UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

MODELO BASADO EN APRENDIZAJE PROFUNDO PARA EL ANÁLISIS DE SENTIMIENTO DE TUITS EN ESPAÑOL

TESIS

PRESENTADA POR:

DANITZA YVETTE BERMEJO ESCOBAR

GERSON WALDYR VIZCARRA AGUILAR

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

PUNO - PERÚ

2020

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

MODELO BASADO EN APRENDIZAJE PROFUNDO PARA EL ANÁLISIS DE SENTIMIENTO DE TUITS EN ESPAÑOL

TESIS PRESENTADA POR:

DANITZA YVETTE BERMEJO ESCOBAR

GERSON WALDYR VIZCARRA AGUILAR

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

APROBADO POR EL JURADO REVISOR CONFORMADO POR:

PRESIDENTE :

M.Sc. WILLIAM EUSEBIO ARCAYA COAQUIRA

PRIMER MIEMBRO :

Ing. ALDO HERNAN ZANABRIA GALVEZ

SEGUNDO MIEMBRO:

M.Sc. PABLO CESAR TAPIA CATACORA

DIRECTOR / ASESOR :

Mg. OLIVER AMADEO VILCA HUAYTA

TEMA: Redes Neuronales
ÁREA: Inteligencia Artificial Y Sistemas Bio-inspirados

FECHA DE SUSTENTACIÓN 06 NOVIEMBRE DEL 2019

A Dios, por todo lo que me ha dado. A mis padres Mary y Hugo, por su amor y haberme educado como la persona que soy. A mis hermanos Paola y Brandon, por enseñarme a luchar por mis seres queridos.

Danitza.

A Dios, por guiar mis pasos. A mi mamá Otilia, por sus palabras sinceras y amor incondicional y a mi hermano Edson por estar siempre a mi lado.

Gerson.

Agradecimientos

En primer lugar deseamos agradecer a Dios por habernos guiado a lo largo de estos años de estudio.

Agradecemos a la Universidad Nacional del Altiplano, nuestra alma máter, por habernos cobijado y brindado la formación que ahora nos permitirá ayudar a construir una mejor sociedad.

Agradecemos de forma muy especial a nuestro asesor Mg. Oliver Amadeo Vilca Huayta por habernos guiado y alentado en el transcurso del proceso de investigación.

Deseamos agradecer de forma especial a los ingenieros: Mayenka Fernandez Chambi, Alberth Mendizabal Flores por habernos apoyado en distintos aspectos de la vida académica y laboral.

Deseamos agradecer a nuestros amigos Antoni y Jorge por su apoyo incondicional, a Jean Pierre y Juan Carlos por sus consejos.

Finalmente, queremos agradecer a la Universidad Católica San Pablo por el reconocimiento al artículo científico que es base de la presente tesis.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

CNN Red Neuronal Convolucional (Convolutional Neural Network)

CPU Unidad Central de Procesamiento (Central Processing Unit)

GPU Unidad de Procesamiento Gráfico (Graphics Processing Unit)

InterTASS International TASS Corpus

LSTM Memoria de Corto y Largo Plazo (Long Short-Term Memory)

MLE Estimación de Máxima Verosimilitud (Maximum Likelihood Estimation)

NLP Procesamiento de Lenguaje Natural (Natural Language Processing)

ReLU Unidad Lineal Rectificada (Rectified Linear Unit)

RNN Red Neuronal Recurrente (Recurrent Neural Network)

SEPLN Sociedad Española para el Procesamiento del Lenguaje Natural

STOMPOL Corpus of Spanish Tweets for Opinion Mining at aspect level about Politics

TASS Taller de Análisis de Sentimiento en la SEPLN

TPU Unidad de Procesamiento Tensorial (Tensor Processing Unit)

XML Lenguaje de Marcado Extensible (eXtensible Markup Language)

RESUMEN

El análisis de sentimiento es una tarea de investigación en el campo del Procesamiento del Lenguaje Natural cuyo objetivo principal es determinar la polaridad de sentimiento de un texto. El análisis de sentimiento ha obtenido buenos resultados en el idioma inglés. Sin embargo, las métricas en español aún son bajas. La presente tesis propone tres modelos basados en aprendizaje profundo para abordar la tarea de análisis de sentimiento de tuits en español. El objetivo es mejorar los resultados obtenidos por métodos anteriores. Para ello, se ha realizado el preprocesamiento de los datos y la generación de representaciones de palabras que serán las entradas de los modelos. Seguidamente, se implementaron las redes neuronales recurrente, convolucional y un híbrido de ambos. Para evaluar los modelos propuestos, se utilizó el corpus InterTASS con cuatro clases. La métrica principal que se consideró es la exactitud. Los resultados muestran que los modelos propuestos son competitivos frente a previos métodos de referencia.

Palabras clave: Aprendizaje Profundo, Análisis de Sentimiento, Red Neuronal Convolucional, Red Neuronal Recurrente, Tuits en Español

ABSTRACT

Sentiment Analysis is a task in Natural Language Processing, whose main objective is to determine the sentiment polarity of a text. Sentiment analysis has obtained excellent results in English. However, the metrics in Spanish are still quite low. This thesis proposes three models based on deep learning to solve the Spanish tweets sentiment analysis task. The objective is to improve the results of prior methods. To do so, we perform the pre-processing of the data and the generation of word representations which will be the inputs of our model. Then, we implement the recurrent and convolutional neural network and a hybrid model of both. To evaluate the proposed models, we use the InterTASS dataset with four classes. The principal metric that we consider is the accuracy. The results show that the proposed models get competitive results against reference baselines.

Keywords: Deep Learning, Sentiment Analysis, Convolutional Neural Network, Recurrent Network. Spanish Tweets

CAPÍTULO I

INTRODUCCIÓN

Debido al creciente uso de internet, a diario se generan millones de datos en forma de texto en diversas redes sociales y blogs. Estos datos contienen información valiosa la cual no puede ser procesada manualmente debido a su gran cantidad. Gracias al aumento de poder computacional, se han desarrollado algoritmos más complejos en el área de NLP; el cual permite poder tratar texto de forma automática y rápida. Debido a que la semántica y sintáctica de un texto son complejas, NLP se ha subdividido en múltiples tareas con fines específicos. Algunas tareas de NLP son: resumen automático de textos, traducción de idiomas, extracción de relaciones, análisis de sentimiento, reconocimiento del habla, clasificación de artículos por semántica, entre otros.

El análisis de sentimiento es una tarea que ha cobrado mucha importancia para la comunidad científica y empresarial. Así, el análisis de sentimiento puede ser utilizado para realizar diagnósticos de una población con respecto a temas específicos. Tales diagnósticos pueden ser aprovechados por empresas u organizaciones a fin de mejorar sus productos o servicios, o por otros investigadores para hacer estudios sociales.

El análisis de sentimiento es una tarea de clasificación de textos. Dado un texto, se debe determinar la polaridad a la que pertenece. Generalmente esta clasificación considera

tres categorías: positiva, negativa y neutra. Sin embargo, también se considera análisis de sentimiento a la clasificación en otras clases tales como: alegría, tristeza, etc. Esta tarea se puede realizar a niveles de oración, de documento, o de aspectos,

El análisis de sentimiento ha sido tratado utilizando distintos métodos desde su introducción como tarea de NLP. Actualmente, los mejores resultados han sido obtenidos por métodos basados en aprendizaje profundo (Zhang, Wang, y Liu, 2018). Sin embargo, los resultados en español aún están muy por debajo de los obtenidos en inglés. Además, no se pueden utilizar modelos entrenados con textos en inglés para el procesamiento de texto en español debido a que las reglas gramaticales varían entre idiomas.

La presente investigación aporta una posible solución a este problema mediante el uso de varios modelos de aprendizaje profundo. Específicamente, se toma en cuenta a la Red Neuronal Convolucional (Convolutional Neural Network, CNN), Red Neuronal Recurrente (Recurrent Neural Network, RNN) y sus variantes. Además, se establece como un antecedente para futuros trabajos de similar naturaleza.

La presente tesis se relaciona principalmente con el artículo revisado por pares **A Deep Learning Approach for Sentiment Analysis in Spanish Tweets** (Vizcarra, Mauricio, y Mauricio, 2018) ver Anexo A.

Los posteriores capítulos se han organizado de la siguiente manera: El Segundo capítulo contiene una revisión literaria de conceptos, definiciones y antecedentes de la investigación. En el Tercer capítulo se detalla los métodos utilizados. En el Cuarto Capítulo se presenta los resultados de la investigación y la comparación de los resultados frente a otros modelos utilizando los mismos recursos y métricas. Finalmente en el quinto y sexto capítulo se presentan las conclusiones y recomendaciones a considerar, respectivamente.

1.1 OBJETIVOS

1.1.1 Objetivo general

Desarrollar un modelo basado en aprendizaje profundo para el análisis de sentimiento de tuits en español que obtenga resultados competitivos frente a anteriores métodos utilizando los mismos recursos y métricas.

1.1.2 Objetivos específicos

- Implementar un algoritmo de preprocesamiento que mejore la calidad de las entradas para los modelos de aprendizaje profundo.
- Implementar modelos basados en aprendizaje profundo que potencialmente puedan resolver la tarea de análisis de sentimiento de tuits en español.
- Evaluar los modelos propuestos en relación a anteriores métodos, considerando métricas estándar.

CAPÍTULO II

REVISIÓN DE LITERATURA

Este capítulo contiene literatura para familiarizar al lector con el tema de investigación. La primera sección (2.1) comprende la introducción a los términos y conceptos que serán mencionados a lo largo de la tesis. La segunda sección (2.2) incluye los antecedentes que dieron origen a esta investigación. Los antecedentes se encuentran categorizados en nacionales e internacionales.

2.1 BASE TEÓRICA

Esta sección contiene una revisión de definiciones de los términos que se utilizan en las siguientes secciones y capítulos. Primero, se abarcan los temas relacionados a la variable de análisis de sentimiento de tuits en español. Seguidamente se definen los métodos relacionados al aprendizaje profundo.

2.1.1 Análisis de sentimiento

2.1.1.1 Procesamiento de Lenguaje Natural

El NLP es un área de investigación en ciencia de la computación e inteligencia artificial relacionada con el procesamiento del lenguaje humano (Lane, Howard, y Hapke,

2019). Desde una perspectiva científica, el NLP tiene como objetivo modelar mecanismos cognitivos en el lenguaje escrito, y en tareas humanas que implican el comprender y manipular el lenguaje humano (Deng y Liu, 2018).

Por otro lado, el NLP no es un campo de estudio nuevo, sino que comenzó a estudiarse desde que surgió el interés humano por desarrollar una interacción fluida entre un humano y un computador. Si se desea especificar un punto de inicio, este podría ser el momento en que Alan Turing propuso en su artículo Intelligence (Machinery, 1950) un test para determinar si una entidad es un computador o un humano.

Actualmente, se puede medir la importancia del NLP gracias a los beneficios obtenidos por sus aplicaciones. La gran cantidad de información existente pública o privada, que, en conjunto a la mejora en la potencia de calculo de los ordenadores posibilita que los computadores puedan procesar el lenguaje natural humano.

Niveles de Análisis en NLP El NLP trabaja utilizando diferentes métodos y técnicas para interpretar el lenguaje humano. Los niveles en esta área ayudan a comprender y analizar oraciones secuenciales. Por otro lado, para desarrollar un modelo de NLP no es necesario implementar todos los niveles, sino se determinan los niveles que deben ser desarrollados en el algoritmo en concordancia a las funciones del modelo (Covington, Grosz, y Pereira, 1994). Estos niveles son:

a) **Nivel de análisis morfológico:** en este nivel se extraen las raíces, sufijos, prefijos y otros elementos de las palabras. El objetivo de este nivel es entender como se constituyen las palabras como la unidad más pequeña denominada morfemas.

b) **Nivel de análisis sintáctico:** se analiza la estructura de las oraciones en base a un

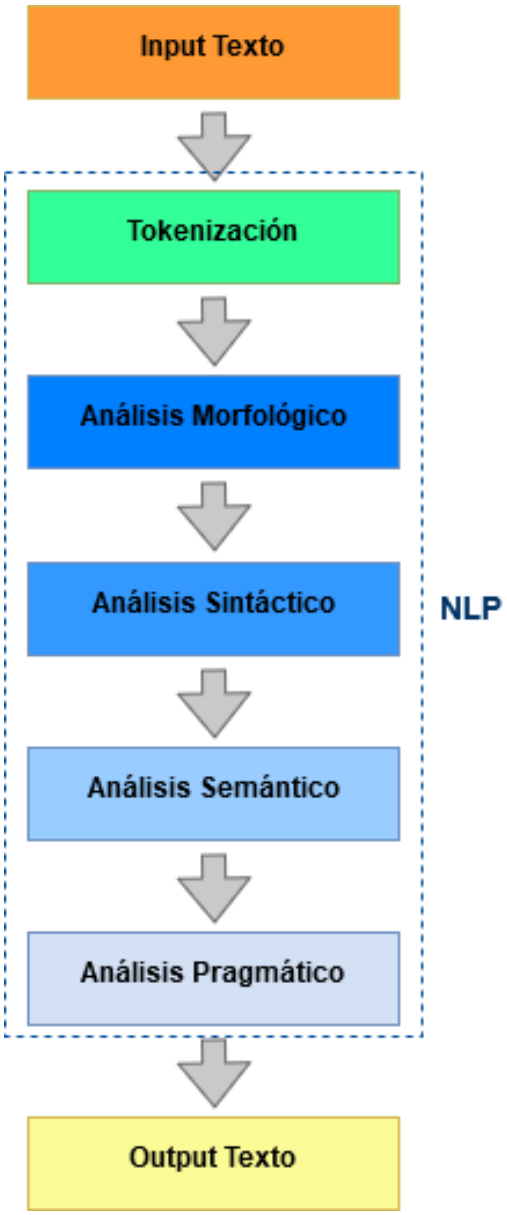
modelo gramatical. El objetivo es conocer el cómo se unen y ordenan las palabras para formar oraciones.

c) **Nivel de análisis semántico:** se analiza la semántica para proporcionar sentido a las oraciones y otorgarles un significado, además busca resolver los problemas estructurales y léxicos que pudieran surgir.

d) **Nivel de análisis pragmático:** se analiza textos compuestos por más de una oración, tomando de referencia aquellas oraciones inmediatamente anteriores y su relación entre ellas.

Un ejemplo de la implementación de un solo nivel es el trabajo de Abdel-Monem, Shaalan, Rafea, y Baraka (2019), en el cual se desarrolló un modelo de traducción automático multilingüe donde solo se implementó el nivel de análisis morfológico.

Figura 2.1: Flujo de trabajo en NLP según sus niveles



Elaborado por el equipo de trabajo

En la figura 2.1 se puede observar el flujo de trabajo en NLP con sus diferentes niveles de análisis anteriormente descritos. Usualmente al inicio del flujo, el texto es procesado por una técnica denominada **tokenización**. Con la tokenización se identifica las unidades mínimas de información conocidas como **tokens**, dividiendo las oraciones en unidades más pequeñas que como las palabras, signos de puntuación y otros elementos.

Tareas de NLP Las aplicaciones típicas del NLP incluyen (Pouyanfar et al., 2019; Yang, Luo, Chueng, Ling, y Chin, 2019): etiquetado del discurso, reconocimiento de entidades, traducción automática, búsqueda de respuestas, análisis de sentimiento, corrección automática de errores, extracción de información, búsqueda y recuperación de información, clasificación de texto, generación de lenguaje natural, entendimiento del lenguaje natural, análisis léxico, resumen de textos, entre otras. En el Tabla 2.1 se aprecia una lista categorizada de las aplicaciones de NLP según Hapke, Lane, y Howard (2019).

Tabla 2.1: Aplicaciones de NLP

Tarea	Web	Documento	Auto-completado
Edición	ortografía	gramar	estilo
Dialogo	chatbot	asistencia	planificación
Escritura	índice	concordancia	tabla de contenidos
E-mail	filtro de spam	clasificación	priorización
Extracción de texto	resumen	extraer conocimiento	diagnóstico médico
Leyes	inferencia legal	buscar precedentes	clasificar de citaciones
Noticias	detección de eventos	comprobar hechos	composición del titular
Atribución	detección de plagio	literatura forense	entrenamiento
Análisis de Sentimiento	seguimiento moral	reseña de productos	atención al cliente
Predicción	finanzas	elecciones	marketing
Escritura	guiones de películas	poesía	letra de canciones

Fuente: Hapke et al. (2019)

Algunas de las tareas de NLP más importantes en la literatura son:

a) **Análisis de sentimiento**

Se trata del análisis de textos con el objetivo de procesar y analizar las actitudes, opiniones, sentimientos y la subjetividad del texto de las personas (H. Liu y Zhang, 2018).

b) **Extracción de información**

Esta tarea se encarga de analizar un texto a fin de extraer información estructurada o semiestructurada que sea de interés. Por ejemplo Elton et al. (2019) en su investigación, usaron artículos y reportes científicos para extraer los conocimientos prácticos más relevantes.

c) **Traducción automática**

La traducción automática se trata de realizar la traducción de textos de un lenguaje a otro, para realizar esta tarea existe diversos tipos de técnicas el cual se definirá según el contexto de la traducción y de su enfoque.

d) **Recuperación de Información**

La recuperación de información abarca de distintas subtareas como son: organizar, almacenar, recuperar y evaluar información archivada en documentos textuales. Para realizar esta tarea, se hace uso de métodos estadísticos y NLP (Voorhees, 1999). Esto con el fin de que el usuario que haga uso de esta tarea encuentra información de manera más eficaz.

2.1.1.2 Análisis de sentimiento

El análisis de sentimiento es el proceso computacional para la identificación y clasificación del sentimiento, opinión, evaluación, actitud, valoración, afecto, emociones, subjetividad, detección de sarcasmo y otros que un sujeto expresa sobre una entidad en particular (Zhao, Liu, y Xu, 2016), y a su vez, lo asocia al lenguaje. Esta tarea principalmente se realiza mediante el análisis del lenguaje escrito, por lo que es un tema activo en NLP. Sin embargo, también ha sido abordado de otras formas como el reconocimiento de emociones basadas en las expresiones faciales en una modalidad visual y auditiva (Angadi y Reddy, 2019).

El término de análisis de sentimiento y minería de opinión son usados indistintamente para el análisis automático de opiniones sobre una entidad en particular. Según la Real Academia Española, se define sentimiento como el estado afectivo del ánimo, mientras que la opinión se define como el juicio o valoración que forma una persona respecto a un asunto en particular. Sin embargo, el término usado en los estudios en el idioma español es análisis de sentimiento, el cual es una clara referencia al único *workshop* sobre el campo en español Taller de Análisis de Sentimiento en la SEPLN (TASS), que desde el 2012 está bajo la responsabilidad de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN).

El análisis de sentimiento se centra principalmente en opiniones que expresan o implican sentimientos positivos o negativos. Una definición ampliamente extendida entre la comunidad científica es la enunciada por los investigadores Pang, Lee, et al. (2008) que definen el análisis de sentimiento como:

"Tratamiento computacional de las opiniones, sentimientos y fenómenos subjetivos en

textos."

El análisis de sentimiento se puede realizar en general, de dos maneras (D'Andrea, Ducange, Bechini, Renda, y Marcelloni, 2019):

a) **Opinión Regular**, el cual se divide en:

   - Directa, determina una opinión positiva o negativa sobre una entidad/elemento en específico.
   - Indirecta, da una opinión ambigua que afecta a una entidad/elemento.

b) **Comparación**, se hace una opinión relacionando las similitudes o diferencias entre dos o más entidades u elementos, esta comparación suele hacerse entre aspectos que ambas partes poseen.

Usualmente, para identificar un sentimiento se extrae los siguientes atributos:

- **Polaridad**: en donde el sujeto expresa una opinión positiva o negativa.
- **Subjetividad**: que representa el objeto del que se está argumentando.
- **Actor**: la persona o la entidad el cual expresa la opinión.

Por ejemplo, en el lanzamiento del teléfono móvil " Iphone 11". Un usuario, realiza un tuit mencionando que el teléfono le gustó. El actor es la persona que realizó el tuit, la polaridad es positiva y la subjetividad es alta.

Opinión La información de un texto puede clasificarse principalmente en: hechos y opiniones. Un hecho es la expresión objetiva sobre algo. Una opinión expresada de manera

textual generalmente es subjetiva ya que describe los sentimientos, emociones y valoraciones de una entidad hacia un tema. Además, una opinión textual comparte algunas características claves con una opinión que son, el objetivo y la polaridad. Una oración objetiva u subjetiva, se clasifica según su subjetividad. Por otro lado, una oración positiva, negativa o neutra, se clasifica mediante su nivel de polaridad.

Muchos estudios (H. Liu y Zhang, 2018) han observado que una opinión esta expresada por adjetivos, seguidos por verbos y expresiones compuestas. Esta observación clasifica a los adjetivos como la base de una opinión. Sin embargo, si solo se considera a los adjetivos surgen otros problemas los cuales son analizados por su polaridad.

Una palabra, al igual que una oración, puede ser positiva, negativa o neutra, pero esto depende de su contexto. Por ejemplo, si se considera el adjetivo alto. Si se habla de la duración de vida de una batería, tiene una orientación positiva; mientras si se refiere al precio de la misma batería tendría una orientación negativa.

Aplicaciones del análisis de sentimiento El análisis de sentimiento es un tema de gran interés el cual posee muchas aplicaciones ya desarrolladas. En los últimos años, ha sido usado por empresas, organizaciones, usuarios individuales los cuales realizan estudios sobre opiniones de personas para incrementar sus beneficios. La principal razón es que pueden ser usados para aplicaciones como: servicio al cliente, marketing, puntuación de productos, comentarios, relaciones públicas, etc. En efecto, son muchas las empresas que desarrollan herramientas para el análisis de sentimiento, tales como: Google, Microsoft, Amazon, eBay, SAS, Oracle, Adobe, Aylien, Bloomberg, NVidia, Facebook, Pinterest, Twitter, entre otras.

Asimismo, las aplicaciones del análisis de sentimiento se han diversificado en dis-

tintos campos posibles como son: salud, turismo, consumo, servicios, eventos sociales, y hasta elecciones políticas. Además de sus aplicaciones en tareas de la vida real, también existen otras aplicaciones que están orientadas a investigaciones. Los cuales usan información no estructurada extraída de foros, blogs y redes sociales.

Por ejemplo: Apple desea saber la opinión de su público objetivo después del lanzamiento de su teléfono móvil " Iphone 11". Para ello, se disponen a realizar análisis de sentimiento a nivel de aspecto en Twitter. Con ello, obtienen un reporte de las opiniones acerca de las características de su producto; que servirá de referencia para reorientar sus estrategias de marketing.

Niveles El análisis de sentimiento, bajo un enfoque granular, proporciona un marco conceptual para tareas de clasificación en el aprendizaje profundo (Li, Yuan, Wu, Xue, y Hu, 2019; H. Liu y Zhang, 2018; Zhang et al., 2018). Como es el caso de los clasificadores de imágenes, los cuales basados en información granulada hacen uso de técnicas de reconocimiento de los patrones de dicha imagen.

Figura 2.2: Clasificación de imagen y texto usando información granulada

Elaborado por el equipo de trabajo

En la imagen 2.2 se puede apreciar la clasificación tanto de imagen, como de texto

usando información granulada. En el caso de la imagen se hace el reconocimiento de un ave, en el caso del texto se realiza la clasificación por polaridad siendo catalogada como negativa.

En el campo del NLP un texto puede granularse en distintas unidades siendo su unidad básica la palabra (Zhao et al., 2016). En combinación con el análisis de sentimiento básicamente se cuenta con tres niveles de granularidad : nivel de documento, nivel de oración, nivel de aspecto.

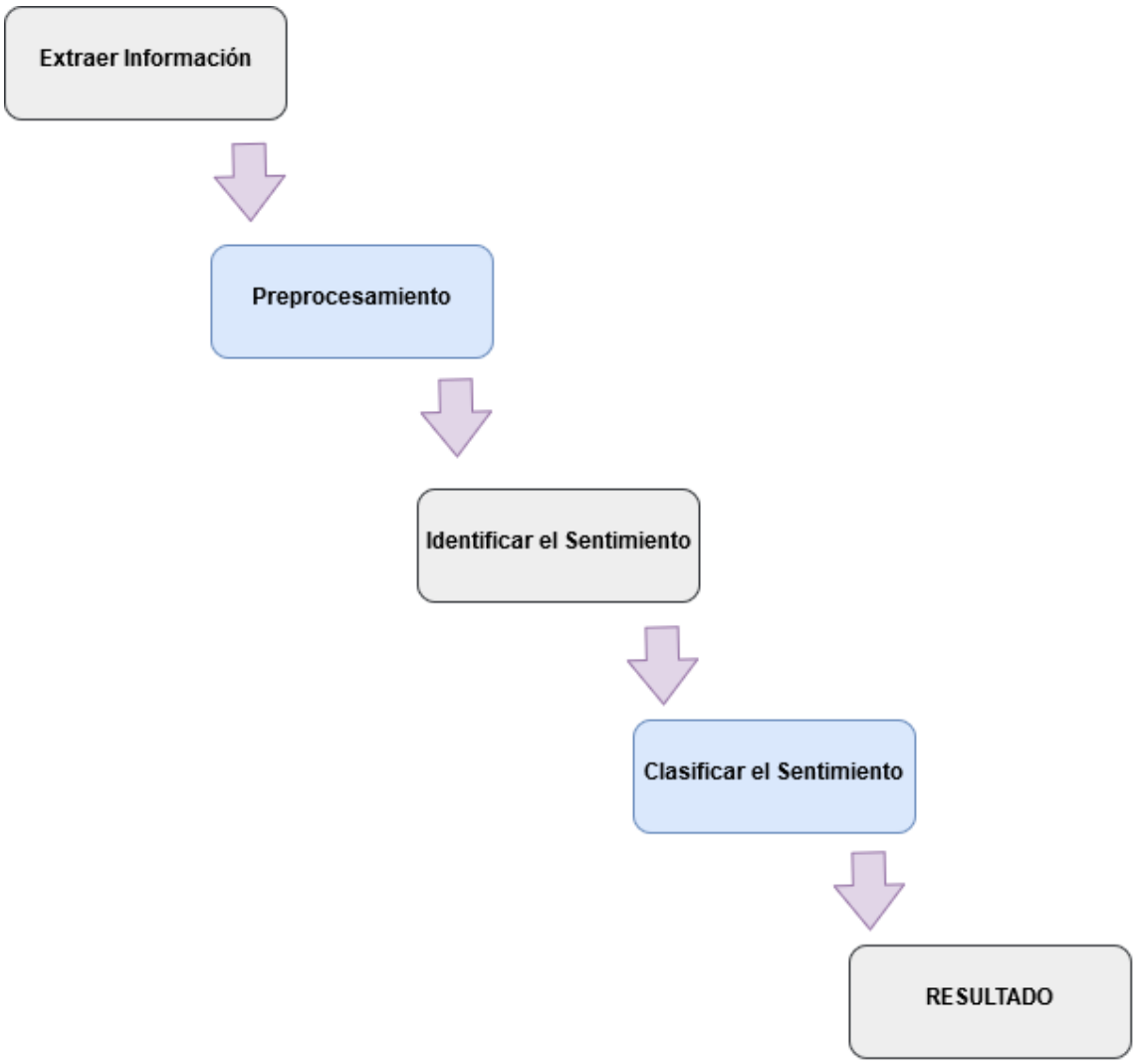
a) **Nivel de documento**, determina si un documento completo tiene una opinión general positiva o negativa.

b) **Nivel de oración**, clasifica la opinión de cada oración individual de un documento. Para esta tarea, primero se clasifica si la oración es subjetiva o objetiva. Por lo general, las oraciones objetivas son establecidas como una opinión neutral, y las oraciones subjetivas son clasificadas como opiniones positivas o negativas.

c) **Nivel de aspecto**, consiste en varias subtareas como son: extracción de aspecto, extracción de entidad y finalmente la clasificación de sentimiento. Un aspecto se refiere a un conjunto pequeño de palabras que definen a una entidad. Por ejemplo, de la oración: “Respecto al Restaurante Mandarín: el buffet criollo estuvo excelente, pero la atención pésima.” Se tiene al Restaurante Mandarín como la identidad, y como aspectos al buffet y la atención. La clasificación por aspectos deja a entender cuales son los aspectos por los que una persona prefiera un elemento más que otro.

Métodos clásicos Son muchas las metodologías de extraer y clasificar un sentimiento. La mayoría coincide que la forma general debe ser organizada siguiendo los pasos mostrados en la Figura 2.3:

Figura 2.3: Metodología general para el análisis de sentimiento



Elaborado por el equipo de trabajo

Preprocesamiento Antes de clasificar un texto es necesario realizar un preprocesamiento. El objetivo es limpiar el ruido. Esta tarea es importante aun más cuando lo datos son extraídos de redes sociales ya que estos suelen contener palabras con errores gramati-

cales o de caligrafía. Básicamente, esta tarea se dedica a la eliminación de espacios vacíos, eliminación palabras repetidas, normalización de palabras mayúsculas, eliminación de tildes, eliminación de números, eliminación de retuits, normalización de de risas, enlaces, hashtags, normalización de jergas.

Identificar el sentimiento Se trata de localizar la o las palabras posibles que contengan un sentimiento. Para realizar esta tarea se cuenta con reglas, algoritmos generales, y entre otros.

Clasificación de un sentimiento Una clasificación de polaridad puede ser en las categorías: positivo, negativo y neutro. Los pasos utilizados para resolver la tarea de análisis de sentimiento varían según el método que se plantea aplicar; ya sea clásico, de aprendizaje automático o aprendizaje profundo.

2.1.1.3 El problema con el análisis de sentimiento

Los problemas en su mayoría que se presentan en el análisis de sentimiento son heredados del NLP, sin embargo, son problemas propios del campo los cuales deben ser solucionados por los investigadores para poder obtener mejores resultados.

La **connotación**, es uno de los problemas en análisis de sentimiento que se refiere al significado de una palabra el cual varia según el contexto. Por ejemplo, el verbo " comer"no tiene el mismo significado que la frase " es pan comido".

Otro problema, es la **detección de sarcasmo e ironía** los cuales incluso no son detectados por humanos en ocasiones. Por ejemplo, "¡Estupendo, mi batería cada día dura menos!". El sentimiento por la batería no es positiva sino negativa.

2.1.1.4 Análisis de sentimiento en Twitter

Twitter es un *microblogging* que permite a sus usuarios enviar mensajes cortos sobre cualquier tema en un máximo de 280 caracteres llamados tuits o *tweets*. Twitter posee de una gran popularidad mundial, esto debido a su sencillo manejo, a que publica más de 65 millones de tuits al día y abarca diversos temas de interés. Por lo que, Twitter ha sido usada en una variedad de propósitos. Los mensajes en Twitter son públicos y para tener acceso a su información no es necesario presentar ningún tipo de identificación. Por lo tanto, Twitter representa una gran fuente de información que puede ser aprovechada por los miles de usuarios de internet.

En tal sentido, el análisis de sentimiento en Twitter es una tarea que se dedica a determinar la polaridad de sentimiento de los mensajes publicados a través de esta red social. En efecto, el análisis de sentimiento en Twitter ofrece a las organizaciones una forma rápida y eficaz de monitorear el sentimiento público respecto a la marca, negocio, dirección, entre otros (Saif, He, y Alani, 2012).

2.1.1.5 Análisis de sentimiento en tuits en español

El objetivo de esta tarea consiste en determinar la polaridad de un tuit en español. La polaridad tiene una escala de cuatro niveles: P, N, NEU y NONE. La mayor dificultad que tiene esta tarea es el análisis de texto informal. Esto conlleva a errores de ortografía y el uso de emoticonos, así como la falta de contexto debido al límite en la longitud de caracteres.

Pese a que TASS no publica un corpus nuevo cada año, existen cuatro corpus que son usados para el entrenamiento y para la evaluación del modelo como clasificador.

a) **Corpus general del TASS**, incluye 7219 tuits en español, escritos por personalidades y celebridades conocidas dentro del mundo de la política, economía, comunicación y cultura. Incluye 60798 tuits de testing.

b) **Social-TV Corpus**, este corpus fue recolectado durante el 2014 en la final de Copa del Rey en España entre el Real Madrid y F.C. Barcelona. Durante aquel partido, se generaron más de un millón de tuits en el lapso de 15 minutos antes del partido hasta 15 minutos después de culminado. Los tuits generados fueron filtrados quedando un total de 2773 tuits. La polaridad en este corpus consta de tres niveles: positivo, negativo y neutro. Para el entrenamiento se tiene 1773 tuits y para la validación 1000 tuits.

c) **Corpus of Spanish Tweets for Opinion Mining at aspect level about Politics (STOMPOL)**, este es un corpus para el análisis de sentimiento a nivel de aspectos. El tema abarcado en los tuits es la campaña política en las elecciones regionales y locales del 2015 en España. Los tuits fueron recolectados del 23 al 24 de Abril enfocándose temas de interés electoral: economía, sanidad, educación, propio partido y otros aspectos. Respecto a la polaridad, se consideraron tres niveles: positivo, negativo y neutro. El corpus consiste de 1 284 tuits, dividido en dos partes 784 tuits para el entrenamiento y 500 tuits para la validación.

d) **International TASS Corpus (InterTASS)**, en este corpus la polaridad se anota en cuatro niveles: P, N, NEU y NONE. Está compuesto por 1008 tuits para el entrenamiento, 506 tuits para la validación y 1899 tuits para test.

2.1.1.6 Métricas

Para determinar el rendimiento de un algoritmo es necesario contar con medidas, en el caso del análisis de sentimiento se considera cuatro posibles estados los cuales podemos visualizar en la tabla 2.2 en el caso de una correcta o incorrecta clasificación, pudiendo tener un etiquetado acertado o no. Para entender estos cuatro estados sea una Clase X.

- **Verdaderos Positivos** (True Positives, TP), que han sido etiquetados de manera correcta y pertenecen a la Clase X.
- **Falsos Positivos** (False Positives, FP), que fueron etiquetados en la Clase X, pero esta clasificación es incorrecta.
- **Verdaderos Negativos** (True Negatives, TN), aquellos que no pertenecen a la Clase X, y su clasificación es correcta.
- **Falsos Negativos** (False Negatives, FN), son aquellos que fueron etiquetados de manera incorrecta y deberían pertenecer a la Clase X.

Tabla 2.2: Visualización de los estados de clasificación.

	Positivo (1)	**Negativo (0)**
Positivo (1)	TP	FP
Negativo (0)	FN	TN

Fuente: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Al evaluar la calidad de un trabajo en análisis de sentimiento se cuenta con métricas ya definidas las cuales son: Exactitud (*Accuracy*), Precisión (*Precision*), Exhaustividad (*Recall*), Valor F1 (*F1 score*).

Exactitud esta medida es el número de ejemplos positivos clasificados correctamente dividido por el número de total de ejemplos etiquetados por el modelo. **Exhaustividad** es el número de ejemplos positivos clasificados correctamente dividido por el número total de ejemplos positivos en los datos. **Valor-F1** es usado medir la eficiencia de un modelo y corregir el error de distancia en los casos de exactitud y exhaustividad. Para ello el Valor F1 combina las dos medidas anteriores ponderadas por el parámetro β, por lo general el valor de β tiene un valor igual a 1 tanto en exhaustividad como en precisión (Sokolova y Lapalme, 2009)

Basándonos en los cuatro anteriores estados, podemos definir las medidas como se muestra en el siguiente Tabla 2.3:

Tabla 2.3: Cálculo de las métricas de análisis de sentimiento

Métrica	**Fórmula**
Exactitud	$\frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$
Precisión	$\frac{\text{TP}}{\text{TP} + \text{FP}}$
Exhaustividad	$\frac{\text{TP}}{\text{TP} + \text{FN}}$
Valor F1	$\text{F}\beta = (1 + \beta * \frac{\text{Precisión* Exhaustividad}}{(\beta * \text{Precisión}) + \text{Exhaustividad}}$

Elaborado por el equipo de trabajo.

2.1.2 Aprendizaje profundo

El aprendizaje profundo es un método de aprendizaje automático que ha obtenido mejores resultados que los métodos clásicos en distintas áreas de aplicación. En esta sección, se presenta conceptos de los términos relacionados al aprendizaje profundo.

2.1.2.1 Aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial, la cual se caracteriza porque sus algoritmos aprenden de los datos. Los algoritmos de aprendizaje automático se basan en modelos matemáticos y se plantean para resolver una **tarea** específica. El proceso de aprendizaje de un algoritmo es comúnmente llamado **entrenamiento**. El conjunto de datos de los cuales se aprende se denomina **dataset** o **corpus**. Estas bases de datos contienen un conjunto de n ejemplos o **entradas**. Cada entrada es representada como un vector de características, en el que cada elemento describe un atributo de la entrada (Ruder, 2019). Estos ejemplos son independientes entre sí y conforman la distribución p_{data}.

La base de datos utilizada para el aprendizaje se divide en tres partes: **datos de entrenamiento**, **datos de prueba** y **datos de validación**. El conjunto de entrenamiento es donde se aplica el algoritmo de aprendizaje. Los datos de validación se utilizan para medir el progreso del aprendizaje durante el entrenamiento. Y los datos de prueba se utilizan para medir el desempeño final del algoritmo entrenado. Los datos de validación y prueba no se utilizan para entrenar el algoritmo.

Para evaluar las capacidades de un algoritmo de aprendizaje automático, se debe realizar una medida cuantitativa de su desempeño (Goodfellow, Bengio, y Courville,

2016). A esta medida se le denomina generalmente **métrica** y varía según la tarea que se desea resolver.

2.1.2.2 Tipos de aprendizaje

Se puede clasificar a los algoritmos de aprendizaje automático según la forma de aprendizaje, lo cual hace cambiar también los datos que el algoritmo necesita para aprender; principalmente, la clasificación consiste en **aprendizaje supervisado** y **aprendizaje no supervisado**.

En el aprendizaje supervisado se optimiza la probabilidad de que una distribución de datos pertenezca a una **etiqueta** determinada. De este modo, su conjunto de datos de entrenamiento contiene un ejemplo asociado a una etiqueta. Esta etiqueta es la que se desea aprender a predecir. Así, la etiqueta es discreta en tareas de clasificación, y continua en problemas de regresión.

El aprendizaje no supervisado trata de aprender las etiquetas de cada ejemplo sin tener acceso a ellas. Para tal efecto, sus algoritmos se enfocan en descubrir las similitudes y diferencias entre los datos de entrenamiento, para así poder agruparlos en sus respectivas clases.

2.1.2.3 Función objetivo

Los algoritmos de aprendizaje automático necesitan una función objetivo que deben minimizar o maximizar para aprender. Lo que se desea maximizar es la probabilidad de predecir la etiqueta correcta (p_{model}) dada una entrada, esto es equivalente a minimizar el término en negativo. De acuerdo a Ruder (2019), es la función objetivo de la Estimación

de Máxima Verosimilitud (Maximum Likelihood Estimation, MLE)

$$\theta_{MLE} = \arg\min_{\theta} -\mathbb{E}_{x\sim\hat{p}_{data}}[\log p_{model}(x_i;\theta)] \tag{2.1}$$

Además, siguiendo a Ruder (2019) esto es equivalente también a minimizar la entropía cruzada entre la distribución correcta p_{data} y la salida del algoritmo p_{model}

$$\theta_{MLE} = \arg\min_{\theta} H(\hat{p}_{data}, p_{model}) \tag{2.2}$$

Cuando la función objetivo debe ser minimizada se le llama también **función de costo** o **pérdida**.

2.1.2.4 Descenso de gradiente

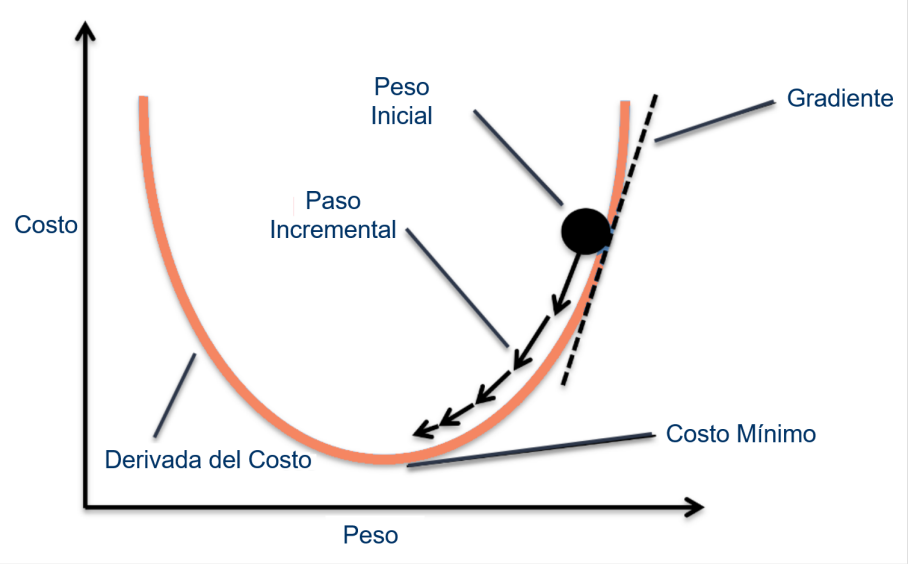
Para que el algoritmo pueda aprender, se debe optimizar una función objetivo. Asimismo, el descenso de gradiente optimiza los parámetros de un modelo de aprendizaje automático con respecto al mínimo de una función de costo. Estos parámetros son también llamados **pesos** en una red neuronal y **coeficientes** en una regresión lineal.

El proceso iterativo de optimización consta de dos partes principales: primero, se calcula la gradiente de la función de costo en el punto actual, obteniendo la pendiente. Seguidamente, se actualiza los parámetros del modelo en la dirección opuesta a la gradiente calculada. Este proceso se repite hasta obtener un costo mínimo.

La Figura 2.4 muestra el proceso de descenso de gradiente considerando solo un parámetro. Cabe resaltar que las funciones de costo respecto a los parámetros usadas en los algoritmos de aprendizaje automático son mucho más complejas, dependiendo del

número de parámetros del modelo y de la complejidad de la tarea.

Figura 2.4: Procedimiento del descenso de gradiente



Fuente: Lanham (2018)

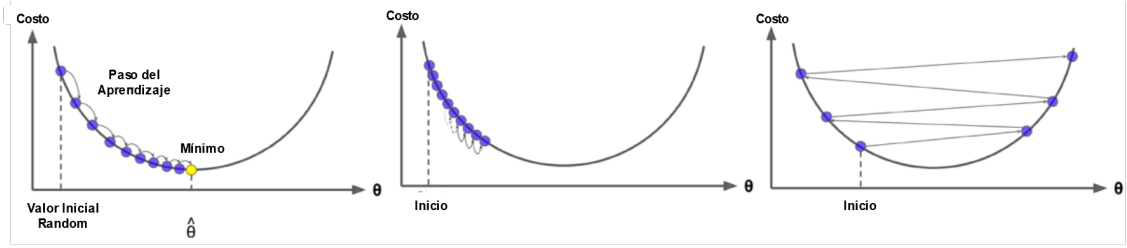
La actualización de los parámetros θ está definida como

$$\theta = \theta + \alpha \nabla_\theta J(\theta) \tag{2.3}$$

donde $J(\theta)$ es la función de costo a minimizar, ∇_θ es la gradiente con respecto a los parámetros y α es la tasa de aprendizaje del modelo. La **tasa de aprendizaje** define en cuánto se van a actualizar los parámetros. Es por esta razón que es muy importante definir una tasa de aprendizaje adecuada para lograr la optimización de los parámetros.

En la Figura 2.5 se muestra la variación en el proceso de optimización de parámetros al cambiar la tasa de aprendizaje. Una tasa de aprendizaje muy grande hará que el modelo diverja y sea muy variable, en contraste, una tasa de aprendizaje muy pequeña hará que el modelo aprenda muy lentamente.

Figura 2.5: Optimización usando diferentes tasas de aprendizaje



Fuente: Géron (2017)

La actualización de parámetros y cálculo de la gradiente se pueden realizar de varias formas.

- El **descenso de gradiente por lote**, también llamado **por *batch***, calcula la gradiente de la función de costo y actualiza los parámetros usando todos los ejemplos del dataset. Esto es más costoso computacionalmente, debido a que se calculan las gradientes para cada ejemplo del dataset a la vez. Sin embargo, hace que el entrenamiento sea más estable.

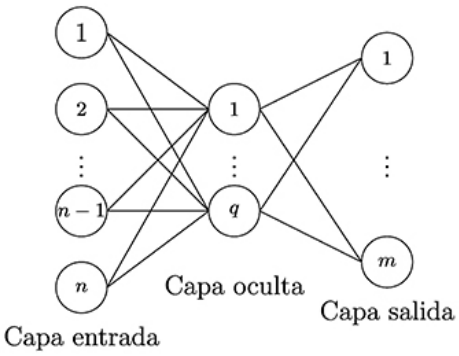
- El **descenso de gradiente estocástico** procesa el costo y hace la actualización usando un único ejemplo. De este modo, este procedimiento se repite para cada ejemplo en el dataset. Esto es menos costoso, pero puede producir más ruido en el entrenamiento y hacerlo inestable.

- El **descenso de gradiente por mini-lotes**, denominado también **por mini-batches**, realiza un procesamiento de varios ejemplos a la vez, tomando en cuenta una cantidad mayor a uno y menor a la del dataset. Es el tipo de descenso de gradiente más utilizado. Debido a esto, se ha hecho común llamarlo únicamente **descenso de gradiente por lotes** o **por batches**.

2.1.2.5 Red neuronal

La red neuronal es el método base del aprendizaje profundo. Una red neuronal es un conjunto de funciones lineales intercaladas con funciones de activación no lineales. La unidad básica de una red neuronal es la **neurona**. Una neurona realiza una combinación lineal de sus entradas y una función de activación. Al conjunto de neuronas que reciben el mismo grupo de entradas se le denomina **capa**. Las neuronas de una misma capa también generan salidas que servirán como entrada de la siguiente capa.

Figura 2.6: Ejemplo de un perceptrón multicapa



Fuente: Vivas, Martínez, y Pérez (2014)

La Figura 2.6 muestra la arquitectura de un perceptrón multicapa. Esta es la red neuronal más representativa y consta de tres tipos de capas:

- La **capa de entrada** recibe la representación numérica de los datos que se procesarán.
- La **capa intermedia** o **capa oculta** realiza una combinación lineal de las salidas de la anterior capa y luego ejecuta una función de activación. Cabe señalar que una red puede tener varias capas ocultas en su arquitectura.
- La **capa de salida** de la misma forma recibe como entradas el resultado de la ante-

rior capa. Sin embargo utiliza una función de activación diferente. Generalmente las funciones de activación de la capa de salida realizan una normalización las salidas en una distribución de probabilidades.

El proceso que consiste en la entrada a la red, el cálculo secuencial de todas las capas ocultas de la red hasta obtener un resultado de la capa de salida se conoce como **propagación hacia adelante**.

2.1.2.6 Aprendizaje profundo

El aprendizaje profundo es un método dentro del aprendizaje automático, el cual aprende características de los datos de forma jerárquica. Al respecto, LeCun, Bengio, y Hinton (2015) indican que las características de menor nivel se agrupan para formar características de alto nivel. Así, para aprender características de alto nivel se debe haber aprendido las características de menor nivel. De este modo, se conforma una jerarquía de características que, de acuerdo a Goodfellow et al. (2016), es profunda y de ahí proviene su nombre.

Al procesar una imagen con aprendizaje profundo, las primeras capas de la red neuronal aprenden las características más simples tales como vértices y bordes. Las siguientes capas aprenden las agrupaciones de estas carácterísticas, formando representaciones más complejas. De esta forma se logra abstraer todo lo que contiene la imagen.

El perceptron multicapa es la arquitectura más básica y representativa en el aprendizaje profundo. A partir de este modelo, se han generado modificaciones y nuevos modelos de acuerdo a la naturaleza de los datos y a la tarea a resolver.

2.1.2.7 Retro-Propagación

En la sección anterior, se ha mencionado acerca del uso de la gradiente para la optimización de los parámetros en un algoritmo de aprendizaje automático. En ese mismo contexto, la red neuronal utiliza la **retro-propagación** (Rumelhart, Hinton, Williams, et al., 1988) para optimizar sus parámetros. La retro-propagación utiliza las derivadas parciales y la regla de la cadena del cálculo con un orden específico de operaciones.

Sea una red neuronal de profundidad D con parámetros $W, b \in \theta$ que procesa una entrada x y produce una salida $\hat{y}$. W_i y b_i son los pesos de la capa $i \in \{1, ..., D\}$. La función de costo $L(\hat{y}, y)$ es calculada usando las etiquetas y. La propagación hacia adelante está definida como:

$$\begin{aligned} a_i &= W_i h_{i-1} + b_l \\ h_i &= \sigma(a_i) \end{aligned} \tag{2.4}$$

donde $h_0 = x$ y $\hat{y} = h_D$. Para calcular las gradientes en cada capa, se comienza con la capa de salida.

$$\nabla_{\hat{y}} J = \nabla_{\hat{y}} L(\hat{y}, y) \tag{2.5}$$

Después de eso, se calcula la gradiente de la función previa a la activación usando a regla de la cadena

$$\nabla_{a^{(i)}} J = \nabla_{\hat{y}} J \odot \sigma'(a^{(i)}) \tag{2.6}$$

Por esa razón las funciones de activación deben ser diferenciables. Seguidamente se cal-

cula las gradientes para los parámetros W_i y b_i

$$\begin{aligned} \nabla_{W^{(i)}} J &= \nabla_{\hat{y}} J \; h^{(i-1)\top} + \lambda \nabla_{b^{(i)}} \\ \nabla_{b^{(i)}} J &= \nabla_{\hat{y}} J + \lambda \nabla_{b^{(i)}} \end{aligned} \tag{2.7}$$

Finalmente se propagan las gradientes a la siguiente capa de activación.

$$\nabla_{h^{(i-1)}} J = W^{(i)\top} \; \nabla w_i J \tag{2.8}$$

2.1.2.8 Funciones de activación

Las funciones de activación, también llamadas funciones de transferencia, son las que definen la salida de una neurona. Estas funciones son no lineales y deben ser diferenciales para optimización. Una función de activación puede ayudar en el aprendizaje de distribuciones complejas y proveer predicciones acertadas. Dentro de los tipos de funciones de activación se encuentran:

ReLU La función Unidad Lineal Rectificada (Rectified Linear Unit, ReLU) o función rampa, fue utilizada por primera vez en el área de inteligencia artificial por Nair y Hinton (2010). Esta función es similar a una función identidad, sin embargo utiliza un límite inferior definido en cero para los números negativos. Matemáticamente, la función ReLU se define como:

$$f(x) = \text{máx}(0, x) \tag{2.9}$$

Donde x es la entrada. Si $x < 0, R(x) = 0$ y $x > 0, R(x) = x$.

Sigmoidea La función sigmoidea fue propuesta por primera vez por Verhulst (1838) y se ha utilizado en el campo del aprendizaje automático debido a su derivación simple. Esta función es mayormente utilizada en las neuronas de salida para realizar clasificación binaria. La función sigmoidea se define por la siguiente fórmula:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.10}$$

Donde la función toma de entrada cualquier números real y su salida se sitúa en un intervalo de [0,1].

Softmax La función softmax fue propuesta por Bridle (1990). Esta función es similar a la sigmoidea debido a la normalización que aplica. Esta función no solamente realiza un mapeado de los valores entre cero y uno, sino que también hace que los valores normalizados sumen un total de uno (1). Esta función se utiliza como salida probabilística en problemas de clasificación con múltiples clases. La función softmax se define de la siguiente forma:

$$f(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.11}$$

Donde z es un vector de las entradas de la capa de salida y i indexa a las unidades de salida como $i = 1, 2, 3, ..., j$

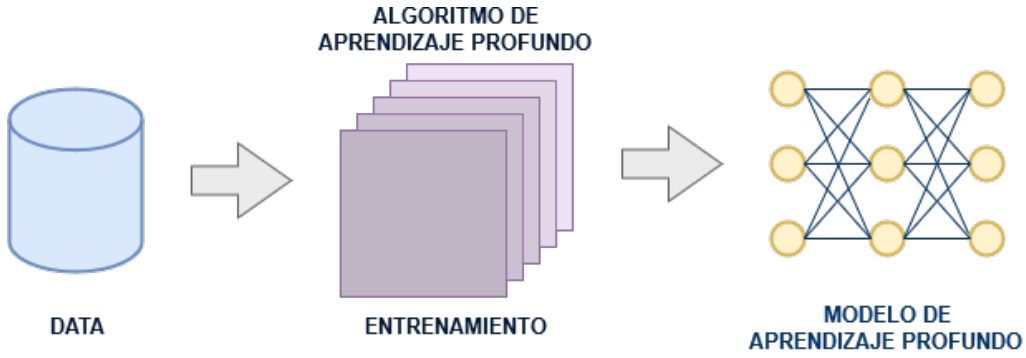
2.1.2.9 Modelo en aprendizaje profundo

Actualmente, no existe una definición exacta para un modelo en aprendizaje profundo. Sin embargo, suele ser interpretado como una representación matemática de un proceso del mundo real. Por lo general, un modelo para aprender lo hará mediante un entrenamiento haciendo uso de grandes conjuntos de datos etiquetados y arquitecturas de redes neuronales. Algunas de las definiciones de un modelo de aprendizaje profundo, son:

- Para TensorFlow (2019) en su guía para principiantes en Aprendizaje Profundo. Define un modelo como la relación entre las características y la etiqueta.
- En el contexto de redes neuronales, un modelo de acuerdo con Jones (2017) consiste en capas de redes interconectadas a través de sus pesos que en consecuencia alteran las entradas.
- Según Amazon (2019), un modelo de aprendizaje profundo es un modelo matemático. Dicho modelo genera predicciones utilizando padrones extraídos de sus datos. Amazon a su vez subdivide un modelo de aprendizaje profundo en tres tipos:
  - **Modelo de clasificación binaria** predice resultados binarios. Para entrenar este tipo de modelo se utiliza la regresión logística.
  - **Modelo de clasificación multiclase** busca generar predicciones de multiclases, es decir, predecir más de uno de dos resultados. Para entrenar este tipo de modelo se utiliza la regresión logística multinomial.
  - **Modelo de Regresión** predice un valor numérico. Para entrenar este tipo de modelo se utiliza la regresión lineal.

Para crear un modelo de aprendizaje profundo se debe seguir una serie de procedimientos los cuales se pueden ver en la Figura 2.7, en donde previamente se debe crear una fuente de datos. Posteriormente, se crea el modelo el cual surge tras ejecutar el algoritmo de aprendizaje profundo sobre los datos utilizados para el entrenamiento con el fin de aprender.

Figura 2.7: Definición de un modelo de aprendizaje profundo



Elaborado por el equipo de trabajo

Se puede crear un nuevo modelo de aprendizaje profundo usando el mismo algoritmo con datos distintos. También, es posible crear un modelo diferente usando los mismos datos, pero utilizando un algoritmo de aprendizaje profundo distinto.

Para construir un modelo de aprendizaje profundo, es necesario observar y analizar el problema que se plantea resolver y los datos que se utilizarán para entrenar la red neuronal. Esto se debe a que la arquitectura y complejidad de la red se diseña exclusivamente por las características del problema y la naturaleza de sus datos.

Tener una métrica para el modelo es muy importante, debido a que es la única forma de evaluar el desempeño del modelo y plantear mejoras a su diseño. La elección de la métrica que se utiliza depende principalmente del problema o tarea que se plantea resolver. Por ejemplo, el error cuadrático medio es una buena métrica para un problema de regresión, sin embargo no es una buena métrica para un problema de clasificación.

2.1.3 Herramientas de desarrollo

Para diseñar un modelo de aprendizaje profundo es necesario utilizar un lenguaje de programación que tenga la versatilidad para manejar distintos tipos de datos de forma eficiente. De este modo, C++ es uno de los lenguajes de programación más eficientes al momento de realizar algoritmos y por otro lado Java es uno de los más ineficientes, por lo tanto, C++ se consideraría una buena opción. Sin embargo, también es necesario contar con el respaldo de una comunidad activa que contribuya desarrollando paquetes y librerías para tal lenguaje. Estas librerías sirven para que los desarrolladores e investigadores no tengan la necesidad de implementar todos los algoritmos desde cero y se puedan concentrar en realizar mejoras. Actualmente, las librerías que tienen mayor actividad y popularidad en el desarrollo de redes neuronales son: Pytorch, TensorFlow y Keras. Todos ellos son desarrollados en el lenguaje de programación Python. En el presente trabajo de investigación se utilizó el lenguaje de programación Python.

2.1.3.1 Python

Python es un lenguaje de programación simple pero potente que gracias a sus funcionalidades es óptimo para tareas de procesamiento de datos lingüísticos. Originalmente Python fue diseñado a principios de 1990 por Guido van Rossum en el Centrum Wiskunde & Informatica como sucesor de un lenguaje de programación llamado ABC (Van Rossum y Drake Jr, 2009). Actualmente, es gratuito y se puede descargar desde: https://www.python.org, en donde también se puede acceder a la documentación e interactuar con su comunidad.

Su rápido y fácil desarrollo además como su evaluación lo hace menos complejo

comparado a otros lenguajes de programación, lo cual, lo convierte en un lenguaje de programación sencillo de usar para principiantes y expertos en NLP (Thanaki, 2017). Asimismo, Python al ser un lenguaje de programación multiparadigma es accesible para distintos estilos de programación como la programación orientada a objetos. Otra ventaja es su entorno el cual fue heredado de un interfaz escrito en C, C++ y otros lenguajes lo cual permite una programación e interfaz paralela.

Además, Python tiene un núcleo pequeño y puede ampliarse importando librerías externas. Python tiene muchas librerías orientadas para casi todas las tareas en un proyecto de Inteligencia Artificial. Estas librerías ayudan al ahorro de tiempo a los programadores en la codificación de algoritmos básicos. Las librerías de redes neuronales que actualmente cuentan con actualizaciones y soporte son:

- **TensorFlow**: Es la librería de código abierto desarrollada por Google. Esta librería se especializa en el manejo de datos matemáticos que se usan en aprendizaje automático como las redes neuronales. Entre sus funcionalidades principales, se encuentra la implementación de diferenciación automática y de la retro-propagación. Utiliza un grafo estático que permite acelerar el aprendizaje.

- **Pytorch**: Es la librería de código abierto para aprendizaje automático desarrollado por Facebook AI. Se basa en la librería Torch (implementada en el lenguaje de programación Lua). Tiene funcionalidades muy similares a TensorFlow. A diferencia de TensorFlow, Pytorch utiliza un grafo dinámico que le permite flexibilidad en el proceso de aprendizaje.

- **Keras**: Es una librería de alto nivel de código abierto para redes neuronales. Se utiliza como un frontend de TensorFlow, lo que permite una rápida implementación

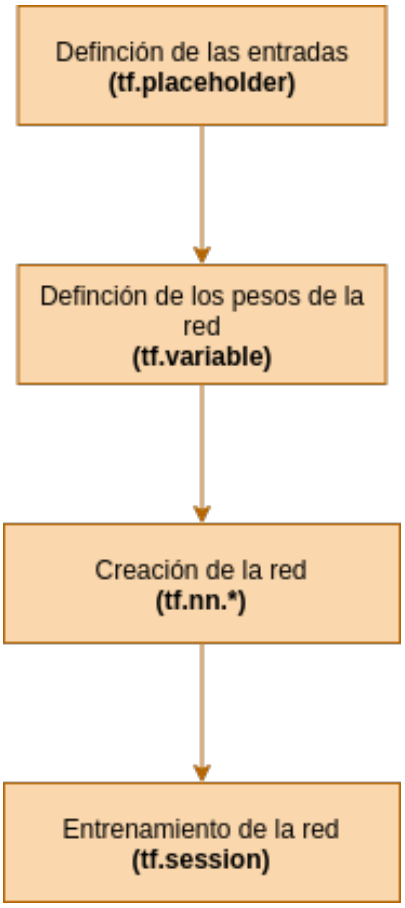
de redes neuronales profundas. Sin embargo, no brinda libertad para hacer mejoras o implementar módulos nuevos.

A fin de tener libertad en la implementación para poder realizar mejoras y más experimentos en los modelos, se descartó Keras. Para tener un adecuado manejo de datos y un grafo estático para definir las operaciones adecuadamente, en esta tesis se utilizó la librería TensorFlow.

2.1.3.2 Tensorflow

Tensorflow (Abadi et al., 2016) es una librería que maneja un flujo de trabajo basado en sesiones, placeholders y variables (pesos) para aprender. Asimismo, está optimizada para el procesamiento y entrenamiento de modelos de aprendizaje automático en varios dispositivos computacionales: Unidad Central de Procesamiento (Central Processing Unit, CPU) multicore, Unidad de Procesamiento Gráfico (Graphics Processing Unit, GPU), y Unidad de Procesamiento Tensorial (Tensor Processing Unit, TPU).

Figura 2.8: Creación de una red neuronal en TensorFlow



Elaborado por el equipo de trabajo

El procedimiento que se sigue para la creación y entrenamiento de una red neuronal en TensorFlow se describe en la Figura 2.8, donde "tf.es la librería TensorFlow. Los procesos mencionados en la Figura 2.8 se describen a continuación:

- **Definición de entradas**: Como primer paso, se define la forma de la matriz de entradas a la red. Para ello, la clase **tensorflow.placeholder** se utiliza para colocar las entradas de la red, estas entradas no son modificadas en el momento de la actualización de pesos.

- **Definición de los pesos de la red**: La red neuronal posee pesos los cuales se irán modificando durante el entrenamiento. La clase **tensorflow.variable** implementa

una matriz que se ve afectada por la actualización de pesos.

- **Creación de la red**: Luego de definir las entradas y los pesos de la red, se configuran las operaciones que se realizarán en la propagación hacia adelante hasta obtener una salida. En esta etapa también se define el algoritmo de optimización que se utilizará, las métricas que servirán para monitorear a la red y otros hiperparámetros. El paquete **tensorflow.nn** implementa un conjunto de varias operaciones de distintos tipos y complejidades.

- **Entrenamiento de la red**: Finalmente se realiza la creación de la sesión, la inicialización de variables y el entrenamiento de la red. La clase **tensorflow.session** maneja el entrenamiento de una red neuronal en TensorFlow.

2.2 ANTECEDENTES

2.2.1 Antecedentes nacionales

Ramos, Rafael, y Huamaní (2017) realizaron un estudio de clasificación de tuits. Tal clasificación se realizó bajo dos criterios: clasificación de subjetividad y análisis de sentimiento. Para ambas tareas, utilizaron clasificadores basados en máquinas de vectores de soporte (SVM) y para la clasificación de subjetividad agregando reglas creadas manualmente. Utilizaron dos clasificadores para cada tarea, uno para el texto del tuit y otro para su localizador de recursos uniforme (URL). Se infiere que el clasificador de URL ayude a mejorar los resultados debido a que existen usuarios más propensos a expresar alguna clase de opinión específica. Sin embargo, no existe antecedentes que utilicen URL como parte de la clasificación.

Olivares (2016) reportó los beneficios alcanzados en el análisis de sentimiento haciendo uso de ontologías de dominio, para lo cual, analizó el estado del arte respecto al aprovechamiento de la variedad expresiva orientado a la representación de las emociones humanas. Concluyendo, que al aplicar ontologías de dominio a distintas investigaciones de análisis de sentimiento obtenemos 9 conflictos y 22 beneficios. Finalmente, de ellos resalta 3 beneficios: soporte en representaciones estructuradas de opiniones, respecto a la polaridad mayor precisión e integridad y soporte en la representación de modelos emocionales complejos.

2.2.2 Antecedentes internacionales

Souma, Vodenska, y Aoyama (2019) exploraron el poder predictivo del análisis de sentimiento usando el aprendizaje profundo y definiendo las polaridades de sentimientos en función al stock de un producto antes y después de la publicación de la noticia del producto. Para tal tarea, usando un corpus de Wikipedia y Gigaword de 2014 los cuales fueron convertidos en *Word Embeddings* utilizando la técnica de vectores globales (GloVe) los cuales fueron usados como entradas en la arquitectura híbrida de aprendizaje profundo entre una CNN y una RNN con celdas de Memoria de Corto y Largo Plazo (Long Short-Term Memory, LSTM) para el entrenamiento de datos. Sus resultados muestran que el método de creación de datos de entrenamiento propuesto es eficiente frente a otras noticias similares. Además, proponen cambios a las entradas de la red y otros modelos basados en aprendizaje profundo como futuras mejoras a su modelo.

Montanés, Aznar, del Hoyo, y de Big Data (2018) analizaron los modelos actuales en el estado del arte del aprendizaje profundo usados para el análisis de sentimiento. Partiendo de diversas pruebas realizadas con la intención de lograr un híbrido con mayor

efectividad, se obtuvo que este debía ser un modelo híbrido entre una CNN y una red LSTM. Los resultados muestran que un algoritmo híbrido entre ambos modelos presenta una mayor exactitud total (*accuracy*) en comparación a cuando son trabajados individualmente. Sin embargo, el modelo propuesto no es capaz de generalizar correctamente a los datos de validación y test. Mencionan que su modelo, al ser más complejo, requiere más datos de entrenamiento.

Luque y Pérez (2018) trabajaron en la clasificación de la polaridad de tuits en el idioma español. Se enfocaron en el preprocesamiento y representación de las palabras. Entrenaron *Word Embeddings* basados en subpalabras. Adicionalmente, utilizaron la estrategia de aumento de datos basados en traducción inversa con el fin de lidiar el sobreajuste. Sus experimentos con clasificadores lineales muestran resultados competitivos frente a arquitecturas más complejas.

Sobrino Sande (2018) explicó los fundamentos teóricos, aplicaciones y la relación entre NLP y el análisis de sentimiento, ofreciendo adicionalmente mediante un análisis de trabajos publicados el estado del arte en español, así como los métodos utilizados en estas tareas. En su trabajo se implementaron clasificadores de polaridad basados en algoritmos de aprendizaje supervisado (excluyendo a modelos de redes neuronales), para posteriormente comparar los resultados de los métodos más utilizados. Concluye, según los resultados obtenidos, que el avance en el área es notable y admirable. Sin embargo, las métricas en el idioma español aún son bajas. Además, considera que dicha tarea no es muy aplicada en el sector empresarial y su aplicación tendría un impacto positivo en el mismo.

González Barba (2017) hizo un estudio de los modelos basados en aprendizaje profundo y las representaciones de palabras en relación a la tarea de análisis de sentimiento

y otras derivaciones. Para ello, hizo comparaciones de distintos modelos de redes neuronales cambiando las representaciones de palabras en diferentes corpus. Resaltando lo que cada representación aporta al problema. Finalmente, concluye que los modelos implementados obtuvieron resultados competitivos frente a otros enfoques.

2.3 CONSIDERACIONES FINALES

En el presente capítulo se hizo una revisión literaria de conceptos y definiciones que ayudarán al lector a entender los capítulos siguientes. Tales conceptos se agruparon según las dos principales variables de esta tesis. Asimismo, se realizó una revisión de algunos antecedentes a la presente investigación. En el siguiente capítulo se detalla los métodos que se aplicaron para resolver la tarea de análisis de sentimiento de tuits en español.

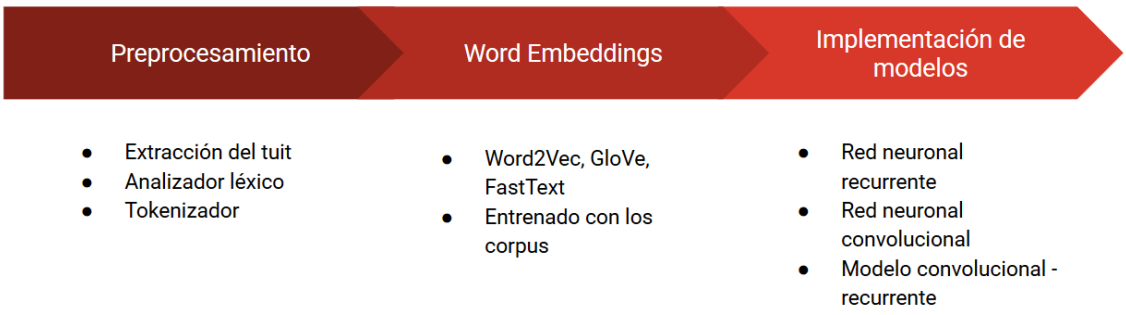
CAPÍTULO III

MATERIALES Y MÉTODOS

En este capítulo se presentan los métodos y modelos de aprendizaje profundo para resolver la tarea de análisis de sentimiento. En la primera sección se enfoca principalmente al preprocesamiento de los datos. La segunda sección contiene la obtención de las representaciones de palabras. Finalmente, la tercera sección contiene el detalle de los modelos propuestos y la descripción de sus arquitecturas.

La figura 3.1 muestra a detalle los pasos que se siguieron para la implementación del método propuesto. Primero se realizó la implementación de un algoritmo de preprocesamiento. Posteriormente, se hizo la conversión de las palabras en sus representaciones matemáticas, *word embeddings*. Finalmente, se implementaron los modelos de aprendizaje profundo propuestos.

Figura 3.1: Proceso de implementación del método.



Elaborado por el equipo de trabajo

3.1 PREPROCESAMIENTO DE LOS DATOS

La base de datos utilizada en el presente trabajo consiste en una lista de tuits en un archivo de formato Lenguaje de Marcado Extensible (XML) que tiene la siguiente estructura:

Listado 3.1: Estructura de la base de datos

```
<?xml version="1.0" encoding="UTF-8"?>
<tweets>
   <tweet> ... </tweet>
   <tweet> ... </tweet>
   <tweet> ... </tweet>
   ...
   <tweet> ... </tweet>
</tweets>
```

Cada tuit tiene la información de su identificador, el usuario que lo escribió, el contenido del tuit, el idioma, y la polaridad del sentimiento que se etiquetó. Un tuit de la base de datos posee la siguiente estructura:

Listado 3.2: Ejemplo de tuit

```
<tweet>
      <tweetid>770973163920453632</tweetid>
      <user>DeseoLiterario</user>
      <content>@Lenavg3 jajajaja si si. Buscare en Pinterest algo asi gracias :*</content>
      <date>2016-08-31 13:15:00</date>
      <lang>es</lang>
      <sentiment>
            <polarity><value>P</value></polarity>
      </sentiment>
</tweet>
```

Con el objetivo de dar a la red neuronal información más valiosa, se realizó el

preprocesamiento del texto de cada tuit. Para ello, basándose en Severyn y Moschitti (2015) y Navas-Loro y Rodrıguez-Doncel (2017), se creó un analizador léxico con el propósito de lidiar con términos triviales y estandarizar las entradas para la red neuronal. El analizador léxico sigue las siguientes reglas:

- **Cambiar todas las palabras a minúscula:** para asociar una palabra a su representación se debe crear una estructura de datos basada en claves. De esta manera, las palabras deben estar totalmente uniformizadas para un correcto almacenamiento. Gracias a esto se recuperan con mayor eficiencia las representaciones de palabras que se tienen.

- **Borrar URLs, espacios en blanco extra, caracteres especiales y palabras repetidas más de una vez:** este tipo de datos no aportan información que sirva para determinar la polaridad de sentimiento que expresa un texto, debido a esto, son eliminados.

- **Reemplazar expresiones de risa (tales como 'jajaja', 'haha', 'LOL', etc.) por 'ja':** también se realiza para estandarizar el texto. De este modo, cualquier expresión de risa se recupera de una única forma.

- **Corregir coloquialismos y vicios del lenguaje comunes en internet (por ejemplo 'por' en vez de 'x'):** estos errores agregan ruido al texto, debido a ello se deben uniformizar.

- **Crear un diccionario de palabras vacías (Stopwords)**: las palabras vacías son las más frecuentes en los textos, por lo que provocan ruido al clasificador y no ayudan en la definición de la polaridad de sentimiento en un texto. Por ello, dichas palabras quedan eliminadas de la etapa de preprocesamiento. Un diccionario de palabras

vacías suele estar conformado por pronombres, artículos, preposiciones, adverbios e incluso algunos verbos. En tal sentido, en base al corpus usado las palabras más comunes son las mostradas en la Tabla 3.1 y en el Anexo B donde se proporciona la lista completa de palabras vacías que se consideró en este trabajo.

Tabla 3.1: Fragmento de lista de Stopwords

Stopwords	Stopwords	Stopwords
a	acá	aca
ahi	ahí	al
algo	algún	algun
alguna	algunas	alguno
algunos	allá	alla
allí	alli	ambos
ante	antes	aquel

Elaborado por el equipo de trabajo.

Sumado al analizador léxico. se sustituyó los emoticonos del dataset por una palabra que representara su significado estadístico. Se utilizó una clasificación basada en modelo emoticons-clusters propuesto por Wang y Castanon (2015). La Tabla 3.2 muestra la clasificación utilizada y las palabras que se consideraron se encuentran en **negrita**.

La sustitución de emoticonos se debe a que los emoticonos representan una alta expresividad y significado en un texto. Esta información generalmente es descartada. Un emoticono puede cambiar la totalidad del sentimiento que expresa un texto. Por ejemplo, al decir "Mañana lloverá :(" denota tristeza o pena; sin embargo, al decir "mañana lloverá :D" denota alegría o esperanza. Cabe resaltar que se realizó la sustitución por las palabras en inglés para que no se confundan con las palabras del dataset original.

Tabla 3.2: Emoticons-clusters y su significado estadístico de Wang y Castanon (2015)

Cluster	Emoticonos	Significado estadístico
A	:) :D =)	**good** thanks happy fantastic lovely wonderful amazing ...
B	;) :-) ;-) :-D =D	**smile** friends face music favorite pic kind
	;P =] XD	coffee pleasure positive exciting healthy ...
C	:(:/ :') :'(:-(	**miss sorry bad** hate sad omg sick late
	D: ;(:-/ :| :/	mad ugh ugly broke
D	:P ;D :-P :] :p	**what lol** don't no know think can't why ever never look ...
E	(:	**love** follow please hey wish goodnight ...
F	XP	**stuck** shoot **fatally**
H	8)	**best fun** coming week playing top happiness weekend ...

Elaborado por el equipo de trabajo.

Por último, se separa la oración en palabras para convertirlas en sus representaciones. Esto se debe a que los *word embeddings* utilizan una sola palabra a la vez para transformarla en una representación numérica. El Algoritmo 1 muestra el procedimiento de preprocesamiento utilizado.

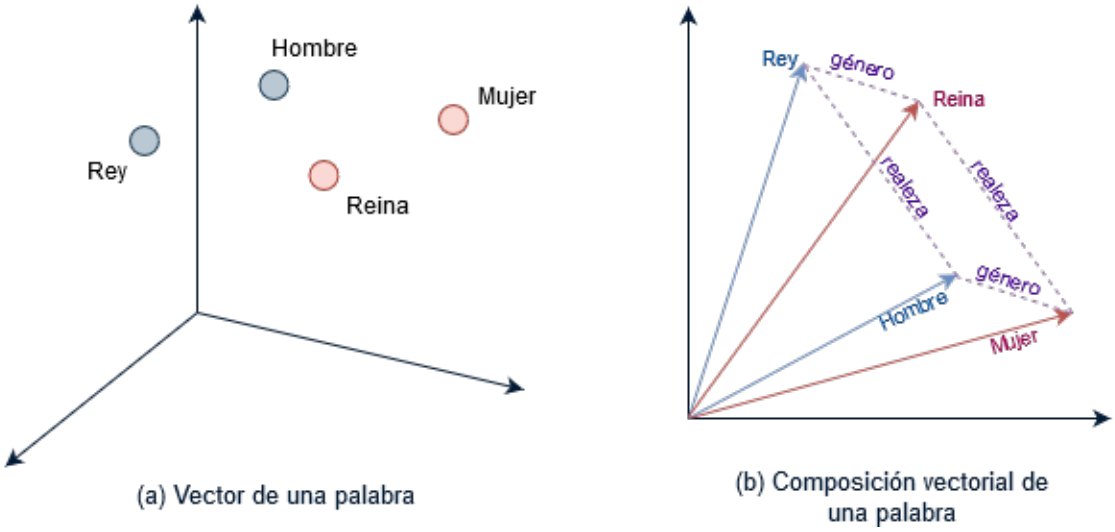
3.2 REPRESENTACIONES DE PALABRAS

La entrada a la red propuesta consiste en *word embeddings*. Un *word embedding* es una representación vectorial de una palabra. El término introducido por Bengio, Ducharme, Vincent, y Jauvin (2003), y Collobert et al. (2011) unificaron una arquitectura para NLP y *word embedding*.

Algoritmo 1: Preprocesamiento de datos

Datos: Tuit no procesado
Resultado: Lista de tokens procesados
para *i en lista de letras* **hacer**
  si *i es mayúscula* **entonces**
    cambiar a minúscula
  fin
  si *i es carácter especial o espacio extra* **entonces**
    borrar *i*
  fin
fin
para *i en palabras del tuit* **hacer**
  seleccionar *i* **hacer**
    caso *URL, palabra repetida, dentro de la lista de StopWords* **hacer**
      borrar *i*
    fin
    caso *i es una expresión de risa* **hacer**
      reemplazar *i* por token Jaja
    fin
    caso *i representa un vicio de lenguaje* **hacer**
      corregir *i*
    fin
    caso *i dentro de diccionario de emoticonos* **hacer**
      reemplazar *i* por su significado
    fin
  fin
fin

Figura 3.2: Representación vectorial de una palabra en un plano



(a) Vector de una palabra

(b) Composición vectorial de una palabra

Fuente: S. Liu et al. (2017)

Por ejemplo, haciendo referencia a la representación simplificada visual vectorial de las palabras *Rey, Reina, Hombre y Mujer* según S. Liu et al. (2017) se obtiene la Figura 3.2 en el punto **(a)** se observa la representación vectorial de las palabras ya nombradas en un plano de tres dimensiones y en el punto **(b)** se muestra la representación vectorial en dos dimensiones y las relaciones entre las palabras (Rey, Reina, Hombre y Mujer) respecto a su género y su denominación respecto a un linaje real.

Por otro lado, al hacer una representación vectorial de las palabras *Rey, Reina, Princesa, Hombre, Mujer,Realeza y Edad* con valores hipotéticos, se obtiene una representación similar a la Tabla 3.3. En el cual el mapeo de una palabra se extiende a todos los elementos en el vector, y a su vez cada elemento contribuye a la definición de las demás palabras.

Tabla 3.3: Representación vectorial de una palabra

	Rey	**Reina**	**Mujer**	**Princesa**
Realeza	0.99	0.99	0.02	0.98
Masculino	0.99	0.05	0.01	0.02
Femenino	0.05	0.93	0.99	0.94
Edad	0.70	0.60	0.50	0.10

Fuente: Jácome Galarza (2016)

Para obtener los *word embeddings* que sirvieron de representación de palabras en esta investigación, se aplicaron tres modelos:

- **Word2Vec:** Word to Vector (Mikolov, Chen, Corrado, y Dean, 2013), es un modelo basado en el método *skipgram*. Consiste en el cálculo de la probabilidad condicional de predecir una palabra contexto, dada otra palabra central. Utiliza el principio de

estimación de máxima verosimilitud para entrenarse. Luego del entrenamiento, se utilizó los vectores centrales de cada palabra como representación.

- **GloVe:** Global Vectors (Pennington, Socher, y Manning, 2014), este modelo varía el *skipgram* y le agrega otro tipo de entrenamiento. Utiliza ratios de probabilidad entre 2 palabras condicionales y una palabra objetivo central. A diferencia de Word2Vec, GloVe utiliza el error cuadrático para entrenarse. Al finalizar el entrenamiento, se usa el vector objetivo central como representación de palabra.

- **FastText:** (Bojanowski, Grave, Joulin, y Mikolov, 2017), es una extensión del método Word2Vec. Trata a cada palabra como una composición de grupos de n letras. De esta forma obtiene los vectores de una palabra mediante la suma de los vectores de los grupos de letras que contiene. Esto ayuda a generar mejores *embeddings* de palabras raras o que no estén dentro de un vocabulario establecido. Para obtener los vectores de los grupos de letras utiliza un entrenamiento muy similar al de Word2Vec.

Para el entrenamiento de los *embeddings* se compuso un dataset que agrupaba los corpus "General", "Social TV", "STOMPOL" e "InterTASS". De este modo, se aplicaron los 3 métodos en el dataset unificado. Finalmente, se obtuvo tres representaciones vectoriales para cada palabra, que marcaban el punto de partida para la el entrenamiento de los modelos de aprendizaje profundo.

3.3 MODELOS

En esta sección se presentan los modelos de aprendizaje profundo utilizados para resolver el problema.

3.3.1 Red neuronal recurrente

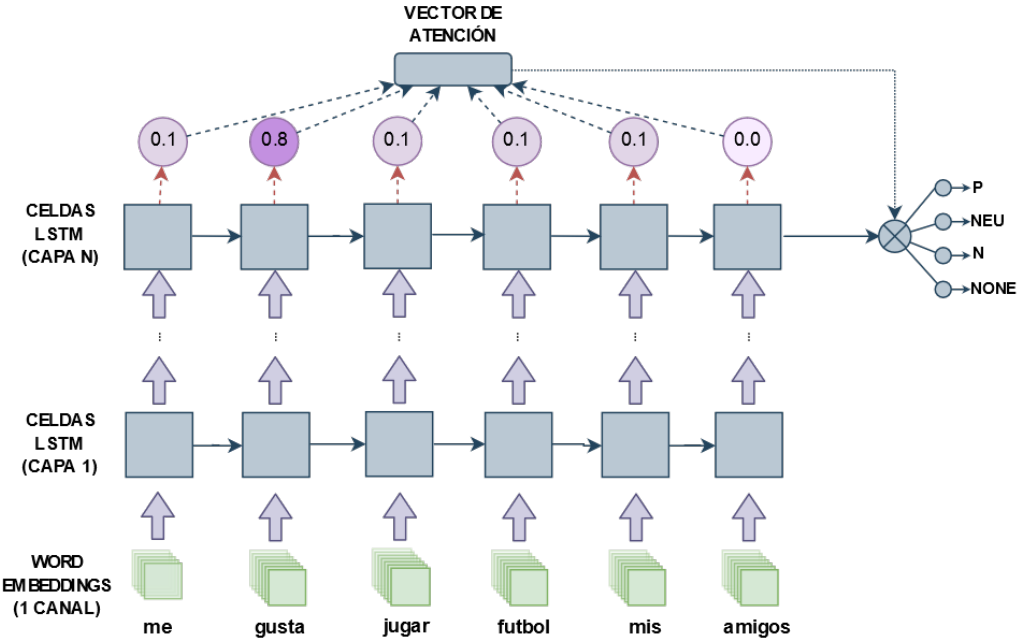
Inicialmente propuesto por Rumelhart et al. (1988) y presentado también por Elman (1990), es el modelo de aprendizaje profundo por excelencia para los datos secuenciales. Gracias a ello, ha obtenido muy buenos resultados resolviendo tareas de NLP. Se basa en la ejecución de las mismas operaciones secuencialmente, es decir, su grafo computacional es cíclico. En líneas generales, el valor de las representaciones va cambiando a medida de que se recibe más entradas. En NLP la red procesa una entrada a la vez y aplica las mismas operaciones a cada miembro de la secuencia, empezando por el primer elemento.

Debido a que la salida va cambiando con el procesamiento de nuevos elementos, aumenta la posibilidad de que la salida de la red pueda estar altamente parcializada por los elementos finales de la entrada. Aunado a esto, si la secuencia es suficientemente grande puede provocar que las gradientes se vuelvan muy pequeñas (**desvanezcan**) o muy grandes (**exploten**). Esto provoca que la optimización de parámetros falle, haciendo así imposible el aprendizaje.

Las celdas de LSTM propuestas por Hochreiter y Schmidhuber (1997) son muy utilizadas para corregir los problemas de una RNN. Esto se debe principalmente a que contienen compuertas para manejar la información que procesan. Asimismo, la compuerta de olvido f_t (Gers, Schmidhuber, y Cummins, 2000) ayuda a controlar si se debe recordar u olvidar los datos anteriores.

La Figura 3.3 muestra la arquitectura utilizada en el presente trabajo. La red consta de dos partes principales: Las celdas LSTM y el vector de atención.

Figura 3.3: Arquitectura de la red neuronal recurrente



Elaborado por el equipo de trabajo

En cada paso de la RNN se procesa una palabra del texto de entrada. En este modelo se usa únicamente un tipo de *embedding* de los 3 disponibles. Las celdas LSTM procesan los *embeddings* de las palabras de entrada secuencialmente, a esto se le denomina capa oculta. Sin embargo, se agregan más capas para hacer que la arquitectura tenga mayor profundidad. De esta manera se procesan las salidas de la capa anterior un numero determinado de veces.

Como se había mencionado anteriormente, esta arquitectura puede dar un resultado muy parcializado por los últimos elementos de entrada. Debido a esto se utiliza un modelo de atención, el cual pondera las salidas de cada sección temporal y la utiliza para calcular la salida final de la red. Esta idea está inspirada en el modelo de atención (Bahdanau, Cho, y Bengio, 2014) de modelos secuencia a secuencia (Sutskever, Vinyals, y Le, 2014).

Finalmente se utiliza una función softmax en la capa de salida para determinar las

probabilidades de pertenencia a cada clase P, Neu, N, None.

3.3.2 Red neuronal convolucional

Inicialmente propuesto por LeCun, Bottou, Bengio, Haffner, et al. (1998), es un modelo originalmente diseñado para el procesamiento de imágenes. Este se basa en la operación de convolución del cual viene su nombre. La operación de convolución puede ser definida como la ponderación promediada de las partes de una imagen. Tiene dos elementos principales: una imagen de entrada a procesar y una **matriz de convolución** o **kernel** que pondera las partes de la imagen. La operación de convolución está definida como:

$$C(m,n) = \sum_i \sum_j I(m-i, m-j)K(i,j) \tag{3.1}$$

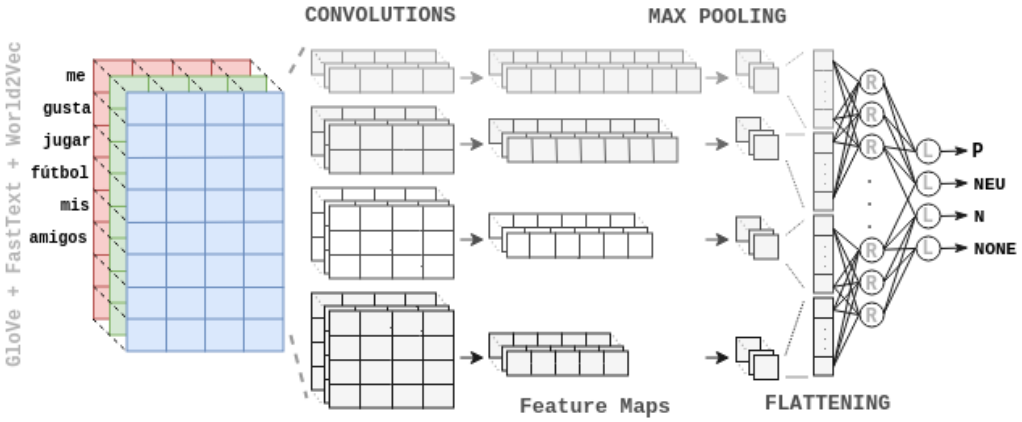
Donde I es la imagen de entrada, K es el kernel, i, j son las coordenadas del kernel y m, n son las coordenadas de la imagen. La salida de una convolución es llamada **mapa de características**.

Una ventaja del uso de las CNNs es que utiliza menos parámetros que una red neuronal recurrente y un perceptron multicapa. Asimismo, la CNN presenta una propiedad de invariabilidad con respecto a la ubicación de elementos en la imagen, lo que significa que puede reconocer los mismos elementos en las imágenes sin importar su ubicación. Finalmente, las operaciones de convolución se pueden realizar en paralelo, lo cual acelera el entrenamiento de este tipo de modelos. Por las razones anteriormente expuestas, la CNN es el modelo por excelencia del procesamiento de imágenes.

En el área de NLP, Kim (2014) propuso el uso de las CNNs para la clasificación de textos. Utilizando como base este modelo, se propuso la CNN aplicada en la presente

tesis. La Figura 3.4 muestra la arquitectura del modelo utilizado en el presente trabajo.

Figura 3.4: Arquitectura de la red neuronal convolucional



Fuente: Vizcarra et al. (2018)

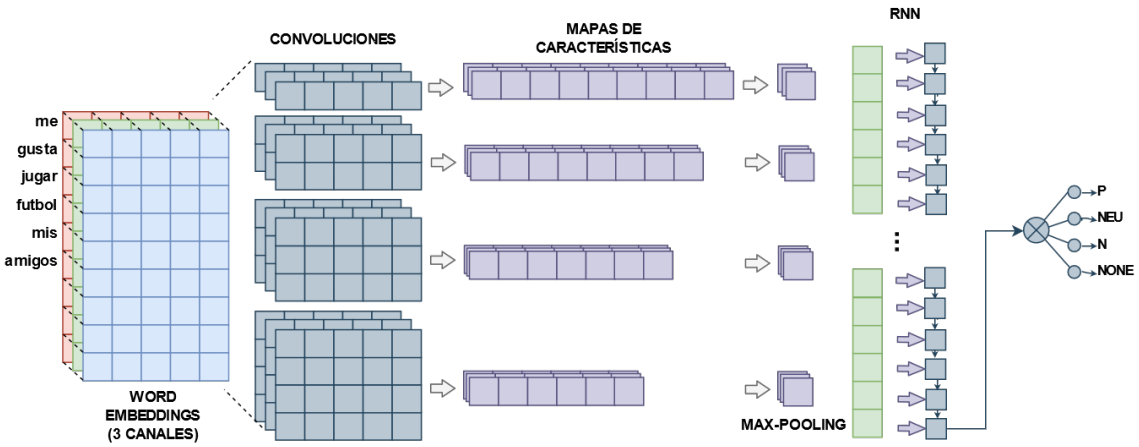
La CNN propuesta utiliza los 3 vectores de *embedding*. Esto es similar al caso de los canales de color de una imagen (Red, Green, Blue). Seguidamente se realiza varias convoluciones paralelas de 1 dimensión sobre las entradas (en la Figura 3.4, se realizan convoluciones de kernels $1, 2, 3, 4$). Cada convolución debe captar el significado de k palabras, donde k es el tamaño de la convolución. Esto resulta un mapa de características por cada kernel. A continuación se realiza una operación de **max-pooling** (extraer el elemento mayor) en cada mapa de características. Luego se usan dos capas densas (perceptrón multicapa). Finalmente, se realiza una función softmax para determinar las probabilidades de las salidas.

3.3.3 Modelo convolucional-recurrente

Este modelo utiliza los 3 algoritmos de embedding. Este modelo realiza convoluciones sobre la entrada y la operación *max-pooling*, pero no las capas densas. En vez de eso, se ejecuta el modelo recurrente sobre las salidas del *max-pooling*. Finalmente, se aplica la función softmax para determinar la polaridad de sentimiento a la que pertenece la oración

de entrada. La Figura 3.5 muestra la arquitectura del modelo.

Figura 3.5: Arquitectura del modelo convolucional-recurrente



Elaborado por el equipo de trabajo.

3.4 CONSIDERACIONES FINALES

En este capítulo se detallaron los métodos que se aplicaron para resolver la tarea de análisis de sentimiento de tuits en español. Se hizo una descripción de los algoritmos de preprocesamiento. Asimismo, se describieron las arquitecturas de los modelos propuestos. En el siguiente capítulo, se presentan al detalle los experimentos realizados y los resultados obtenidos.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

En este capítulo se presenta los resultados obtenidos de acuerdo a los objetivos planteados. De esta forma primero se presenta los resultados obtenidos en el preprocesamiento. Seguidamente, los resultados conseguidos en la implementación del modelo. Finalmente, se muestran los resultados alcanzados al evaluar los modelos implementados.

4.1 PREPROCESAMIENTO

Como primer paso, se creó un script para la extracción del contenido y la polaridad del tuit con el fin de crear los conjuntos de datos para entrenar la red neuronal. Esto resultó en una estructura de pares que contiene un tuit y su polaridad. La figura 4.1 muestra algunos de los tuits extraídos en pares.

El fragmento de código que realiza el preprocesamiento se presenta en la figura 4.2; en él se utiliza la librería 're' de python para hacer emparejamientos y detecciones con expresiones regulares.

Así, el preprocesamiento de una oración de entrada se realiza de la forma mostrada en la figura 4.3.

Figura 4.1: Conjunto de tuits extraídos.

	content	polarity
0	Salgo de #VeoTV , que día más largooooooo...	NONE
1	@PauladeLasHeras No te libraras de ayudar me/n...	NEU
2	@marodriguezb Gracias MAR	P
3	Off pensando en el regalito Sinde, la que se v...	N
4	Conozco a alguien q es adicto al drama! Ja ja ...	P
5	RT @FabHddzC: Si amas a alguien, déjalo libre....	NONE
6	Toca @crackoviadeTV3 . Grabación dl especial N...	P
7	Hoy asisitiré en Madrid a un seminario sobre l...	NONE
8	Buen día todos! Lo primero mandar un abrazo gr...	P
9	Desde el escaño. Todo listo para empezar #endi...	P
10	Bdías. EM no se ira de puente. Si vosotros os ...	P
11	Un sistema económico q recorta dinero para pre...	P
12	#programascambiados caca d ajuste	N
13	Buen viernes	P
14	“@adri_22_22: #programascambiados es TT gracia...	P
15	Noooooos días! Me he dormidooooooo ya estoy en ...	NONE
16	Vamos a por el viernes (@ Ayuntamiento de Mála...	NONE
17	La Universidad confía en De la Calle para enca...	P
18	¿Me ayudáis a que #indultoneiro sea TT? Por si...	P
19	abcdesevilla.es: Recio no tiene «indicios pote...	N
20	abcdesevilla.es: Cuatro altos cargos de Empleo...	N
21	La marcha atrás del PP en posponer devolución ...	N
22	Viernes negro: Aumenta el paro en Noviembre y ...	N
23	Accidente en BUS-VAO A-6 km. 12. Motorista de ...	N
24	"La verdadera seriedad es cómica". Niall Binns...	P
25	#FF a ti, que deseas desesperadamente hacerme ...	N
26	Mis #FF xa @merpastor x volver a sonreír, a @G...	P
27	Agradezco a trabajadores y sindicatos la desco...	P
28	#FF a @RdscubreNavidad por ser una gran inicia...	P
29	"Hasta ahora, los filósofos han tratado de com...	NONE
...	...	...

Elaborado por el equipo de trabajo.

Figura 4.2: Fragmento de código del tokenizador.

```
def tokenizer(text):
    text = str(text)
    text = re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', ' ', text)
    text = re.sub(r">?[:|8|=]'?-?[\]|)|3|>|\}]", 'smile', text)
    text = re.sub(r">?[:|8|x|X|=|;]'?-?[D|3|<|>|]", 'laugh', text)
    text = re.sub(r">?[:|8|=]-?[(|<|\[|\{|]", 'sad', text)
    text = re.sub(r">?[:|8|=]'-?[(|<|\[|\{|]", 'cry', text)
    text = re.sub(r"D-?'?[:|8|=|;|X|x]<?", 'horrorified', text)
    text = re.sub(r">?[:|8|=]-?[0|o|O]", 'surprise', text)
    text = re.sub(r">?[;]-?[\])|3|>|\}]", 'wink', text)
    text = re.sub(r">?[:|8|=|x|X]-?[P|/|p|þ]", 'playful', text)
    text = re.sub(r">?[:|8|=|x|X]-?[/|\\|L|S]", 'skeptical', text)
    text = re.sub(r">?[:|8|=|x|X]-?[/|\\|L|S]", 'skeptical', text)

    text = re.sub('<[^>]*>', ' ', text)
    text = re.sub(r'\b-\b', ' ', text)
    text = re.sub('[^|][0-9]+', ' ', text)
    text = re.sub('[\s!/,\\.?¡¿""":/();]+', ' ', text) #”
    text = re.sub("[\s!/,\\.?¡¿'"":/();]+", ' ', text)
    text = re.sub('[W]+', ' ', text)
    text = re.sub('[\s]+', ' ', text.lower())
    text = re.sub(r'\b(gg+\b)+', 'por', text)
    text = re.sub(r'(.)\1{2,}', r'\1', text)
    text = re.sub(r'\b(d*xd+x*[xd]*\b|\ba*ha+h[ha]*\b|\ba*ja+j[ja]*|o?l+o+l+[ol]*)\b', 'ja', text)
    text = re.sub('^\s', '', text)
    text = re.sub('@([a-z0-9_]+)', '@user', text)
    text = re.sub(r'\b(x\b)+', 'por', text)
    text = re.sub(r'\b(q\b)+|\b(k\b)+|\b(ke\b)+|\b(qe\b)+|\b(khe\b)+|\b(kha\b)+', 'que', text)
    text = re.sub(r'\b(xk\b)+|\b(xq\b)+', 'porque', text)
    text = re.sub(r'\b(xk\b)+|\b(xq\b)+', 'porque', text)
    text = re.sub(r'\b(ai+uda\b)+', 'ayuda', text)
    text = re.sub(r'\b(hostia\b)+', 'ostia', text)
    text = re.sub(r'\b(d\b)+', 'de', text)
    text = re.sub(r'\b(\w+)( \1\b)+', r'\1', text)

    #text = re.sub('\b(?:a*(?:ha)+h?|:a*(?:ja)+j?|(?:l+o+)+l+)\b', 'jeje', text)

    text = re.findall(r'[^\s!,.?¡¿""":;]+', text)
    text = [w for w in text if w not in stop]
    return text
```

Elaborado por el equipo de trabajo.

Figura 4.3: Preprocesamiento de una oración del corpus.

```
data.at[26, "content"]
```

```
'Mis #FF xa @merpastor x volver a sonreír, a @GFVara x ser tan autentico, a @24horas_rne x su subidòn y a @Carmende
lRiego x su victoria'
```

```
tokenizer(data.at[26, "content"])
```

```
['#ff',
 'xa',
 '@user',
 'por',
 'volver',
 'sonreír',
 '@user',
 'por',
 'ser',
 'tan',
 'autentico',
 'horas_rne',
 'por',
 'subidòn',
 '@user',
 'por',
 'victoria']
```

Elaborado por el equipo de trabajo.

4.2 MODELOS DE APRENDIZAJE PROFUNDO

En esta sección se presenta la representación matemática de cada modelo y detalles de la implementación de los modelos.

Se define el tuit de entrada como $X = \{x_1, x_2, ..., x_n\}$ donde x_t es un vector que representa a la palabra v_j que se encuentra en un vocabulario definido. La red neuronal produce una salida $\hat{y}$ que es contrastada con la clase $y \in \{P, N, NEU, NONE\}$.

4.2.1 Red neuronal recurrente

La RNN implementada se define de la siguiente forma:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \odot \sigma_c(c_t)
\end{aligned}
\tag{4.1}
$$

Donde $x_t \in X$ es la entrada de la celda, W y U son los pesos de la celda y b es el *bias*. i_t es la compuerta de entrada, o_t es la compuerta de salida. h_t es el estado oculto de la celda actual, h_{t-1} es el estado oculto de la anterior celda. c_t es la memoria de la celda, c_{t-1} es la memoria de la celda anterior. Los valores de c_0 y h_0 son 0. Adicionalmente se hace el uso de las funciones de activación sigmoidea (σ_g) y tangente hiperbólica (σ_c). Finalmente cuando $t = n$, se sigue el siguiente procedimiento:

$$
\begin{aligned}
f_n &= \sigma_g(W_f x_n + U_f h_{n-1} + b_f) \\
i_n &= \sigma_g(W_i x_n + U_i h_{n-1} + b_i) \\
o_n &= \sigma_g(W_o x_n + U_o h_{n-1} + b_o) \\
c_n &= f_n \odot c_{n-1} + i_n \odot \sigma_c(W_c x_n + U_c h_{n-1} + b_c) \\
h_n &= o_n \odot \sigma_c(c_n) \\
\hat{y} &= \sigma_s(h_n)
\end{aligned}
\tag{4.2}
$$

donde σ_s es la función softmax.

4.2.2 Red neuronal convolucional

La capa convolucional de la CNN implementada se define de la siguiente manera:

$$C(t)_z = ReLU(\sum_i X(t-i)K_z(i) + b) \tag{4.3}$$

donde C es el resultado de la convolución, $k \in K$ es un vector kernel dentro del conjunto K de vectores utilizados, z es un número entero en el intervalo de $[0, longitud(K)]$ y b es el vector bias. Luego se realiza la operación de max pooling p y concatenación $concat$.

$$\begin{aligned} p(C_k) &= max(C(t)_k) \\ concat &= p(C_{[0,len(K)]})) \end{aligned} \tag{4.4}$$

Finalmente, se calcula la salida de la red $\hat{y}$ con las capas densas utilizando las matrices de pesos W_{fc1}, W_{fc2} y los vectores bias b_{fc1}, b_{fc2}

$$\begin{aligned} h_{fc1} &= ReLU(p \odot W_{fc1} + b_{fc1}) \\ \hat{y} &= ReLU(h_{fc1} \odot W_{fc2} + b_{fc2}) \end{aligned} \tag{4.5}$$

4.2.3 Modelo convolucional-recurrente

Este modelo es un híbrido de los modelos anteriormente presentados y utiliza funciones similares. La red CNN-RNN se define como:

$$
\begin{aligned}
C(t)_z &= ReLU(\sum_i X(t-i)K_z(i)+b) \\
p(C_k) &= max(C(t)_k) \\
concat &= p(C_{[0,len(K)]})) \\
f_t &= \sigma_g(W_f concat_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i concat_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o concat_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \sigma_c(W_c concat_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \odot \sigma_c(c_t)
\end{aligned}
\tag{4.6}
$$

Luego, la salida de la red se calcula de la siguiente forma:

$$\hat{y} = \sigma_s(h_n) \tag{4.7}$$

4.2.4 Detalles de implementación

La entrada de cada red neuronal es de dimensiones [tamaño de minibatch, longitud de la oración, numero de canales de embeddings, dimensión del embedding]. Para todas las redes neuronales se utiliza un minibatch de tamaño 55, la oración más larga como estándar (33 palabras), 50 como dimensión de los embeddings. Para las redes CNN y CNN-RNN se utilizan tres canales de embeddings y para la red RNN, un canal.

Para la CNN, cada convolución requiere 100 kernels para entrenar y genera 400 mapas de características. El perceptron multicapa (capa densa) tiene 200 neuronas y usa la

función de activación ReLU. Las neuronas de salida son cuatro y usan la función logística de activación. Se añadió un dropout del 25 % de probabilidad para evitar el sobreajuste.

Para la RNN, se utilizó una capa de celdas LSTM. El modelo de atención se basa en la multiplicación con las salidas. Para el modelo CNN-RNN se utilizó las configuraciones del modelo CNN. Para todos los modelos se utilizó la entropía cruzada como función de costo y el optimizador ADAM con una tasa de aprendizaje de $1e-4$.

Para ejecutar los experimentos se utilizó una PC con las siguientes características: procesador Intel Core i7 3.6 GHz, memoria 16 GB 2400 MHz DDR4 y tarjeta de vídeo NVidia GTX 1080ti. Finalmente, para realizar la implementación se usó el lenguaje de programación Python y la librería TensorFlow-1.5.

4.3 EVALUACIÓN DE LOS MODELOS

Para evaluar correctamente los modelos propuestos y hacer una comparación justa, es necesario especificar los recursos utilizados y definir las métricas estándar que se aplican. En esta sección se especifica los detalles del corpus utilizado, el cálculo de la métrica y los trabajos referenciales que utilizan tales corpus y métrica.

4.3.1 Corpus

A fin de realizar las pruebas en el presente trabajo se hizo uso de un corpus ya existente en el idioma español correspondiente al TASS 2017 (Martınez-Cámara, Dıaz-Galiano, Garcıa-Cumbreras, Garcıa-Vega, y Villena-Román, 2017). Dicho taller plantea una serie de tareas relacionadas con el análisis de sentimiento en Twitter. Las tarea 1 se refiere a la determinación de polaridad en tuits, la cual se aborda en esta investigación.

Los textos en los corpus están clasificados en cuatro categorías: Positivo (P), Negativo (N), Neutro (NEU) y Sin sentimiento (NONE).

Tabla 4.1: Distribución de tuits en los corpus InterTASS según su polaridad

	P	N	NEU	NONE	∑
InterTASS	318	418	133	139	1008
Corpus General TASS	2464	1877	588	1207	6136

Elaborado por el equipo de trabajo.

Para el entrenamiento del corpus InterTASS y el corpus general del TASS se ha distribuido los tuits según su polaridad (positivo, negativo, neutro y sin sentimiento) como se muestra en la Tabla 4.1 según los datos especificados en el documento resumen (Martınez-Cámara et al., 2017) del evento TASS.

4.3.2 Métricas y trabajos referenciales

Para evaluar los modelos propuestos, se consideró como métrica principal la exactitud del método, es decir, que porcentaje de los tuits fueron clasificados correctamente. Como se ha mencionado en el capítulo II, Tal métrica se obtiene usando la siguiente fórmula:

$$\text{Exactitud} = \frac{\text{TP} + \text{TN} + \text{TNeu} + \text{TNone}}{\text{TP} + \text{FP} + \text{TN} + \text{FN} + \text{TNeu} + \text{FNeu} + \text{TNone} + \text{FNone}} \tag{4.8}$$

Donde TP, TN, TNeu y TNone son los ejemplos clasificados correctamente en las categorías Positivo, Negativo, Neutro y NONE respectivamente. FP, FN, FNeu y FNone son los ejemplos clasificados erróneamente en las categorías Positivo, Negativo, Neutro y NONE respectivamente. Así, se considera mejor al método que pueda clasificar correctamente la mayor cantidad de ejemplos en un grupo aleatorio. Formalmente podemos decir que:

$$\text{Exactitud} = \frac{\text{Número de ejemplos clasificados correctamente}}{\text{Total de ejemplos}} \tag{4.9}$$

En el presente trabajo de investigación, por ejemplo en la clasificación de los sentimientos de dos clases: *positivos* y *negativos*, se obtiene una matriz de confusión de 2 x 2 como se muestra en la Tabla 4.2.

Tabla 4.2: Matriz confusión de los sentimientos positivos y negativos.

	Verdadero (T)	**Negativo (F)**
Positivo (P)	**Verdadero Positivo(TP)** Realidad: *Positivo* Clasificación: *Positivo*	**Falso Positivo(FP)** Realidad: *Positivo* Clasificación: *Negativo*
Negativo (N)	**Verdadero Negativo(TN)** Realidad: *Negativo* Clasificación: *Positivo*	**Falso Negativo(FN)** Realidad: *Negativo* Clasificación: *Negativo*

Elaborado por el equipo de trabajo.

Los trabajos referenciales que se consideraron para comparar los métodos propuestos son los siguientes:

- ELiRF-UPV-run1 (Hurtado, Pla, y González, 2017) Multilayer perceptron utilizando embeddings skipgram.
- RETUYT-svm cnn (Rosá, Chiruzzo, Etcheverry, y Castro, 2017) Clasificador híbrido usando las salidas de una CNN y una máquina de vector soporte.
- ELiRF-UPV-run3 (Hurtado et al., 2017) Multilayer perceptron utilizando embeddings de polaridad.
- jacerong-run-2 (Cerón-Guzmán, 2017) Combinación de 19 clasificadores entrenados en InterTASS corpus.
- jacerong-run-1 (Cerón-Guzmán, 2017) Combinación de 3 clasificadores entrenados en InterTASS corpus.
- INGEOTECevodag-001 (Moctezuma et al., 2017) Algoritmo genético.

4.3.3 Configuración de kernels

Debido a que en una RNN hay menor cantidad de hiperparámetros a optimizar, se puso mayor énfasis a las pruebas de la CNN. Principalmente en la definición de Kernels de convolución.

Tabla 4.3: Mejores resultados de exactitud de combinatoria de kernels en el corpus InterTASS

Prueba	Primero		Segundo		Tercero	
	Combinación	Exactitud	Combinación	Exactitud	Combinación	Exactitud
1	$<1,2>$	0.6182	$<2,4>$	0.6131	$<1,2,3,4>$	0.6125
2	$<1,2,3>$	0.6124	$<1,2,4>$	0.6112	$<1,2>$	0.6099
3	$<1,2>$	0.6163	$<1,4>$	0.6128	$<1,3>$	0.6118
4	$<1,2>$	0.6156	$<1,3>$	0.6120	$<1,2,3,5>$	0.6114
5	$<1,2>$	0.6214	$<1,2,3,4>$	0.6144	$<1,2>$	0.6134

Fuente: Vizcarra et al. (2018).

Para definir cuántos y cuáles deberían ser los kernels de convolución en los modelos basados en CNN, se realizaron dos experimentos. En el primero, se probó con las combinatoria de $\{1,2,3,4,5,6,7\}$ kernels sin repetición. Los resultados de este experimento se muestran en la Tabla 4.3

Tabla 4.4: Resultados estadísticos de las combinaciones de kernels en el corpus InterTASS

Kernels	Mejor prueba	Peor prueba	Promedio
$< 1,2 >$	**0.6219**	**0.6124**	**0.6158**
$< 1,3 >$	0.6163	0.6035	0.6094
$< 1,2,3 >$	0.6175	0.6029	0.6126
$< 1,2,3,4 >$	0.6118	0.5908	0.6008

Fuente: Vizcarra et al. (2018).

En el segundo experimento, se seleccionó a las mejores combinaciones de kernels de la Tabla 4.3. Luego, para cada kernel, se ajustaron hiperparámetros para obtener sus mejores, peores y resultados promedios. Los resultados de este experimento se muestra en la tabla 4.4.

Finalmente, en la Figura 4.4 se muestra el proceso de entrenamiento de la red neuronal convolucional con los filtros de convolución $< 1,2 >$.

4.3.4 Comparación con trabajos referenciales

La Tabla 4.5 muestra los resultados de la exactitud en los corpus InterTASS y General. En el concurso, los datos de entrenamiento y prueba se encuentran en diferentes paquetes. De esta forma, los resultados mostrados refieren a la exactitud en el conjunto de test. Los resultados de la presente investigación están marcados en negrita y el estado del

Figura 4.4: Resultados del entrenamiento de la red neuronal convolucional.

```
with tf.Session() as sess:
    saver = tf.train.Saver()
    sess.run(tf.global_variables_initializer())
    for i in range(n_epoch):
        x_train, y_train = WE.get_minibatch(batch_size, i, x_3d, y_d)
        if i % 100 == 0:
            train_accuracy = accuracy.eval(feed_dict={x_input: x_train, y_output: y_train, keep_prob: 0.5})
            print('step %d, eficiencia del training %g' % (i, train_accuracy))
        train_function.run(feed_dict={x_input: x_train, y_output: y_train, keep_prob: 0.5})

    # testing
    ef_test = accuracy.eval(feed_dict={x_input: x_test, y_output: y_test, keep_prob: 1})
    print('eficiencia del test %g' % ef_test)
```

```
step 0, eficiencia del training 0.327273
step 100, eficiencia del training 0.454545
step 200, eficiencia del training 0.472727
step 300, eficiencia del training 0.509091
step 400, eficiencia del training 0.454545
step 500, eficiencia del training 0.490909
step 600, eficiencia del training 0.345455
step 700, eficiencia del training 0.454545
step 800, eficiencia del training 0.545455
step 900, eficiencia del training 0.490909
step 1000, eficiencia del training 0.563636
step 1100, eficiencia del training 0.490909
step 1200, eficiencia del training 0.636364
step 1300, eficiencia del training 0.6
step 1400, eficiencia del training 0.545455
step 1500, eficiencia del training 0.545455
step 1600, eficiencia del training 0.563636
step 1700, eficiencia del training 0.563636
step 1800, eficiencia del training 0.654545
step 1900, eficiencia del training 0.4
step 2000, eficiencia del training 0.618182
step 2100, eficiencia del training 0.545455
step 2200, eficiencia del training 0.545455
step 2300, eficiencia del training 0.6
step 2400, eficiencia del training 0.527273
step 2500, eficiencia del training 0.6
step 2600, eficiencia del training 0.654545
step 2700, eficiencia del training 0.618182
step 2800, eficiencia del training 0.6
step 2900, eficiencia del training 0.690909
step 3000, eficiencia del training 0.618182
step 3100, eficiencia del training 0.709091
eficiencia del test 0.614404
```

Elaborado por el equipo de trabajo

arte en cada corpus se encuentra marcado con un asterisco (*).

Se incluyeron cuatro pruebas del presente trabajo. Tres de ellos son de cada modelo utilizando el preprocesamiento indicado en el Capítulo III; el cuarto CNN se refiere al modelo que obtuvo mejores resultados sin aplicar el preprocesamiento de los emoticones.

4.4 DISCUSIÓN

Los resultados muestran que la red neuronal convolucional es eficiente en el análisis de sentimiento a nivel de tuits en español. Los experimentos mostraron que la combina-

Tabla 4.5: Resultados comparativos en TASS-2017 para análisis de sentimiento

Método	Corpus	
	InterTASS	General
CNN-EMOTIC	**0.615**	**0.741**
CNN	**0.593**	**0.707**
RNN-EMOTIC	**0.604**	**0.712**
CNN-RNN-EMOTIC	**0.558**	**0.642**
ELiRF-UPV-run1	0.607	0.666
RETUYT-svm cnn	0.596	0.674
ELiRF-UPV-run3	0.597	0.725*
jacerong-run-2	0.602	0.701
jacerong-run-1	0.608*	0.706
INGEOTECevodag-001	0.507	0.514

Elaborado por el equipo de trabajo.

ción de kernels $< 1, 2 >$ hace que la red neuronal convolucional obtenga mejores resultados. El kernel 1 es la ponderación de palabras individuales, el kernel 2 es el análisis de grupos de dos palabras. Esto coincide con la descripción de la tarea, ya que un tuit es un texto corto.

Los tres canales de *embeddings* utilizados ayudan a tener una mejor representación del texto de entrada. También, las mejoras fueron posibles gracias al preprocesamiento de emoticones, coincidiendo con el estudio de Wang y Castanon (2015) acerca de la información que expresa un emoticon. Esto significó una importante mejora en ambos datasets (de 59.3 % - 70.7 % a 61.58 % - 74.14 % en los corpus InterTASS y General respectivamente).

Finalmente, es importante resaltar que el método basado en CNN es más robusto

que anteriores métodos, debido a que mantiene la mejor exactitud en ambos corpus. Por otra parte, los métodos que conforman el estado del arte muestran un buen resultado en un corpus que se contrasta con su resultado en el otro corpus. Esto se respalda con el estudio de Hassan y Mahmood (2017), donde señala que los modelos convolucionales obtienen mejores resultados que los modelos recurrentes en el procesamiento de textos cortos.

CAPÍTULO V

CONCLUSIONES

PRIMERA

Con el objetivo de obtener resultados competitivos frente a anteriores métodos, se ha propuesto tres modelos de aprendizaje profundo que potencialmente puedan resolver la tarea de análisis de sentimiento de tuits en español. Se concluye que los tres modelos propuestos presentan un nivel de exactitud competitivo frente a métodos que conforman el estado del arte en los datasets utilizados. Entre ellos, se destaca el modelo basado en redes neuronales convolucionales, que obtiene 61.58 % - 74.14 % de exactitud en los corpus InterTASS y General, logrando así superar a los métodos que conforman el estado del arte en los dos corpus evaluados.

SEGUNDA

Se ha implementado un algoritmo de preprocesamiento para mejorar la calidad de las entradas de los modelos de aprendizaje profundo. El algoritmo tiene dos partes principales: a) la extracción del texto y la polaridad del archivo XML de entrada; b) la utilización de un tokenizador para eliminar el ruido, extraer emoticonos y separar las palabras. Los resultados muestran que, gracias al preprocesamiento realizado, el modelo basado en redes neuronales convolucionales presentó una mejora de 2.2 % y 3.4 % respectivamente

en los corpus evaluados, lo cual ha permitido superar el estado del arte. Se concluye que el preprocesamiento de los datos es una etapa crítica en los métodos basados en aprendizaje profundo. En ese mismo contexto, el preprocesamiento de emoticonos es muy útil en la tarea de análisis de sentimientos ya que un emoticon representa gran cantidad de información subjetiva en un texto, la cual es valiosa en esta tarea. Al mismo tiempo, el modelo basado en redes neuronales convolucionales toma ventaja de la obtención de tres canales de embeddings para mejorar la calidad de representación de palabras que tiene por entrada.

TERCERA

Se han implementado tres modelos basados en aprendizaje profundo, cada uno con características diferentes. El primer modelo se basa en una red neuronal recurrente, ya que es la red por excelencia del procesamiento secuencial. La implementación contiene un modelo de atención para considerar todas las palabras de entrada, y celdas de memoria a corto y largo plazo a fin de evitar el desvanecimiento o explosión de gradientes. El segundo modelo está basado en una red neuronal convolucional, debido a que extrae y procesa el significado de varias palabras a la vez. La implementación realizada incluye convoluciones en paralelo con el objetivo de acelerar el entrenamiento, operaciones de max-pooling con el fin de considerar elementos representativos, y contiene un perceptrón multicapa al final para hacer la red más profunda. La tercera arquitectura es un híbrido de las dos arquitecturas anteriores que realiza las operaciones de la red convolucional sustituyendo el perceptrón multicapa por una red neuronal recurrente. Se considera una combinación de ambas redes con el fin de aprovechar las ventajas de cada una.

CUARTA

El rendimiento de los tres modelos es diferente uno de otro.

a) La red neuronal convolucional obtiene exactitudes de 61.5 % y 74.1 % que lo coloca como el mejor modelo en los corpus InterTASS y General. Se concluye que la capacidad que tiene el modelo de poder procesar grupos de palabras ayuda a mejorar los resultados previos. Aunado a esto, los kernel de convolución $\{1, 2\}$ representan la ponderación para una y dos palabras respectivamente, lo cual concuerda con la tarea de análisis de textos cortos como lo es un tuit.

b) La red neuronal recurrente obtiene exactitudes de 60.4 % y 71.2 % que lo coloca como el tercer lugar frente a anteriores trabajos de referencia. Uno de los inconvenientes que se detectó, es que solamente utiliza una representación de palabras lo cual dificulta la mejora de sus resultados, especialmente en tareas con pocos datos como la de este corpus.

c) El modelo convolucional-recurrente obtiene exactitudes de 55.8 % y 64.2 %, lo cual lo ubica en penúltimo lugar de los métodos de referencia. Aún así, dicho modelo se puede considerar competitivo, debido a que otros métodos del TASS-2017 obtienen resultados aún más bajos. Se concluye que este modelo no tiene resultados a la altura de sus partes debido a que la salida de los filtros de convolución no tiene una relación secuencial entre sí. Esto hace que no se pueda aprovechar las capacidades de la capa recurrente colocada al final del modelo.

CAPÍTULO VI

RECOMENDACIONES

- Al momento de diseñar un modelo de aprendizaje profundo se recomienda revisar la naturaleza de los datos que se procesarán. Esto ayudará a elegir la arquitectura adecuada para la tarea, además de dar ideas para la modificaciones a la arquitectura original.

- Se recomienda priorizar el preprocesamiento de los datos. Debido a que tiene un alto impacto en los resultados obtenidos. Llama la atención que para los experimentos llevados a cabo, el preprocesamiento hizo que los resultados de un modelo competitivo, lograsen superar el estado del arte.

- Debido al uso de nuevas arquitecturas como los transformers, el procesamiento de lenguaje natural actualmente se encuentra en una etapa de mejora continua. Se recomienda proponer nuevos modelos a partir de este tipo de arquitecturas. Asimismo, los modelos pre-entrenados podrían ser de mucha ayuda para este tipo de tareas que cuentan con un corpus pequeño.

Referencias

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... others (2016). Tensorflow: A system for large-scale machine learning. En *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)* (pp. 265–283).

Abdel-Monem, A., Shaalan, K., Rafea, A., y Baraka, H. (2019). A proposed approach for generating arabic from interlingua in a multilingual machine translation system. En *Language engineering conference.*

Amazon. (2019). *Amazon machine learning key concepts.* Amazon Machine Learning. Recuperado de https://docs.aws.amazon.com/machine-learning/latest/dg/amazon-machine-learning-key-concepts. (01-11-2019)

Angadi, S., y Reddy, R. V. S. (2019). Survey on sentiment analysis from affective multimodal content. En *Smart intelligent computing and applications* (pp. 599–607). Springer.

Bahdanau, D., Cho, K., y Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y., Ducharme, R., Vincent, P., y Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137–1155.

Bojanowski, P., Grave, E., Joulin, A., y Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguis-*

tics, *5*, 135–146.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. En *Neurocomputing* (pp. 227–236). Springer.

Cerón-Guzmán, J. A. (2017, September). Classifier ensembles that push the state-of-the-art in sentiment analysis of spanish tweets. En *Proceedings of tass 2017: Workshop on semantic analysis at sepln (tass 2017)* (Vol. 1896). Murcia, Spain: CEUR-WS.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., y Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, *12*(Aug), 2493–2537.

Covington, M. A., Grosz, B. J., y Pereira, F. C. (1994). *Natural language processing for prolog programmers*. Prentice hall Englewood Cliffs (NJ).

D'Andrea, E., Ducange, P., Bechini, A., Renda, A., y Marcelloni, F. (2019). Monitoring the public opinion about the vaccination topic from tweets analysis. *Expert Systems with Applications*, *116*, 209–226.

Deng, L., y Liu, Y. (2018). *Deep learning in natural language processing*. Springer.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Elton, D. C., Turakhia, D., Reddy, N., Boukouvalas, Z., Fuge, M. D., Doherty, R. M., y Chung, P. W. (2019). Using natural language processing techniques to extract information on the properties and functionalities of energetic materials from large text corpora. *arXiv preprint arXiv:1903.00415*.

Géron, A. (2017). *Hands-on machine learning with scikit-learn and tensorflow: concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Inc.

Gers, F. A., Schmidhuber, J., y Cummins, F. (2000). Learning to forget: Continual pre-

diction with lstm. *Neural Computation*, *12*(10), 2451–2471.

González Barba, J. Á. (2017). *Aprendizaje profundo para el procesamiento del lenguaje natural* (Tesis de Master). Universitat Politecnica de Valencia.

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep learning*. MIT Press. (`http://www.deeplearningbook.org`)

Hapke, H. M., Lane, H., y Howard, C. (2019). *Natural language processing in action*. Manning.

Hassan, A., y Mahmood, A. (2017). Deep learning approach for sentiment analysis of short texts. En *Control, automation and robotics (iccar), 2017 3rd international conference on* (pp. 705–710).

Hochreiter, S., y Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hurtado, L.-F., Pla, F., y González, J.-A. (2017). Elirf-upv en tass 2017: Análisis de sentimientos en twitter basado en aprendizaje profundo. En *Proceedings of tass.*

Jácome Galarza, L. R. (2016). Building a predictive model for kaggle's "home depot product search relevance" competition.

Jones, M. T. (2017). *Models for machine learning.* IBM Developer. Recuperado de https://developer.ibm.com/articles/cc-models-machine-learning. (01-11-2019)

Kim, Y. (2014). Convolutional neural networks for sentence classification. En *Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP 2014, october 25-29, 2014, doha, qatar, A meeting of sigdat, a special interest group of the ACL* (pp. 1746–1751).

Lane, H., Howard, C., y Hapke, H. M. (2019). *Natural language processing in action: Understanding, analyzing, and generating text with python*. Manning.

Lanham, M. (2018). *Learn arcore-fundamentals of google arcore: Learn to build augmented reality apps for android, unity, and the web with google arcore 1.0*. Packt Publishing Ltd.

LeCun, Y., Bengio, Y., y Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, H., Yuan, T., Wu, H., Xue, Y., y Hu, X. (2019). Granular computing-based multi-level interactive attention networks for targeted sentiment analysis. *Granular Computing*, 1–9.

Liu, H., y Zhang, L. (2018). Fuzzy rule-based systems for recognition-intensive classification in granular computing context. *Granular Computing*, *3*(4), 355–365.

Liu, S., Bremer, P.-T., Thiagarajan, J. J., Srikumar, V., Wang, B., Livnat, Y., y Pascucci, V. (2017). Visual exploration of semantic relationships in neural word embeddings. *IEEE transactions on visualization and computer graphics*, *24*(1), 553–562.

Luque, F. M., y Pérez, J. M. (2018). Atalaya at tass 2018: Sentiment analysis with tweet embeddings and data augmentation. En *Tass@ sepln* (pp. 29–35).

Machinery, C. (1950). Computing machinery and intelligence-am turing. *Mind*, *59*(236), 433.

Martınez-Cámara, E., Dıaz-Galiano, M., Garcıa-Cumbreras, M., Garcıa-Vega, M., y Villena-Román, J. (2017). Overview of tass 2017. *Proceedings of TASS*, *1896*.

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

Moctezuma, D., Graff, M., Miranda-Jiménez, S., Tellez, E. S., Coronado, A., Sánchez, C. N., y Ortiz-Bejar, J. (2017). A genetic programming approach to sentiment

analysis for twitter: Tass'17. En *Proceedings of tass 2017: Workshop on sentiment analysis at sepln co-located with 33nd sepln conference (sepln 2017)* (Vol. 1896).

Montanés, R., Aznar, R., del Hoyo, R., y de Big Data, G. (2018). Aplicación de un modelo hıbrido de aprendizaje profundo para el análisis de sentimiento en twitter. *Proceedings of TASS*, *2172*.

Nair, V., y Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. En *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 807–814).

Navas-Loro, M., y Rodrıguez-Doncel, V. (2017). Oeg at tass 2017: Spanish sentiment analysis of tweets at document level.

Olivares, C. A. (2016). *Revisión sistemática sobre la aplicación de ontologías de dominio en el análisis de sentimiento* (Tesis de Master). Pontificia Universidad Católica del Perú.

Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, *2*(1–2), 1–135.

Pennington, J., Socher, R., y Manning, C. (2014). Glove: Global vectors for word representation. En *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., ... Iyengar, S. (2019). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, *51*(5), 92.

Ramos, A., Rafael, N., y Huamaní, E. (2017). *Minería de opiniones subjetivas aplicado a una red social de microblogging usando técnicas de minería de textos y máquina vector soporte* (Tesis de Grado). Universidad Nacional de Ingeniería.

Rosá, A., Chiruzzo, L., Etcheverry, M., y Castro, S. (2017). Retuyt en tass 2017: Análisis de sentimientos de tweets en espanol utilizando svm y cnn. En *Proceedings of tass*.

Ruder, S. (2019). *Neural transfer learning for natural language processing* (Tesis Doctoral no publicada). National University of Ireland, Galway.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*(3), 1.

Saif, H., He, Y., y Alani, H. (2012). Semantic sentiment analysis of twitter. En *International semantic web conference* (pp. 508–524).

Severyn, A., y Moschitti, A. (2015). Twitter sentiment analysis with deep convolutional neural networks. En *Proceedings of the 38th international acm sigir conference on research and development in information retrieval* (pp. 959–962).

Sobrino Sande, J. C. (2018). *Análisis de sentimientos en twitter* (Tesis de Master). Universitat Oberta de Catalunya.

Sokolova, M., y Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, *45*(4), 427–437.

Souma, W., Vodenska, I., y Aoyama, H. (2019). Enhanced news sentiment analysis using deep learning methods. *Journal of Computational Social Science*, *2*(1), 33–46.

Sutskever, I., Vinyals, O., y Le, Q. V. (2014). Sequence to sequence learning with neural networks. En *Advances in neural information processing systems* (pp. 3104–3112).

TensorFlow. (2019). *Tensorflow programming*. Custom training: walkthrough. Recuperado de https://www.tensorflow.org/tutorials. (01-11-2019)

Thanaki, J. (2017). *Python natural language processing*. Packt Publishing Ltd.

Van Rossum, G., y Drake Jr, F. L. (2009). Tutorial python. *Editorial: Fred L. Drake Jr*.

Verhulst, P.-F. (1838). Notice sur la loi que la population suit dans son accroissement.

Corresp. Math. Phys., *10*, 113–126.

Vivas, H., Martínez, H. J., y Pérez, R. (2014). Structured secant method for the multilayer perceptron training. *Revista de Ciencias*, *18*(2), 131–150.

Vizcarra, G., Mauricio, A., y Mauricio, L. (2018). A deep learning approach for sentiment analysis in spanish tweets. En *International conference on artificial neural networks* (pp. 622–629).

Voorhees, E. M. (1999). Natural language processing and information retrieval. En *International summer school on information extraction* (pp. 32–48).

Wang, H., y Castanon, J. A. (2015). Sentiment expression via emoticons on social media. En *2015 ieee international conference on big data (big data)* (pp. 2404–2408).

Yang, H., Luo, L., Chueng, L. P., Ling, D., y Chin, F. (2019). Deep learning and its applications to natural language processing. En *Deep learning: Fundamentals, theory and applications* (pp. 89–109). Springer.

Zhang, L., Wang, S., y Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), e1253.

Zhao, J., Liu, K., y Xu, L. (2016). Sentiment analysis: Mining opinions, sentiments, and emotions bing liu (university of illinois at chicago) cambridge university press, 2015 isbn 9781107017894. *Computational Linguistics*, *42*, 1–4.

ANEXOS

ANEXO 1 PUBLICACIÓN

En el presente anexo se presenta la publicación publicada durante el desarrollo de la presente tesis, así como el premio recibido por dicha publicación.

Título: A deep learning approach for sentiment analysis in Spanish Tweets (Vizcarra et al., 2018)

Autor: Vizcarra Aguilar, Gerson
Mauricio, Antoni
Mauricio, Leonidas

Editorial: Springer Verlag

DOI: https://doi.org/10.1007/978-3-030-01424-7_61

ISBN: 978-3-030-01424-7

Conferencia: 27th International Conference on Artificial Neural Networks, ICANN2018; Rodas; Grecia; 4 de Octubre 2018 al 7 de Octubre del 2018.

Universidad Católica San Pablo

Premios a la Excelencia en Investigación Científica (PEIC) 2018

N°	Departamento	Autores	Título de la publicación	Revista / evento	Scopus	Web of Science	Scielo	H5	SJR	SNIP	Métrica PEIC
1	Ingeniería Eléctrica y Electrónica	Khan, Zain Ahmed; **Zenteno, Efrain**; Handel, Peter; Isaksson, Magnus	Extraction of the Third-Order 3 x 3 MIMO Volterra Kernel Outputs Using Multitone Signals	IEEE Transactions on Microwave Theory and Techniques	X	X		57	1.000	1.956	93.95
2	Ingeniería Eléctrica y Electrónica	**Ludeña-Choez J., Choquehuanca-Zevallos J.J., Ma**	Sensor nodes fault detection for agricultural wireless sensor networks based on NMF	Computers and Electronics in Agriculture	X			43	0.814	1.563	74.24
3	Psicología	Caycho-Rodríguez, T., Ventura-León, J., García-Cadena, C.H., Tomás, J.M., Domínguez-Vergara, J., Daniel, L., **Arias-Gallegos, W.Y.L.**	Psychometric evidence of a brief measure of resilience in non-institutionalized Peruvian older adults	Psychosocial Intervention	X	X		18	0.381	1.054	42.06
4	Ingeniería Eléctrica y Electrónica	Khan, Zain Ahmed; **Zenteno, Efrain**; Handel, Peter; Isaksson, Magnus	Multitone Design for Third Order MIMO Volterra Kernels	IEEE MTT-S International Microwave Symposium Digest	X	X		28	0.363	0.780	39.10
5	Ingeniería Eléctrica y Electrónica	**Castillo-Aranibar P.**, García-Lampérez A., Segovia-	Omnidirectional compact dual-band antenna based on dual-frequency unequal split ring resonators for WLAN and WiMAX applications	Progress In Electromagnetics Research M	X			34	0.187	0.524	31.85
6	Psicología	**Huamani Cahua, Julio Cesar; Arias Gallegos, Walter L.; Nunez Cohello, Ana Lucia**	Predictive Model of Purpose of Life in Adolescents of Public Educational Institutions from Arequipa City	Cuadernos de Neuropsicologia-Panamerican Journal of Neuropsychology		X		7	1.319	0.000	28.07
7	Ciencia de la Computación / Psicología	**Santisteban J., Santisteban-Muñoz J.**	Psychometric computational thinking test (poster)	Annual Conference on Innovation and Technology in Computer Science Education (ITICSE 2018)	X			20	0.896	0.000	25.75
8	Grupo de Ciencia y Tecnología Ambiental	**Montalvo Andia, Javier Paul**; Yokoyama, Lidia; Cesar Teixeira, Luiz Alberto	Study of the Equilibrium, Kinetics, and Thermodynamics of Boron Removal from Waters with Commercial Magnesium Oxide	International Journal of Chemical Engineering	X	X		14	0.327	0.489	24.84
9	Ciencia de la Computación	**Vizcarra G., Mauricio A.**, Mauricio L.	A deep learning approach for sentiment analysis in Spanish Tweets	Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	X			0	0.295	0.655	22.33
10	Ciencia de la Computación	**Menejes Palomino N.**, Cámara Chávez G.	Abnormal event detection in video using motion and appearance information	Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	X			0	0.295	0.655	22.33
11	Ciencia de la Computación	**Ochoa-Luna J.**, Ari D.	Deep neural network approaches for Spanish sentiment analysis of short texts	Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	X			0	0.295	0.655	22.33
12	Ciencia de la Computación	García G., Gallardo J., **Mauricio A., López J.**, Del Ca	Detection of Diabetic Retinopathy Based on a Convolutional Neural Network Using Retinal Fundus Images	Lecture Notes in Computer Science, Conferencia: 26th International Conference on Artificial Neural Networks (ICANN)	X	X		14	0.295	0.655	28.48
13	Ciencia de la Computación	**Mauricio A., López J.**, Huauya R., Diaz J.	High-resolution generative adversarial neural networks applied to histological images generation	Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	X			0	0.295	0.655	22.33
14	Ciencia de la Computación	Mantilla, Luis; **Yari, Yessenia; Meza-Lovon, Graciela**	A Novel Fuzzy Probabilistic Clustering Algorithm for Satellite Image Segmentation	IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2018)		X		20	0.270	0.000	13.89
15	Psicología	**Arias W.L.**	La frenología y sus implicancias: Un poco de historia sobre un tema olvidado	Revista Chilena de Neuro-Psiquiatria	X		X	8	0.137	0.254	12.60
16	Humanidades: Instituto para el Matrimonio y la Familia	**Viera, R.A.S., Calcina, R.R.**	Determinantes sociodemográficos de la alta fecundidad en mujeres Peruanas	Revista Chilena de Obstetricia y Ginecología	X			14	0.139	0.142	12.40
17	Psicología	Tomás Caycho-Rodríguez, José Ventura-León, Martín Noe-Grijalva, Miguel Barboza-Palomino, **Walter L. Arias Gallegos**, Mario Reyes-Bossio, Claudio Rojas-Jara	Evidencias psicométricas iniciales de una medida breve sobre preocupación por el cáncer	Psicooncología	X			10	0.164	0.186	12.25
18	Ingeniería Eléctrica y Electrónica	Ernesto Palo Tejada, Victoria Campos Falcon, **Eber**	Design and implementation of a low cost particulate material transducer	IEEE International Instrumentation and Measurement Technology Conference (I2MTC 2018)	X			16	0.178	0.000	10.39
19	Psicología	Adrian Zegarra-Valdivia, Jonathan; **Nadia Chino-Vilca, Brenda**	Social Cognition and Executive Function Impairment in Young Women with Anorexia Nervosa	Clínica y Salud		X	X	13	0.173	0.000	8.98

Premios por departamentos

1ro. en Ingeniería Eléctrica y Electrónica
Prof. Efraín Zenteno Balaños

1ro. en Psicología
Prof. Julio Huamani Cahua, Prof. Walter Arias Gallegos y alumna Ana Lucia Núñez Cohello

1ro. en la línea de Ciencia y Tecnología Ambiental
Prof. Javier Montalvo Andia

1ro. en Ciencia de la Computación
Alumno Antoni Gerson Vizcarra (maestría)

1ero. en Humanidades
Rosa Seperak Viera, Renzo Rivera Calcina

Premios generales

1ro. en el general
Prof. Efraín Zenteno Bolaños

2do. en el general
Prof. Jimmy Ludeña Choez, Prof. Juan J. Choquehuanca Zeballos y Prof. Efraín Mayhua López

ANEXO 2 LISTA DE STOP WORDS

Tabla 0.1: Lista de Stopwords

a	acá	aca	ahi	ahí	al	algo
algún	algun	alguna	algunas	alguno	algunos	allá
alla	allí	alli	ambos	ante	antes	aquel
aquella	aquello	aquellas	aquellos	aquí	aqui	arriba
así	asi	atrás	aun	aunque	cada	casi
como	cómo	con	cual	cuál	cuáles	cuales
cualquieres	cualquier	cualquieras	cualquiera	cuan	cuán	cuándo
cuando	cuantos	cuántos	cuanto	cuánto	de	del
demás	desde	donde	dónde	dos	el	él
ella	ellas	ello	ellos	en	eres	esa
esas	eso	esos	ese	esta	este	éste
esto	estos	etc	etcétera	etcetera	ha	hasta
la	las	lo	los	me	le	les

mis	mi	mío	mio	míos	mios	mía
mías	mia	mias	mientras	muy	ni	nosotras
nosotros	nuestra	nuestro	nuestras	nuestros	os	otra
otras	otros	para	pues	q	que	k
qué	siendo	sino	so	sobre	sr	sra
sres	sta	sus	su	te	tu	tú
tus	un	una	uno	unas	unos	usted
ustedes	vos	vosotras	vosotros	vuestra	vuestro	vuestras
vuestros	y	yo				

Elaborado por el equipo de trabajo.