Procesamiento de características sintácticas y léxicas para análisis de sentimientos usando TextVectorization y capas LSTM bidireccionales

Pablo Andrés Pertuz Duran[1], Iván Daniel Zapata Flórez[2]

[1]Ingeniería Mecatrónica, Universidad Tecnológica de bolívar, Cartagena, Colombia
[2]Ingeniería de sistemas y computación, Universidad Tecnológica de bolívar, Cartagena, Colombia
[*]ppertuz@utb.edu.co

Resumen: El objetivo de este trabajo fue el procesamiento de características sintácticas y léxicas para análisis de sentimientos(SA) usando capas de LSTM y TextVectorization, con el dataset EmoEvent que contiene tweets en español relacionados con diferentes dominios como, por ejemplo: entretenimiento, catástrofes, huelgas globales y política, y unas clases asociadas a cada tweet que vendrían siendo: ira, disgusto, miedo, alegría, tristeza y sorpresa. Luego de hacer el preprocesamiento, y pasar por las distintas metodologías a utilizar ya sea, la codificación de la variable objetivo, el textvectorization, y la creación del modelo con arquitectura NN, el resultado que obtuvimos fue un sobreentrenamiento, para evitar esto usamos el modelo *early stopping*, el modelo detuvo el entrenamiento en el epoch 7 y así evito el sobreentrenamiento, dando como resultado que la mejor pérdida para el dataset de validación fue de 0,2788 y la mejor accuracy de 89%.

Link del repositorio: https://github.com/ppertuzduran/BidirectionalLSTMForSA.git

1. Introducción

El análisis de sentimiento (SA) es una técnica de minería contextual ampliamente utilizada para extraer información útil y subjetiva de datos basados en texto. Se aplica al procesamiento del lenguaje natural (NLP), análisis de texto, biometría y lingüística computacional para identificar, analizar y extraer respuestas, estados o emociones de los datos. La polaridad de sentimiento de un elemento determinado indica las emociones de un usuario asociado con un fragmento de texto, es decir, si el texto representa la actitud positiva, negativa o neutral del usuario hacia el elemento especificado. En esta nueva era de la digitalización, se ha vuelto casi que fundamental la necesidad de compartir emociones, estados de ánimos, pensamientos, etc. En el caso de Facebook con los estados y en el caso de Twitter con los tweets. Con la llegada de toda esta tecnología en línea, se abrieron nuevas puertas para las empresas ya que utilizaron diferentes modelos para conocer los comentarios y las actitudes de las personas para análisis corporativos, inteligencia, vigilancia de actividades no éticas y SA de la opinión del cliente. SA, que a menudo se conoce como minería de sentimientos, es un componente

clave de la PNL, con la intención de ayudar a los usuarios a analizar y reconocer las emociones incluidas en los textos subjetivos.

2. Dataset

EmoEvent dataset, contiene 8.409 tweets anotados escritos en español. Se basa en eventos que tuvieron lugar en abril de 2019 relacionados con diferentes dominios: entretenimiento, catástrofes, política, conmemoración y huelga globales. Trabajamos con dos archivos, uno para entrenar al modelo y otro para realizar las pruebas finales; El conjunto de datos se anotó mediante crowdsourcing. Las etiquetas doradas se asignaron teniendo en cuenta el acuerdo de tres anotadores, las menciones de usuarios de Twitter se sustituyeron por @USUARIO y los hashtags se sustituyeron por #HASHTAGS Y cada tweet está etiquetado con una emoción única. El nombre de las columnas del dataset son: id, tweet y label; Y estos labels de la columna label mencionada anteriormente son los siguientes:

- Ira (también incluye molestia y rabia).
- Disgusto (también incluye desinterés y aversión).
- Miedo (también incluye aprensión, ansiedad, preocupación y terror).
- Alegría (también incluye serenidad y éxtasis).
- Tristeza (también incluye pensatividad y pena).
- Sorpresa (también incluye distracción y asombro).

3. Preprocesamiento

Luego de importar el dataset, se revisó su información con las diferentes funciones que nos ofrece la librería *pandas*, por ejemplo: .head() y .info(), se determinó que no había valores NaN. Se examinó también la distribución de las clases en el dataset, en dónde existía una clase con un solo dato, por lo cual se borró; Al terminar la revisión del dataset, se observó que habían tweets que contenían urls, menciones a usuarios, hashtags, menciones a videos o audios y emojis, estos elementos fueron limpiados para que el modelo procesara la información relevante en los tweets; Para eso se creó una función que elimine cada una, que serían en total cinco funciones, y además creamos una sexta para que aplique todo ese preprocesamiento de las cinco funciones anteriores a cada tweet. Para eliminar las urls, las menciones a usuarios y las menciones a videos o audios usamos el módulo RE ya incorporado

por Python, y para eliminar los emojis usamos la librería cleantext. Ya con la función lista, se aplicó a todos los tweets limpiándolos y preparándolos la extracción de características.

4. Metodología utilizada

4.1 Codificación de la variable objetivo

Al ser un problema de clasificación multiclase, se codificó la variable objetivo con la codificación *one-hot* utilizando el método *LabelEncoder()*. Se aplicó el codificador a las clases de la variable objetivo "label", reformando la estructura de la columna o Serie a una matriz de valores booleanos por cada clase, indicando si el tweet pertenece a esa clase o no.

```
others
joy
sadness
anger
surprise
disgust
fear
```

others	*joy*	*sadness*	*anger*	*surprise*	*disgust*	*fear*
0	*1*	*0*	*0*	*0*	*0*	*0*
1	*0*	*0*	*0*	*0*	*0*	*0*
0	*0*	*1*	*0*	*0*	*0*	*0*
0	*0*	*0*	*0*	*1*	*0*	*0*
0	*0*	*0*	*0*	*1*	*0*	*0*
0	*0*	*0*	*0*	*0*	*1*	*0*

4.2 TextVectorization

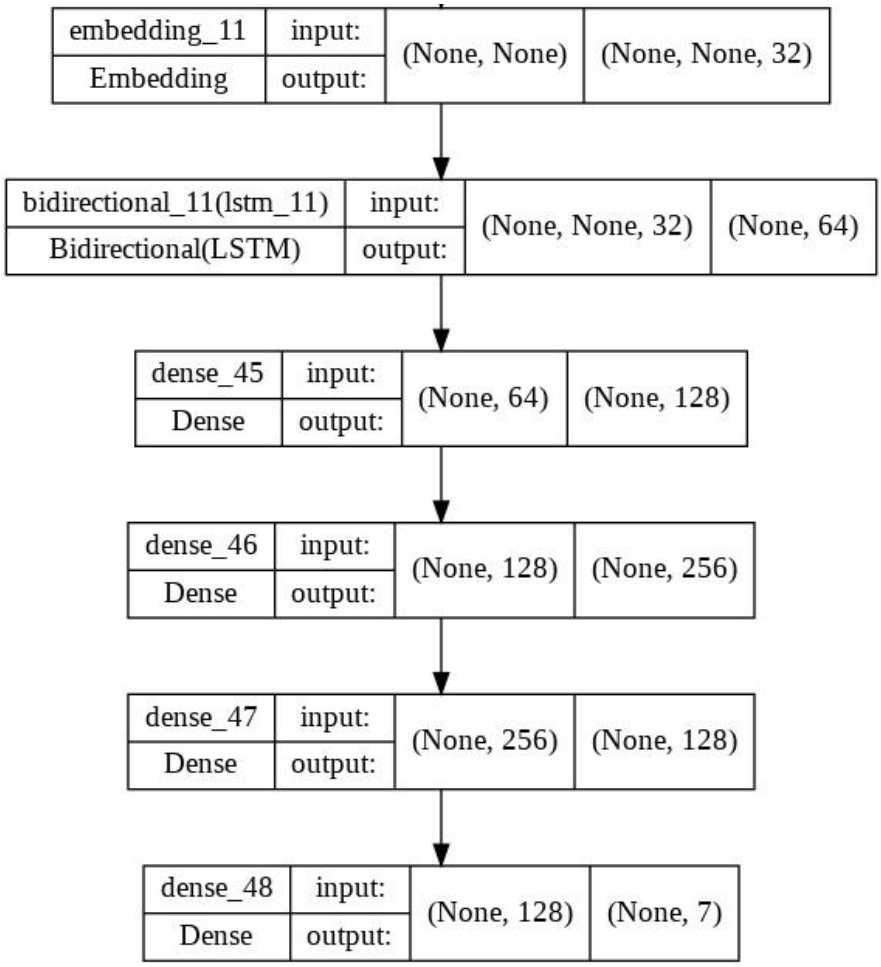
Para convertir las palabras de los tweets a vectores, se aplicó la capa TextVectorization propio de la librería de tensorflow. TextVectorization es una capa de preprocesamiento que asigna características de texto a secuencias enteras.

Se definió el número de palabras en el vocabulario del vectorizador como 200000 (MAX_FEATURES), con una longitud máxima de 1800 palabras por tweet a vectorizar. Si el tweet no abarca el límite de 1800 palabras, el vector será completado con ceros *(zero padding).*

El vectorizador fue adaptado (generación del vocabulario) con los tweets procesados del dataset de entrenamiento, y fue usado para vectorizar los tweets del dataset de entrenamiento y el dataset de validación.

4.3 Creación del modelo (NN architecture)

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, None, 32)          6400032

 bidirectional_11 (Bidirecti  (None, 64)               16640
 onal)

 dense_45 (Dense)            (None, 128)               8320

 dense_46 (Dense)            (None, 256)               33024

 dense_47 (Dense)            (None, 128)               32896

 dense_48 (Dense)            (None, 7)                 903

=================================================================
Total params: 6,491,815
Trainable params: 6,491,815
Non-trainable params: 0
```



El modelo consiste en una capa E*mbedding* que se encarga de otorgarle características a cada palabra del vector de entrada. En este caso, se le otorgan 32 características a cada palabra (ya vectorizada) para un mejor entendimiento de la naturaleza de esta. De esta forma, el modelo será capaz de reconocer y asociar los vectores de características (vectores correspondientes al *embedding* de cada palabra) con los labels objetivo del problema.

El número de parámetros de esta capa equivale a las 200000 palabras del vocabulario + 1 (para encasillar palabras desconocidas) por el número de características asignadas a cada palabra por el *embedding layer*.

Luego se agregó la capa de nuestra aplicación, la capa LSTM bidireccional preparada para recibir entradas de vectores de características de 32 elementos. La función de activación usada fue ‘tanh’ debido a los requerimientos de tensorflow para trabajar con GPUs.

Luego se agregaron capas densas totalmente conectadas con funciones de activación ‘ReLU’, una de 128 unidades, otra de 256 unidades y otra de 128 unidades.

Finalmente, se agregó otra capa densa de 7 unidades con función de activación sigmoide la cual genera una salida para el modelo similar a una distribución probabilística.

El modelo fue entrenado con la función de pérdida *BinaryCrossentropy*, el optimizador *Adam* y como métrica de evaluación *binary accuracy*.

5. Resultados

Gracias a la función *early stopping* el modelo detuvo el entrenamiento en el epoch 7 para evitar el overfitting. La mejor pérdida para el dataset de validación fue de 0,2788 y la mejor *accuracy* de 89%.

```
Best Validation Loss: 0.2788
Best Validation Accuracy: 0.8900
```