseño de la investigación

Este trabajo de investigación fue realizado en cinco fases:

III.6.1. Revisión de la literatura.

La revisión bibliográfica fue enfocada en diferentes investigaciones en el área de redes neuronales, temas de desarrollo de software con inteligencia artificial y la documentación de las API de las redes sociales en cuestión y de los *frameworks* con los que se trabajó. Este proceso de revisión fue elaborado a lo largo de toda la investigación y durante el desarrollo del software.

III.6.2. Selección de población y muestra.

Nuestra población para el caso de estudio, comprende los comentarios en español de la red social *Twitter*. Para definir la cantidad de datos a utilizar, Dixon, Thain y Wulczyn (2017) sugieren que la muestra sea representada por alrededor de 8000 comentarios verificados de los cuales al menos la mitad son calificados con anterioridad como ofensivos, esto con el fin de que la red neuronal tenga una base de comentarios ofensivos suficientemente grande para aprender a diferenciarlos de los no ofensivos.

III.6.3. Recolección de datos

La recolección de datos se realizó a través de la API de *Twitter* y de una aplicación desarrollada por los autores que extrae una muestra significativa de comentarios de *Twitter* dado una palabra u oración.

Posteriormente, estos datos fueron verificados por los investigadores con el fin de filtrar aquellos comentarios no relevantes como cadenas de

caracteres aleatorios, mensajes conocidos como *spam* y aquellos que no estén en el idioma español. Adicionalmente, se utilizó una interfaz gráfica en la cual los usuarios comunes seleccionados calificaron los comentarios recolectados para establecer un aproximado de la probabilidad de que se califique como ofensivo o tóxico. Dixon, Thain y Wulczyn (2017) sugieren que cada comentario puede tener hasta un máximo de 40 calificaciones de personas diferentes, motivo por el cual se seleccionaron 40 usuarios (cuyos emails se encuentran en el apéndice A). Luego de calificados los comentarios, se utilizaron las calificaciones para el cálculo de la probabilidad de que sea calificado como ofensivo dividiendo la cantidad de veces que fue calificado ofensivo entre la cantidad total de personas que lo calificaron.

III.6.4. Procesamiento de los datos

Utilizando la guía de Yao y Tan (2001) que explica una manera de trabajar con las redes neuronales, se expone que los datos se deben dividir en 3 grupos: 1. 70% de los datos, se utilizarán para el entrenamiento de la red. 2. 15% de los datos para validación del funcionamiento. 3. El 15% restante de los datos hacer una evaluación final de la red neuronal.

La red neuronal, mediante el análisis de oraciones en lenguaje natural presentado en N-gramas, determina la probabilidad de que un comentario sea calificado como ofensivo. Las probabilidades se clasifican como “Poco probable de ser ofensivo”, “relativamente neutro” y “Bastante probable de ser ofensivo” dependiendo del resultado obtenido.

Una vez definida la topología de la red, se procedió a desarrollar el modelo planteado y finalmente se realizó el debido entrenamiento con los datos recolectados en la fase anterior. Luego, se evaluó el funcionamiento de la red neuronal utilizando el bloque de datos designado para validación, con la finalidad de verificar que el modelo detecta correctamente datos externos al entrenamiento.

III.6.5. Comunicación de los resultados

Luego de finalizado el proceso de desarrollo y entrenamiento, se procedió a verificar el funcionamiento de la red neuronal con el fin de determinar si logran detectar acertadamente la probabilidad de que los comentarios sean calificados como ofensivos. Esto se realizó a través de una interfaz gráfica que permite a los usuarios solicitar estadísticas de múltiples comentarios de *Twitter*. El software se encarga de procesar la data y enviarla a la red neuronal y una vez que ésta determine los resultados se le muestran al usuario concluyendo con una herramienta que cumple con el objetivo principal de la investigación.

Capítulo IV. Resultados

En este capítulo se expone el análisis de los resultados obtenidos durante la investigación. Los mismos se encuentran agrupados bajo los objetivos específicos del trabajo.

IV.1 Evaluación de la carga ofensiva

El primer objetivo de la investigación fue definir los criterios para evaluar la carga ofensiva de los comentarios. Para poder llevar a cabo el objetivo, fue indispensable especificar lo que significa una ofensa. Según la Real Academia Española (2019) una ofensa es el acto de "humillar o herir el amor propio o la dignidad de alguien, o ponerlo en evidencia con palabras o hechos. Ir en contra de lo que se tiene comúnmente como bueno, correcto o agradable". Tomando en cuenta esta definición y el contexto en línea de las redes sociales podemos interpretar que también se puede humillar o herir personas poniéndolo en evidencia a través de palabras escritas en comentarios en internet que van en contra de lo comúnmente agradable. Este tipo de actos es conocido como ciberacoso (derivado del término en inglés, *Cyberbullying)*.

Cabe destacar que estudios psicológicos como el realizado por Rodríguez y Moreno (2017) han demostrado exitosamente que existe una diferencia en la percepción de lo que es ofensivo dependiendo del género y la edad de las personas. Rodríguez y Moreno toman como caso de estudio a jóvenes adolescentes y demostraron que los varones son más propensos a sentirse ofendidos por agresiones físicas mientras que las hembras se ofenden más cuando se les atribuyen rasgos negativos.

Tomando lo anterior en cuenta, se pudo concluir que un criterio para evaluar la carga ofensiva de comentarios es utilizando la perspectiva de diferentes personas para determinar un aproximado de la probabilidad de que cualquier persona pueda identificar determinado comentario como ofensivo. Dixon, Thain y Wulczyn (2017) respaldan esta hipótesis en su trabajo

relacionado a la evaluación de la carga ofensiva de comentarios en inglés hechos en foros de discusiones en línea.

IV.2 Clasificación de datos

El segundo objetivo fue calificar los datos extraídos de una red social seleccionada, a través de los criterios definidos. Para lograr este objetivo se seleccionó una red social que pudiese proporcionar los datos de manera programática tomando en cuenta los derechos de privacidad de las personas. Luego, se desarrolló un algoritmo que extrae los comentarios de la red social aplicando una búsqueda por palabras claves. Una vez extraída la data se seleccionó un grupo de usuarios de redes sociales de diferentes edades para la calificación a través de una página web elaborada durante la investigación.

IV.2.1 Selección de la red social.

Durante el proceso de selección se tomaron en consideración YouTube, Instagram, Facebook y *Twitter.* Se seleccionó la red social *Twitter* basándose en que su contenido principal son cadenas de caracteres de longitud limitada, lo cual se adapta perfectamente para el caso de estudio en cuestión. Las redes sociales como *Facebook* e *Instagram*, a pesar de ser buenas alternativas debido a que en su contenido es común que se generen discusiones en la sección de comentarios de cada post, se descartaron en una etapa temprana de la investigación debido a que las políticas de privacidad de la empresa, *Facebook* (también dueña de *Instagram),* hacen que la extracción de los datos sea bastante restringida. Con respecto a los comentarios realizados en *YouTube*, luego de una extensa búsqueda, se concluyó que generalmente son comentarios que carecen de sentido polémico, en su mayoría son mensajes de saludos y de spam.

IV.2.2 Extracción de la data

Se realizó la extracción de la data programáticamente a través de la interfaz de programación de la aplicación (en adelante API por sus siglas en ingles) de *Twitter* llamada *Twitter Developers Platform* la cual posee la capacidad de recibir solicitudes por medio del protocolo HTTP. Las solicitudes se hicieron mediante la plataforma estándar de búsqueda de *Tweets*, esta recibe una solicitud con una cadena de caracteres y devuelve una lista con todos los tweets realizados en los últimos 7 días que contengan lo datos proporcionados.

Para poder interactuar con la plataforma de *Twitter*, las solicitudes tienen que ser realizadas desde un servidor, por lo que se utilizó un módulo realizado en Django como puente entre la comunicación de un módulo web realizado en Angular con la API de *Twitter*. Los detalles de la implementación de ambos módulos se exponen en la sección IV.4 de este capítulo.

El flujo de una solicitud de búsqueda de comentarios es realizado desde el módulo web hacia el servidor Django, luego este procede a realizar la solicitud ante la API de *Twitter,* una vez que recibe la respuesta procede a responder al módulo web. Finalmente, este módulo se encarga de limpiar la data y mostrarla como en la figura 8.

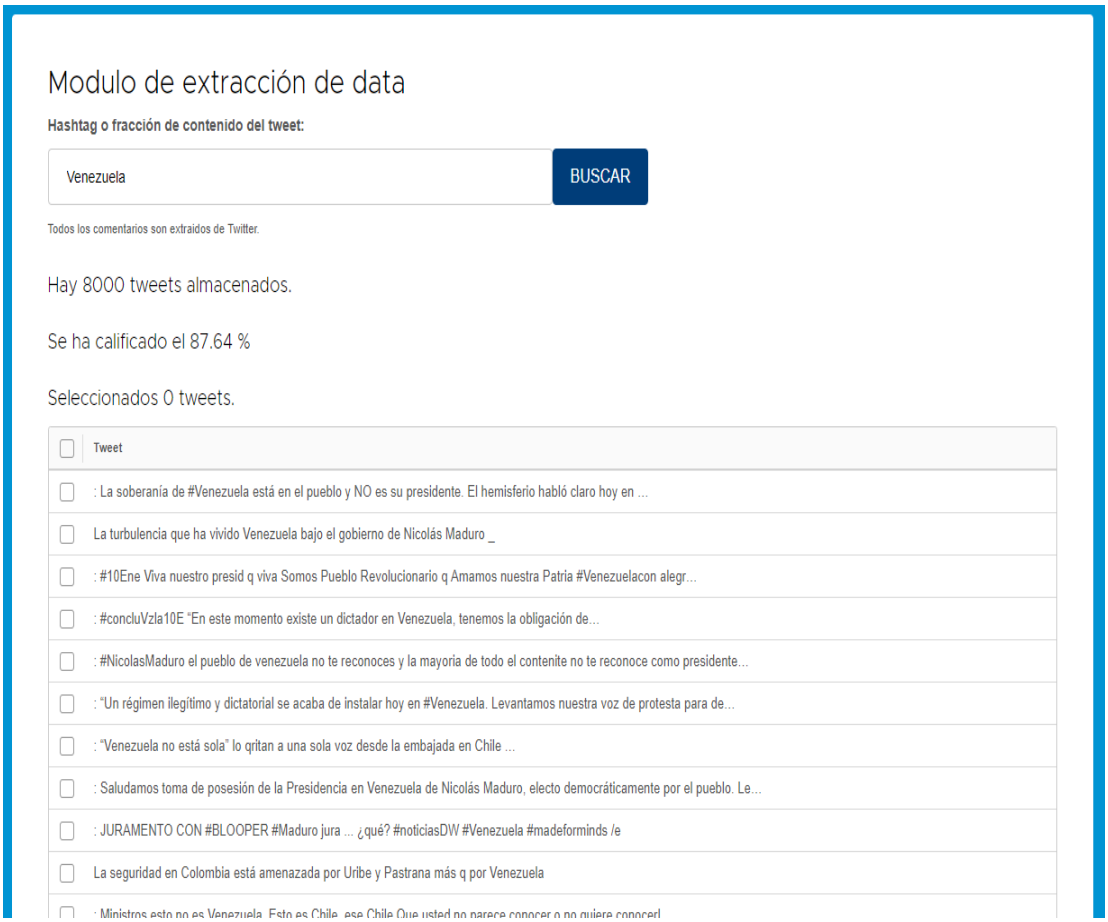

Figura 8. Módulo de extracción de datos.

Fuente: elaboración propia

Luego de recibidos los datos, se procedió a seleccionar los comentarios bajo los criterios mencionados en la sección III.6.3 y se almacenaron en una base de datos agregando dos propiedades adicionales, el total de calificaciones y la cantidad de calificaciones negativas, ambas inicializadas en cero. Este proceso de búsqueda, selección y almacenado se repitió múltiples veces realizando búsquedas de tweets con palabras claves hasta llegar a almacenar ocho mil comentarios. Los primeros cuatro mil fueron buscados utilizando frases y palabras con contenido denigrante, sexual, humillante y/o amenazante para tener una base de datos con suficiente contenido para que

la red neuronal desarrollada aprendiera a diferenciar comentarios ofensivos de no ofensivos.

IV.2.3 Método de calificación

Para la calificación de la data, se realizó una aplicación web en la cual se mostraron los comentarios traídos de la base de datos. Dichos comentarios fueron mostrados de uno en uno a los usuarios seleccionados para calificar. Los mismos pudieron seleccionar para cada comentario 3 opciones, “no, no es ofensivo”, “no sé, paso” y “si, es ofensivo” como se muestra en la figura 9. La interacción con cada botón de la vista ejecutaba una función diferente en el controlador:

- “No, no es ofensivo”: suma uno al total de calificaciones del comentario, dejando la cantidad de calificaciones negativas sin ningún cambio.
- “No sé, paso”: esta acción salta al siguiente comentario en la lista sin modificar ninguno de los datos asociados al comentario. Sin embargo, esta calificación no se pierde, cuando el usuario vuelve a iniciar sesión le aparece de nuevo los comentarios que saltó anteriormente.
- “Si, es ofensivo”: suma uno al total de calificaciones y a la cantidad de calificaciones negativas del comentario.

Los usuarios seleccionados procedieron a realizar la calificación de los comentarios a medida que se iban almacenando en la base de datos hasta que todos los comentarios fueron calificados. Este proceso fue realizado durante el período comprendido entre diciembre 2018 y febrero 2019.

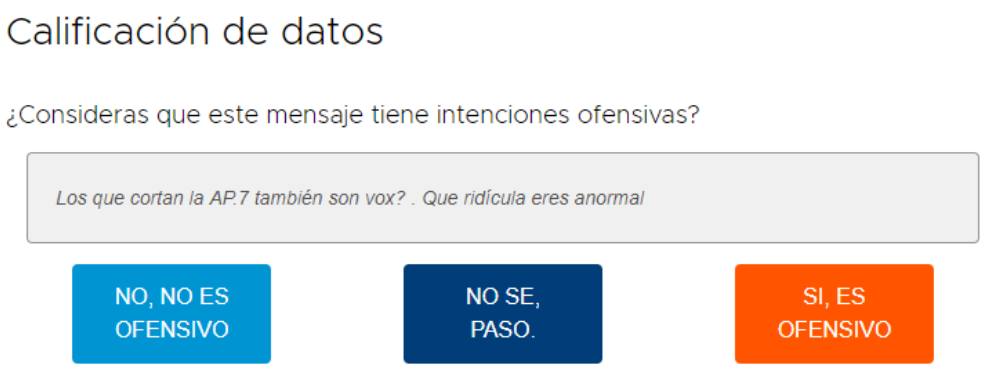

Figura 9. Módulo de calificación de comentarios.

Fuente: elaboración propia

IV.3 Diseño de red neuronal

El siguiente objetivo propuesto fue diseñar el software basado en la red neuronal que detecta la carga ofensiva de comentarios en una red social. Para el desarrollo del modelo se realizó una investigación sobre los diferentes tipos de redes, sus beneficios en diferentes aplicaciones y las topologías recomendadas para tratar con reconocimiento de lenguaje natural.

La función principal de la red neuronal es analizar los comentarios y aprender a diferenciar los ofensivos del resto. Esto se logró luego de una etapa de entrenamiento donde la red neuronal utilizó los comentarios calificados como ofensivos por el grupo de personas designado y aprendió las cualidades que poseen aplicando cálculos matemáticos y algoritmos sobre la base de datos de comentarios designada para entrenamiento y validación. Cabe destacar que los modelos de redes neuronales pueden desarrollar implícitamente un sesgo no intencional debido a que están sujetas a aprender en base a la perspectiva de humanos. Dixon, Li, et al (2017) definen que el sesgo no intencional en aplicaciones de clasificación de texto sucede cuando “el modelo funciona mejor para comentarios que contienen ciertos términos en

comparación con otros". También exponen que existen formas de reducir este tipo de sesgo, pero que aún no se ha descubierto cómo eliminarlo por completo.

Luego de la etapa de entrenamiento se procedió a evaluar la red con la base de datos designada para pruebas con el fin de verificar el correcto funcionamiento de la red con datos diferentes al entrenamiento y evaluar sus resultados.

IV.3.1 Definición de la topología y parámetros.

El tipo de red utilizado durante la investigación fue una red neuronal recurrente LSTM debido a sus destacadas cualidades en el análisis de texto en lenguaje natural. Aunado a esto, utilizando *Adam* y *Binary Cross Entropy* la eficiencia del aprendizaje y la precisión de los resultados incrementaron de forma significativa.

IV.3.1.1 Validación del funcionamiento

El funcionamiento de la red se evaluó tomando en cuenta la precisión de los resultados con respecto a la data de entrenamiento y de validación en la fase de entrenamiento. Además, se evaluó la cantidad de falsos(F)/verdaderos(V) positivos(P) y negativos(N) durante la etapa de prueba mediante una matriz, concluyendo los siguientes resultados:

- Precisión de entrenamiento: 99.23%
- Precisión de validación: 79.42%
- Matriz de confusión: [695 VN, 103 FN, 139 FP, 213 VP]

IV.3.1.2 Estructuración de los datos de entrada

Los datos recolectados fueron oraciones o frases representadas como cadenas de caracteres. Para la introducción de estos datos a la red primero se

debe pasar por un proceso de *tokenización*, para el cual debe existir una estructura de datos semejante a un diccionario que es denominado como el vocabulario de la red. Luego de *tokenizar*, cada comentario se ajusta a una longitud “N” de tokens fija y cada *token* se vectoriza mediante el proceso de *Embedding.* De forma que la red neuronal recibe cada comentario como una matriz NxM en donde cada vector es la representación vectorial de dimensión “M” de cada palabra en el comentario.

IV.3.1.2.1 Vocabulario

Para la elaboración del vocabulario de la red se tomó la lista con todos los comentarios obtenidos para la etapa de entrenamiento. A cada comentario se le eliminaron todos los símbolos (como &, #) y signos de puntuación que tenían exceptuando los acentos en las palabras. Esto con la finalidad de tener solo palabras en cada comentario. Luego de esto, se contaron las ocurrencias de todas las palabras en la lista, se ordenaron de mayor a menor y se almacenaron en el vocabulario en memoria principal. Cada elemento del vocabulario está relacionado con un índice, este es un indicador de la posición en la que se encuentra la palabra. Finalmente, se decide un valor para la cantidad máxima de palabras en el vocabulario con la finalidad de eliminar palabras con poca ocurrencia ya que no aportan nada al aprendizaje de la red. El valor máximo tomado durante la investigación fue de treinta mil palabras. Este se obtuvo luego de probar con diferentes valores y se seleccionó el que brindó mejores resultados en el aprendizaje en cuanto a verdaderos negativos y falsos positivos. Los resultados se muestran en la tabla 1.

IV.3.1.2.2 Tokenización

El proceso de tokenización consiste de convertir cada comentario a una lista de números (Keras, 2015.), en donde cada número representa al índice de la palabra en el vocabulario. Por ejemplo, asumiendo que el vocabulario tiene la siguiente lista: [2: el, 4: perro, 6: es, 8: bonito]. El proceso de

tokenización sobre un comentario como “El perro es bonito” retorna una lista de la siguiente forma [2, 4, 6, 8]. Por defecto en el vocabulario existen dos índices reservados, que son el cero (0) y uno (1). El cero no se le asigna a ninguna palabra y el uno es el valor que toman aquellas palabras que no se encuentran en el vocabulario al momento del tokenizado. Finalizado el proceso de tokenización, cada lista se ajustó a una longitud de cuarenta tokens debido a que la red necesita un tamaño fijo en las entradas. Los casos en donde los comentarios no llegan a cuarenta palabras, se rellenaron con ceros y los casos donde los comentarios se exceden, se truncaron. Se tomó como longitud máxima cuarenta palabras debido a que el 95% de los comentarios de la muestra se encuentran por debajo de este límite. La figura 10 representa la gráfica donde se puede apreciar la cantidad de comentarios por cantidad de palabras.

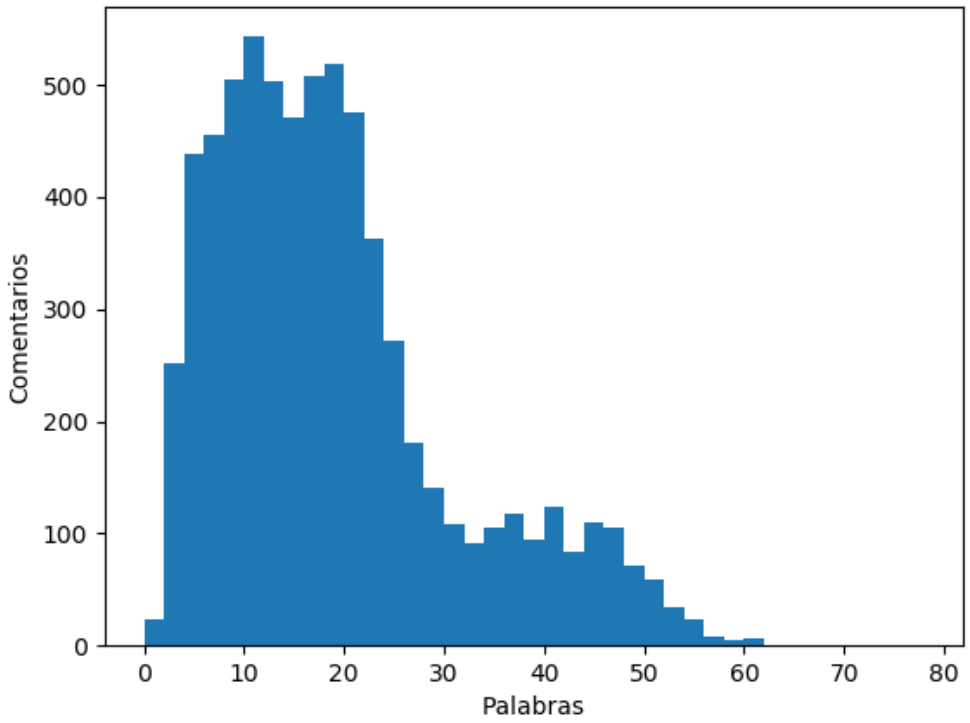

Figura 10. Gráfica de cantidad de comentarios por cantidad de palabras. Fuente: elaboración propia.

IV.3.1.2.3 Embebido

El embebido es utilizado porque “la representación de palabras en un espacio vectorial ayuda a los algoritmos de aprendizaje a lograr un mejor rendimiento en tareas que implican procesar lenguaje natural al agrupar palabras similares” (Mikolov, Sutskever, et al. 2013). La dimensión utilizada para los vectores según TensorFlow (20 de noviembre de 2017) es un número seleccionado a conveniencia del autor de la red, es decir, el número que ofrezca mejores resultados. Luego de evaluar diferentes cantidades se seleccionó un tamaño del vector de 28 dimensiones. Los resultados se encuentran en la tabla 1.

Tabla 1: Cantidad máxima de palabras y sus resultados

Tamaño vocabulario / dimensión de embebido	Verdaderos negativos	Falsos negativos	Falsos Positivos	Verdaderos Positivos
10.000 / 28	637	90	165	222
10.000 / 64	637	84	165	228
10.000 / 86	628	77	174	235
30.000 / 28	695	103	139	213
30.000 / 64	649	79	185	239
30.000 / 86	574	55	260	263
80.000 / 28	627	75	207	243
80.000 / 64	617	73	217	245

Fuente: elaboración propia.

IV.3.1.3 Capas internas de la red

La estructura interna de las capas de la red que obtuvo mejores resultados fue la siguiente: Capa de entrada, Capa de embebido, Capa LSTM Bidireccional, Capa de Pooling Máximo Global, Capa de activación tanh, Capa de salida con activación sigmoide (Diagrama de estructura de la red en Apéndice C).

IV.3.1.4 Parámetros de optimización

Para la optimización del aprendizaje de la red se utilizó la función de pérdida *Binary Cross Entropy* (en adelante BCE, por sus siglas en inglés) en conjunto con el optimizador *ADAM.*

El BCE permite calcular el valor de pérdida cuando los valores de salida de la red representan probabilidades (valores decimales entre 0 y 1) asociados a respuestas binarias. Por ejemplo, tomando que un comentario ofensivo tenga el valor binario 1, los comentarios no ofensivos tendrían el valor 0. Si la red clasifica un comentario ofensivo con una probabilidad de 0.98, el valor de pérdida es bastante pequeño. Sin embargo, si la red clasifica un comentario ofensivo con una probabilidad de 0.02, el valor de pérdida debe ser bastante grande de forma que la red pueda corregir el error adecuadamente.

El optimizador *ADAM* fue seleccionado debido a su eficiencia manejando grandes cantidades de datos y parámetros de múltiples dimensiones (Kingma, Ba. 2015) como lo son los las matrices embebidas que representan a los comentarios dentro de la red.

IV.4 Desarrollo del software

El último objetivo de la investigación consistió en desarrollar el software bajo los *frameworks Django y Angular*. El objetivo se llevó a cabo utilizando los patrones de diseño Modelo-vista-controlador (en adelante MVC) y Cliente-

Servidor dentro de los frameworks seleccionados. El patrón MVC utilizado en Angular facilitó que las interacciones de los usuarios en las vistas de la aplicación web se comunicaran a través de los controladores con el servidor realizado en Django y la base de datos. Concluyendo con un software completo cuya comunicación cliente-servidor es rápida y eficiente brindando resultados a gran velocidad.

IV.4.1 Arquitectura MVC

El software desarrollado en *Angular* fue estructurado en 3 componentes principales: extracción, clasificación y resultados. Cada componente consta de una vista y un controlador único mientras que el modelo es global para los tres y se encarga de manejar la comunicación con la base de datos.

IV.4.1.1 Componente de extracción.

Este componente cumple con cuatro funciones principales: solicitar al servidor Django la extracción de comentarios de *Twitter*, limpiar los comentarios que recibe, almacenarlos en la base de datos y descargar un archivo ".csv" con la información de los comentarios. La vista del componente se puede apreciar en la sección 2.2 de este capítulo en la figura 8.

IV.4.1.2 Componente de clasificación

El componente de clasificación cumple con su principal propósito de permitir a los usuarios seleccionados calificar los comentarios uno a uno. Esto se logró mediante dos funciones principales: buscar uno por uno los comentarios previamente almacenados en la base de datos y luego de calificado actualizar los valores en la base de datos. En la sección 2.3 de este capítulo, se muestra la figura 9 con la vista del componente.

IV.4.1.3 Componente de resultados

Este componente se encarga de la interacción con la red neuronal a través del servidor Django. Cumple con dos métodos, el primero consiste de una clasificación sencilla en la cual el usuario del software puede ingresar un texto cualquiera y lo envía a la red neuronal para ser analizado. El segundo requiere que el usuario ingrese una frase o palabras clave para buscar en *Twitter*. Una vez que se obtienen los comentarios, se envían a la red neuronal para que se analicen, finalizado este proceso, se muestran todos los comentarios en la vista con un indicador de cuánto fue la calificación que obtuvo, al igual que un *slider* para decidir hasta qué nivel de ofensivo se desea ver. La vista del componente puede observarse en la figura 11.

IV.4.2 Arquitectura Cliente-Servidor

La comunicación con el servidor realizado en Django se hizo a través de dos puntos de acceso principales: uno para la manejar la comunicación externa con *Twitter* y otro para la comunicación interna con la red neuronal.

IV.4.2.1 Comunicación externa con *Twitter*

La comunicación con *Twitter* fue realizada a través de su portal *Twitter Developer Platform.* Para la interacción con los servidores de Twitter fue necesario registrarse como desarrollador en la plataforma y solicitar claves de acceso. Finalizado el proceso de registro, la comunicación con el servidor de *Twitter* se hizo desde nuestro servidor Django mediante solicitudes en el protocolo HTTP hacia el punto de acceso dedicado a la búsqueda de *tweets.* Cada solicitud envía las claves de acceso y las palabras claves por las cual se buscarán los *tweets*. La respuesta de Twitter, si las credenciales son válidas, es una lista con todos los tweets que contienen las palabras claves de los últimos 7 días.

Figura 11. Vista de componente de resultados.

Fuente: elaboración propia.

IV.4.2.2 Comunicación interna con la red neuronal.

Una vez finalizado el proceso de entrenamiento, validación y pruebas de la red neuronal, el modelo de red se integró en el servidor Django de forma que cada vez que se inicialice, la red neuronal se construye en tiempo de compilación y está lista para ser utilizada localmente por los puntos de acceso del servidor cuando llegue una solicitud.

Al recibir una solicitud, el servidor se asegura de que la solicitud cumpla con el protocolo adecuado HTTP, luego verifica que exista al menos un texto para calificar y los introduce en la red. El resultado, en caso de ser exitoso, es enviado como respuesta al cliente que realizó la solicitud. En caso contrario, se envía un mensaje de error con el motivo.

Capítulo V. Conclusiones y recomendaciones

Este trabajo fue realizado con el objetivo principal de “Implementar un sistema de detección de la carga ofensiva de comentarios y mensajes en español en redes sociales”. Las conclusiones obtenidas del trabajo son las siguientes:

- Las calificaciones realizadas en los comentarios apoyan la teoría de que la gravedad de una ofensa es relativa y depende de la perspectiva de cada persona debido a que en un grupo de personas existen quienes calificaron un comentario como ofensivo mientras otras, el mismo comentario, lo calificaron como no ofensivo.
- Un medidor de qué tan ofensivo puede ser una frase u oración es la probabilidad de que el mismo sea percibido como ofensivo cuando es calificada por un grupo de personas de diferentes edades que puedan tener diferentes perspectivas sobre lo que es una frase ofensiva.
- Las redes neuronales son herramientas poderosas capaces de aprender patrones en los datos y realizar clasificaciones acertadas sobre nueva información utilizando métodos de corrección de errores como la retro propagación en conjunto con el descenso de gradiente y las celdas LSTM.
- Las redes neuronales LSTM Bidireccionales son una herramienta eficaz en el análisis de intenciones en lenguaje natural escrito como cadenas de caracteres debido a su capacidad de aprender intenciones en datos que contienen una estructura secuencial.
- Los frameworks *Django* y *Angular* en conjunto, permiten elaborar aplicaciones web progresivas capaces de realizar cálculos matemáticos complejos, con una carga computacional exigente, de manera sencilla y escalable debido a la capacidad de *Django* de funcionar como servidor e integrarse apropiadamente con un cliente desarrollado en *Angular*.

Dando por finalizado el proyecto se presentan las siguientes recomendaciones para trabajos futuros:

- Realizar una recolección de datos más grande para ampliar el vocabulario y el aprendizaje de la red.
- Elaborar una API pública que permita que usuarios puedan interactuar con la red neuronal de forma segura y privada.
- Investigar sobre diferentes topologías de red que puedan incrementar la precisión de los resultados.
- Utilizar *embeddings* públicos pre-entrenados durante el entrenamiento de la red para estudiar los cambios en la precisión de los resultados.
- Elaborar un web socket en el servidor Django de forma que se puedan enviar cambios no solicitados al cliente a través de observables.

Bibliografía

Abadi, M., Agarwal, A., Barhamn, P., Brevdo, E., Chen, Z. Citro, C., et al. (2015) TensorFlow: Large-Scale Machine Learning on heterogeneous Distributed systems. Recuperado el 11 de enero de 2019 de: http://download.tensorflow.org/paper/whitepaper2015.pdf

Adventures in Machine Learning (2017, octubre). *Recurrent neural networks and LSTM tutorial in Python and TensorFlow*, [en línea]. Recuperado el 22 de febrero de 2019, de: https://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/.

Angular (2018) Angular: One framework. Mobile & Desktop. Recuperado el 12 de enero de 2019 de: https://angular.io/docs

Barrera, J. (S. F.) Redes Neuronales. Universidad de Guadalajara, Recuperado el 18 de febrero de 2019, de: http://www.cucei.udg.mx/sites/default/files/pdf/toral_barrera_jamie_areli.pdf

Basualdo, M., Ruiz, C. (2001). Redes Neuronales: Conceptos Básicos y aplicaciones. Recuperado el 12 de febrero de 2019, de: https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

Bass, L., Clements, P. y Kazman, R. (2004). *Software architecture in practice.* Boston: Addison-Wesley

Bartolomé, A. (2008): E-Learning 2.0 - Posibilidades de la Web 2.0 en la Educación Superior. Recuperado el 12 de febrero de 2019, de: http://www.lmi.ub.es/cursos/web20/2008upv/

Beck, K., Cunningham, W. (1987). Using Patterns Languages for Object-Oriented Programs. Specification and Design for Object-Oriented Programming. Recuperado el 16 de febrero de 2019, de: https://c2.com/doc/oopsla87.html

Bruguier, A., Damavandi, B., Kumar, S., Shazeer, N. (2016) *NN-grams*: Unifying neural network and n-gram language models for speech recognition. San Francisco, USA. Recuperado el 15 de enero de 2019 de: https://pdfs.semanticscholar.org/5b7a/b3fe5d4684289280f52a82a5709a92ab2caf.pdf?_ga=2.184404644.320230059.1527201006-591438879.1527201006-

Botev, Z., Kroese, D., Taimre, T. (2006) Generalized cross-entropy methods. Recuperado el 14 de febrero de 2019 de: https://people.smp.uq.edu.au/DirkKroese/ps/bokrta.pdf

Bonaccorso, G. (2017) Machine Learning Alhorithms: A reference guide to popular algorithms for data science and machine learning**.** (pp 248-300)

Boyd, D., Ellison, N. (2007): Social network sites: Definition, history and scholarship. Recuperado el 16 de febrero de 2019, de : http://gabinetedeinformatica.net/wp15/2008/06/12/sitios-de-redes-sociales-definicion-historia-y-ayuda-a-su-estudio-i/

Bullinaria, J. (2004). Learning with Momentum, Conjugate Gradient Learning.

Burgos, J. E. (2002) Herencia genética, sistema nervioso y conducta. Psicología del Aprendizaje (pp. 43-78) México.

Castrillón, C., Perlaza, J., Van Schoonhove, A., Owen, E., (2001) La red neuronal backpropagation como interpolador. Recuperado el 12 de febrero de 2019, de: http://bdigital.unal.edu.co/10623/14/19259573.Parte4.pdf

Camacho, C (2016) Desarrollo de un sistema de reconocimiento de habla natural basado en redes neuronales profundas, recuperado el 16 de febrero de 2019, de: http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20160905CarolinaCamachoCostumero.pdf

Beebe, N (1993): Accurate Hyperbolic Tangent Computation. Recuperado el 19 de febrero de 2019, de: https://www.math.utah.edu/~beebe/software/ieee/tanh.pdf.

Cui, Z., Ke, R., Wang, Y. (2018) Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. Recuperado el 12 de febrero de 2019, de: https://arxiv.org/ftp/arxiv/papers/1801/1801.02143.pdf

Dahou, A., Xiong, S., Zhou, J., Houcine, M., Duan, P. (2018) Word Embeddings and Convolutional Neural Network for Arabic Sentiment Classification. Recuperado el 11 de febrero de 2019 de: https://www.researchgate.net/publication/312331309_Word_Embeddin

gs_and_Convolutional_Neural_Network_for_Arabic_Sentiment_Classification

Dixon, L., Li, J., Sorensen, J., Thain, N. y Vasserman, L. (2017). *Measuring and mitigating unintended bias in text classification.* Jigsaw. Recuperado el 22 de febrero de 2019 de: https://github.com/conversationai/unintended-ml-bias-analysis/blob/master/presentations/measuring-mitigating-unintended-bias-paper.pdf

Dixon, L., Thain, N., Wulczyn, E. (2017) *Ex Machina: Personal attacks seen at scale*. Ithaca, New York: Cornell University. Recuperado el 15 de enero de 2019 de: https://arxiv.org/abs/1610.08914.

Escobar, R (2014) Redes neuronales, procesos cognoscitivos y análisis de la conducta. Recuperado el 16 de febrero de 2019, de: http://conductual.com/sites/default/files/pdf-articles/Redes%20neuronales_Escobar_0.pdf

Gal, Y., Ghahramani, Z. (2016). A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. Recuperado el 12 de febrero de 2019 de: https://arxiv.org/pdf/1512.05287.pdf

García, M (2018) Aplicación de modelos de redes neuronales al modelado y predicción del precio de la electricidad en España. Recuperado el 16 de febrero de 2019, de: http://oa.upm.es/52027/1/TFG_MANUEL_GARCIA_LOPEZ.pdf

Gers, F., Schmidhuber, J., Cummins, F. (1999). *"Learning to forget: continual prediction with LSTM",* Lugano, Suiza. Recuperado el 8 de enero de 2019 de: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=1660ECB8AAEFCE2517AB996099A3F8C1?doi=10.1.1.55.5709&rep=rep1&type=pdf

Gers, F., Schmidhuber, J., Cummins, F. (2000) "Learning to forget: continual prediction with LSTM", Neural Computation (pp. 2451-2471)

Gers, F., Schmidhuber, J. (s.f.). LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages. Manno, Suiza. Recuperado el 14 de enero de 2019 de: ftp://ftp.idsia.ch/pub/juergen/L-IEEE.pdf

Gómez, J., González, F. (1998). Curso Básico de redes neuronales. Universidad Nacional de Colombia. Recuperado el 16 de febrero de 2019, de: http://dis.unal.edu.co/~jgomezpe/docs/conferences/notes/Redes%20Neuronales%20cursillo%20congreso%2095.doc

González, J., Tudurí, J., Rul-lan, G. (2017). Análisis de Series Temporales Usando Redes Neuronales Recurrentes. Recuperado el 18 de febrero de 2019, de: https://www.apsl.net/documents/1/Analaisi_con_LSTM.pdf

Graves, Alex (s.f.) Supervised Sequence Labelling with Recurrent Neural Networks. (pp18-38)

Gurney, K. (1997). *An introduction to Neural Networks.* Recuperado el 22 de febrero de 2019, de: https://www.inf.ed.ac.uk/teaching/courses/nlu/assets/reading/Gurney_et_al.pdf

Hernández, R., Fernández, C. & Baptista, P. (2014). Metodología de la investigación. 6ta ed. México: McGraw-Hill

Hochreiter, S (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.* Recuperado el 18 de febrero de 2019, de: https://www.bioinf.jku.at/publications/older/2304.pdf

Hochreiter, S., Schmidhuber, J. (1997) *"Long short-term memory", Neural Computation* (pp.1735-1780)

Hoff, M. E., Widrow, B. (1960). Adaptive switching circuits. (pp.96-104). California: Los Angeles. http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf

Holovaty, A., Kaplan-Moss, J. (2008). El libro de Django. Recuperado el 16 de febrero de 2019 de: http://bibing.us.es/proyectos/abreproy/12051/fichero/libros%252Flibro-django.pdf

Islas, O., Ricaurte, P. (2013). *Investigar las redes sociales*: Comunicación total en la sociedad de la ubicuidad. Recuperado el 16 de febrero de 2019 de: http://editorialrazonypalabra.org/pdf/ryp/InvestigarRedesSociales.pdf

Keras (2015) *Keras:* The Python deep learning library. Recuperado el 8 de enero de 2019 de: https://keras.io/

Kingma, D., Ba, J. (2015). ADAM: A Method for Stochastic Optimization. Recuperado el 16 de febrero de 2019 de: https://arxiv.org/pdf/1412.6980v8.pdf

Kohonen, T (1989) *Self-organization and associative memory*. Springer-Verlag, New York, Recuperado el 20 de enero de 2019 de: https://ia801904.us.archive.org/28/items/SelfOrganizationAndAssociativeMemory/Self%20Organization%20and%20Associative%20Memory_text.pdf

Korva, J. (2016) Developing a web application with Angular 2. Recuperado el 16 de febrero de 2019 de: https://www.theseus.fi/bitstream/handle/10024/121905/Korva_Jukka.pdf?sequence=1

Kyurkchiev, N., Markov, S. (2015). Sigmoid functions: some approximation, and modelling aspects.

Martínez, I (2007) Introducción a las redes neuronales, Recuperado el 12 de febrero de 2019, de: http://www.gurugames.es/people/pedro/aad/ivan_martinez.pdf

Mcculloch, W., Pitts, W. (1943) A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical Biophysics,* (pp 115-133).

Mikolov, T., Sutskever, H., Chen, K., Corrado, G., Dean, J. (2013). *Distributed representations of words and phrases and their compositionality.*

Google Inc. Mountain View, California. Recuperado el 22 de febrero de 2019 de: https://arxiv.org/pdf/1310.4546.pdf

Python (2018) The Python Tutorial. Recuperado el 18 de enero de 2019 de: https://docs.python.org/3/tutorial/index.html

Real Academia Española. (2019). Diccionario de la lengua española (22.aed.). Consultado en: http://www.rae.es/rae.html

Rebollo, M., (2018) Quora: *¿Cuáles son los mejores lenguajes de programación para machine learning?* [comentario en un foro en línea]. Recuperado el 5 de noviembre de 2019 de: https://es.quora.com/Cu%C3%A1les-son-los-mejores-lenguajes-de-programaci%C3%B3n-para-machine-learning

Sánchez, D., Revuelta, J., González, A., Corchado, J. (2017). Hybridizing metric learning and case-based reasoning for adaptable clickbait detection. Recuperado el 16 de febrero de 2019 de: https://bisite.usal.es/archivos/hybridizing_metric_learning_and_case-based_reasoning_for_adaptable.pdf

Scherer, D., Muller, A., Behnke, S. (2010) Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. Recuperado el 12 de febrero de 2019 de: http://ais.uni-bonn.de/papers/icann2010_maxpool.pdf

Smith P., Mahdavi, J., Carvalho, M. (2006) An investigation into cyberbullying, its form, awareness and impact, and the relationship between age and gender in cyberbullying. London.

Stamateas, B. (2008). *Gente toxica*: Las personas que nos complican la vida y cómo evitar que sigan haciéndolo. (1era edición). Barcelona, España: Ediciones B. Recuperado el 15 de enero de 2019 de: https://www.fisterra.com/mbe/investiga/cuanti_cuali/cuanti_cuali.asp

Rodríguez, L., Moreno, J. (2017). *Percepción de ofensas o agravios en adolescentes*, [en línea]. Argentina: Pontifica Universidad Católica Argentina. Recuperado el 18 de febrero de 2019 de: https://www.redalyc.org/articulo.oa?id=83654004002

Vásquez, J. (2014). Red Neuronal Feedforward como estimador de patrones de corrientes en el interior del puerto de manzanillo sujeto a la acción de tsunamis. Recuperado el 13 de febrero de 2019 de: https://www.imt.mx/archivos/Publicaciones/PublicacionTecnica/pt406.pdf

Vandebosch, H., Van-Cleemput, K. (2008). Defining Cyberbullying a quialitative research into the perceptions of youngster. (PP. 499-503).

Vivas, H. (2014) Optimización en el entrenamiento del Perceptrón Multicapa, Recuperado el 12 de febrero de 2019, de : http://www.unicauca.edu.co/matematicas/investigacion/gedi/optimizacion/TesisVivas.pdf

Williams, R., Zipser, D. (1989). “A learning algorithm for continually training recurrent neural networks”. *Neural Computation*. (pp. 270)

Yao, J. & Tan, C. (2001). Guidelines for financial forecasting with neural networks. Proc. International Conference on Neural Information Processing, Shanghai, China, pp 757-761.

Apéndice A: Emails del grupo de personas dedicadas a calificar los comentarios.

ehernandez@cxn.agency	estefanygonzalez@correo.unimet.edu.ve	nascimentg@gmail.com
14-10984@usb.ve	alemvangrieken@gmail.com	juan15597l@gmail.com
die.herrera.diaz@gmail.com	lucianopinedo@gmail.com	deborah.pm790@gmail.com
darkstein647@gmail.com	andressaade95@gmail.com	antonyfigueira26@gmail.com
ajad350@gmail.com	zerourahara@gmail.com	gtroncone@correo.unimet.edu.ve
andrea.beatriz.scardino.rodriguez@gmail.com	reaqa1@gmail.com	troncone.aniello@gmail.com
sgcarvallo@gmail.com	joseaguerrero6@gmail.com	isabelladesantis12@gmail.com
carlos.fontes.99@gmail.com	valenvv123@gmail.com	jose.quevedo2011@gmail.com
croquerml@gmail.com	klorn690@gmail.com	delellisgabriel@gmail.com
abraham.chang@correo.unimet.edu.ve	brablawaxplagabrwabla@gmail.com	kikecalabuigcolmenares@gmail.com
davidmartinezsanchez96@gmail.com	augustocasale@gmail.com	meoiswa@gmail.com
andreskovacs@correo.unimet.edu.ve	alejandropenad32@gmail.com	viviana.gomez@correo.unimet.edu.ve
valentinaaguileralara@gmail.com	luisgraterol97@gmail.com	edonellister@gmail.com

guillermoleon0610@gmail.com		

Apéndice B: Funciones de activación

B.1 Función Sigmoide

Según Pérez (2002) son un conjunto de funciones crecientes, monótonas y acotadas que provocan una transformación no lineal de su argumento. Siendo la más utilizada la función logística o sigmoide:

$$g_L(x) = \frac{1}{1 + e^{-x}}$$

Figura 12: Función Logística sigmoidal.

Elaboración: Pérez (2002, pp11)

En la figura 13, se muestra el comportamiento de la función sigmoide entre los valores -1 y 1, evidenciando que se encuentra acotada entre 1 y 0.

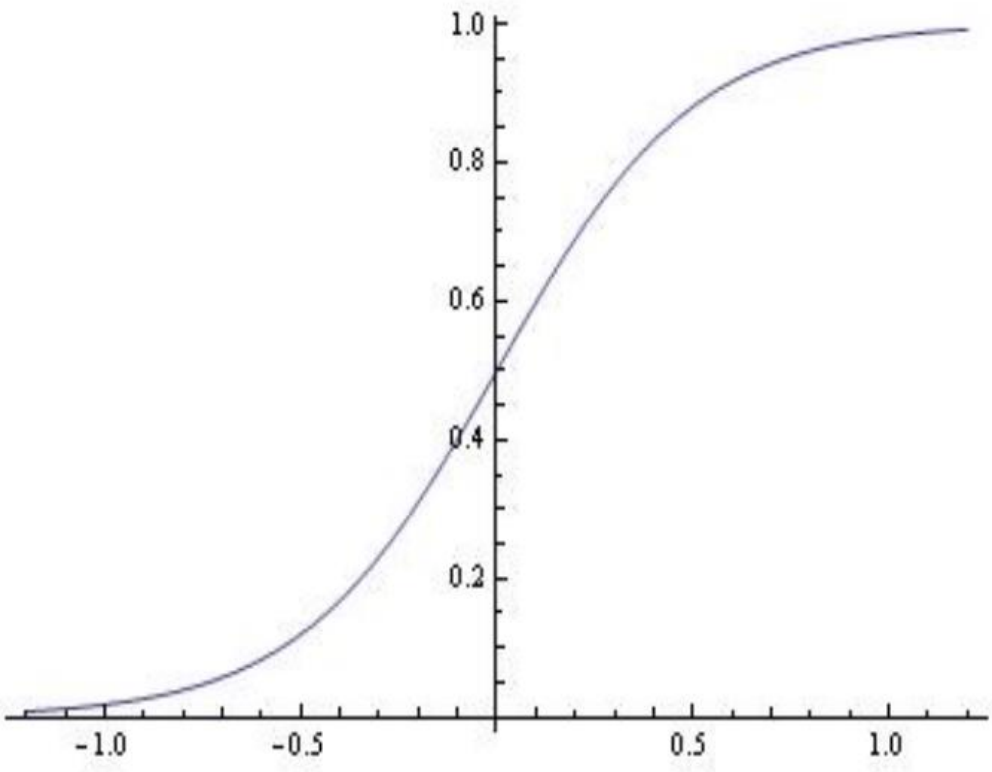

Figura 13: Gráfica de la función logística sigmoide

Elaboración: Kyurkchiev, Markov (2015, pp. 12)

B.2 Tangente hiperbólica

La función Tangente hiperbólica se comporta de manera similar a la sigmoidea:

$$g_T(x) = \tanh(x)$$

Figura 14: Ecuación Tangente hiperbólica

Fuente: Pérez (2002, pp12)

La función se encuentra acotada entre 1 y -1. La función sigmoide y la tangente hiperbólica se relacionan mediante la ecuación:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Figura 15: Relación función sigmoide y función tangente hiperbólica

Fuente: Pérez (2002, pp12)

En la siguiente figura 16, se muestra la gráfica de la función tangente hiperbólica para los valores entre -20 y 20, estando acotada entre -1 y 1.

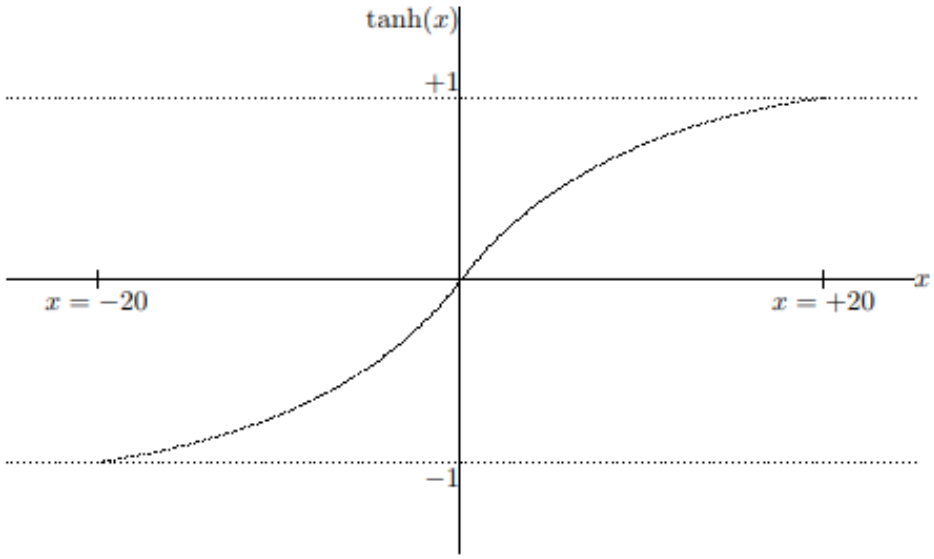

Figura 16: Gráfica de la Función Tangente hiperbólica

Fuente: Beebe (1993, pp 2)

Apéndice C: Diagrama de la estructura de la red neuronal

En la figura 17 se muestra el diagrama de la estructura de la red neuronal desarrollada durante la investigación, su disponibilidad se encuentra en el repositorio: https://github.com/maximilianoc94/tesis.

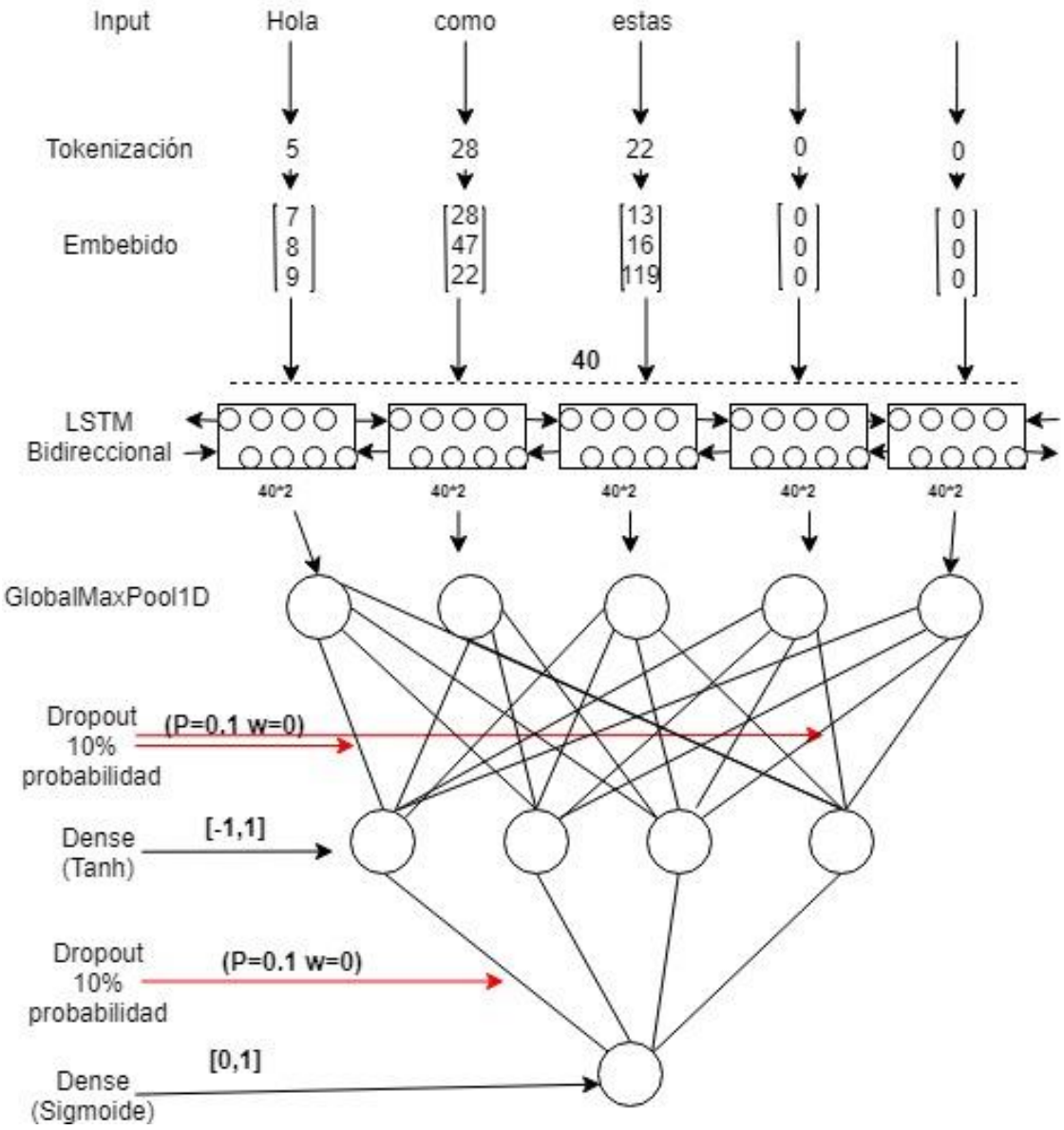

Figura 17: Diagrama de la estructura de la red neuronal.

Fuente: elaboración propia.