Detección de palabras clave en el análisis de sentimiento de tweets usando técnicas de ML

Julián David Mazo Correa

Trabajo de grado presentado para optar al título de Especialista en Analítica y Ciencia de Datos

Asesor
Julián David Arias Londoño, Doctor (PhD)

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos
Medellín, Antioquia, Colombia
2021

Cita	Mazo Correa [1]
Referencia Estilo IEEE (2020)	[1] J. D. Mazo Correa, “Detección de palabras clave en el análisis de sentimiento de tweets usando técnicas de ML”, Trabajo de grado especialización, Especialización en Analítica y Ciencia de Datos, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2021.

Especialización en Analítica y Ciencia de Datos, Cohorte II.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTAS DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

ML	Machine Learning
NLP	Natural Language Processing
UDEA	Universidad de Antioquia
BERT	Bidirectional Encoder Representations from Transformers
RoBERTa	Robustly Optimized BERT Pretraining Approach
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network

RESUMEN

El desarrollo del presente trabajo se basa en identificar las palabras claves que manifiestan la carga de sentimiento clasificados como positivo, negativo y neutral sobre un conjunto de tweets. Se diseñaron 7 modelos, dos de ellos con redes neuronales convolucionales y redes neuronales recurrentes para el etiquetado de sentimiento como ejercicio de ilustración ya que los tweets están previamente anotados, un baseline que implementan 3 estrategias de preprocesamiento de texto para la salida e implementación de arquitecturas con capas de BERT y RoBERTa, estos para detectar las palabras claves del tweet que poseen la carga de sentimiento el cual es objetivo principal de la tarea. El modelo que mejor resultados arrojó según la métrica del índice de Jaccard fue el implementado con RoBERTa con un valor de 0.749, es importante anotar que uno de los baseline implementado con una capa de embedding a partir de una matriz de coocurrencia generada por la técnica Glove haya obtenido un valor de 0.586 del promedio del índice de Jaccard por encima de la implementación con BERT que fue del 0.549 sobre la misma métrica. La exploración de los datos evidenció que la eliminación de caracteres especiales y palabras con poco significado semántico permiten mejorar el desempeño de los modelos además la creación de estrategias de preprocesamiento de texto para la variable objetivo ayuda a implementar un baseline con un buen desempeño para regirse como punto de referencia en el diseño de modelos más robustos.

Palabras Clave — Twitter, sentimiento, palabras clave, modelos, procesamiento del lenguaje natural

ABSTRACT

The present work is focused on identifying the keywords that manifest the sentiment load classified as positive, negative, and neutral on a set of tweets. 7 models were designed, two of them with convolutional neural networks and recurrent neural networks for the labeling of sentiment as an illustration exercise since the tweets are previously annotated, a baseline that implements 3 text preprocessing strategies for the output and implementation of architectures with layers of BERT and RoBERTa, these to detect the keywords of the tweet that have the sentiment load which is the main objective of the task. The model that produced the best results according to the Jaccard index metric was the one implemented with RoBERTa with a value of 0.749, it is important to note that one of the baselines implemented with an embedding layer from a co-occurrence matrix generated by the Glove technique obtained a value of 0.586 from the average of the Jaccard index above the implementation with BERT which was 0.549 on the same metric. The exploration of the data showed that the elimination of special characters and words with little semantic meaning allow to improve the performance of the models, and the creation of text preprocessing strategies for the target variable helps to implement a baseline with a good performance to be governed as benchmark for the design of more robust models.

Keywords — twitter, sentiment, keywords, models, natural language processing

I. INTRODUCCIÓN

A. Contexto del problema.

Twitter es una de las redes sociales más populares de esta década, donde cada tweet es la idea particular de un usuario que, como es normal, puede diferir de las opiniones de otros usuarios en la red. En Twitter se expresan opiniones de diversos temas como política, religión, deportes, ciencia, entre otros, y cada una de ellas puede tener una carga de sentimiento en su concepción, lo cual, puede producir una imagen favorable con una mención positiva o por el contrario una percepción adversa si se postea un mensaje negativo sobre el sujeto en cuestión.

Es apenas entendible que, debido al alcance de Twitter, muchas empresas se hayan ocupado por desarrollar o adquirir soluciones para la detección o asociación de sentimientos de tweets dirigidos hacia sus compañías, con el objetivo de mejorar o por lo menos mitigar cualquier daño colateral que la queja de un usuario pueda tener sobre su imagen o producto.

Existen diversas aplicaciones en las cuales se puede evidenciar el análisis de sentimiento sobre los posts de los usuarios en la red social Twitter, tales como opiniones asociadas a microbiología [1], elecciones presidenciales en Colombia periodo 2014 [2], Industria del retail [3], deportes [4], religión [5], entre otros aspectos relevantes que se discuten y se mencionan a diario. La gran mayoría de las aplicaciones se enfocan en comprender la visión de los usuarios sobre un tema en específico y su influencia en el desarrollo de una discusión, quizás donde se evidencia un esfuerzo más considerable es en la industria ya que el posicionamiento de las marcas ayuda a la venta de sus productos y servicios, como se ilustra en el sector del turismo, uno de los más sensibles por estas publicaciones [6].

Atendiendo a lo anterior se abordará un reto donde el objetivo primordial es entender la carga de sentimiento de un tweet, seleccionando el texto que lo representa. Un factor fundamental para resolver esta tarea es conocer el etiquetado de cada tweet, cabe aclarar que el dataset a analizar tiene una variable con este ítem, sin embargo, se profundizará en hallar esta etiqueta para posteriormente proceder a seleccionar ese trozo de tweet que permita discernir si efectivamente hay cierta coherencia con la carga de sentimiento.

Es importante aclarar que se analizarán en total 3 sentimientos: Positivo, Negativo y Neutral, debido a que es posible analizar más clases de sentimiento derivados de los anteriores, es decir, para un sentimiento positivo se podrían encontrar derivaciones como felicidad, amor, perdón, etc, para el negativo: odio, decepción, tristeza, etc., y para el neutral, descripción, anuncio, respuesta lógica, etc.

La tarea u objetivo en cuestión es traído desde un reto de la página Kaggel, la fuente del dataset utilizado en el reto es la plataforma appen cuyo título es: "Sentiment Analysis: Emotion in Text tweets with existing sentiment labels" y es utilizado bajo *creative commons attribution 4.0. International License.* [7]

B. Análisis de sentimiento del texto

El análisis de sentimiento es un problema del aprendizaje automático cuyo objetivo es determinar el tipo de emoción de una persona a partir de información gestual o escrita; por lo tanto, existen trabajos de análisis de sentimiento empleando imágenes [8], voz [9] o texto [10]. En el caso particular del análisis de sentimiento en texto, la idea básica es que, a partir de la información encontrada en una frase, párrafo o incluso artículo, se pueda obtener de manera automática la carga de sentimiento, es decir, si es positivo, negativo o neutral, éste es un problema clásico en la rama del procesamiento de lenguaje natural que se ha abordado en el área de machine learning desde algoritmos clásicos hasta redes neuronales. Este tipo de tareas generalmente son afrontadas como ejercicios de clasificación que hacen parte de la rama de aprendizaje supervisado. Es importante mencionar que se puede afrontar también como ejercicio de clustering perteneciente a la rama del aprendizaje no supervisado si es que no se tiene anotado cada tweet.

Para ilustrar de forma granular el problema, se ha recurrido a la generación de un modelo el cual se basa en la implementación de redes neuronales recurrentes y otro con redes neuronales convolucionales, para ambos se definirán métricas [11] que permitirán conocer el performance del modelo y a partir de ellas concluir cuál es el más adecuado para resolver dicha tarea.

C. Detección de las palabras claves que identifican el sentimiento anotado

El objetivo principal del trabajo es diseñar un modelo que permita extraer la porción del tweet que contiene la carga de sentimiento, para ello, se diseñarán diversos tipos de arquitecturas de redes neuronales (Convolucional, Recurrente) donde se elegirá la que obtenga los mejores resultados de acuerdo con la métrica seleccionada. Se empleará como única métrica la función de Jaccard [12]. Como punto de partida se tendrá un baseline como referencia para validar el desempeño de los modelos y se concluirá entonces cual es el más apropiado para desempeñar la tarea.

Todos los experimentos serán ambientados en Notebooks de Jupyter, si en caso dado el poder computacional empleado no es óptimo se importará cada archivo en Google Colab [13] para emplear un entorno de ejecución que contenga los requerimientos necesarios para las ejecuciones (GPU, TPU).

II. OBJETIVOS

A. General

Diseñar un modelo que permita extraer de un tweet la porción de texto que contenga la carga de sentimiento etiquetado con la mayor exactitud posible según la evaluación de la métrica de Jaccard.

B. Específicos

- Desarrollar estrategias que permitan la ejecución de modelos utilizando redes neuronales para la clasificación de sentimiento, brindando una mayor claridad sobre la utilización de arquitecturas de este tipo en un problema de clasificación.

- Implementar una capa de embedding en los modelos diseñados, empleando un entrenamiento local (Corpus del dataset) y la técnica de transfer learning para concluir cuál método emplear en el modelo.

- Diseñar estrategias de preprocesamiento en los datos de entrada para la correcta ejecución de los modelos a implementar.

- Modelar una arquitectura base donde sea posible seleccionar el texto que caracteriza el sentimiento etiquetado.

- Diseñar arquitecturas partiendo de bases como BERT y RoBERTa para construir modelos robustos brindando mayor precisión en la consecución del objetivo principal.

III. METODOLOGÍA

A. Estrategias de preprocesamiento

Se plantearon varias consideraciones para la implementación de todos los modelos, las cuales permitieron utilizar el framework TensorFlow de Python para la ejecución de las arquitecturas gracias a los cálculos simbólicos a través de tensores [14].

Antes de continuar cabe aclarar que el texto o variables de tipo ‘string’ no son aptos como entradas de un modelo de machine learning, por lo cual se deben transformar según las reglas del Procesamiento del Lenguaje Natural para ser operadas y proveer sentido a los modelos mencionados [15].

A continuación, se mencionan 4 pasos básicos aplicados al preprocesamiento de los datos para los modelos de detección de sentimiento.

1) Base: En este apartado se mencionan puntos básicos realizados en los datos para todos los modelos en la detección de sentimiento y baseline en la extracción de palabras claves.

- Remover caracteres especiales, incluyendo signos de puntuación que puedan alterar la composición de algunas palabras.

- Eliminación de stopwords sobre cada uno de los tweets y el texto objetivo. Se llaman stopwords a todas las palabras que por sí solas no poseen mucho sentido semántico, un ejemplo de ellas son las preposiciones y artículos, los cuales ostentan mayor frecuencia en cualquier corpus [16]. Se utilizó el módulo corpus de la librería NLTK (Natural Language Toolkit) para limpiar aquellas palabras sin contenido semántico en los tweets.

- Se crea un vocabulario interno con todas las palabras contenidas en los tweets, posteriormente cada tweet es codificado como una secuencia de enteros, donde cada número es el índice de un token en el diccionario [17].

- Se realiza un proceso de Padding a todos los tweets del dataset, el cual consiste en establecer un tamaño fijo para todos los tweets, y la secuencia que no posea dicha longitud será completada con ceros a la izquierda [18].

2) Análisis de sentimiento: Para la característica 'Sentiment' que es la variable objetivo, al ser categórica se transforma mediante la técnica de 'One-Hot-Encoding' en un arreglo de vectores cada uno de 3 posiciones (cada posición representando un sentimiento) donde la posición que representa el sentimiento tiene un valor de uno y las demás un cero. [19]

Se divide el dataset en 3 conjuntos: Entrenamiento, validación y prueba, esto en aras de optimizar el procesamiento cuando se ejecute el modelo, ya que una validación cruzada tendría mucho costo computacional además se considera que el dataset tiene suficientes observaciones que pueden ayudar a que el resultado del ejercicio sea válido sin obtener un tiempo de ejecución elevado.

3) Baseline detección de palabras claves con carga de sentimiento: Se abarcaron tres estrategias para codificar la variable objetivo, la primera consiste en crear un vector de longitud igual al número máximo de palabras encontradas en la variable 'text' para cada tweet, éste tendrá valores establecidos igual a uno en las posiciones de las palabras donde se haya seleccionado texto significativo, las restantes serán establecidas en cero, como se aprecia en el siguiente ejemplo:

Máxima longitud: 15
Texto: "Cuando veo tus ojos creo que brilla mi corazón de amor"
Texto seleccionado: "brilla mi corazón de amor"
Entrada: [0,0,0,0,0,0,1,1,1,1,1,0,0,0,0]

Como se observa en el ejemplo anterior para completar el tamaño del vector según la longitud máxima se establecen en cero las posiciones a la derecha. Se requiere que todos los elementos del tensor tengan un tamaño fijo para que el framework de Tensorflow procese de forma exitosa la variable objetivo.

En la segunda estrategia se codificaron dos salidas en la variable objetivo, una que contiene la posición inicial del texto seleccionado y otra que contiene la posición final del mismo, al igual que en la estrategia anterior, se crean vectores de longitud fija en los cuales se pone valor de uno en la posición inicial y las demás serán cero y para la otra salida se establecerá el valor de 1 en la posición final y las restantes serán cero como se muestra a continuación:

Máxima longitud: 15
Texto: “Cuando veo tus ojos creo que brilla mi corazón de amor”
Texto seleccionado: “brilla mi corazón de amor”
Entrada 1: [0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]
Entrada 2: [0,0,0,0,0,0,0,0,0,0,1,0,0,0,0]

Similar a la estrategia uno, según la longitud máxima establecida el resto de posiciones a la derecha se completarán con valor de cero.

Para la tercera estrategia se emplea una metodología similar que la anterior donde se crean dos entradas una que representa la posición inicial del trozo de texto que posee la carga de sentimiento y otra que representa la posición final, esta difiere de la segunda en que no se emplea un vector de unos y ceros, sino que se tomará un número entero para ambas salidas iniciando en la posición cero como se muestra a continuación:

Máxima longitud: 15
Texto: “Cuando veo tus ojos creo que brilla mi corazón de amor”
Texto seleccionado: “brilla mi corazón de amor”
Entrada 1: 6
Entrada 2: 10

B. Modelos empleados

Todos los modelos presentados en esta sección pueden encontrarse en el github empleado para esta monografía [22]

En el análisis de sentimiento y el baseline para la selección de texto clave se utilizó una capa de embedding pre entrenada, la cual es generada a partir de una representación de vectores para cada palabra. La generación de dichos vectores se realiza a través de la técnica Glove que es un algoritmo no supervisado entrenado con estadísticas globales de coocurrencia palabra-palabra agregada de un corpus y las representaciones resultantes muestran interesantes sub-estructuras lineales de un espacio vectorial de palabras. [23]

1) Análisis de sentimiento: Se diseñaron dos modelos en los cuales se utilizaron redes neuronales recurrentes y convolucionales ambos tienen como entradas la variable 'text' que como ya se mencionó en la sección 3.1.1 se tokenizó y se le realizó padding. La variable de salida es 'selected_text' a la cual se le realizó codificación "One-hot-Encoding" para efectos de procesamiento del modelo.

a) Modelo secuencial con red neuronal recurrente bidireccional:

- Capa de embedding: Consta de una matriz de dimensión igual a 12001 x 100 que contiene la matriz de coocurrencia palabra a palabra ya sea de un corpus pre entrenado (Transfer learning) o de un entrenamiento sobre el dataset empleado en este ejercicio.

- Capa LSTM Bidireccional: Posee 50 unidades, y una activación de tangente hiperbólica.

- Capa Densa: Con 128 neuronas y una activación tipo 'relu'

- Capa de dropout: Esta capa ayuda a reducir el sobre ajuste, es una manera de regularizar la actualización de los pesos a una tasa de 0.2.

- Capa Densa: Salida de la red, compuesta por 3 neuronas cada una de ella por cada sentimiento y una activación tipo softmax para que los valores arrojados por cada una de las neuronas sumen el valor de 1.

El optimizador es “Adam”, la función de pérdida es “categorical-crossentropy” para predecir más de dos categorías y las métricas establecidas son Recall, Precisión y F1, en la figura 1. se aprecia el esquema de dicha red.

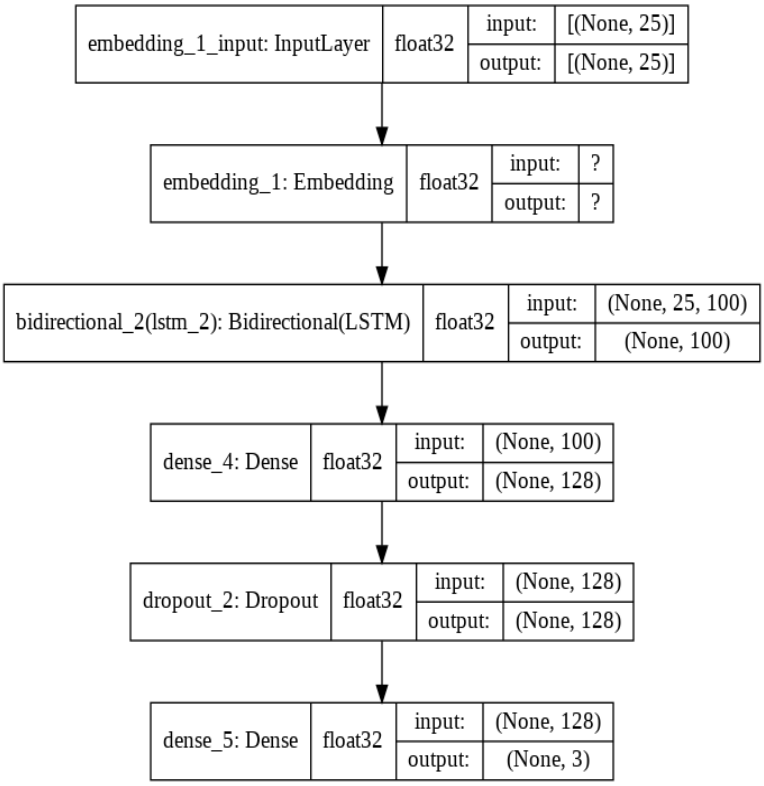

Fig. 1. Arquitectura modelo funcional RNN para análisis de sentimiento

b) Modelo funcional con redes neuronales convolucionales

- Capa Input: Forma (None,25), determinando la entrada del tensor, en este caso el conjunto de entrenamiento que consta de los tweets y la carga de sentimiento asociada.

- Capa de embedding: Compuesta por una matriz de coocurrencia palabra-palabra la cual puede ser generada a través de un proceso de transfer learning o entrenada directamente con el dataset del ejercicio.

- Capa convolucional 1D: capa convolucional que posee 256 neuronas y un Kernel de dimensión 5, con una activación tipo ‘relu’.

- Capa MaxPooling1D: esta capa tiene un pooling de 2 para reducir los mapas de activación al ser ingresados a la siguiente capa convolucional.

- Capa Convolucional 1D: idénticas características de la capa convolucional anterior, con un número de neuronas igual a 256 y un Kernel con dimensión 5.

- Capa MaxPooling1D: Pooling size de 8 el cual reduce el tamaño de los mapas de activación generados en la capa convolucional anterior.

- Capa Flatten: modifica la dimensión del tensor para que la secuencia pueda ser procesada por una capa densa.

- Capa densa: configurada con 256 neuronas y tipo de activación ‘relu’..

- Capa densa: capa de salida sobre la secuencia configurada con 3 neuronas o unidades, y una activación tipo Softmax.

Se compila el modelo con una función de pérdida ‘categorical-crossentropy ‘ un optimizador ‘rmsprop’ y las métricas Recall, Precision y F1.

Cabe aclarar que en el entrenamiento se incluye la técnica de early stopping el cual tiene como parámetro monitor = ‘loss’ con respecto a la función de pérdida y patience en 1, con esto se está mitigando el desvanecimiento del gradiente descendiente cuando se ejecute dicho entrenamiento del modelo. En la figura 2. se puede apreciar la arquitectura del modelo y lo descrito anteriormente.

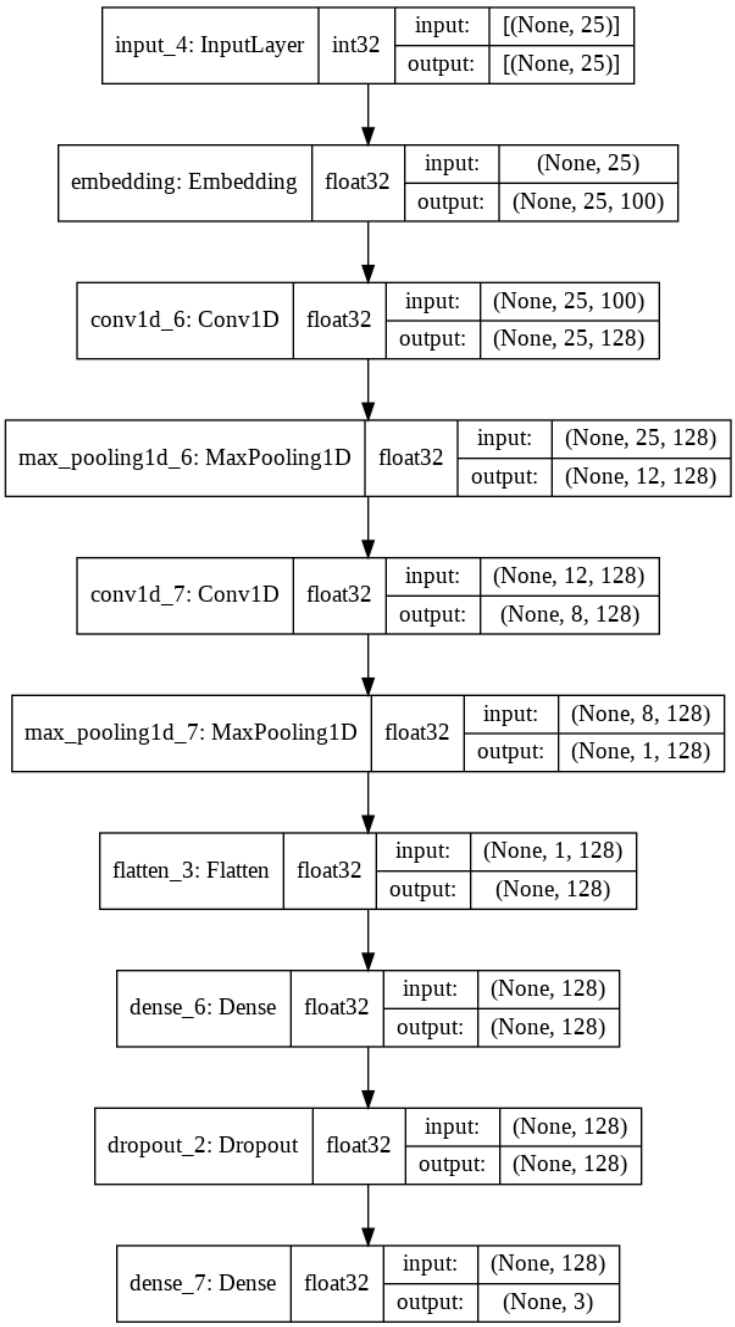

Fig. 2. Modelo funcional con CNN para análisis de sentimiento

2) Extracción de palabras clave con carga de sentimiento asociada

a) Modelo Baseline Estrategia 1: Con este modelo se pretende tener dos entradas, una que corresponde al conjunto de tweets, los cuales fueron tokenizados para posteriormente agregarle padding según la máxima longitud de un tweet establecida, esta se envía a una capa de embedding donde se implementa transfer learning sobre una matriz de coocurrencia generada por la técnica Glove, y está el sentimiento categorizado en un rango de cero a dos la cual es enviada a una capa de embedding entrenada en el modelo donde luego pasa a una capa Flatten, posteriormente ingresa a una capa RepeatVector donde se enviará una secuencia en 25 oportunidades, luego se concatenan la salida de la capa RepeatVector y la primera capa de embedding, posteriormente pasará por dos capas bidireccionales LSTM, se ingresará a una capa densa de 128 unidades para finalizar en otra capa densa con una neurona y activación tipo sigmoide, arrojando única salida.

- Capa Input: Capa de entrada que recibe las secuencias del conjunto correspondiente a la variable text (tokenizada) y la variable objetivo (tensor one-hot-encoding) con forma (25,).

- Capa Input2: Recibe la categorización del sentimiento.

- Capa Embedding 1: Capa de embedding previamente configurada donde se establece la matriz de coocurrencia generada a través de la técnica Glove con dimensión 100. Se ingresa el input 1.

- Capa embedding 2: Capa de embedding de dimensión 3 y longitud de 20, se ingresa el sentimiento.

- Capa Flatten: Redimensiona la salida de la capa de embedding 2.

- Capa RepeatVector: Con factor de repetición en 25 para concatenar el sentimiento con la salida de la capa de embedding 1.

- Capa Concatenate: Concatena el sentimiento (salida de la capa RepeatVector) con la capa de embedding 1. Añade la carga de sentimiento a cada secuencia que ya ha pasado por la capa de embedding.

- Capa LSTM Bidirecctional 1: Capa bidireccional LSTM con 50 unidades, Return Sequences en True.

- Capa LSTM Bidirecctional 2: Capa bidireccional LSTM con 50 unidades, Return Sequences en False.

- Capa TimeDistributed Dense 1: 128 neuronas y activación tipo relu.

- Capa TimeDistributed Dense 2: con 1 neurona y activación tipo Sigmoid.

El modelo se compila con una función de pérdida “binary_crossentropy”, optimizador ADAM y las métricas utilizadas son Recall, Precisión y Jaccard.

En la figura 3 se aprecia el modelo descrito anteriormente.

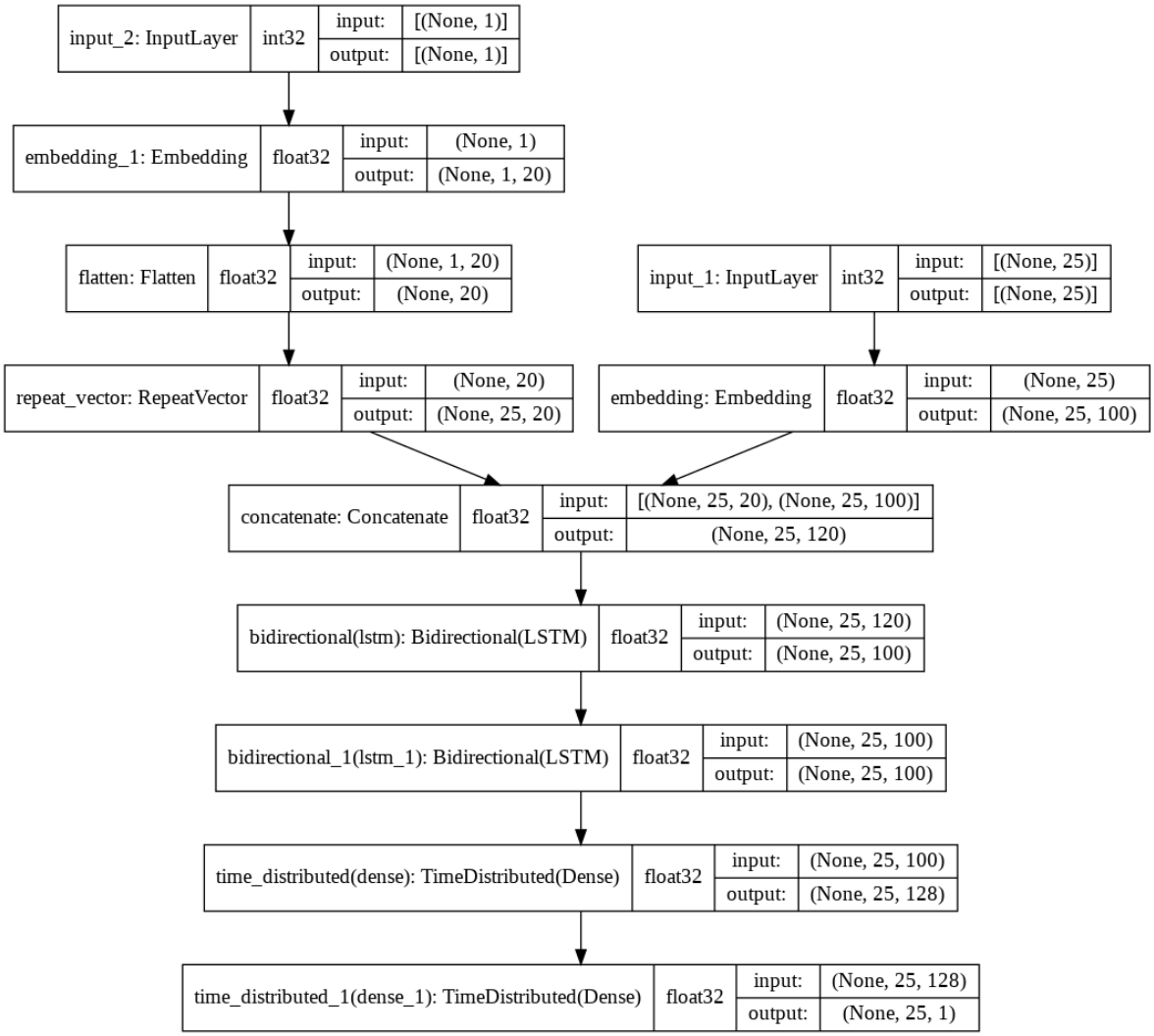

Fig. 3. Modelo baseline con estrategia 1

b) Modelo Baseline Estrategia 2: Al igual que el modelo anterior posee las mismas entradas y una arquitectura de red similar, sólo que este modelo a diferencia del anterior no posee una salida sino dos y estás se conforman por una capa densa con 25 neuronas para activar cada elemento de la secuencia, donde la primera salida dará a conocer el índice de la palabra inicial en el texto seleccionado y la segunda la posición de la palabra final. Estas salidas con un formato One-hot-Encoding, a continuación, se describe con más detalle:

- Capa Input: Capa de entrada que recibe las secuencias del conjunto correspondiente a la variable text (tokenizada) y la variable objetivo (tensor one-hot-encoding) con forma (25,).

- Capa Input2: Recibe el sentimiento con forma (1,).

- Capa Embeding 1: Capa de embedding previamente configurada donde se establece la matriz de coocurrencia generada a través de la técnica Glove con dimensión 100. Se ingresa el input 1.

- Capa embedding 2: Capa de embedding de dimensión 3 y longitud de 20, se ingresa el sentimiento.

- Capa Flatten: Redimensiona la salida de la capa de embedding 2.

- Capa RepeatVector: Con factor de repetición en 25 para concatenar el sentimiento con la salida de la capa de embedding 1.

- Capa Concatenate: Concatena el sentimiento (salida de la capa RepeatVector) con la capa de embedding 1. Añade la carga de sentimiento a cada secuencia que ya ha pasado por la capa de embedding.

- Capa LSTM Bidirecctional 1: Capa bidireccional LSTM con 50 unidades, return Sequences en True.

- Capa LSTM Bidirecctional 2: Capa bidireccional LSTM con 50 unidades, return Sequences en False.

- Capa Dense 1: 128 neuronas y activación tipo relu.

- Capa Dense 2: 25 neuronas y activación tipo Softmax, para detectar cualquiera de las 25 clases para el índice de la palabra inicial.

- Capa Dense 2: 25 neuronas y activación tipo Softmax, para detectar cualquiera de las 25 clases para el índice de la palabra final.

El modelo se compila con una función de pérdida “categorical_crossentropy” para ambas salidas de la red, optimizador ADAM y las métricas utilizadas son Recall, Precisión y Jaccard. En la figura 4 se aprecia el modelo descrito anteriormente.

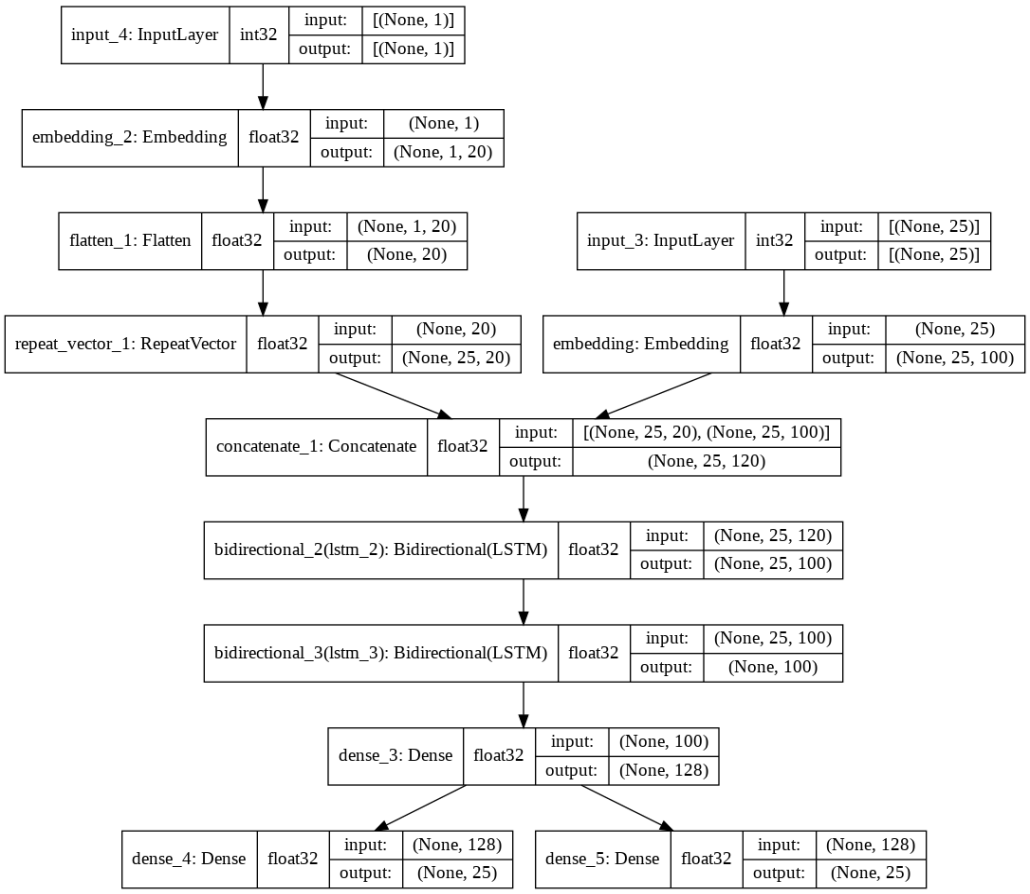

Fig. 4. Modelo baseline con estrategia 2

c) Modelo Baseline Estrategia 3: Brindando entradas iguales que el modelo anterior y una arquitectura de red similar, solo difieren que este modelo presenta dos capas densas correspondientes a cada salida y las componen una neurona cada una para realizar la predicción sobre una variable continua a través de una técnica de regresión. A continuación, se describe el modelo:

- Capa Input: Capa de entrada que recibe las secuencias del conjunto correspondiente a la variable text (tokenizada) y la variable objetivo (tensor one-hot-encoding) con forma (25,).

- Capa Input2: Recibe el sentimiento con forma (1,).

- Capa Embeding 1: Capa de embedding previamente configurada donde se establece la matriz de coocurrencia generada a través de la técnica Glove con dimensión 100. Se ingresa el input 1.

- Capa embedding 2: Capa de embedding de dimensión 3 y longitud de 20, se ingresa el sentimiento.

- Capa Flatten: Redimensiona la salida de la capa de embedding 2.

- Capa RepeatVector: Con factor de repetición en 25 para concatenar el sentimiento con la salida de la capa de embedding 1.

- Capa Concatenate: Concatena el sentimiento (salida de la capa RepeatVector) con la capa de embedding 1. Añade la carga de sentimiento a cada secuencia que ya ha pasado por la capa de embedding.

- Capa LSTM Bidirecctional 1: Capa bidireccional LSTM con 50 unidades, return Sequences en True.

- Capa LSTM Bidirecctional 2: Capa bidireccional LSTM con 50 unidades, return Sequences en False.

- Capa Dense 1: 128 neuronas y activación tipo relu.

- Capa Dense 2: 1 neurona, no requiere activación ya que lo que se predice es una variable continua y no de clasificación para la posición de la palabra inicial del texto seleccionado.

- Capa Dense 2: 1 neurona, no requiere activación ya que lo que se predice es una variable continua y no de clasificación para la posición de palabra final del texto seleccionado.

El modelo se compila con una función de pérdida “mse” para ambas salidas de la red, optimizador ADAM y las métricas utilizadas son MAE y MSE, recordando que se utiliza una predicción mediante regresión por el tipo de variable objetivo ya que es continua. Para efectos del ejercicio al final se hace un post proceso para determinar el índice de Jaccard.

La figura 4 muestra la distribución del modelo.

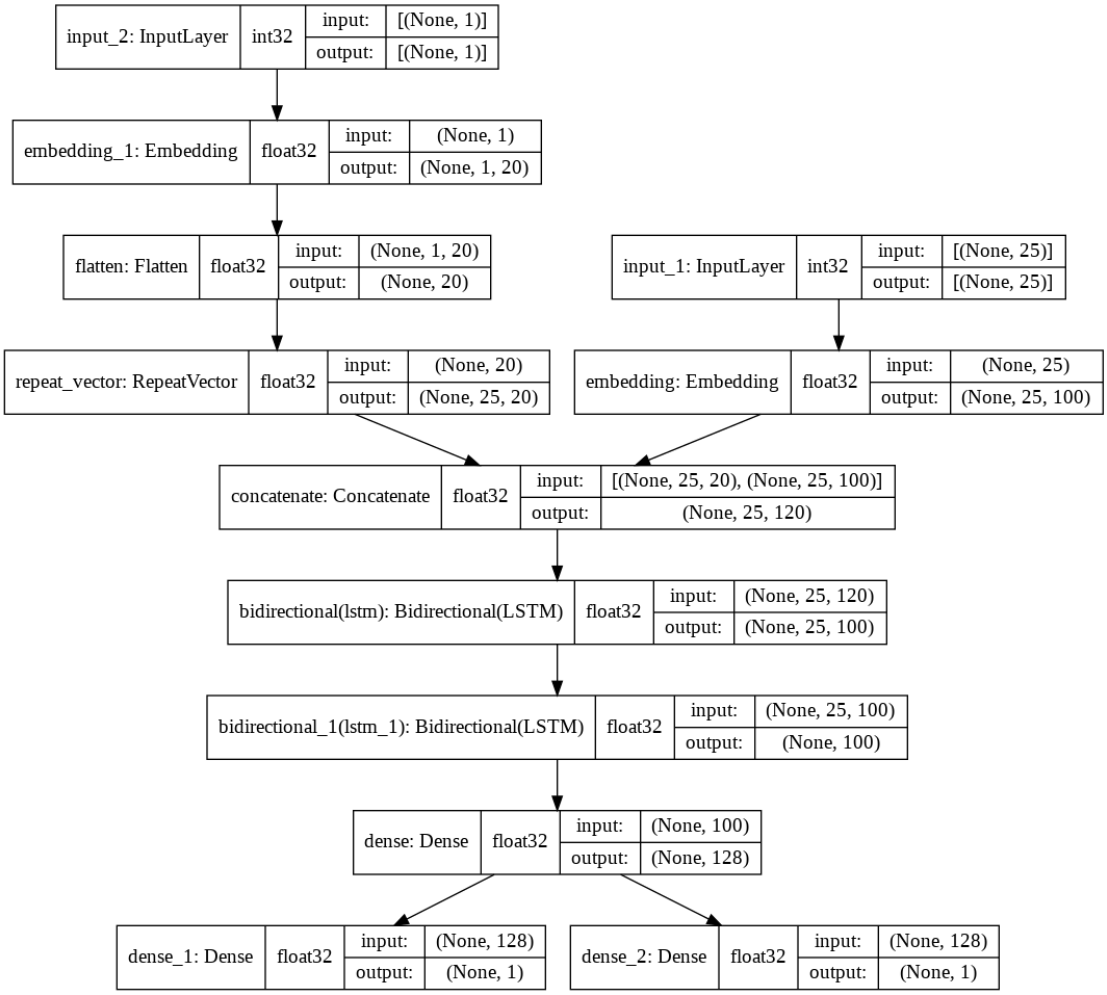

Fig. 5. Modelo baseline con estrategia 3

d) BERT estrategia 2: Este modelo consta de cuatro entradas, tres corresponden al conjunto de tweets completo, que se tokenizan con una función propia de BERT que es un estado del arte publicado por el equipo de Google IA para NLP con múltiples aplicaciones en el área de NLP. Con los tweets tokenizados se obtienen los IDS, máscaras y segmentos [20] (Formato propio de BERT), esto ingresa a la capa bert_model. El sentimiento categorizado entra a una capa de embedding propiamente entrenada, posteriormente pasa a una capa flatten, luego a una capa RepeatVector, esto para ser concatenado con la salida de la capa BERT, esta salida se dirige a una capa LSTM de 50 unidades, pasa una capa densa de 128 neuronas donde consecutivamente pasan a dos capas densas integradas por 25 neuronas, cada una arroja una secuencia donde en la primera activa la posición inicial de la palabra del texto seleccionado y la segunda la final, como se describe a continuación:

- Capa Input 1: Donde se ingresan los Ids que son las palabras del tweet tokenizadas y con padding a la máxima longitud del tweet.

- Capa input 2: Se ingresa el masking, característica de entrada para la utilización de la capa de BERT en el modelo.

- Capa input 3: Establecimiento de los segmentos que al igual que la capa 2 es una característica necesaria para la implementación de la capa de BERT en el modelo.

- Capa input 4: Se ingresa la carga de sentimiento para cada tweet.

- Capa Bert: Se ingresan los Input 1,2,3.

- Capa embedding: Capa de embedding de dimensión 3 y longitud de 20, se ingresa el sentimiento (input 4).

- Capa Flatten: Redimensiona la salida de la capa de embedding.

- Capa RepeatVector: Con factor de repetición en “max_seq_length” para concatenar el sentimiento con la salida de la capa BERT.

- Capa Concatenate: Concatena el sentimiento (salida de la capa RepeatVector) con la capa bert. Añade la carga de sentimiento a cada secuencia que ya ha pasado por la capa BERT.

- Capa LSTM: 50 unidades, return Sequences en False.

- Capa Dense 1: 128 neuronas y activación tipo relu.

- Capa dropout: con una tasa de 0.3

- Capa Dense 2: 25 neuronas y activación tipo Softmax, para detectar cualquiera de las 25 clases para el índice de la palabra inicial.

- Capa Dense 2: 25 neuronas y activación tipo Softmax, para detectar cualquiera de las 25 clases para el índice de la palabra final.

El modelo se compila con una función de pérdida “categorical_crossentropy” para ambas salidas de la red, optimizador ADAM y las métricas utilizadas son Recall, Precisión y Jaccard. En la figura 6 se aprecia el modelo descrito anteriormente.

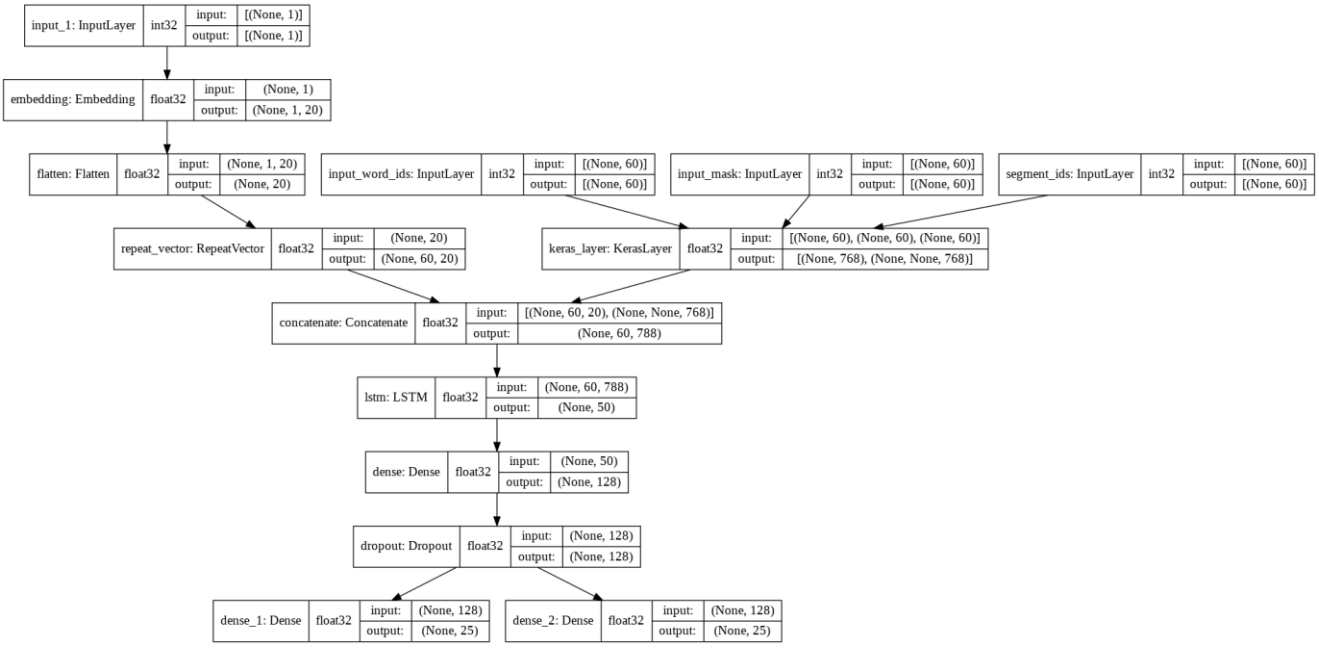

Fig. 6. Modelo implementación BERT

e) RoBERTa: Tiene 3 entradas con el formato propio de RoBERTa [21], ingresan a la capa TFRobertaModel, en la cual se envían a dos segmentos de la red neuronal idénticos para dos salidas diferentes el cual uno corresponde al índice inicial y el otro al final del texto seleccionado, constan de una capa Dropout con una tasa de 0.1, una capa convolucional de una dimensión compuesta de un kernel de tamaño uno, una capa flatten y posteriormente una de activación con función softmax, a continuación se describe lo mencionado:

- Capa Input 1: Donde se ingresan los Ids que son las palabras del tweet tokenizadas y con padding a la máxima longitud del tweet.

- Capa input 2: Se ingresa el attention mask, característica de entrada para la utilización de la capa de RoBERTa en el modelo.

- Capa input 3: Establecimiento de los tokens que representan los tipos de Ids donde al igual que la capa 2 es una característica necesaria para la implementación de la capa de RoBERTa en el modelo.

- Capa RoBERTa: Se ingresan el input 1, 2, 3.

- Capa Dropout: Establecido en una tasa de 0.1.

- Capa Convolucional de 1 dimensión: Se establece un kernel de una unidad.

- Capa Flatten: Redimensiona la salida de la capa convolucional.

- Capa de Activación 1: Se activa una salida con la función Softmax.

- Capa de Activación 2: Se activa una salida con la función Softmax.

El modelo es compilado con una función de pérdida “Categorical_crossentropy”, con optimizador ADAM, y se integra la métrica de jaccard directamente en el entrenamiento del modelo.

En la figura 6 se puede apreciar el grafo del modelo.

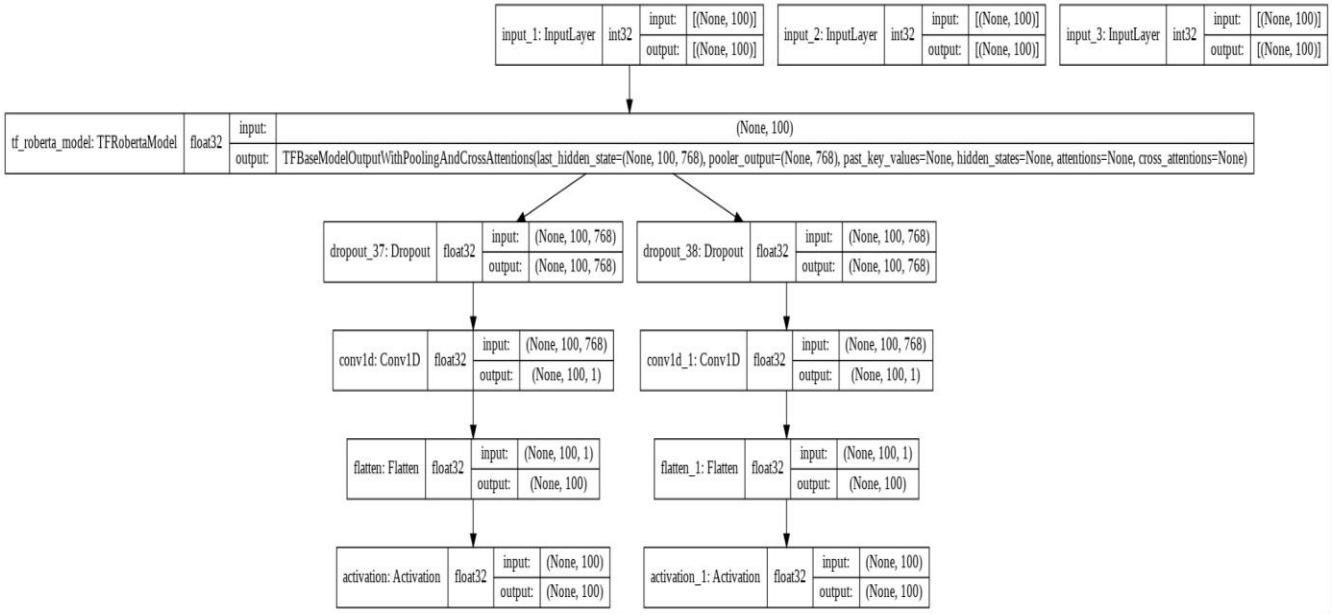

Fig. 7. Modelo implementación RoBERTa

IV. EXPERIMENTACIÓN Y RESULTADOS

A. Análisis Exploratorio

Los primeros resultados se obtienen desde el análisis exploratorio del dataset, se obtienen características importantes como la distribución de las clases de sentimiento y palabras con más frecuencia en cada clase, entre otros.

Antes de realizar cualquier procesamiento sobre los datos se deberá obtener la distribución por etiquetado de sentimiento, donde es relevante rescatar que la frecuencia de los sentimientos negativo y positivo posee un valor similar, además los tweets etiquetados con sentimiento neutral poseen una frecuencia un poco más elevada. En la figura 7 se aprecia la distribución de la data según el sentimiento:

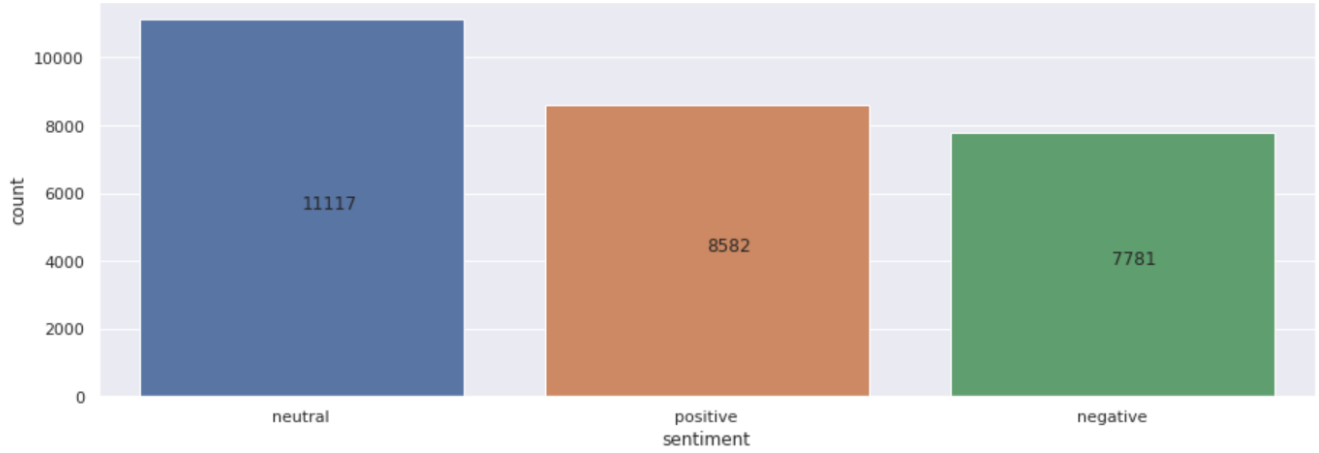

Fig. 8. Distribución de los datos según la clase de sentimiento.

Dada la distribución del etiquetado de tweets según la carga de sentimiento se decide no realizar ningún preprocesamiento de balanceo del dataset pues se determina que la diferencia de observaciones de las diversas categorías no es muy desproporcionada una con respecto a las restantes.

La frecuencia de las palabras en cada una de las 3 etiquetas es un punto de partida para determinar la necesidad de una limpieza de stopwords o de caracteres especiales, esto debido a que palabras con poco o nada de contenido semántico podrían intervenir en una predicción acertada del tweet dado, encontrando que para cada etiqueta predominan pronombres, artículos y preposiciones, como se evidencia en la tabla 1.

TABLA I
FRECUENCIA DE PALABRAS EN LA VARIABLE ‘TEXT’ CLASIFICADOS POR SENTIMIENTO

Positivo		Negativo		Neutral	
Palabra	Frecuencia	Palabra	Frecuencia	Palabra	Frecuencia
i	3687	i	4506	i	4872
the	2982	to	2870	to	4119
to	2954	the	2442	the	3522
a	2375	my	2016	a	2506
you	1759	a	1803	my	1992
and	1639	and	1544	and	1827

Según lo presentado en la tabla 1, se procede a realizar una limpieza de stopwords y puntuación irrelevante tanto a la variable ‘text’ que contiene el tweet completo como a las palabras claves que representan el sentimiento, calculando nuevamente la frecuencia se tiene la tabla 2 para los tweets completos y la tabla 3 para el texto seleccionado.

TABLA II
FRECUENCIA DE PALABRAS EN LA VARIABLE ‘TEXT’ POR SENTIMIENTO POSTERIOR A LA LIMPIEZA

Positivo		Negativo		Neutral	
Palabra	Frecuencia	Palabra	Frecuencia	Palabra	Frecuencia
day	1223	like	476	get	623
good	1046	dont	469	go	575
love	875	cant	465	day	503
happy	840	get	436	dont	493
mothers	629	miss	422	going	479

TABLA III
FRECUENCIA DE PALABRAS EN LA VARIABLE ‘SELECTED_TEXT’ POR SENTIMEINTO POSTERIOR A LA LIMPIEZA

Positivo		Negativo		Neutral	
Palabra	Frecuencia	Palabra	Frecuencia	Palabra	Frecuencia
good	826	miss	358	get	612
happy	730	sad	343	go	568
love	697	sorry	300	day	492
day	456	bad	246	dont	483
thanks	439	hate	230	going	472

B. Análisis de sentimiento

1) Modelos ejecutados: Se diseñaron dos arquitecturas basadas en redes neuronales recurrentes (RNN) y redes neuronales convolucionales (CNN) como se describió en el capítulo III apartado B. Para ambos modelos se probaron dos tipos de capas de embedding, una con transfer learning implementado el método Glove de Stanford y otra con un entrenamiento local sobre los tweets de entrenamiento. Para ambas capas de embedding se establecieron las métricas de Recall, F1 y Accuracy las cuales se eligieron para este análisis.

En aras de elegir el modelo más adecuado para obtener el etiquetado de sentimiento con una mayor precisión se realizó una búsqueda de malla [24] para identificar los hiper parámetros más acordes en la ejecución de ambos modelos.

Se segmenta la búsqueda de hiperparámetros en dos experimentos globales, uno para la arquitectura de redes recurrentes (tabla 4) y otra para la compuesta por redes convolucionales (tabla 5).

En la tabla 4 se aprecian 12 columnas las cuales corresponden al número de épocas, “Neuronas” que representa la búsqueda de neuronas en la capa LSTM Bidireccional, batch donde se busca el número de batch óptimo y Dropout para encontrar la tasa más apropiada. Luego están los resultados de las métricas Accuracy, Recall y F1 Score junto el resultado en el conjunto de validación y para finalizar está el campo TF que marca si el experimento contiene transfer learning o no.

TABLA IV

BÚSQUEDA DE HIPERPARÁMETROS PARA LA ARQUITECTURA DE REDES RECURRENTES.

Épocas	Neuronas	Batch	Dropout	Accuracy	Accuracy en val	Recall	Recall - en val	F1 Score	F1 Score en val	TF
3,5,7,10 Mejor(5)	fijas - 50, 128, 3	32,128 Mejor (32)	0.2	0,7432	0,7033	0,697	0,6561	0,738	0,6945	Si
3,5,10,12 Mejor(5)	Capa LSTM 10,30,50 Mejor (50)	32	0.2	0,7322	0,7033	0,6721	0,6561	0,7164	0,6945	Si
3,5,10 Mejor(5)	Capa LSTM 50,100 Mejor (50)	32	0.2	0,7334	0,7045	0,6763	0,6601	0,7199	0,6953	Si
3,5,7 Mejor (5)	Capa LSTM 50,70,90 Mejor (70)	32	0.2	0,7352	0,7059	0,6778	0,649	0,7204	0,693	Si
3,5,7,10 Mejor(3)	fijas - 50, 128, 3	32,128 Mejor (128)	0.2	0,8186	0,6689	0,7884	0,6467	0,8136	0,663	No
3,5,10,12 Mejor(3)	Capa LSTM 10,30,50 Mejor (10)	32,128 Mejor (128)	0.2	0,861	0,6807	0,8335	0,6538	0,8568	0,6749	No
3,5,10 Mejor(3)	Capa LSTM 5,7,10 Mejor (5)	128	0.2	0,785	0,6533	0,7228	0,6447	0,7663	0,6491	No
3,5,10 Mejor (3)	Capa LSTM 3,5,7,10 Mejor (7)	32,128 Mejor (128)	0.2,0.3 Mejor (0.3)	0,8408	0,6417	0,7956	0,6323	0,828	0,6419	No

La tabla 5 contiene 12 columnas con idéntico formato y descripción que la técnica del modelo anterior.

TABLA V

BÚSQUEDA DE HIPER PARÁMETROS PARA LA ARQUITECTURA CON RED CONVOLUCIONAL

Épocas	Neuronas	Batch	Dropout	Accuracy	Accuracy en val	Recall	Recall en val	F1 Score	F1 Score en val	TF
3,5,10,12 - Mejor(3)	128,128,128,3 fijas	32,128 - Mejor (32)	0.2	0,739	0,6875	0,6746	0,6786	0,724	0,6786	Si
3,5,10 - Mejor(3)	32,64,128 variables en las de 128 fija la de 3 Mejor(128)	32,128 - Mejor (128)	0.2	0,724	0,6776	0,6618	0,668	0,7105	0,668	Si
3,5,10 Mejor (3)	64,128,256 variables en las de 128 fija la de 3 Mejor (256)	32,128 - Mejor (128)	0.2	0,747	0,6591	0,697	0,6408	0,737	0,6408	Si
3,5,10 Mejor (3)	128,256,512 variables en las de 128 fija la de 3 Mejor (256)	32,128 - Mejor (128)	0.2	0,7458	0,6739	0,6898	0,6564	0,734	0,6564	Si
3,5,10,12 - Mejor(3)	128,128,128,3 fijas	32,128 - Mejor (128)	0.2	0,9026	0,6224	0,8915	0,6188	0,9014	0,6247	No
3,5,10 - Mejor(3)	32,64,128 variables en las de 128 fija la de 3 Mejor(128)	32,128 - Mejor (128)	0.2	0,936	0,6259	0,9298	0,6221	0,9359	0,626	No

3,5,10 Mejor (3)	64,128, 256 variable s en las de 128 fija la de 3 Mejor (128)	32,128 - Mejor (128)	0.2	0,9118	0,5989	0,901	0,5979	0,9115	0,6009	No

Luego de realizar todos los experimentos en la búsqueda de hiperparámetros y verificando los resultados de las métricas tanto en el conjunto de entrenamiento como en el de validación se concluye que empleando un entrenamiento local de la capa de embedding propicia un sobreajuste en el modelo, por lo cual se valorarán solamente los experimentos que contengan Transfer learning para ambas arquitecturas utilizando la métrica en el conjunto de validación como ideal para la elección de los hiperparámetros mostrada en la tabla 6.

TABLA VI

DESEMPEÑO DE LOS MODELOS EN LA BÚSQUEDA DE HIPER PARÁMETROS

Modelo	Accuracy	Recall	F1 Score
Modelo RNN	0,7045	0,6601	0,6953
Modelo CNN	0,6875	0,6786	0,6786

Posteriormente se aprecian los hiperparámetros en la tabla 7.

TABLA VII

HIPERPARÁMETROS ENCONTRADOS PARA AMBOS MODELOS

Parámetros	Modelo RNN	Modelo CNN
Épocas	5	3
Neuronas	50	128
Dropout	0.2	0.2
Batch	32	32

C. Extracción de palabras claves con carga de sentimiento

Se ejecutaron 5 modelos diferentes que varían tanto en arquitectura como en tipo de entrada como el Baseline con estrategias 1, 2, 3, BERT y RoBERTa (ver sección 3.2.2). Cada modelo tuvo un desempeño diferente con relación a las métricas establecidas. Para este trabajo se prioriza en índice de Jaccard ya que también entraron en la evaluación Accuracy, Precisión, Recall y F1 Score, en este caso no se utilizó la búsqueda de hiperparámetros ya que se incurre en un costo computacional alto debido a la complejidad de las arquitecturas, sin embargo, se logró obtener los desempeños esperados según la complejidad de los modelos, en la tabla 8 se puede apreciar el valor de las métricas en cada uno de ellos.

TABLA VIII
PERFORMANCE DE LOS MODELOS EJECUTADOS PARA LA EXTRACCIÓN DE PALABRAS CLAVES

Modelos	Promedio del índice de Jaccard
RoBERTa	0,749
Baseline Estrategia 2	0,586
Baseline Estrategia 3	0,558
BERT	0,549
Baseline Estrategia 1	0,013

Analizando la tabla se evidencia que el modelo roBERTa es el más apropiado para abordar el objetivo principal del proyecto, se esperaba además que la arquitectura que emplea BERT tuviera un desempeño más alto de acuerdo con el baseline con las 3 estrategias planteadas, pero sólo pudo superar a la estrategia 1 que realmente no debería ser considerada debido al bajo desempeño con relación al índice de Jaccard. El baseline que emplea la estrategia 3 de preprocesamiento brinda un

desempeño aceptable y se podría considerar como punto de partida en este ejercicio para futuras implementaciones.

Los resultados demuestran la importancia de una buena estrategia de preprocesamiento cuando se diseña un modelo, para el presente caso un modelo basado en NLP, ya que se emplea la misma arquitectura en 3 de los 5 modelos y el desempeño varía dependiendo de los datos ingresados al modelo, además contemplar si es posible transformar una variable objetivo de modo que la técnica de machine learning varíe a clasificación o regresión si es aprendizaje supervisado.

V. CONCLUSIONES

- La tarea de implementar un modelo de NLP implica ambientar una arquitectura robusta con múltiples capas, también ejecutar diversas estrategias que permiten concretar el objetivo propuesto, evidencia de ello fue el primer paso de exploración de los datos, donde inicialmente se pudieron detectar palabras con poco significado semántico que entorpecen el modelo. Gracias al diseño de una técnica de limpieza se puede ingresar al modelo una información mucho más concisa, esto es eliminando stop words y signos de puntuación irrelevantes para el etiquetado de un tweet.

- Las operaciones simbólicas entre tensores permiten la generación de múltiples estrategias para el preprocesamiento de los datos de entrada y salida, lo cual brinda un amplio espectro de posibilidades para implementar un modelo de machine learning y obtener diversos resultados. Se comprende entonces que es importante la creación de un modelo baseline con una arquitectura definida y a partir de ello generar diversas estrategias de procesamiento de texto para la variable objetivo con el fin de brindarle la salida más apta según la métrica definida. Con este punto de referencia se pueden diseñar e implementar modelos muchos robustos que puedan ser comparados con los resultados del baseline definiendo si es un modelo para tomar en cuenta o puede ser descartado.

- Se aprecia la influencia del entorno físico (hardware) para la ejecución de los modelos, debido a que se optimizó el tiempo de entrenamiento incluso en la búsqueda de hiperparámetros cuando se configuró un entorno con GPU [25] gracias a Google Colaboratory, lo que evidencia la importancia de elegir los recursos adecuados para la implementación de modelos más robustos como BERT o roBERTa que requieren un entorno apropiado con tarjetas gráficas o un cluster de máquinas ya que el tiempo de ejecución se reduce considerablemente si es comparado con un entorno básico sólo con CPU

- Los problemas de procesamiento del lenguaje natural siempre van a estar en foco de investigación como parte fundamental de la industria, en este caso se utilizó un modelo con características de RoBERTa el cual se pudo implementar de gran manera arrojando el resultado más sobresaliente, es del caso mencionar que tan solo en el 2018 sale a luz el artículo de BERT que es base de roBERTa siendo un punto de quiebre en la implementación de soluciones a problemas de NLP. Se espera entonces que las nuevas investigaciones contribuyan al mejoramiento tanto en calidad de resultados como en uso de procesamiento de máquina, incentivando las buenas prácticas y quizás brindando una mejor modelo para el objetivo abordado en este trabajo.

REFERENCIAS

[1] S. Bhatnagar, N. Choubey, “Making sense of tweets using sentiment analysis on closely related topics”, Soc. Netw. Anal. Min. **11,** 44, 2021, doi: https://doi.org/10.1007/s13278-021-00752-0

[2] J. A. Cerón-Guzmán, A Sentiment Analysis Model of Spanish Tweets. A Sentiment Analysis Model of Spanish Tweets. Case Study: Colombia 2014 Presidential Election. [tesis de maestría]. Bogotá D.C (Colombia): Universidad Nacional, 2016.

[3] T. P. Souza, O. Kolchyna, P. C. Treleaven, T. Aste. “Twitter Sentiment Analysis Applied to Finance: A Case Study in the Retail Industry”, 2015, arXiv:1507.00784v3, https://arxiv.org/pdf/1507.00784.pdf.

[4] Wunderlich, Fabian, and D. Memmert, "Innovative Approaches in Sports Science—Lexicon-Based Sentiment Analysis as a Tool to Analyze Sports-Related Twitter Communication", 2020, *Applied Sciences* 10, no. 2: 431, doi: https://doi.org/10.3390/app10020431.

[5] H. Noka, “Text Mining Tweets on Religion.” 2020. [En línea]. https://bit.ly/3HLJkq1

[6] G. Gupta, P. Gupta, “"Twitter Mining for Sentiment Analysis in Tourism Industry.”, 2019, pp. 302-306, doi: 10.1109/WorldS4.2019.8903940.

[7] Kaggel, “Tweet Sentiment Extraction, Extract support phrases for sentiment labels”, 2020. https://bit.ly/3FL1OVI.

[8] U. Doshi, V. Barot, and S. Gavhane, “Emotion Detection and Sentiment Analysis of Static Images”, 2020, pp. 1-5, doi: 10.1109/ICCDW45521.2020.9318713.

[9] V. Anandan, “Sentiment Analysis on Voice Data”, 2021. [En línea].https://bit.ly/3DMZsW1.

[10] E. Boiy, P. Hens, K. Deschacht, MF. Moens, “Automatic Sentiment Analysis in On-line Text.”, 2007, ELPUB2007. Openness in Digital Publishing: Awareness, Discovery and Access - Proceedings of the 11th International Conference on Electronic Publishing held in Vienna, Austria 13-15 June 2007 / Edited by: Leslie Chan and Bob Martens. ISBN 978-3-85437-292-9, 2007, pp. 349-360. https://bit.ly/3FGkki1

[11] R. Agarwal, “The 5 Classification Evaluation metrics every Data Scientist must know.”. 2019. [En línea]. https://bit.ly/3oRN2WF

[12] S. Glen, "Jaccard Index / Similarity Coefficient." 2016. [En línea].https://bit.ly/3E29vGZ

[13] Google, "Google Colaboratory [Software]", Google, 2021

[14] TensorFlow, "TensorFlow Federated: Machine Learning on Decentralized Data [Software]", Tensorflow, 2021

[15] G. Mordecki, "Word embeddings: how to transform text into numbers.", 2017. [En línea] https://bit.ly/3cHPV6Q.

[16] S. Teja, "Stop Words in NLP.", 2020. [En línea].https://bit.ly/30TNNXb.

[17] Tensorflow, "tf.keras.preprocessing.text.Tokenizer [Software].". TensorFlow, 2021

[18] Tensorflow, "tf.keras.preprocessing.sequence.pad_sequences [Software].". TensorFlow, 2021

[19] Tensorflow, "tf.keras.utils.to_categorical [Software].". TensorFlow, 2021

[20] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.", 2019, arXiv:1810.04805v2. https://arxiv.org/pdf/1810.04805.pdf

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, M. Levy, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach", 2019. arXiv:1907.11692v1. https://arxiv.org/pdf/1907.11692.pdf.

[22] Github, "Jmazo25/Monografía [Software]", https://github.com/Jmazo25/Monografia

[23] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation", 2014. [En línea]. https://nlp.stanford.edu/projects/glove/

[24] C. Hansen, "How to use Grid Search CV in sklearn, Keras, XGBoost, LightGBM in Python", 2019. [En línea]. https://bit.ly/30Oihdj

[25] Nvidia, "¿Qué es la computación acelerada por GPU?". 2020. https://bit.ly/3xh8mIM