Instituto Tecnológico de Buenos Aires

Especialización en Ciencia de Datos

Trabajo Final Integrador

Comparación de Modelos para el Análisis de Sentimiento

utilizando Redes Neuronales LSTM y Word Embeddings pre-entrenados

por Gonzalo Julián Poch

Tutora: Dra. Marcela Riccillo

RESUMEN

Resumen

El Procesamiento del Lenguaje Natural (NLP) es una rama del análisis de datos que tiene como objetivo la interpretación por parte de las máquinas de los métodos comunicacionales de los humanos. En profundidad, el Análisis de Sentimiento surge como un desprendimiento de los problemas de clasificación, en los cuales se busca asociar una muestra dada con una clase. Dicho análisis permite obtener una noción generalizada de la positividad o negatividad de una serie de textos dados.

En el presente Trabajo se estudia la aplicación de la técnica de Análisis de Sentimiento sobre un conjunto de datos extraído de la red social Twitter, donde los usuarios realizan publicaciones de textos cortos, generalmente sobre temáticas actuales en tiempo real.

Además, se detalla la aplicación e influencia de diversas técnicas de Word Embeddings para la inicialización de los pesos sinápticos de la Red Neuronal Recurrente con Arquitectura LSTM de forma no aleatoria mediante la comparación de distintos modelos basados en Word2Vec y GloVe. Para este último, se tiene en consideración la aplicación de Transfer Learning, técnica de Aprendizaje Profundo que busca aprovechar la información relacionada con la resolución de un problema y utilizarla sobre otro.

Palabras Clave

Procesamiento del Lenguaje Natural, Análisis de Sentimiento, Redes Neuronales Recurrentes, Arquitectura LSTM, Twitter.

ABSTRACT

Abstract

Natural Language Processing (NLP) is a field in data analysis that aims at the interpretation by machines of the communicational methods of humans. In depth, Sentiment Analysis emerges as a detachment from classification problems, in which it seeks to associate a given sample with a class. Such analysis allows obtaining a generalized notion of the positivity or negativity of a series of given texts.

In this paper, the application of the Sentiment Analysis technique is studied on a set of data extracted from the social network Twitter, where users publish short texts, generally on current topics in real time.

In addition, the application and influence of various Word Embeddings techniques for the initialization of the synaptic weights of the Recurrent Neural Network with LSTM Architecture in a non-random way by comparing different models (based on Word2Vec and GloVe) is detailed. For the latter, the Transfer Learning application is taken into consideration, a Deep Learning technique that seeks to take advantage of the information related to the resolution of a problem and use it on another.

Keywords

Natural Language Processing, Sentiment Analysis, Recurrent Neural Networks, LSTM Architecture, Twitter.

Índice

1. INTRODUCCIÓN

El contexto global de pandemia por coronavirus condujo al gobierno de la República Argentina a tomar medidas tan inéditas como influyentes: durante los últimos días de marzo de 2020 se dispuso una cuarentena obligatoria, se cerraron las fronteras tanto provinciales como internacionales y se suspendieron todas las actividades no esenciales. Estas medidas, con diversos niveles de estrictez, se vieron replicadas a lo largo de los cinco continentes, con el objetivo de mitigar la propagación del virus.

Cuando las políticas influyen sobre las libertades individuales, no solo es relevante hacer foco en el impacto económico, sino también en el impacto psicológico desde una perspectiva de salud pública. El médico psiquiatra y psicoanalista Pedro Horvat describe síntomas que van desde "*la angustia, la ansiedad, la incertidumbre, hasta las crisis emocionales en mayor y en menor medida de acuerdo a cada persona*" (D´Ambra, 2020).

Por diversos motivos, entre los cuales se encuentra el mismo aislamiento, resulta de gran dificultad tener una percepción generalizada de los sentimientos de la sociedad y de sus reacciones respecto a las medidas comunicadas. Mediante el presente estudio, con motivación en los puntos previamente expuestos, se desea comprender los alcances del abordaje de dicha

problemática haciendo uso de técnicas de Procesamiento Natural de Texto, aplicando las mismas sobre datos (textos, en este caso) publicados en redes sociales de microblogueo.

En Argentina, la red social Twitter (que permite publicar mensajes de texto plano con un máximo de 280 caracteres, llamados tweets) es la plataforma de microblogueo más utilizada: una de cada dos personas de entre 16 y 64 años afirma haberla visitado en el último mes (Global Web Index, 2020) y cuenta con 5 millones de usuarios activos (Kemp, 2020). De obtenerse resultados satisfactorios en cuanto a la validez del modelo, se podría generar información valiosa acerca del sentimiento generalizado a partir de los datos disponibles en la red social mencionada previamente.

La información disponible en las redes sociales encapsula conocimiento acerca de la opinión y sentimientos de la población. Por ejemplo, durante una situación de emergencia, algunos usuarios generan información ya sea proporcionando observaciones en primera persona, o aportando conocimientos relevantes de fuentes externas. Twitter, con su naturaleza *real-time,* se ha utilizado con éxito como sensor de terremotos e incendios forestales. Además, se ha demostrado que los datos de Twitter geolocalizados son una fuente fiable para detectar desastres e investigar la respuesta (Lu et al., 2015).

Si bien el potencial hallazgo de tendencias respecto a los sentimientos, positivos o negativos, no signifique la resolución de ninguna de las cuestiones antes mencionadas, podría interpretarse como una herramienta orientativa para conocer el impacto generado sobre, al menos, una porción de la sociedad.

2. Estado del Arte

El Análisis de Sentimiento puede definirse como una técnica que tiene como objetivo identificar el estado afectivo o la intención comunicativa emocional de un interlocutor (o usuario) dado, asegurando "*la polaridad del lenguaje natural mediante la clasificación supervisada y/o no supervisada*" (Rasool et al., 2019). La misma, se enmarca en el campo de las Ciencias de la Computación, de la Inteligencia Artificial y de la Lingüística, llamado Procesamiento del Lenguaje Natural (PLN o NLP, por sus siglas en inglés), el cual se ocupa de formular e investigar mecanismos para comprender el lenguaje, como también aspectos generales cognitivos humanos (Liddy, 2001).

El creciente uso de las herramientas digitales como métodos de expresión pública trae consigo una oportunidad de obtención de información agregada respecto a la opinión popular. Como Liu (2012) afirma, existe una coincidencia entre el crecimiento del campo del análisis de

datos, que en conjunto con el creciente uso de las redes sociales brindan un escenario completo para la puesta en práctica de las técnicas de Análisis de Sentimiento.

Sin embargo, existen ciertas limitaciones a la hora de llevar a cabo este tipo de análisis sobre redes sociales, en particular si se considera que las publicaciones suelen ser breves y no estructuradas. El Análisis de Sentimiento se ha estudiado extensamente en diversas áreas, como puede ser el caso de las reseñas de productos o marcas. Sin embargo, el rendimiento de estas metodologías sobre datos de redes sociales puede resultar insatisfactorio debido a las características que estos datos poseen (Lu et al., 2015).

Recientemente, la técnica de Word Embedding (técnica de aprendizaje en PLN) y las Redes Neuronales tales como las Redes Neuronales Convolucionales (CNN), Redes Neuronales Recurrentes (RNN) y la Arquitectura LSTM (Long Short-Term Memory) fueron empleadas satisfactoriamente para realizar Análisis de Sentimiento.

Existen distintos trabajos en donde se demuestra la utilización de estructuras de RNN en conjunto de una Arquitectura LSTM para la confección de un modelo clasificador que trabaje específicamente sobre textos cortos. Entre ellos, se destaca (J. Wang et al., 2016) que investiga la efectividad de las Redes Neuronales LSTM para la clasificación de sentimiento, basado en *Word2Vec.* Además, existen diversos *papers* donde se estudia específicamente la aplicación de *Word Embeddings* como (Rudkowsky et al., 2018), donde se utilizan como parte de un procedimiento de aprendizaje automático supervisado que estima los niveles de negatividad en discursos parlamentarios.

3. Definición del Problema

Según lo descripto previamente, resulta de importancia comprender la potencialidad de la aplicación de modelos computacionales para el procesamiento de datos y Análisis de Sentimiento sobre la información que se hace pública a través de las redes sociales.

4. Justificación del estudio

Este estudio busca proporcionar evidencia, a partir de la aplicación práctica de técnicas de Análisis de Sentimiento y, en particular, de la Arquitectura LSTM (Long Short-Term Memory), para determinar la validez del método a la hora de determinar tendencias de sentimiento.

5. Alcances del trabajo y limitaciones

El presente estudio pretende exponer los alcances de la aplicación de Redes Neuronales Recurrentes con Arquitectura LSTM para el Análisis de Sentimiento sobre un conjunto de datos

etiquetados y su potencial validez para la determinación de la polaridad de las opiniones esbozadas por los usuarios de las redes sociales. Para ello, se desea acceder a datos públicos (publicaciones en Twitter) previamente etiquetados, los cuales deberán ser pre procesados para su tratamiento mediante un modelo de clasificación.

Por otra parte, existen ciertas limitaciones a la hora de llevar a cabo este tipo de análisis sobre redes sociales, en particular si se considera que las publicaciones suelen ser breves y no estructuradas (Lu et al., 2015). Es por ello por lo que, dentro del presente trabajo, se plantea como objetivo la descripción de las limitaciones prácticas que se presentan al aplicar Análisis de Sentimiento mediante una Arquitectura de Redes Neuronales LSTM sobre un *dataset* que contiene *tweets*.

Además, con motivos de disponibilidad de información, todas las etapas experimentales se desarrollarán considerando textos en idioma inglés. Si bien el proceso es análogo, podría ocurrir que las métricas seleccionadas para determinar la validez del modelo se vean afectadas por cuestiones propias del lenguaje.

6. Hipótesis

Mediante técnicas de Procesamiento Natural de Texto (Natural Language Processing o NLP) se podría identificar la variación del sentimiento de los usuarios de las redes sociales.

Variable independiente: tiempo.

Variable dependiente: sentimiento (nominal).

Variable contextual: redes sociales.

Definición nominal de la variable. Sentimiento: estado afectivo o intención comunicativa emocional de un interlocutor o usuario con respecto a algún tema.

Definición operacional de la variable. Sentimiento: es la etiqueta (positivo, negativo o neutro) que se le asigna a un texto particular según el análisis de las palabras que lo conforman.

7. Objetivos

7.1 Objetivo General

Construir una herramienta basada en Redes Neuronales Recurrentes, que permita determinar la polaridad de un conjunto de palabras pertenecientes a una oración o a un conjunto de oraciones y determinar su validez mediante las métricas correspondientes a los modelos de clasificación.

7.2 Objetivos Específicos

- Realizar una limpieza y adaptación de los datos de modo que su abordaje mediante técnicas de Procesamiento Natural de Texto sea factible.
- Construir una herramienta de Sentiment Analysis basada en Redes Neuronales Recurrentes de tipo LSTM para clasificar las distintas publicaciones según su polaridad (positivo o negativo).
- Comparar la performance de diferentes modelos según la parametrización definida para la capa de Word Embeddings.

8. METODOLOGÍA

8.1 Técnicas

El Análisis de Sentimiento puede definirse como una técnica que tiene como objetivo identificar el estado afectivo o la intención comunicativa emocional de un interlocutor (o usuario) dado, asegurando la polaridad del lenguaje natural mediante la clasificación supervisada y/o no supervisada. A partir de dicha técnica, se desea identificar la expresión del sentimiento de los usuarios mediante sus publicaciones en las redes sociales.

La estrategia por utilizar consiste en seis grandes pasos:

- **Obtención de los datos.** Se deberá acceder a una colección lo suficientemente voluminosa de *tweets,* idealmente clasificados. Esta clasificación se espera que especifique el sentimiento y será útil para el entrenamiento del modelo.
- **Preparación de los datos.** Para llevar a cabo un procesamiento de texto utilizando Redes Neuronales, se debe realizar una preparación de los datos (técnicamente, un preprocesamiento). El mismo consiste en la *Tokenización* (dividir el texto en oraciones y luego las oraciones en palabras), la Normalización (estandarización de las palabras utilizadas, por ejemplo, mediante la eliminación de mayúsculas) y, si aplicara, la eliminación del ruido (se omiten o eliminan palabras redundantes).
- **Word Embedding.** Mediante esta técnica de aprendizaje, se logra representar a las palabras del lenguaje natural como vectores de números reales. De esta manera, es

posible aumentar el rendimiento de las tareas en el Análisis de Sentimiento (Socher et al., 2013).

- **Entrenamiento del modelo.** A partir de la colección etiquetada, se realiza un entrenamiento del modelo clasificador, el cual será el encargado de asignarle una polaridad a los datos proporcionados. Este modelo consiste en una Arquitectura LSTM, que surge como una variación de las Redes Neuronales Recurrentes tradicionales, la cual permite hacer frente a los fallos a la hora de aprender, seleccionando de forma dinámica la representación multidimensional de cada palabra analizada (Gers et al., 2000).
- **Evaluación del modelo.** Una vez obtenido el modelo, se lleva a cabo una evaluación de este a través de métricas que permiten definir la *performance.* En esta instancia se muestran métricas tales como *accuracy* (predicciones correctas sobre el total)*, precision* (precisión de una clase en particular) y *recall* (capacidad del modelo de detectar una clase particular).

8.2 Herramientas

Se realizará mediante Python (Python Software Foundation, 2021) el desarrollo de una solución que permita el preprocesamiento y entrenamiento de los datos, como así también la evaluación del modelo obtenido con métricas propias del mismo.

Se prevé el uso de las siguientes librerías, todas ellas de carácter *open source.*

- *Pandas:* librería de software para manipulación y análisis de datos.
- *NumPy:* librería de software que brinda soporte para el manejo de matrices y arreglos multidimensionales de gran dimensión, junto con una gran colección de funciones matemáticas de alto nivel para operar entre ellos.
- *TensorFlow:* librería para Machine Learning desarrollada por Google, con especial foco en el entrenamiento y la inferencia de Redes Neuronales Profundas (Deep Neural Networks).
- *Keras:* librería que provee una interfaz para trabajar con Redes Neuronales Artificiales. Esta se complementa con TensorFlow y está enfocada en ser amigable para el usuario.
- *Matlplotlib:* es una librería para realizar gráficos en Python y se complementa con NumPy.

8.2.1 Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (PLN, o NLP por sus siglas en inglés) se enmarca en las ciencias lingüísticas, las cuales tienen como objetivo central caracterizar y explicar los modos de comunicación que la sociedad utiliza, ya sea de forma verbal, escrita o por cualquier otro medio.

La caracterización de los modos de comunicación puede interpretarse de diversas maneras. No solo se trata de comprender el lenguaje como herramienta de interacción entre los humanos, sino también sus estructuras intrínsecas, las cuales nacen tanto explícita como implícitamente a modo de estandarización para facilitar la misma comunicación. Sin embargo, tal como se indica en (Manning et al., 2002), particularmente en este último siglo, las miradas de los lingüistas fueron reticentes hacia esta idea: "*no es posible proveer una caracterización basada en reglas y enunciados para describir cualquier secuencia de palabras*". Esto se debe a que la sociedad se encuentra forzando continuamente estas 'reglas' para solventar sus necesidades comunicacionales.

En función de las dificultades mencionadas para interpretar y explicar el lenguaje, nacen nuevas formas de aproximarse a la resolución de este problema, acompañadas por las nuevas tecnologías y el crecimiento significativo de las Ciencias de Datos. Este enfoque que combina el potencial de procesamiento de grandes volúmenes de datos junto con fundamentos estadísticos tiene como objetivo la detección de patrones comunes que ocurren a la hora del uso del lenguaje.

La principal dificultad en PLN es la insoslayable ambigüedad localizada en todos los niveles del problema: ambigüedad léxica simple (una misma palabra puede ser un sustantivo, un adjetivo o un verbo), ambigüedad estructural o sintáctica (por ejemplo, en la frase 'vi a un hombre con un telescopio', el telescopio podría haber sido utilizado para ver al hombre o bien el hombre observado podría estar sosteniéndolo), ambigüedad semántica (una misma palabra puede tener numerosos significados), ambigüedad pragmática (por ejemplo, la pregunta 'puedes levantar esa piedra?' puede ser dicotómica o bien un pedido para levantar la piedra) y ambigüedad referencial (no siempre es claro quién es el sujeto cuando se utilizan pronombres personales) (Allen, 2003).

De esta manera, resulta claro que las limitaciones crecen de manera exponencial cuando se desea lograr una interpretación computacional de los textos semejante a la humana. Es por ello por lo que, en vistas de la detección de patrones previamente mencionada, surgen

estrategias analíticas más simplistas que plantean la comprensión de características del texto como un problema de clasificación.

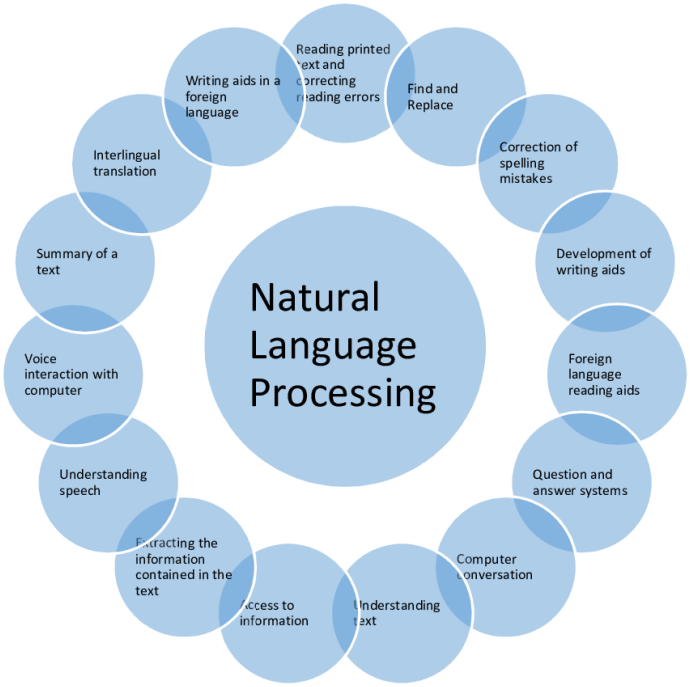


Figura 1. Tópicos del Procesamiento del Lenguaje Natural (Adalı, 2012).

8.2.2 Análisis de Sentimiento

Teniendo en consideración las limitaciones expresadas nacen técnicas como el Análisis de Sentimiento, que busca determinar la polaridad de un texto independientemente del significado real que tenga.

Esta tarea recibió un interés creciente de la comunidad de investigación en los últimos años. El trabajo realizado hasta hoy tiene en cuenta la forma en la que el "sentimiento" puede ser clasificado en textos asociados con diferentes géneros y lenguajes, en el contexto de diferentes aplicaciones (Pang & Lee, 2008).

El resultado de estos análisis mostró que los distintos tipos de textos requieren métodos especializados para el Análisis de Sentimiento. Esto significa que, por ejemplo, no se pueden afrontar los sentimientos de la misma manera con textos de un diario que con textos de blogs, reseñas, foros u otro tipo de contenido generado por usuarios. A la luz de estos hallazgos, lidiar con el Análisis de Sentimiento en Twitter requiere de un análisis adicional de las características de dichos textos y el diseño de métodos adaptados (Balahur et al., 2010).

8.2.3 Análisis de Sentimiento en Twitter

La tarea de identificar la polaridad de un texto de corta longitud (Twitter establece un máximo de 280 caracteres por publicación) puede parecer sencilla para una persona. Sin embargo, pueden existir ciertas discrepancias cuando un mismo texto se presenta a más de una persona. Estas discrepancias pueden deberse a diferentes razones: cuestiones propias del vocabulario, subjetividades ligadas a la opinión del observador o simplemente por un error a la hora de realizar la clasificación (Mozetič et al., 2016).

Adicionalmente, se pueden generar controversias en las clasificaciones de oraciones según el contexto que se tiene de ellas. En términos generales, los observadores tienden a clasificar más oraciones como neutras cuando tienen menos información del contexto discursivo (Kumari Yeruva et al., 2020).

Las publicaciones en Twitter son, en particular, extremadamente dependientes del tiempo (o, justamente, del contexto). En otras palabras, esto quiere decir que los usuarios tienen tendencia a comentar sobre sucesos que ocurrieron en el corto plazo, muchas veces haciendo referencias tácitas a los mismos. Como problemática adicional a la hora de automatizar la identificación de patrones en el contenido publicado en la red social, se suele escribir con un lenguaje coloquial, lo cual puede interferir con aquellos modelos que tienen sustento en un diccionario finito de palabras.

Para obtener una expresión cuantitativa que describa la calidad de las clasificaciones humanas (en inglés, *human annotations*), se utilizan distintos coeficientes que permiten contrastar la concordancia entre dos observadores respecto a una muestra: coeficiente *kappa de Cohen* para medir la concordancia entre dos observadores y los coeficientes *kappa de Fleiss* y *Alpha de Krippendorff* para el caso de más de dos observadores (Bobicev & Sokolova, 2017). Si bien no es el objeto del presente informe estudiar las limitaciones en concordancia de observadores humanos, es importante destacar que las mismas generan una cota superior al rendimiento (HLP, siglas en inglés de *Human Level Performance*) de los clasificadores modelados mediante técnicas de *Machine Learning* cuando justamente los datos con los que se alimentan fueron etiquetados por humanos (Ng, 2017).

Teniendo en consideración las limitaciones propias del Análisis de Sentimiento, es posible generar modelos que se aproximen al nivel humano de clasificación. La gran ventaja de realizarlo de forma automatizada es que permite obtener resultados direccionales para comprender el sentimiento generalizado, considerando que desde una computadora hogareña es posible procesar millones de *tweets* en cuestión de minutos. De esta manera, se

podrían inferir las sensaciones respecto a un suceso en particular al momento en el que el mismo está ocurriendo.

8.2.4 Redes Neuronales

Tal como sucede con la mayoría de los desarrollos en el área de cómputo, las Redes Neuronales Artificiales (RNA) se inspiran en nosotros, los seres humanos. Particularmente, este tipo de modelos computacionales toman como punto de partida las estructuras neuronales biológicas que se encuentran en el cerebro humano. Estas redes contienen capas organizadas de unidades interconectadas o nodos. Las RNA se utilizan principalmente para tareas en las que es difícil derivar restricciones lógicas de forma explícita, como el reconocimiento de patrones y el análisis predictivo (Yegnanarayana, 1994).

Las neuronas de las diferentes capas se encuentran interconectadas entre sí mediante pesos sinápticos. De esta manera, la salida de una neurona puede convertirse en la entrada de la siguiente. En función de la fuerza y el significado de la conexión, se asignan los pesos sinápticos: cuanto más significativa sea la ponderación, mayor será la influencia que una neurona podrá ejercer con respecto a otra contigua. La Figura 2 muestra un ejemplo de una Red Neuronal con una capa de entrada, una oculta y una de salida.

Las neuronas localizadas en la capa de entrada reciben información del mundo exterior en forma de patrones o señales. Por otra parte, las neuronas que forman parte de la capa oculta, situadas entre las de entrada y las de salida, mapean patrones de información externa. Por último, las neuronas de la capa de salida transmiten información en forma de señales como resultado.

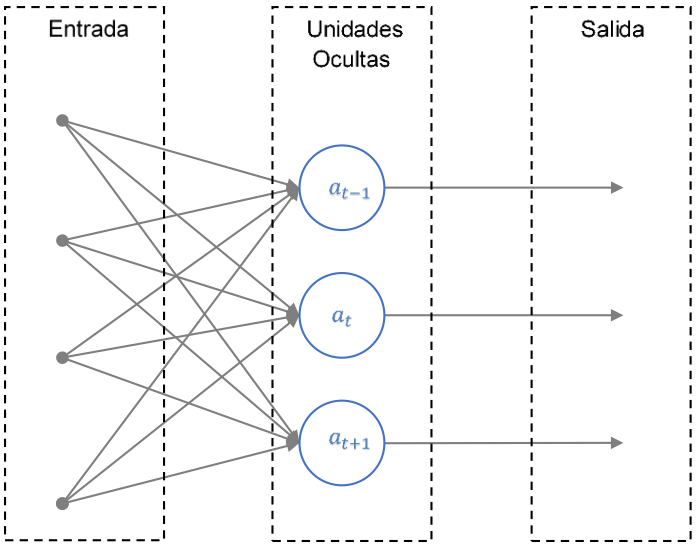


Figura 2. Estructura de una Red Neuronal sencilla.

En una Red Neuronal, a las capas intermedias situadas entre la entrada y la salida se les denomina capas ocultas. Cada una de estas unidades intermedias es el resultado de aplicar una función a la suma ponderada de cada elemento de la entrada (cada salida de las unidades de la capa anterior, conectada mediante los pesos a dicha unidad). De esta misma forma se calcula la salida, solo que esta vez tendrá el resultado de las unidades ocultas en vez de la entrada para realizar las operaciones.

A la hora de interpretar un texto, el ser humano se basa indefectiblemente en el contexto alrededor del cual se encuentra. En efecto, existen numerosas palabras llamadas homógrafas, las cuales se escriben exactamente de la misma manera, pero pueden asociarse a diferentes significados según la oración en la que se las utiliza. Las Redes Neuronales tradicionales no son capaces de realizar este procesamiento basado en contexto y es por ello por lo que surgieron las Redes Neuronales Recurrentes.

Una Red Neuronal Recurrente (RNN), es un modelo o tipo de arquitectura de Red Neuronal creado para procesar y obtener información de datos secuenciales. El análisis de video, la subtitulación de imágenes, el Procesamiento del Lenguaje Natural (PLN) y el análisis de la música son ejemplos a aplicación de las capacidades de las Redes Neuronales Recurrentes. A diferencia de las Redes Neuronales Artificiales tradicionales, que asumen la independencia entre los datos de entrada, las RNN capturan activamente sus dependencias secuenciales y temporales.

Uno de los atributos más definitorios de las Redes Neuronales Recurrentes es que comparten sus parámetros. Sin compartir parámetros, el modelo asignaría parámetros únicos

para representar a cada dato en una secuencia y, por lo tanto, no podría realizar inferencias sobre secuencias de longitud variable. El impacto de esta limitación puede observarse de forma notoria en el Procesamiento del Lenguaje Natural. Una Red Multicapa tradicional fallaría porque crearía una interpretación del lenguaje con respecto a los parámetros únicos establecidos para cada posición (palabra) en una frase. Las RNN, sin embargo, serían más adecuadas para la tarea, ya que comparten pesos entre los datos espaciados secuencialmente (Arana, 2021).

La estructura de la red es similar a una red estándar de varias capas, con la diferencia de que permite conexiones a las unidades ocultas que están asociadas con un retardo en el tiempo. "*A través de estas conexiones, el modelo es capaz de retener información del pasado, permitiendo descubrir correlaciones entre eventos que están muy separados los unos de los otros en el tiempo*" (Pascanu et al., 2013). En la Figura 3 se muestra la estructura básica de una RNN.

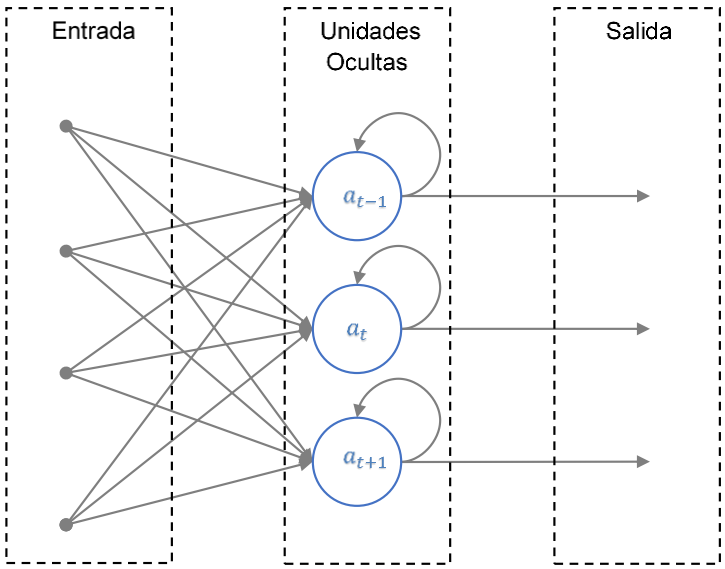


Figura 3. Estructura de una RNN.

A diferencia de las arquitecturas de Redes Neuronales con propagación hacia adelante o de tipo Perceptrón (Feed-forward Neural Networks) donde las conexiones entre las unidades no conforman un ciclo, estas contienen conexiones de retroalimentación. Estos ciclos constituyen la memoria interna de la red que se utiliza para evaluar las propiedades del dato actual con respecto a los datos del pasado inmediato. También es importante tener en cuenta que la mayoría de las Redes Neuronales del tipo perceptrón están limitadas a un mapeo uno a

uno de la entrada y la salida. Las RNN, sin embargo, pueden realizar mapeos de uno a muchos, de muchos a muchos (por ejemplo, traducir el habla) y de muchos a uno (por ejemplo, identificar la voz) (Goodfellow et al., 2016).

8.2.5 Arquitectura LSTM

A pesar de que en principio el modelo de una RNN parece simple y potente, en la práctica es muy difícil entrenar estas redes y conseguir los resultados deseados. Como se ha explicado con anterioridad, las Redes Neuronales Recurrentes basan su funcionamiento en el procesado de secuencias de información, agregando al cómputo de cada elemento de la secuencia, el resultado del análisis del elemento anterior. Es decir, el resultado final de una RNN será una función que agregará el procesamiento de todos los elementos de la secuencia. Sin embargo, a medida que la red procesa más elementos de la cadena, tiene problemas en recordar información pasada, lo que se conoce como desvanecimiento de gradiente (*vanishing gradient)* (Nielsen, 2019).

Este problema surge como consecuencia del proceso matemático mediante el cual las Redes Neuronales Recurrentes son capaces de entrenarse a partir de ciclos de un corpus (*back-propagation through time*). A medida que la RNN recibe elementos de la secuencia, el gradiente del error se propaga hacia atrás. Es por esto que las RNN más sencillas no son capaces de aprender patrones muy extendidos en el tiempo, si no que solo son eficaces en rangos cortos, como por ejemplo secuencias de 10 elementos (Williams & Zipser, 1995).

LSTM (del inglés Long Short-Term Memory) es una arquitectura utilizada en el campo de Deep Learning para la confección de Redes Neuronales Recurrentes (RNN), especialmente diseñada para sobrellevar los problemas antes mencionados. Una red LSTM puede retener información en intervalos de tiempo de más de 1000 pasos incluso en el caso de secuencias de entrada ruidosas e incompresibles. Esto se logra mediante un algoritmo eficiente basado en gradientes para una arquitectura que mantiene un error constante (por lo tanto, no se genera el desvanecimiento de gradiente) a través de estados internos de en la capa oculta (Hochreiter & Schmidhuber, 1997).

8.2.6 Secuencias de Palabras versus Bolsas de Palabras

En el lenguaje cotidiano pueden existir frases que contienen las mismas palabras, pero en un orden diferente, lo que genera que la frase en sí tome otro significado. Los modelos basados

en bolsas de palabras no pueden tener esto en consideración y, por lo tanto, probablemente clasifiquen estas frases bajo la misma categoría. La técnica de Word Embedding (descrita en el apartado 8.3.1.1) utilizada en conjunto con un modelo de Redes Neuronales sin capacidad de memoria no es suficiente, ya que los vectores de palabras pre-entrenados son una función de una única palabra y no tienen en cuenta la posición de esta en cada oración. Cualquiera de las soluciones que utilizan este enfoque de bolsa de palabras para convertir un corpus en vectores numéricos sufren la misma deficiencia. En base a esta problemática, se plantea un modelo basado en Redes Neuronales LSTM y Word Embeddings, que haga frente a la necesidad de encontrar una solución que tenga la capacidad de interpretar no solo el significado semántico de las palabras, sino también el peso de estas según su ubicación en una oración.

Como se introdujo anteriormente, las Redes Neuronales LSTM son una rama de las Redes Neuronales Recurrentes (RNN) cuyo sello distintivo es aprender y predecir a partir de secuencias, por lo que tienen la capacidad de sobreponerse a las limitaciones mencionadas *ut supra*. Para detectar la polaridad de sentimientos en textos cortos, resulta necesario explorar la semántica más profunda de las palabras utilizando métodos de Aprendizaje Profundo o *Deep Learning* (J. H. Wang et al., 2018), situando cada una de las palabras en un contexto dado.

En el presente trabajo se desea comprender los efectos de la aplicación de la Arquitectura LSTM para la clasificación de sentimientos, justamente en textos cortos como lo son los *tweets*. Primero, los tweets son pre-procesados para eliminar los diferentes componentes que podrían generar ruido y se normalizan las palabras. En segundo lugar, las palabras en las publicaciones se convierten en vectores usando modelos de W*ord Embeddings*. Luego, las secuencias de palabras convertidas en vectores alimentan a la red LSTM para aprender la dependencia contextual entre las palabras. Por último, se define una polaridad para cada una de las publicaciones.

8.3 Parametrización de la Red Neuronal

Los hiperparámetros de una Red Neuronal son aquellos que describen las características que tendrá la arquitectura a utilizar y que deben establecerse para llevar a cabo el entrenamiento del modelo. Estas características brindarán información a la Red Neuronal para determinar de qué manera se gestionarán las secuencias de datos con los cuales se la alimente.

De esta manera, el desarrollador del modelo tiene la capacidad de customizarlo, planteándose como objetivo la obtención de aquellos parámetros que maximicen las métricas observadas.

8.3.1 Capa de Embedding

La capa de Embedding es una de las capas disponibles en la Arquitectura LSTM, integrada en la librería Keras. Mediante esta capa es posible definir los parámetros iniciales para que la Red Neuronal cuente con un espacio vectorial donde se representen las relaciones semánticas entre las palabras del corpus.

Por defecto, los pesos sinápticos se inicializan de forma aleatoria. Sin embargo, existe la posibilidad de alimentar la Red con una matriz de Embeddings pre-entrenados, lo cual facilita el proceso de entrenamiento.

8.3.1.1 Word Embeddings

Una de las partes más importantes del Procesamiento del Lenguaje Natural está relacionada con la representación de las palabras. Para el abordaje de esta problemática, se desea emplear una representación que capture similitudes semánticas y sintácticas entre las palabras. Un paradigma muy común para adquirir tales representaciones se basa en la hipótesis distributiva de (Harris, 1954), que afirma que "*las palabras en contextos similares tienen significados similares*".

Recientemente, con base en la hipótesis distributiva, se propuso la representación de las palabras como vectores generados a través del entrenamiento de modelos basados en arquitecturas de Redes Neuronales, conocidos como Word Embeddings (Bengio et al., 2003). Los Word Embeddings representan el significado a través de la geometría. Tal como se muestra en la Figura 4, proporcionan representaciones vectoriales de palabras de modo que la relación entre dos vectores refleja la relación lingüística entre las palabras. Además, disponen de un número fijo de dimensiones para cada elemento, lo que facilita su integración con la Arquitectura LSTM que, tal como se mencionó previamente, requiere alimentarse con elementos de igual dimensionalidad.

De forma predeterminada, la capa de Word Embeddings en una Red Neuronal Recurrente se inicializan aleatoriamente. Luego, sufren modificaciones de forma gradual durante la fase de entrenamiento gracias a la retropropagación (o *backpropagation)*, descripta en 9.3.5. De este modo, se logra que las palabras con similitud semántica se ubiquen cerca en términos de distancia en el espacio vectorial.

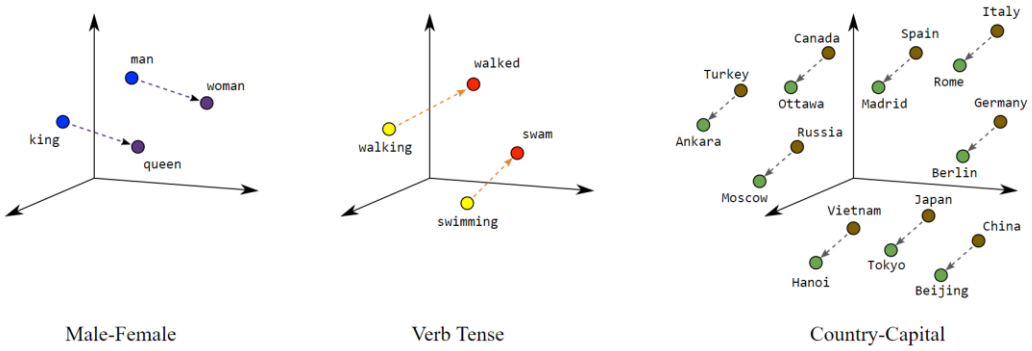


Figura 4. Relaciones lineales entre palabras en el espacio vectorial (Google, 2021).

8.3.2 Activación

La función de activación es una especie de compuerta matemática, situada entre la entrada que alimenta una neurona y su salida hacia la capa siguiente. Esta función define la transformación que sufrirá el peso sináptico mediante el cual fue alimentada.

Si no se tuviera una función de activación, los pesos y el sesgo simplemente sufrirían una transformación lineal, la cual tiene una capacidad limitada para resolver problemas complejos y tiene menos poder para aprender relaciones entre los datos. "*Generalmente, las Redes Neuronales utilizan funciones de activación no lineales, que pueden ayudar a la red a aprender datos complejos, calcular y proporcionar predicciones precisas"* (Gupta, 2020).

8.3.3 Dropout

Durante la etapa de entrenamiento de una Red Neuronal puede ocurrir lo que se conoce como *overfitting* (o sobreajuste). El efecto de esto sobre el modelo es que el mismo aprende el ruido estadístico que puede estar presente en el conjunto de entrenamiento, lo cual se traduce en un mal rendimiento al momento de evaluarlo frente a un conjunto de datos de prueba.

Un enfoque para la reducción de este sobreajuste sería entrenar múltiples Redes Neuronales diferentes, utilizando el mismo conjunto de entrenamiento y promediando los resultados obtenidos. Sin embargo, entrenar arquitecturas diferentes es costoso en cuanto a la dificultad que puede conllevar la búsqueda de hiperparámetros óptimos y, además, en cuanto a la capacidad de cómputo necesaria. Adicionalmente, las Redes Neuronales requieren ser alimentadas con grandes cantidades de datos y, en ciertos casos,

no se cuenta con la cantidad suficiente para subdividirlos en mútliples subconjuntos de entrenamiento.

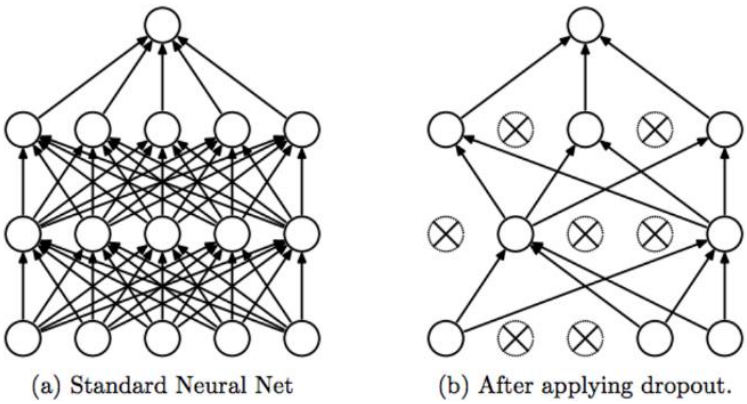


Figura 5. Dropout en Redes Neuronales (Srivastava et al., 2014)**.**

(a) Red Neuronal estándar con dos capas ocultas.

(b) Ejemplo de Red Neuronal resultante al aplicar Dropout sobre la anterior.

La técnica de Dropout previene el sobreajuste y provee una manera de combinar diferentes arquitecturas de una forma eficiente, emulando lo mencionado en previamente. El término Dropout se traduce como abandonar o expulsar: justamente, lo que permite esta técnica es dejar de lado ciertas neuronas y sus conexiones de la Red Neuronal de forma temporal y aleatoria durante la etapa de entrenamiento, obteniendo como resultado una Red como la que se muestra en la Figura 5 (Srivastava et al., 2014).

8.3.4 Optimización

En *Deep Learning* se utiliza el concepto de pérdida o *loss*, que nos dice qué tan mal se está desempeñando el modelo en un determinado momento. Este concepto se utiliza durante la etapa de entrenamiento para tener una noción de la performance del modelo. Optimización es, justamente, el nombre que recibe el proceso de minimizar la pérdida, con el objetivo de obtener el mejor modelo posible.

8.3.4.1 Función de pérdida

Como se mencionó previamente, la función de pérdida es la que indica el rendimiento del modelo y es análoga al error. De esta manera, al momento de calcular el error durante el proceso de optimización, se debe seleccionar una función de pérdida.

De la misma manera que sucede con la función de activación, es relevante que la función de pérdida seleccionada se ajuste a la naturaleza del problema y represente los objetivos del modelo (Hammer, 2001).

8.3.4.2 Optimizador

Los optimizadores son algoritmos o métodos utilizados para minimizar la función de pérdida. Basados en funciones matemáticas, son dependientes de los parámetros que el modelo aprende como, por ejemplo, los pesos sinápticos. Gracias a los mismos, la Red Neuronal comprende de qué manera ajustar los pesos y la tasa de aprendizaje (hiperparámetro configurable que determina qué tan rápido del modelo se adapta al problema).

8.3.4.3 Callbacks

Al momento de entrenar la Red Neuronal existen diferentes parámetros a definir. Uno de ellos es el parámetro de 'epochs' o épocas, bajo el cual se selecciona la cantidad de iteraciones mediante las cuales se alimenta al modelo con la totalidad del conjunto de datos. Cada vez que comienza una época, se actualizan los pesos sinápticos con el objetivo de minimizar la función de pérdida.

Sin embargo, realizar una cantidad sobredimensionada de épocas no solo puede derivar en un consumo de tiempo alto, sino también volverse innecesario si la convergencia de la función de pérdida se detiene con anterioridad. En el peor de los casos, esto podría conllevar a un sobrejauste que aparte al modelo de su rendimiento óptimo y no tenga validez.

La librería Keras (Keras, 2021) ofrece entre sus herramientas los llamados *Callbacks* que buscan abordar esta situación. Funcionan como una especie de reguladores que permiten modificar la tasa de aprendizaje o detener el entrenamiento en base a las métricas que se les asigne monitorear.

9. EXPERIMENTACIÓN

9.1 Características del conjunto de datos

9.1.1 Obtención de los datos

Los datos utilizados para llevar a cabo el presente Trabajo se obtuvieron de la comunidad Kaggle (Kaggle, 2021), la cual es una plataforma en línea de científicos de datos y profesionales del aprendizaje automático perteneciente a Google LLC. La misma, además de hacer las veces de entorno de desarrollo para proyectos de análisis de datos, permite a los usuarios encontrar y publicar *datasets* (conjuntos de datos).

Sentiment140 (Go et al., 2009) es el conjunto de datos obtenido de Kaggle (Kazanova, 2017) para la puesta en práctica de los conceptos previamente descriptos. Este *dataset* fue creado originalmente en la Universidad de Stanford como resultado de un proyecto universitario. El mismo contiene 1,6 millones de *tweets* que fueron extraídos mediante la API de Twitter.

Se aborda la situación a partir de un set de datos preexistente en lugar de incluir la obtención de este dentro del alcance del trabajo en función de dos factores:

- La API de Twitter ya no permite la extracción ilimitada de datos históricos de forma gratuita. Por el contrario, existen diversos esquemas de cotización según lo que el usuario desee (Twitter, 2021).
- Los datos contenidos en Sentiment140 permiten optimizar el tiempo de implementación de los diversos modelos ya que cada uno de los millones de *tweets* que contiene ya se encuentran clasificados como positivos o negativos. De esta manera, se evita el trabajo falible y costoso de clasificación. Al no incluirse una clase neutral en los datos utilizados por definición de los creadores de este conjunto de datos, se hereda como limitación al presente trabajo, el cual se circunscribe en una clasificación binaria.

9.1.2 Análisis exploratorio

A la hora de llevar a cabo un trabajo basado de datos resulta indispensable el desarrollo de un análisis exploratorio de los datos (también conocido como *Exploratory Data Analysis* o, simplemente, EDA).

La suposición subyacente del enfoque exploratorio es que cuanto más se conoce acerca de los datos, más efectivo será el uso que se podrá hacer de los mismos para desarrollar, testear o refinar una teoría. De esta manera, el EDA tiene como objetivo la maximización de la cantidad de información que se puede obtener de un conjunto de datos (Hartwig & Dearing, 1979).

En primer lugar, se desea conocer el esquema del conjunto de datos Sentiment140. Tal como se indica en la fuente de la cual se obtuvo, el mismo cuenta con 1,6 millones de tweets descriptos por las siguientes cinco columnas:

- **Target:** indica la polaridad del *tweet.* Es de tipo entero y contiene un cero para los casos negativos y un cuatro para los casos positivos.
- **IDs:** contiene un número único que identifica a cada *tweet.*
- **Date:** contiene la fecha de cada *tweet.*
- **User:** contiene el id del usuario que publicó el *tweet*.
- **Text:** contiene el *tweet* propiamente dicho.

A continuación, se muestra en la Figura 6 el resultado de mostrar en pantalla los primeros cinco registros del dataset. Por motivos de privacidad, se ocultan los ids de usuario en la Figura.

	target	ids	date	user	text
0	4	2000548978	Mon Jun 01 22:22:06 PDT 2009		I have a follower!....thanks VBrown I though...
1	4	1956514018	Thu May 28 22:02:36 PDT 2009		haha // ma birthday had dhat picture...
2	4	1685520973	Sun May 03 00:45:58 PDT 2009		Wish I was in Vegas... Have f...
3	0	2190910903	Tue Jun 16 04:03:22 PDT 2009		I want to go vacation :] so,i wanna play bon ...
4	0	1793283377	Thu May 14 02:05:22 PDT 2009		my computers down for the count i think

Figura 6. Resultado del comando *head* aplicado sobre la estructura que contiene los datos.

Adicionalmente, dentro del análisis exploratorio, se desea conocer la naturaleza de cada una de las columnas, como así también detectar tempranamente si hubiera datos faltantes para alguno de los casos.

Existen distintas maneras de abordar este problema mediante las funciones que provee *pandas*, por lo que a continuación se muestra, en la Figura 7, lo resultante de utilizar el comando info, que resume características importantes del conjunto de datos: cantidad de registros no vacíos por columna y tipo de dato asociado. En este caso, se puede observar que las cinco columnas cuentan con datos no nulos en todos sus registros. Además, se observan los tipos de datos: dos columnas de tipo número entero con representación en 64 bits resumido como *int64* (target y ids) y las tres restantes (date, user y text) de tipo object, las cuales contienen texto y/o números.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600000 entries, 0 to 1599999
Data columns (total 5 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   target  1600000 non-null  int64
 1   ids     1600000 non-null  int64
 2   date    1600000 non-null  object
 3   user    1600000 non-null  object
 4   text    1600000 non-null  object
dtypes: int64(2), object(3)
memory usage: 61.0+ MB
```

Figura 7. Resultado del comando *info* aplicado sobre la estructura que contiene los datos.

Por último, se generan nubes de palabras para ambas polaridades, las cuales contienen en cada caso las dos mil palabras más frecuentes. De la misma frecuencia se desprende el tamaño asignado a cada uno de los términos. Cabe destacar que para la obtención de estas nubes de palabras se llevó a cabo un proceso de limpieza y normalización previo (eliminación de *stopwords* y aplicación de un *Snowball Stemmer)*, que se explicará a detalle en la sección 9.2 del presente informe. Dichas nubes de palabras se muestran en las Figuras 8 y 9, a continuación.

Figura 8. Nube de palabras generada con *tweets* positivos.

Figura 9. Nube de palabras generada con *tweets* negativos.

9.1.3 Balance de clases

Al momento de modelar una solución para un problema de clasificación puede suceder que el conjunto de datos de entrenamiento cuente con diferencias significativas en cuanto a la cantidad de muestras que hay por cada clase a analizar. Este desbalance puede derivar en resultados adversos y es por ello por lo que cobra relevancia la ausencia de este durante la etapa exploratoria o de preprocesamiento.

Cuando la búsqueda de las reglas de decisión que permiten determinar una clasificación (o polaridad, en este caso) se lleva a cabo en función de un conjunto de entrenamiento cuyas clases se encuentran significativamente desbalanceadas, la regla de decisión produce fronteras de decisión con un sesgo en favor de la clase mayoritaria (es decir, la clase de la cual se tiene un mayor número de muestras). De esta manera, los objetos de la clase minoritaria, que en ciertos casos puntuales de estudio puede ser la más relevante, tienden a ser erróneamente clasificados (Arrieta & Mera, 2015).

Teniendo en consideración que el presente trabajo se enfoca en una clasificación binaria y, por la naturaleza del conjunto de datos, no se observa un motivo aparente para la existencia de una clase mayoritaria. Sin embargo, se valida la certeza de esta hipótesis a partir de la observación de los datos y su clasificación, como se muestra en la Figura 10.

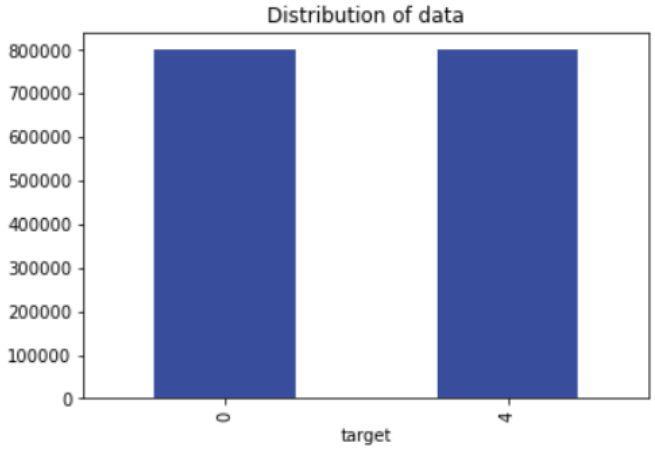


Figura 10. Distribución de las muestras según su polaridad.

9.2 Preprocesamiento de los datos

El preprocesamiento de los datos y, en este caso en particular de los textos, es un paso importante para las tareas de Procesamiento del Lenguaje Natural. A partir de este paso, se transforma el texto de forma tal que tenga las características que el algoritmo requiere para su mejor performance.

Los *tweets* contienen una gran cantidad de información aparte del texto propiamente dicho, como pueden ser URLs o símbolos, los cuales no aportan valor adicional al momento de la búsqueda de la polaridad y podrían alterar los resultados del modelo. Para lidiar con ello, se lleva a cabo una limpieza del conjunto de datos y eliminación del ruido. Luego, se procede con la Tokenización, proceso mediante el cual se divide el texto en palabras a las cuales se les asocia un índice (cada palabra es un *token*). Por último, se analizan los procesos clásicos de normalización y su influencia en los resultados.

9.2.1 Limpieza de datos

Uno de los primeros pasos indispensables al momento de modelar la solución a un problema de Procesamiento del Lenguaje Natural es la limpieza del conjunto de datos. Si bien existen ciertos pasos estandarizados para llevar a cabo este proceso, también existen otros que requieren un mayor dominio de la naturaleza de los datos en cuestión.

En este caso, al tratarse de un dataset clasificado y generado con el objetivo de alimentar modelos de Machine Learning, se deben tener en consideración las diferentes estrategias ya aplicadas por los creadores del mismo, detalladas en (Go et al., 2009) y enunciadas a continuación.

- No se tienen en consideración aquellos tweets que contienen emoticones (secuencia de caracteres que representa una emoción) positivos y negativos al mismo tiempo como, por ejemplo: “Target orientation :(But it is my birthday today :)”.
- Se eliminaron los retweets. Un retweet es una publicación generada a partir de una publicación (tweet) de otro usuario. De esta manera, se evita darle un peso adicional a una publicación en particular de los datos de entrenamiento.
- Se eliminaron los tweets repetidos. Ocasionalmente, la API de Twitter retorna publicaciones duplicadas que pueden influir negativamente en el análisis.

Teniendo en consideración lo desarrollado anteriormente, se pone en práctica una limpieza de los datos acompañada por la eliminación del posible ruido tal como se expresa debajo.

- **Se transforman todas las letras a minúsculas.** Esto permite que no se generen dos o más tokens distintos para una misma palabra en función de las mayúsculas o minúsculas que contiene.
- **Se reemplazan las URLs.** Todos los links que comiencen con ‘http’, ‘https’ o ‘www’ son reemplazados por la palabra ‘<URL>’.
- **Se reemplazan los nombres de usuario.** En Twitter, es posible mencionar a otro usuario en una publicación, anteponiendo su nombre de usuario con una arroba. Todas las palabras que inician con una arroba son reemplazadas con la palabra ‘<user>’.
- **Se reemplazan las letras consecutivas.** En los casos en donde una palabra contiene más de dos letras iguales consecutivas, las mismas se reducen a dos para eliminar el ruido. Por ejemplo, la palabra ‘Hellooooo’ se reemplaza por ‘Helloo’. Cabe destacar que esta solución es una simplificación del abordaje de estos casos y se desarrollará al respecto con mayor profundidad en 12.2.
- **Se reemplazan las contracciones.** Utilizando un nuevo dataset disponible en Kaggle (Adnane, 2020) que contiene un listado de contracciones en inglés junto con sus significados, se reemplazan las mismas. Así, por ejemplo, la contracción “doesn’t” es reemplazada por “does not”.
- **Se eliminan los caracteres no alfabéticos.** Todos los caracteres que no representan letras o dígitos son reemplazados con un espacio en blanco. Esto incluye signos de puntuación, los cuales no son útiles para modelar la solución.

Es relevante destacar que el orden en el que se ejecutan estos pasos no es casual, sino que de gran importancia. Por ejemplo, si se removieran los signos de puntuación antes de reemplazar las URLs, el segundo paso no surtiría ningún tipo de efecto. Lo mismo sucede, como otro ejemplo, con la eliminación de letras repetidas como paso anterior al reemplazo de contracciones: podría suceder que en el texto exista la palabra 'gonnnna' la cual, según la secuencia, primero se reemplaza por la palabra 'gonna' y, más adelante, por la conjunción 'going to'.

Uno de los pasos típicos de la limpieza de un conjunto de datos de texto consiste en la eliminación de las llamadas S*topword*s. Estas, llamadas en español como 'palabras vacías', son aquellas que no tienen un significado como artículos, pronombres, preposiciones, entre otras. Sin embargo, al momento de alimentar un modelo LSTM, el cual se espera que capture significados semánticos y dependencias contextuales entre las palabras, es importante no remover este tipo de palabras (Haddi et al., 2013).

Por otra parte, existen dos procesos que también suelen incluirse como parte del procesamiento de textos. El primero, llamado *Lemmatization*, tiene como intención la estandarización de las palabras: se busca, de manera forzada, el reemplazo convencional de todas las palabras asociadas a una original (lema) por esta misma. Dicho de otra manera, si una palabra se encontrara en el texto en diferentes conjugaciones, numero o género, todas estas repeticiones se reemplazan por un mismo lema. Por ejemplo, 'decir' es el lema de las palabras 'diré' y 'dijéramos'.

En segundo lugar, el proceso llamado *Stemming,* consiste en la eliminación de los sufijos de las palabras basado en reglas gramaticales, conservando únicamente su raíz. De forma similar al proceso anterior, se tiene como objetivo la estandarización del conjunto de datos. A diferencia de la Lematización, este proceso puede generar palabras inexistentes en el diccionario.

Al momento de utilizar métodos de 'Bolsas de Palabras', estos últimos dos procesos mencionados pueden cumplir un papel importante, ya que se fuerzan las palabras a su forma básica y, por ende, se logra reducir el número de palabras únicas. Esta reducción puede generar mejoras en la performance (Krouska et al., 2016). Por el contrario, el Análisis de Sentimiento basado en una estructura LSTM sigue un enfoque diferente que, tal como se explicó anteriormente, tiene en consideración el contexto de cada una de las palabras. En el presente Trabajo se contempla la utilización de W*ord Embeddings*, técnica

mediante la cual el modelo tiene la capacidad de identificar aquellas palabras con un significado similar, por lo que pierde sentido la aplicación de la Lematización.

9.2.2 Tokenización y Padding

La tarea de Tokenización (del inglés, *tokenization*) es otro de los pasos fundamentales, que consiste en la separación de una pieza de texto en unidades de menor tamaño llamados *tokens*. Estos *tokens* facilitan la tarea de comprensión del contexto y la interpretación del significado de cada oración, favoreciendo el desarrollo del modelo para el Procesamiento Natural de Texto: de esta manera, cada palabra podrá ser convertida en un formato numérico, formato idóneo para alimentar la Red Neuronal.

Existen diversas formas de *tokenizar* el conjunto de datos. Para la confección del presente Trabajo se aplica una Tokenización de espacios en blanco, utilizando el método Tokenizer de la librería Keras. Esta división es la que menor tiempo conlleva, ya que simplemente separa las palabras de forma natural, por los espacios en blanco que existen en la oración. Por ejemplo, la oración "I want to break free" se transforma en 'I', 'want', 'to', 'break', 'free'. Luego, se le asigna un índice a cada una de las palabras del conjunto de datos, de forma arbitraria. Considerando la misma oración previamente mencionada a modo de ejemplo, se podrían generar los siguientes índices:

'i': 23

'want': 42

'to': 2

'break': 45

'free': 3

Esto, resultaría en un vector *tokenizado* que se vería de la siguiente manera:

[23, 42, 2, 45, 3]

Es relevante destacar que la validez de las técnicas de Tokenización está estrictamente relacionada con el lenguaje del conjunto de datos a utilizar. Se profundizará más sobre la influencia del lenguaje utilizado en el apartado 12.

Para lograr tiempos de ejecución óptimos, es deseable realizar el proceso de entrenamiento en lotes. De esta manera, el modelo es capaz de procesar más de un elemento (oración o *tweet*) al mismo tiempo, reduciendo en gran medida la demanda temporal. Sin embargo, para la ejecución en lotes, la mayoría de las librerías de Aprendizaje Automático (y en particular, Keras), requiere que todos los elementos dentro de un lote cuenten con la misma dimensionalidad.

El proceso de *Padding* consiste justamente en la normalización de la longitud de cada uno de los vectores de tokens, generados en el paso anterior. Para ello, se define como longitud máxima la del vector que contenga la mayor cantidad de palabras (tokens) en el set de datos. Luego, se añade a cada uno de los vectores tantos tokens nulos (ceros, que no forman parte del diccionario original) como sean necesarios para completar la longitud máxima predefinida. Por ejemplo, si la frase más larga (en cantidad de palabras) del conjunto de datos cuenta con 10 palabras, se obtendrán vectores de longitud 10 en todos los casos. Siguiendo el ejemplo anterior, luego del proceso de *Padding* se obtiene el siguiente resultado:

[0, 0, 0, 0, 0, 23, 42, 2, 45, 3]

9.3 Modelos a Comparar

El proceso de entrenamiento implica inicializar los pesos sinápticos de forma aleatoria para cada una de las dimensiones. De allí, se trata de predecir la salida correspondiente a los valores de entrada usando los valores aleatorios iniciales. Al principio, el error será significativo, pero a través de la comparación de la predicción inicial frente a los resultados correctos, el modelo ajustará los pesos de forma tal que se minimice el error en la predicción.

Teniendo en consideración la magnitud del presente set de datos, se desarrolla la fase de entrenamiento con un subconjunto del set de datos original, de forma tal que sea más ágil la generación y comparación de diversos modelos utilizando diferentes parámetros. Como se detalla en (Prusa et al., 2016), utilizar una muestra de mayor tamaño no implica necesariamente mejoras significativas en cuanto a las métricas resultantes y el modelo óptimo generado con un subconjunto de datos es extrapolable al conjunto de mayor envergadura.

Si bien es posible obtener buenos resultados en la capa de Word Embeddings (principalmente, si se utiliza un conjunto de datos de gran volumen), la etapa de entrenamiento de la misma suele ser demandante en cuanto a tiempos de ejecución. Por este motivo, se considera la aplicación de lo que se conoce como Transfer Learning. Este concepto se centra en la idea de que es posible almacenar conocimiento adquirido durante la resolución de un problema y reutilizarlo al momento de resolver uno nuevo. Específicamente, en lugar de inicializar los pesos de forma aleatoria, se establece una capa de Word Embeddings preentrenados como pesos de inicialización, lo cual ayuda a acelerar el entrenamiento y puede mejorar el rendimiento de los modelos de PNL (Kim, 2014).

En el presente trabajo se comparan tres modelos según lo explicado previamente con distintas variantes asociadas al preentrenamiento de la capa de Word Embeddings. Estos

modelos se generan durante la etapa de entrenamiento, posterior al preprocesamiento y la división del conjunto de datos, que se detallan a continuación.

- **Modelo de Pesos Aleatorios**: modelo que contiene una capa de Embeddings no inicializada (es decir, cuenta con pesos aleatorios al comienzo).
- **Modelo Word2Vec**: modelo con una capa de Embeddings entrenada mediante el algoritmo Word2Vec contenido en la librería Gensim, sobre el conjunto de datos de entrenamiento.
- **Modelo GloVe**: modelo basado en Transfer Learning con una capa de Word Embeddings pre-entrenada mediante GloVe Embeddings. Teniendo en consideración que se obtendrán mejores resultados cuanto más se ajuste la capa pre-entrenada a la naturaleza del conjunto de datos a utilizar, se emplea un dataset de vectores GloVe obtenidos en base a dos mil millones de *tweets,* disponible públicamente en el sitio de la Universidad de Stanford (Stanford University, 2014).

9.3.1 Word2Vec

Word2Vec es un conjunto de técnicas publicadas por un grupo de investigadores de Google, basadas en un modelo de Redes Neuronales que tiene la capacidad de aprender asociaciones entre palabras en base a un corpus (Google, 2013). Propone dos métodos distintos para el aprendizaje de representación:

- CBOW (Bolsa de Palabras Continua), que intenta predecir una palabra según un contexto dado, donde ese contexto no es más que un conjunto de palabras.
- Skip-gram: en sentido inverso, el modelo intenta predecir el contexto en función de una palabra dada.

La librería de Python Gensim, de característica open-source, permite la implementación de una capa de Embeddings basada en Word2Vec, utilizando un conjunto de datos de forma no supervisada. Cuenta con tres parámetros de entrenamiento, que se definen a continuación.

- Size: cantidad de dimensiones asignadas a cada palabra. Un mayor tamaño requiere de mayor entrenamiento, pero puede derivar en un mejor modelo.

- Workers: parámetro que permite la ejecución en paralelo, para minimizar el tiempo de entrenamiento.
- Min_count: mínima cantidad de ocurrencias que debe tener una palabra para no ser descartada del diccionario final. Palabras poco frecuentes pueden adicionar ruido y no aportar valor al entrenamiento.

9.3.2 GloVe

El modelo GloVe (del inglés *Global Vectors*, o Vectores Globales) es una extensión del método Word2Vec desarrollado por la Universidad de Stanford. Mediante este algoritmo se mapean las palabras en un espacio vectorial, donde la distancia entre las palabras está definida por su similitud semántica. El entrenamiento del mismo se lleva a cabo mediante la contabilización de co-ocurrencias entre palabras, en lugar de utilizar una ventana de palabras para definir el contexto (Pennington et al., 2014).

9.3.3 División del conjunto de datos: entrenamiento y prueba

Esta etapa es el nexo entre la preparación de los datos y el entrenamiento del modelo. Durante la misma, se divide el conjunto de datos en dos subconjuntos: el conjunto de entrenamiento y el conjunto de prueba. El conjunto de entrenamiento se corresponde con los datos que se utilizan para construir y entrenar el modelo de clasificación. Por el contrario, el conjunto de prueba comprende datos que el modelo desconoce y se utiliza para comprender la precisión y, por ende, evaluar el clasificador resultante.

Existen diversas maneras de realizar la división del conjunto de datos. A los efectos del presente Trabajo, se realiza una división aleatoria en proporciones 80-20% para obtener los conjuntos de entrenamiento y prueba, respectivamente.

Luego de este proceso, será posible alimentar la capa de W*ord Embedding* con los vectores resultantes pertenecientes al conjunto de entrenamiento, de forma tal que cada índice sea remplazado por su representación vectorial.

9.3.4 Parametrización

La parametrización de los tres modelos generados se lleva a cabo de la misma manera, de forma tal que los resultados sean comparables.

En cuanto a la función de activación, se utiliza una función no lineal sigmoide. Esta función transforma los valores introducidos a una escala (0,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a 0. Si bien no existe una regla general para la función a utilizar, se recomienda en (Goodfellow et al., 2016) la utilización de una única función de activación sigmoide para los problemas de clasificación binaria, como lo es el postulado en este Trabajo.

Luego, al momento de compilar el modelo, se define la función de pérdida, el optimizador a utilizar y los Callbacks.

Al modelar un problema de clasificación binaria, en donde se tienen dos clases, se desea predecir la probabilidad de que cada muestra pertenezca a una de las dos. En estos casos, se sugiere la utilización de la función de pérdida llamada *binary cross-entropy* o entropía cruzada binaria. Esta misma, utilizada en la práctica del presente Trabajo, compara cada una de las probabilidades predichas con la clase real, que puede ser 0 o 1. De ahí, se calcula la diferencia entre lo predicho y la realidad, de forma tal que se minimice el error durante el entrenamiento.

Además, se utiliza un optimizador Adam (del inglés Adaptive Moment Estimation) el cual calcula las tasas de aprendizaje para cada parámetro del modelo. Este algoritmo es uno de los más popular gracias a su potencial y cuenta con las ventajas de ser fácil de implementar, eficiente y poco demandante en cuanto a los requisitos de memoria.

Por último, se utilizan dos C*allbacks* que ofrece la librería Keras descriptos a continuación.

- EarlyStopping: detiene el entrenamiento cuando la métrica monitoreada (en este caso, la función de pérdida) deja de mejorar.
- ReduceLRonPlateau: reduce la tasa de aprendizaje cuando la métrica monitoreada deja de mejorar.

En la Figura 11 se muestra el resultado de ejecutar la función *summary()* sobre el modelo construido, donde se detallan las distintas capas del mismo. Cada una de las capas cuenta con un formato de salida, el cual se muestra en la columna *Output Shape*.

La salida de cada capa se convierte en la entrada de la capa siguiente. Luego, la columna de *param #* indica la cantidad de parámetros (o pesos sinápticos) que el modelo aprende durante el entrenamiento, también conocidos como parámetros entrenables.

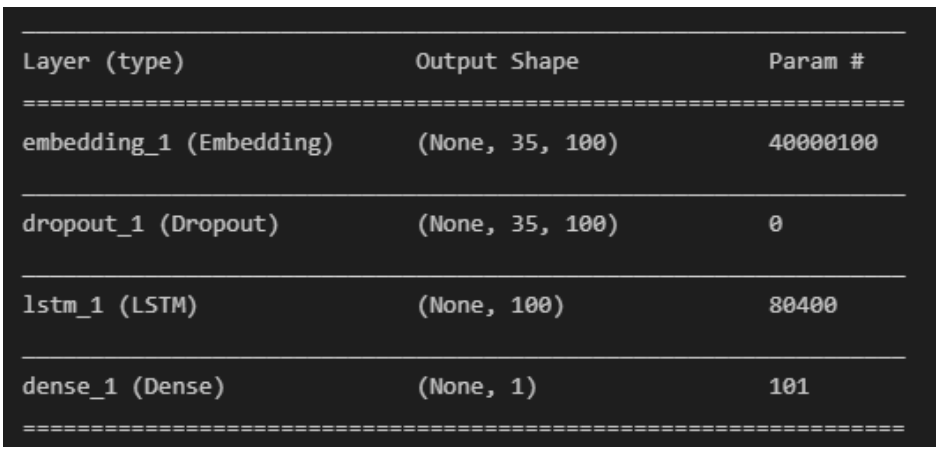
El primer valor de esta columna (*None* en todos los casos) hace referencia a la dimensionalidad de los lotes, descriptos en 9.2.2. A la hora de darle forma al modelo, es posible omitir la definición de esta dimensión, para luego asignarla al momento de entrenamiento y resulta indistinto hacerlo de un modo u otro.

En primer lugar, se tiene una capa de Embeddings que cuenta con más de 40 millones de parámetros y una dimensionalidad de *None (*tamaño de lotes o *batch size*), 35 (longitud máxima en cantidad de palabras de los *tweets* utilizados, en función de lo explicado en la sección de *Tokenización y Padding*) y 100 (dimensión del vector que representa a cada una de las palabras).

Luego, se observa la capa de *Dropout*, la cual no cuenta con parámetros entrenables sino que simplemente selecciona aleatoriamente algunas neuronas y las inactiva durante una etapa del entrenamiento.

Después de la capa de *Dropout*, se encuentra la capa de LSTM, la cual cuenta con 100 neuronas de salida y 80400 parámetros entrenables.

Por último, la estructura cuanta con una Capa Densa o *Dense Layer*, la cual es una capa que está totalmente conectada con la capa que la precede. Esta capa tiene una salida de dimensión 1 y, de esta manera, el modelo puede predecir un valor entre 0 y 1: un valor cercano al 0 indica que se trata de un *tweet* negativo y un valor cercano a 1 indica que se trata de un *tweet* positivo.



Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 35, 100)	40000100
dropout_1 (Dropout)	(None, 35, 100)	0
lstm_1 (LSTM)	(None, 100)	80400
dense_1 (Dense)	(None, 1)	101

Figura 11. Resumen de la estructura de la Red Neuronal Recurrente.

9.4 Evaluación

La evaluación de los distintos modelos se lleva a cabo sobre el conjunto de *testing.* Para todos los casos, se sumariza un total de 16.000 muestras, que representan un 20% del subconjunto de datos utilizado (80.000 *tweets*).

Para llevar a cabo la evaluación de los diferentes modelos se tienen en cuenta en primer lugar diferentes métricas, tanto al final del entrenamiento como durante el mismo, las cuales se describen a continuación. De forma tal que la lectura de las métricas no resulte confusa, se selecciona como clase positiva aquella que contiene los tweets clasificados como positivos.

- Accuracy: representa la cantidad de predicciones correctas del conjunto total de predicciones.
- Precision: indica la probabilidad que tiene el modelo de acertar una clasificación 'positiva'. En otras palabras, permite indicar la proporción de *tweets* identificados como positivos que efectivamente son positivos.
- Specificity: indica la proporción de casos verdaderamente negativos que se predicen como negativos.
- Recall: indica la proporción de los casos verdaderamente positivos que fueron identificados como tales.
- F1 Score: se define como un valor ponderado entre *Precision* y *Recall*. Se incluye en el análisis si bien suele ser de mayor utilidad en aquellos casos en los que existe una distribución desigual de clases.

A continuación, se muestran en la Tabla 1 las diferentes métricas asociadas a los tres modelos planteados.

Tabla 1. Métricas obtenidas para los diferentes modelos planteados.

	Modelo de Pesos Aleatorios	Modelo GloVe	Modelo Word2Vec
Accuracy	0.736	0.728	0.792
Precision	0.789	0.727	0.799
Specificity	0.828	0.724	0.803
Recall	0.644	0.732	0.782
F1 Score	0.709	0.729	0.791

Se puede observar en las métricas que el Modelo Word2Vec es superador a los otros dos en cuanto *a accuracy, precision, recall* y *F1 Score*. En cuanto a *specificity*, el Modelo de Pesos Aleatorios es el que obtuvo un mejor resultado, aunque cuenta con el menor valor de *precision:* el 82.8% de los *tweets* negativos fueron identificados como tales, pero, por el contrario, solo el 64.4% de los *tweets* positivos fueron clasificados como positivos.

Adicionalmente, en la Tabla 2 se analiza el Área Bajo la Curva ROC o AUC (*Area Under Curve*), estadístico mediante el cual se pueden comparar los modelos mediante la comparación de la sensibilidad (*recall*) y la especificidad (*specificity*). Se determina que un modelo performa a niveles de azar cuando AUC ronda el valor de 0.5 y que tiene un rendimiento satisfactorio a partir de los 0.75. Si bien en todos los casos se concluye un rendimiento aceptable, se destaca el Modelo Word2Vec por sobre los otros dos.

Tabla 2. Comparación de estadístico AUC para los distintos modelos.

	Modelo de Pesos Aleatorios	**Modelo GloVe**	**Modelo Word2Vec**
AUC	0.804	0.799	0.877

Además del análisis de las métricas tradicionales, se incluye en el presente informe el análisis de la cantidad de épocas que requirió cada modelo para entrenarse (limitadas mediante el uso de C*allbacks*) como también el tiempo transcurrido en la etapa de entrenamiento. Este tiempo es ciertamente comparable puesto que la ejecución del código se llevó a cabo sobre la misma máquina siendo el único proceso activo en todos los casos.

En la Tabla 3 se puede observar que la cantidad de épocas requeridas se correlaciona, tal como se puede intuir, con los tiempos requeridos para el entrenamiento. A simple vista se puede denotar que los Modelos que cuentan con pesos sinápticos predeterminados para la capa de Embeddings concluyen con anterioridad el entrenamiento. Además, el Modelo Word2Vec logra reducir a más de la mitad el tiempo de entrenamiento, obteniendo métricas superadoras.

Tabla 3. Épocas requeridas y tiempo de entrenamiento de cada modelo.

	Modelo de Pesos Aleatorios	**Modelo GloVe**	**Modelo Word2Vec**
Epochs	57	38	33
Tiempo de Entrenamiento	1h 7min 7s	39min 36s	29min 3s

Mediante las curvas de accuracy y loss es posible tener una noción del aprendizaje que logra progresivamente cada uno de los modelos. En la Figura 12 se muestran dichas curvas, donde se pueden observar las métricas para el conjunto de entrenamiento y para el de validación, que representa un subconjunto aleatorio del 10% de los datos provistos para el entrenamiento y se actualiza en cada época. Es importante analizar el crecimiento de las curvas de validación, ya que nos dan una noción de si el modelo está sufriendo un sobreajuste.

En dichas curvas, se puede apreciar en todos los casos la forma en la que se incrementa gradualmente el valor para la métrica de *accuracy* mientras que disminuye de forma inversamente proporcional el valor para la función de pérdida. Las fluctuaciones apreciadas en las curvas asociadas a los datos de validación pueden explicarse teniendo en consideración la aleatoriedad con la que se construye este subconjunto cada vez que comienza una nueva época.

Idealmente, se espera que las curvas del modelo tengan una trayectoria similar, lo que indica una buena generalización del modelo. Sin embargo, esto no sucede para el caso del Modelo Word2Vec, por lo que más adelante se describirá en 9.5 el proceso de regularización que busca solventar este problema.

Como se mencionó previamente, el proceso de aprendizaje se detiene mediante los Callbacks. En esos casos, observamos que la etapa de entrenamiento se detiene una vez que la función de pérdida alcanzó su valor mínimo para el conjunto de validación.

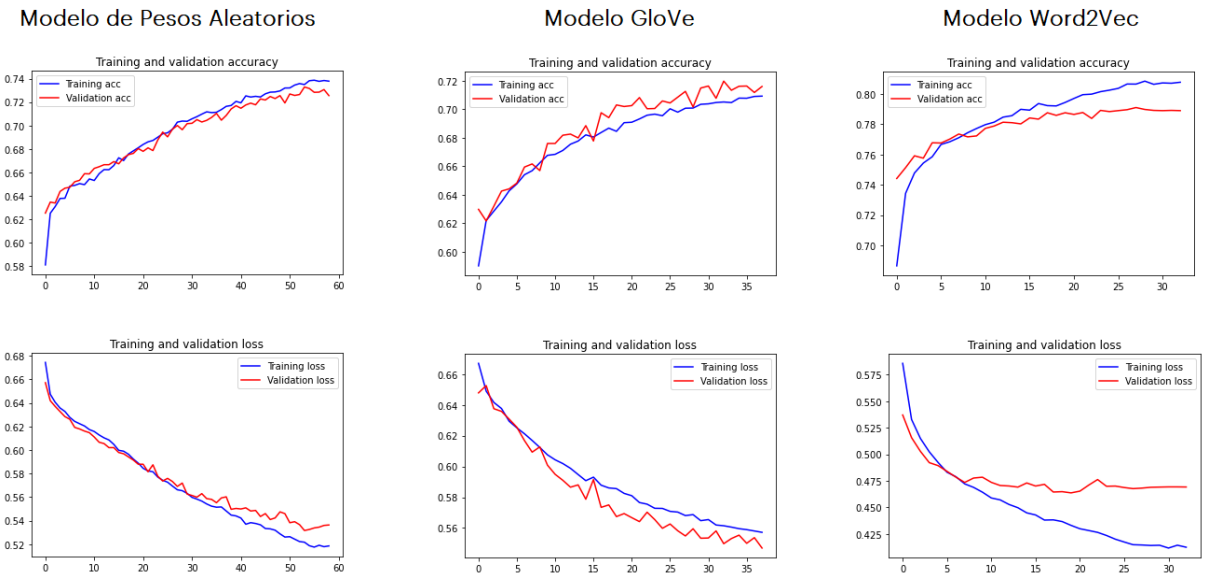


Figura 12. Curvas de accuracy (arriba) y loss (abajo) para el conjunto de training (azul) y de validación (rojo) durante las epochs que conforman el entrenamiento de cada modelo.

Por otra parte, se tiene en cuenta la matriz de confusión, relevante a la hora de comprender la capacidad de clasificación que tiene cada modelo. Esta matriz permite observar gráficamente la cantidad de muestras predichas para cada clase y la magnitud de muestras que fueron bien clasificadas. En la Figura 13 se muestra una matriz de confusión por cada uno de los modelos presentados.

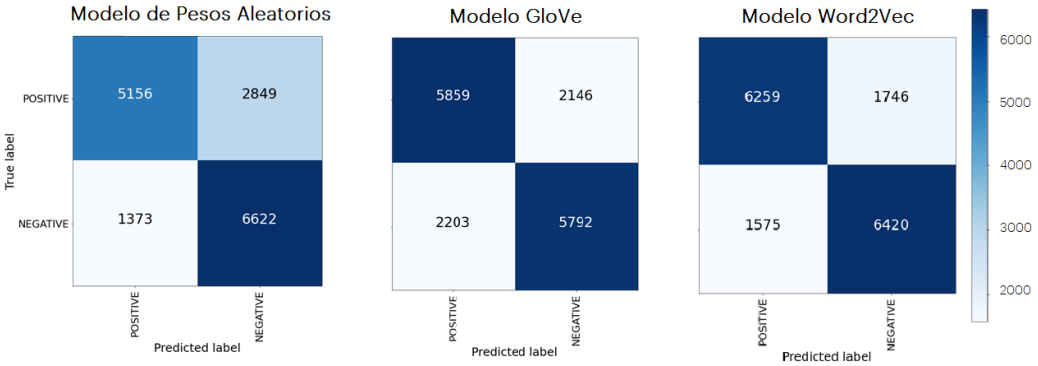


Figura 13. Matrices de confusión para los distintos modelos.

Por último, para comprender la efectividad de los tres modelos, se predice mediante los mismos la polaridad de dos *tweets* que forman parte del conjunto de datos de validación. Según el modelo propuesto, un valor predicho cercano a cero demarca una clasificación negativa y un valor cercano a 1, una positiva.

Los textos a utilizar se enuncian a continuación:

- Negativo: “I hate when I have to call and wake people up”. En español, “odio cuando tengo que llamar y despertar a la gente”.
- Positivo: “I love my saturday mornings!!”. En español, “Amo mis mañanas de sábado”.

A continuación, se muestra en la Tabla 4 los resultados obtenidos de aplicar el modelo sobre los elementos mencionados, para comprender con profundidad el desempeño obtenido. En todos los casos, los diferentes modelos infieren la polaridad de las muestras, si bien se notan mejores resultados para el modelo entrenado en base a Word2Vec.

Tabla 4. Clasificación de los distintos modelos sobre tres nuevos elementos.

	Modelo de Pesos Aleatorios	Modelo GloVe	Modelo Word2Vec
Negativo	0.012	0.014	0.003
Positivo	0.895	0.889	0.986

9.5 Regularización

Si se toma en consideración al Modelo Word2Vec como el de mejor rendimiento entre los tres presentados, resulta de importancia detenerse en observar sus curvas de aprendizaje (Figura 12).

Es posible apreciar un comportamiento en las curvas de pérdida donde el conjunto de entrenamiento continúa la minimización de la función de pérdida mientras que el conjunto de validación toma un comportamiento asintótico. Esta diferencia indica que el modelo está teniendo un buen rendimiento para el conjunto de entrenamiento, pero no tan buen rendimiento para el conjunto de validación.

Cuando lo mencionado anteriormente sucede, es posible afirmar que el modelo no está generalizando lo suficiente para tener el mismo rendimiento sobre cualquier conjunto de datos. Para atacar el problema de generalización, existe un proceso llamado regularización.

La regularización busca que el modelo resultante sea lo menos sensible posible a los valores atípicos que se encuentran en el conjunto de datos de entrenamiento. De esta manera, el modelo resultante generalizaría mejor y se obtendría un rendimiento superior sobre cualquier conjunto de datos desconocido.

Una de las técnicas de regularización que se puede aplicar a este problema es la incorporación de una nueva capa de Dropout. En la Figura 14 se muestra la comparación entre las curvas de accuracy y loss para el Modelo Word2Vec original y un Modelo Word2Vec con una capa adicional de Dropout (llamado de aquí en adelante Modelo Word2Vec II).

Se puede observar con claridad que, si bien las curvas de entrenamiento son similares, se denota una mejora significativa en cuanto a la generalización. Además, gracias a ello, se obtienen métricas ligeramente superadoras, las cuales se detallan a continuación en la Tabla 5.

Tabla 5. Métricas obtenidas para los diferentes modelos Word2Vec.

	Modelo Word2Vec	Modelo Word2Vec II
Accuracy	0.792	0.803
Precision	0.799	0.812
Specificity	0.803	0.812
Recall	0.782	0.811
F1 Score	0.791	0.811

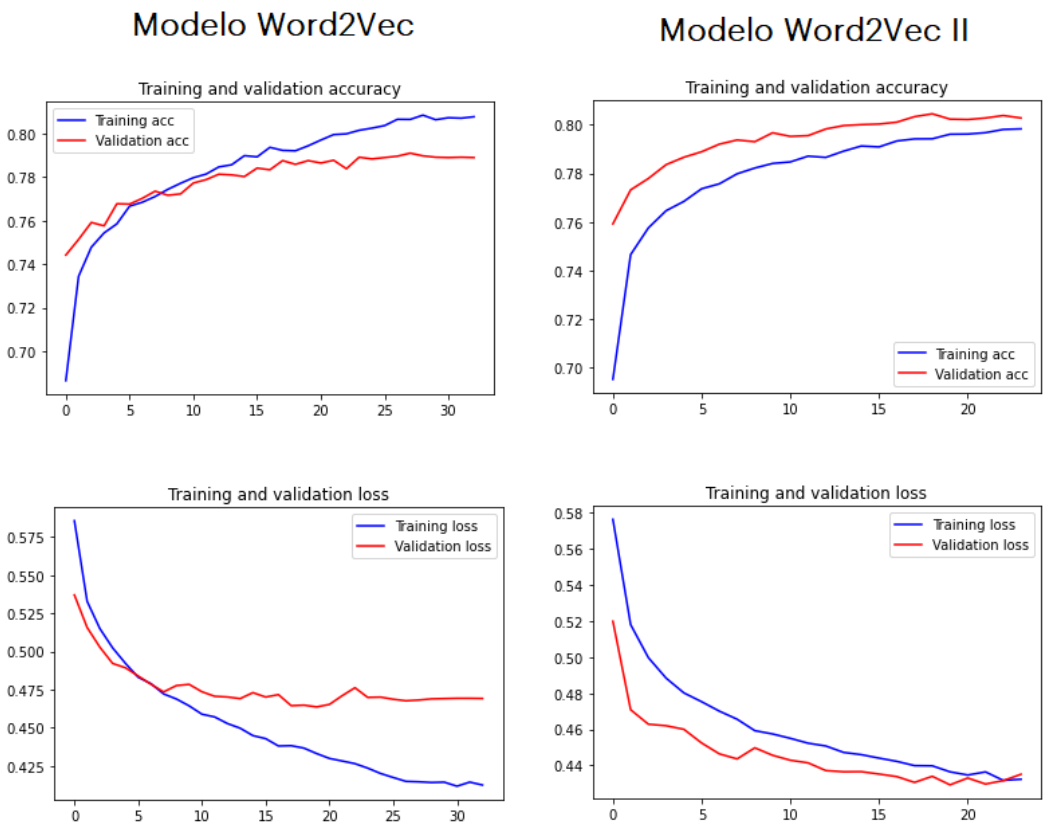


Figura 14. Curvas de accuracy (arriba) y loss (abajo) para el conjunto de training (azul) y de validación (rojo) para los distintos modelos basados en Word2Vec.

10. RESULTADOS

Tomando en consideración los resultados obtenidos, se demuestra que bajo los parámetros definidos en el presente Trabajo y ante modelos de similares características, se obtienen mejores resultados mediante la utilización de una capa de Word Embeddings, inicializada con pesos no aleatorios generados a partir de Word2Vec sobre el conjunto de datos de entrenamiento.

Para comparar ambos modelos generados mediante Word2Vec, se muestra la Curva ROC de ambos (Figuras 15 y 16), la cual permite representar gráficamente la sensibilidad frente a la especificidad del clasificador.

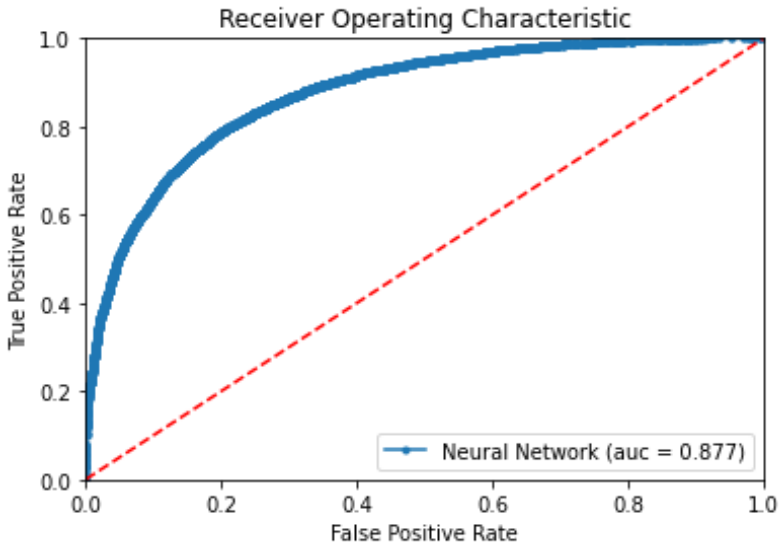


Figura 15. Curva ROC del Modelo Word2Vec junto con su AUC.

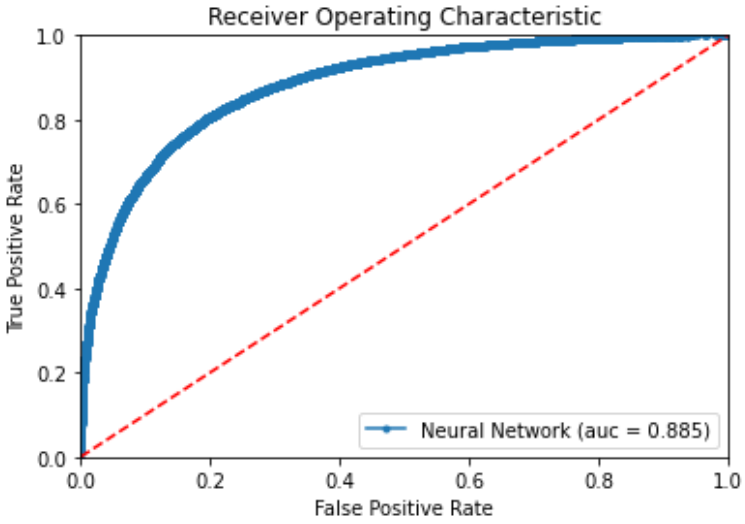


Figura 16. Curva ROC y AUC para el Modelo Word2Vec II.

Gracias a la regularización planteada, se obtuvo una mejora del 10% en cuanto al estadístico AUC, llevando el mismo desde 0.877 a 0.885. Esto nos permite inferir una mejora en las capacidades predictivas del modelo, acompañando el análisis con las métricas que se mostraron en la Tabla 5.

El Modelo Word2Vec II alcanza un AUC de 0.885, tal como se muestra en la Figura 16. Se entiende como altamente satisfactorio aquellos modelos que cuentan con un AUC cercano a 0.9 o superior, por lo que se puede concluir que, si bien no se profundizó sobre la optimización de los hiperparámetros, se obtuvo un modelo robusto con aceptables condiciones.

En cuanto a los tiempos de ejecución, se puede destacar que la utilización de pesos sinápticos no aleatorios en la capa de Word Embeddings permitió reducir el tiempo de entrenamiento en un 50%, en base a lo referenciado en la Tabla 3 de la sección previa.

Al mirar en detenimiento las matrices de confusión, se puede visualizar la presencia de la mayoría de las muestras en la diagonal de la matriz, lo cual indica que, pese a las notables diferencias, todos los modelos obtenidos representan al menos un punto de partida válido para su optimización.

Considerando las curvas de aprendizaje de los tres modelos iniciales se observa que, tal como se esperaba, el modelo que contiene los pesos inicializados a partir de una matriz de Embeddings generada con información propia del conjunto de datos parte de un *accuracy* alto, aún en su primera época. En la Tabla 6 se detalla la métrica de *accuracy* inicial sobre el conjunto de validación al finalizar la primera época de entrenamiento en los distintos modelos planteados.

Tabla 6. Accuracy sobre el conjunto de validación al final de la primera época.

	Modelo de Pesos Aleatorios	Modelo GloVe	Modelo Word2Vec
Val Accuracy en Epoch 1	0.6253	0.6397	0.744

11. CONCLUSIONES

Al momento de llevar a cabo un Análisis de Sentimiento, uno de los pasos más importantes es el de la definición de la herramienta a utilizar para llevar a cabo la clasificación. En el presente Trabajo se plantea la utilización de Redes Neuronales Recurrentes LSTM y su sinergia con las matrices de Word Embeddings para resolver la tarea.

En función de los resultados, se concluye que es posible determinar la polaridad de las publicaciones de textos cortos generadas por los usuarios en *Twitter* mediante el modelo planteado.

Además, se detecta una variación significativa en la performance del modelo en función de las modificaciones realizadas sobre la capa de Word Embeddings utilizada. Al brindarle al modelo una inicialización de los pesos sinápticos en dicha capa, se lograron resultados satisfactorios para la clasificación binaria de los *tweets.*

Cabe destacar que, utilizando un modelo análogo en todos los casos, se obtuvieron mejores métricas resultantes al aplicar el algoritmo Word2Vec sobre el conjunto de datos original, respecto a lo obtenido al inicializar el modelo con pesos sinápticos aleatorios en la capa de Word Embeddings o al utilizar un diccionario GloVe pre-entrenado.

Adicionalmente, queda demostrado que la optimización y regularización de los modelos planteados puede conllevar a aún mejores resultados, sin alterar el preprocesamiento que se realiza sobre el conjunto de datos del cual se parte.

Por otra parte, al basarse en el concepto de Transfer Learning y utilizar una matriz de Word Embeddings pre entrenada basada en GloVe y con datos ajenos al conjunto de datos a clasificar, no se obtuvieron resultados superadores en cuanto a las métricas en comparación de las obtenidas al inicializar la capa de Word Embeddings del mismo modelo con pesos aleatorios. Sin embargo, se logra una disminución del tiempo de entrenamiento superior al 40%, en ambos casos logrando un valor de *accuracy* cercano al 73%.

12. Futuros Trabajos

A través de la realización del presente Trabajo, fue posible encontrar diversos puntos sobre los cuales es posible profundizar.

12.1 Conjunto de datos orientado a COVID-19

La motivación del presente trabajo nace de la potencialidad de las técnicas de Análisis de Sentimiento de determinar la recepción de la población sobre una temática como lo es la pandemia por coronavirus. Como primer trabajo futuro relevante, se destaca la aplicación del modelo planteado sobre un conjunto de datos que contenga *tweets* relacionados con la pandemia y el estudio de la tendencia de sentimiento a lo largo de los días de confinamiento.

Una de las limitaciones para incluir este estudio en el presente Trabajo fue la imposibilidad de recopilar datos de la red social Twitter utilizando palabras claves (como podrían ser 'Covid', 'pandemia', 'confinamiento', entre otras). Además, aún si fuera posible la recopilación, resultaría necesaria la clasificación manual de una parte significativa del conjunto de datos para poder validar la performance del modelo.

12.2 Obtención de datos en tiempo real

Durante el transcurso de la realización del presente Trabajo, la red social *Twitter* publicó una nueva versión de su plataforma para consultar datos. De esta manera, a partir de enero de 2021 se puso a disposición de los investigadores un esquema gratuito para la consulta de datos en tiempo real e histórica (Perez, 2021).

Haciendo uso de este nuevo esquema, sería de interés profundizar sobre la posibilidad de llevar a cabo una clasificación en tiempo real de las publicaciones realizadas sobre una temática en particular y construir, de esta manera, diferentes visualizaciones que muestren la tendencia del sentimiento en los usuarios.

12.3 Introducir clase neutra

Si bien es posible definir un umbral en la clasificación binaria para reclasificar algunas de las muestras como muestras 'neutrales' o sin sentimiento aparente, resultaría de interés la introducción de una clase neutra, llevando el problema a uno de tipo multiclase.

12.4 Optimización de Hiperparámetros

Se determinó inicialmente que la optimización de los parámetros seleccionados a la hora de conformar el modelo LSTM no estaba dentro del alcance del presente trabajo. En efecto, se tuvo como objetivo la confección de un único modelo comparable y estandarizado. Sin embargo, es un proceso valioso aquel que permite determinar cuáles son los parámetros óptimos de forma tal que se obtenga el mejor modelo posible. Por ejemplo, métodos tales como la Optimización Bayesiana pueden ser utilizados para encontrar los mejores parámetros (en este caso podría ser, la dimensión de la capa de Embedding) con la menor cantidad de entrenamientos posibles.

12.5 Influencia del lenguaje en el Análisis de Sentimiento

A lo largo de la investigación y estudio del estado del arte, se destacó la necesidad de brindar especial atención al lenguaje sobre el cual se estaba trabajando. Esto está estrictamente relacionado con que no es sencillo generalizar un modelo que tenga la capacidad de interpretar todos los lenguajes en simultáneo.

La etapa de preprocesamiento es una de las que más relación guarda con el lenguaje a utilizar, ya que procesos tales como el de Stemming, Lemmatization o mismo la introducción de una capa de Word Embeddings pre-entrenada podría carecer de sentido si no se tuviera en cuenta el lenguaje sobre el cual se aplicará el modelo resultante.

Como ejemplo, al momento del preprocesamiento se llevó a cabo una Tokenización por espacios en blanco. Esto fue posible ya que el lenguaje tratado utiliza los espacios para dividir una oración en palabras, pero no hubiese sido de ayuda en otros lenguajes donde las oraciones se construyen de otra manera.

En (Volkova et al., 2013) se exploran las variaciones demográficas del lenguaje para la mejora de Análisis de Sentimiento multilingüe en redes sociales.

12.6 Repetición de letras en las palabras

Como se desarrolló en el presente informe, existen diversos elementos en la estructura lingüística que permiten identificar la tonalidad que el autor de un mensaje desea expresar. En múltiples ocasiones, se encuentra entre los datos propios de redes sociales la presencia de repetición de letras en las palabras. Esta repetición puede influir en la orientación que el emisor quiere brindarle a su mensaje.

Existen pocos trabajos que abordan esta problemática, por lo que la misma se sugiere como un trabajo futuro. Entre esos pocos, se destacan (Pak et al., 2018), que describe la influencia de la repetición en reseñas de sitios en línea y (Kalman & Gergle, 2014), que estudia la presencia de repetición de letras en correos electrónicos.

13. REFERENCIAS BIBLIOGRÁFICAS

Adalı, E. (2012). Doğal Dil İşleme. Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi*.

Adnane, O. (2020). *List of English contractions*. Kaggle. https://www.kaggle.com/ishivinal/contractions

Allen, J. F. (2003). Encyclopedia of computer science. In *Choice Reviews Online* (4th editio). Wiley. https://doi.org/10.5860/choice.38-3650

Arana, C. (2021). *Redes Neuronales Recurrentes: Análsis de los Modelos Especializados en Datos Secuenciales*.

Arrieta, J., & Mera, C. (2015). Estudio Comparativo de Técnicas de Balanceo de Datos en el Aprendizaje de Múltiples Instancias. *LACNEM2015 - Latin American Conference on Networking and Electronic Media*.

Balahur, A., Steinberger, R., Kabadjov, M., Zavarella, V., Van Der Goot, E., Halkia, M., Pouliquen, B., & Belyaeva, J. (2010). Sentiment analysis in the news. *Proceedings of the 7th International Conference on Language Resources and Evaluation, LREC 2010*.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*. https://doi.org/10.1162/153244303322533223

Bobicev, V., & Sokolova, M. (2017). Inter-annotator agreement in sentiment analysis: Machine learning perspective. *International Conference Recent Advances in Natural Language Processing, RANLP*. https://doi.org/10.26615/978-954-452-049-6-015

D´Ambra, A. (2020, July 2). *Psicología y cuarentena: "Es como si tuviéramos una radio de fondo, hay*

días en los que no molesta y otros en los que agobia." https://www.infobae.com/tendencias/2020/07/02/psicologia-y-cuarentena-es-como-si-tuvieramos-una-radio-de-fondo-hay-dias-en-los-que-no-molesta-y-otros-en-los-que-agobia/

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*. https://doi.org/10.1162/089976600300015015

Global Web Index. (2020). *Global Web Index's flagship report on the latest trends in social media*. https://www.globalwebindex.com

Go, A., Bhayani, R., & Huang, L. (2009). *Twitter sentiment classification using distant supervision*. CS224N Project Report, Stanford. http://help.sentiment140.com/for-students/

Goodfellow, I., Benigo, Y., & Courville Aaron. (2016). Deep Learning (Adaptive Computation and Machine Learning series). In *MIT Press*.

Google. (2013). *Word2vec: Tool for computing continuous distributed representations of words.* https://code.google.com/archive/p/word2vec/

Google. (2021). *Machine Learning Crash Course with TensorFlow APIs*. https://developers.google.com/machine-learning/crash-course/images/linear-relationships.svg

Gupta, D. (2020). *Fundamentals of Deep Learning – Activation Functions and When to Use Them?* https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

Haddi, E., Liu, X., & Shi, Y. (2013). The role of text pre-processing in sentiment analysis. *Procedia Computer Science*. https://doi.org/10.1016/j.procs.2013.05.005

Hammer, B. (2001). Neural Smithing – Supervised Learning in Feedforward Artificial Neural Networks. *Pattern Analysis & Applications*. https://doi.org/10.1007/s100440170029

Harris, Z. S. (1954). Distributional Structure. WORD. https://doi.org/10.1080/00437956.1954.11659520

Hartwig, F., & Dearing, B. E. (1979). *Exploratory Data Analysis (Sage University Paper Series on Qualitative Research Methods, Vol. 16)*. SAGE.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*. https://doi.org/10.1162/neco.1997.9.8.1735

Kaggle. (2021). *Kaggle*. kaggle.com

Kalman, Y. M., & Gergle, D. (2014). Letter repetitions in computer-mediated communication: A unique link between spoken and online language. *Computers in Human Behavior*. https://doi.org/10.1016/j.chb.2014.01.047

Kazanova. (2017). *Sentiment140 Kaggle*. 2017. https://www.kaggle.com/kazanova/sentiment140

Kemp, S. (2020). Digital 2020 - July global statshot report. In *Datareportal.com*. https://datareportal.com/reports/digital-2020-july-global-statshot

Keras. (2021). *Callbacks API*. https://keras.io/api/callbacks/

Krouska, A., Troussas, C., & Virvou, M. (2016). The effect of preprocessing techniques on Twitter sentiment analysis. *IISA 2016 - 7th International Conference on Information, Intelligence, Systems and Applications*. https://doi.org/10.1109/IISA.2016.7785373

Kumari Yeruva, V., Chandrashekar, M., Lee, Y., Rydberg-Cox, J., Blanton, V., & Oyler, N. A. (2020). Interpretation of Sentiment Analysis in Aeschylus's Greek Tragedy. *Proceedings of LaTeCH-CLfL*, 138–146. https://aclanthology.org/2020.latechclfl-1.17.pdf

Liddy, E. . (2001). Natural Language Processing. In Encyclopedia of Library and Information Science, 2 Ed. Marcel Decker, Inc. In *Encyclopedia of Library and Information Science, 2nd Ed.*

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*. https://doi.org/10.2200/S00416ED1V01Y201204HLT016

Lu, Y., Hu, X., Wang, F., Kumar, S., Liu, H., & Maciejewski, R. (2015). Visualizing social media sentiment in disaster scenarios. *WWW 2015 Companion - Proceedings of the 24th International Conference on World Wide Web*. https://doi.org/10.1145/2740908.2741720

Manning, C. D., Schütze, H., & Weikurn, G. (2002). Foundations of Statistical Natural Language Processing. *SIGMOD Record*. https://doi.org/10.1145/601858.601867

Mozetič, I., Grčar, M., & Smailović, J. (2016). Multilingual twitter sentiment classification: The role of human annotators. *PLoS ONE*. https://doi.org/10.1371/journal.pone.0155036

Ng, A. Y. (2017). *Why Human-level Performance?* Structuring Machine Learning Projects.

Nielsen, M. (2019). *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com/

OPSA. (2020). *La vida en cuarentena: sentimientos, salud y economía*. Universidad de Buenos Aires. https://www.psi.uba.ar/opsa/informes/La vida en cuarentena.pdf

Pak, I., Teh, P. L., & Cheah, Y. N. (2018). Hidden sentiment behind letter repetition in online reviews. *Journal of Telecommunication, Electronic and Computer Engineering*.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*. https://doi.org/10.1561/1500000011

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *30th International Conference on Machine Learning, ICML 2013*.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. https://doi.org/10.3115/v1/d14-1162

Perez, S. (2021). Twitter's new API platform now opened to academic researchers. *TechCrunch*. https://techcrunch.com/2021/01/26/twitters-new-api-platform-now-opened-to-academic-researchers/

Prusa, J., Khoshgoftaar, T. M., & Seliya, N. (2016). The effect of dataset size on training tweet sentiment classifiers. *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015*. https://doi.org/10.1109/ICMLA.2015.22

Python Software Foundation. (2021). *Python*. https://www.python.org/

Rasool, A., Tao, R., Marjan, K., & Naveed, T. (2019). Twitter Sentiment Analysis: A Case Study for Apparel Brands. *Journal of Physics: Conference Series*. https://doi.org/10.1088/1742-6596/1176/2/022015

Rudkowsky, E., Haselmayer, M., Wastian, M., Jenny, M., Emrich, Š., & Sedlmair, M. (2018). More than Bags of Words: Sentiment Analysis with Word Embeddings. *Communication Methods and Measures*. https://doi.org/10.1080/19312458.2018.1455817

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Stanford University. (2014). *GloVe*. https://nlp.stanford.edu/projects/glove/

Twitter. (2021). *Twitter API*. https://developer.twitter.com/en/docs/twitter-api

Volkova, S., Wilson, T., & Yarowsky, D. (2013). Exploring demographic language variations to improve multilingual sentiment analysis in social media. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*.

Wang, J. H., Liu, T. W., Luo, X., & Wang, L. (2018). An LSTM approach to short text sentiment classification with word embeddings. *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing, ROCLING 2018*.

Wang, J., Yu, L. C., Lai, K. R., & Zhang, X. (2016). Dimensional sentiment analysis using a regional CNN-LSTM model. *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Short Papers*. https://doi.org/10.18653/v1/p16-2037

Williams, R. J., & Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. *Back-Propagation: Theory, Architectures and Applications*.

Yegnanarayana, B. (1994). Artificial neural networks for pattern recognition. *Sadhana*. https://doi.org/10.1007/BF02811896