ARTÍCULO DE INVESTIGACIÓN ORIGINAL

Revista de investigación de sistemas e informática
14(1): 69-81 (2021)
DOI: https://doi.org/10.15381/risi.v14i1.21866

ISSN: 1815-0268 (Impreso) / ISSN: 1816-3823 (Electrónico)
Facultad de Ingeniería de Sistemas e Informática - UNMSM

Análisis de sentimientos en la predicción de resultados de elecciones presidenciales

Sentiment analysis in the prediction of results of presidential election

Vieri E. Garcia-Moreno
https://orcid.org/0000-0002-2338-9456
vieri.garcia@unmsm.edu.pe
Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática. Lima, Perú

Cristhian R. Alvarez-Caicedo
https://orcid.org/0000-0003-2608-4268
cristhian.alvarez1@unmsm.edu.pe
Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática. Lima, Perú

Nataly G. Vásquez-Saenz
https://orcid.org/0000-0003-3807-9685
nataly.vasquez1@unmsm.edu.pe
Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática. Lima, Perú

RECIBIDO: 02/07/2021 - ACEPTADO: 08/08/2021 - PUBLICADO: 28/12/2021

RESUMEN

El presente trabajo tiene por finalidad diseñar un modelo basado en las redes neuronales LSTM y aplicarlo en un caso análisis de sentimientos que pretende predecir los resultados electorales del proceso electoral de los Estados Unidos del año 2016 basándose en el procesamiento de un número significativo de tweets que fueron obtenidos de un Dataset público proporcionado por la plataforma Kaggle. Previamente al diseño, se revisan los antecedentes del problema, métodos existentes y los fundamentos en los que se basa la red LSTM, posteriormente se describe la metodología del proceso de análisis de sentimientos que es aplicado para desarrollar el estudio mencionando detalles en la recolección de datos, el preprocesamiento, la elección del clasificador de polaridad, entre otras etapas para finalmente dar lugar a la implementación, entrenamiento y validación de la red neuronal usando las librerías Tensor Flow y Keras, adicionalmente AxClient para ajustar correctamente los hiperparámetros de la misma en cuya mejor iteración obtuvo una precisión binaria de 98.02% luego de ser entrenada durante 2 épocas utilizando 700000 ejemplos.

Palabras clave: Red Neuronal LSTM; Análisis de Sentimientos; Red Neuronal Recurrente; Tensor Flow; Elecciones Presidenciales.

ABSTRACT

The present work aims to design a model based on the LSTM neural networks and apply it in a sentiment analysis case that aims to predict the electoral results of the 2016 United States electoral process based on the processing of a significant number of tweets that They were obtained from a public Dataset provided by the Kaggle platform. Prior to the design, the antecedents of the problem, existing methods and the foundations on which the LSTM network is based are reviewed, later the methodology of the sentiment analysis process that is applied to develop the study is described, mentioning details in the data collection , the preprocessing, the choice of the polarity classifier, among other stages to finally lead to the implementation, training and validation of the neural network using the Tensor Flow and Keras libraries, additionally AxClient to correctly adjust the hyperparameters of the same in whose best Iteration obtained a binary precision of 98.02% after being trained for 2 epochs using 700000 examples.

Keywords: LSTM Neural Network; Sentiment Analysis; Recurrent Neural Network; Tensor Flow; Presidential Elections.

© Los autores. Este artículo es publicado por la Revista de investigación de sistemas e informática de la Facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional Mayor de San Marcos. Este es un artículo de acceso abierto, distribuido bajo los términos de la licencia Creative Commons Atribución 4.0 Internacional (CC BY 4.0) [https://creativecommons.org/licenses/by/4.0/deed.es] que permite el uso, distribución y reproducción en cualquier medio, siempre que la obra original sea debidamente citada de su fuente original. Para mas información, por favor póngase en contacto con revistarisi.fisi@unmsm.edu.pe

I. INTRODUCCIÓN

En la sociedad actual las redes sociales son uno de los principales medios de comunicación. En estas plataformas un número creciente de personas, expresan sus opiniones y puntos de vista de forma libre, respecto a diversas temáticas incluidas las políticas.

Dicha opinión pública adquiere relevancia en el contexto de los procesos electorales, si los grupos políticos involucrados en el proceso electoral identificaran el estado anímico del público respecto a ellos, sus conclusiones les permitirían ajustar sus estrategias o tomar decisiones más acertadas que si lo hicieran por métodos tradicionales.

Entonces, una vez clarificado la esencialidad de las redes sociales en los procesos electorales la cuestión sería cómo aprovechar la información de estos sitios, ante esto el análisis de sentimientos se ha mostrado en diversos estudios como una herramienta bastante precisa para este cometido; sin embargo, presenta aún ciertas limitaciones con las que se está lidiando, principalmente en la recolección y preprocesamiento de información, donde Chauhan et al. (2020)menciona que para mejorar la predicción e interpretar resultados verdaderos se debe procesar datos válidos por lo que se debe detectar cuentas bot, spam y mensajes sarcásticos.

Debido al gran éxito que tiene el análisis de sentimientos, el presente estudio tiene por objetivo desarrollar un modelo capaz de predecir los resultados electorales de determinado proceso electoral aplicando un modelo de red neuronal LSTM con la finalidad de comprobar su efectividad.

II. MARCO TEÓRICO

2.1. Antecedentes

Una de las redes sociales más populares para networking y microblogging es Twitter, donde los usuarios pueden postear mensajes de una longitud específica (280 caracteres) denominados tweets, los cuales pueden ser respondidos, reenviados, etc. (Ansari et al., 2020)

Las redes sociales están cada vez más cerca de las personas, a modo de ejemplo de su influencia, de los 193 estados miembros de las Naciones Unidas, el 97% tiene una cuenta de Twitter. Asimismo, de la población de los Estados Unidos, el 71% considera Twitter como una fuente de noticias relevante. (Chauhan et al., 2020)

La influencia de las redes sociales es tal, que durante las elecciones del 2016 de los Estados Unidos se abrían debates a través de Twitter, incluso en algunas referencias como Bovet y Makse (2019) se menciona que se hacían publicaciones con cuentas falsas y páginas falsas para influenciar la forma en la que se llevarían las elecciones. Como menciona Grover et al. (2017), una de las razones de la polarización de los votos en las elecciones de Estados Unidos del 2016 fueron los tweets en Twitter

Es bajo este contexto, que podemos utilizar la información proveniente de estas redes sociales, en específico Twitter, para, durante un periodo de elecciones, realizar una predicción de las preferencias de los votantes por los candidatos. La realización de esta predicción se sustenta en la importancia de las redes sociales como medio de comunicación y difusión, el bajo coste del desarrollo de una propuesta de este tipo y el acceso a una población más variada y menos limitada. (Chauhan et al., 2020)

Para poder llevar a cabo esta predicción, el proceso fundamental es la clasificación de los sentimientos de un determinado tweet, este proceso puede ser realizado por diversas técnicas, el objetivo es analizar la positividad o negatividad de la opinión de los usuarios (tweets); para esto existen tres enfoques según menciona Chauhan et al. (2020), el enfoque de machine learning, el enfoque basado en léxico y el enfoque híbrido que combina los dos anteriores.

2.2. Métodos existentes

De entre los enfoques mencionados en una relación de resultados y complejidad, las que dan mejores resultados son las basadas en Deep Learning, entre los métodos de este enfoque tenemos (Ansari et al., 2020):

- *Máquina de Vectores de Soporte (Support Vector Machine)*
- *Árboles de Decisión (Decision Tree Clasificator)*
- *Regresión Logística (Logistic Regression)*
- *Clasificador de bosque aleatorio (Random Forest Classifier)*
- *Long Short Term Memory (LSTM)*

La técnica escogida para desarrollar este trabajo es la LSTM, de entre las técnicas de aprendizaje profundo es la que obtiene mejores resultados para el problema dado, según Bilal et al. (2018) y Ansari et al. (2020), con una precisión de hasta 85%.

2.3. Fundamentos de las Redes Neuronales LSTM

2.3.1. Origen

En los últimos años, según un estudio de Yu et al. (2019), las técnicas de aprendizaje profundo se han desarrollado y adoptado para extraer información de diversos tipos de datos, en base a las características de estos han surgido arquitecturas de aprendizaje profundo que pueden procesarlos de la mejor manera, tales como las redes neuronales recurrentes (RNN), convolucionales (CNN) y las profundas (DNN).

Las redes neuronales recurrentes estándar satisfacen la necesidad que se tiene de procesar datos secuenciales como texto, audio y video gracias a su característica típica, ser una conexión cíclica. Esto hace posible a esta arquitectura la actualización de su estado actual basándose en estados pasados y los datos de entrada actuales; sin embargo, presentan ciertos problemas cuando la brecha entre los datos de entrada relevantes es grande, de acuerdo con Torres (2019), este fenómeno se denomina la desaparición y la explosión de gradiente.

Afortunadamente, tal como lo menciona Yang et al. (2020) en su investigación, el inconveniente que presentaban las redes neuronales tradicionales fue solucionado mediante la introducción de las redes LSTM.

2.3.2. Arquitectura

Antes de describir la topología de la red LSTM, es prudente describir la topología de una red neuronal recurrente estándar simple, pues la red mencionada anteriormente es una variación de este modelo inicial.

De acuerdo con la investigación de Pérez Sanjuán (2019), una red recurrente es aquella que está conformada por neuronas recurrentes, estas a diferencia de las neuronas feedforward, presentan un bucle de retroalimentación, de modo que en cada instante de tiempo esta neurona recibe la entrada perteneciente a dicho instante (x_t) y su salida o estado oculto en un instante de tiempo anterior (s_{t-1}), de modo que la información adquiere carácter persistente lo hace propicio a este tipo de arquitectura procesar secuencias.

La red neuronal recurrente se "desdobla" en función de los instantes de tiempo, tal como se observa en la Figura 1.

De la figura mostrada, se puede observar que en lugar de una matriz de pesos se tienen dos: w_I y w_T, una asociada a las entradas de la red y otra a los estados ocultos. Ambas se utilizan en todos los instantes de tiempo t, esto se debe a que por cada instante de tiempo t no se crea una nueva matriz de pesos.

Respecto a la suma ponderada que suele utilizarse como regla de propagación, sufre una ligera modificación teniendo en cuenta ahora las matrices indicadas, quedando la expresión de la siguiente manera:

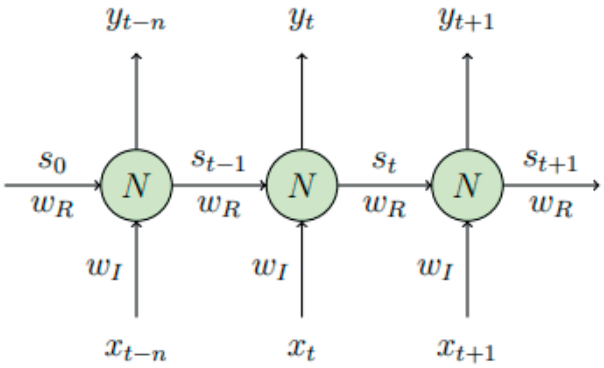
$$s_t = w_I . x_t + w_R . s_{t-1} + \Theta_t$$

Mientras la salida final de la red corresponde a la siguiente expresión, donde σ corresponde a la función de activación de la neurona.

$$y_t = \sigma\,(w_I . x_t + w_R . s_{t-1} + \Theta_t)$$

Una vez descrita la arquitectura de la red neuronal recurrente estándar se procede a describir la de la LSTM, que básicamente al igual que las RNN se componen de una secuencia de neuronas conectadas, pero a diferencia de tener como unidades

Figura 1
Red Neuronal Recurrente



neuronas recurrentes estándar cuenta con las denominadas celdas LSTM. A continuación, la Figura 2 presenta su estructura y se detalla el proceso interno que posibilita lo anteriormente mencionado.

Por cada instante de tiempo t, las celdas LSTM procesan un conjunto de vectores:

1. Una puerta para descartar:

$$f_t = \sigma\,(W_f \times x_t + U_f . h_{t-1} + b_f)$$

2. Una puerta para entrada:

$$i_t = \sigma\,(W_i \times x_t + U_i . h_{t-1} + b_i)$$

3. Un vector de nuevos candidatos para el estado de la unidad:

$$\tilde{C}_t = tanh\,(W_C \times x_t + U_C . h_{t-1} + b_C)$$

4. Una puerta de salida:

$$o_t = \sigma\,(W_O \times x_t + U_O . h_{t-1} + b_O)$$

5. La memoria de la unidad:

$$C_t = i_t \times \tilde{C}_t + f_t \times C_{t-1}$$

6. Una capa de salida oculta:

$$h_t = o_t \times tanh\,(C_t)\tilde{C}$$

Donde $W_f, U_f, W_i, U_i, W_c, U_c, W_o, U_o$ son matrices de pesos y b_i, b_f, b_c, b_o vectores de sesgos.

El proceso interno que se realiza en estas unidades es el siguiente:

1. La unidad decide la información que será olvidada o descartada, para ello calcula f_t. Esto produce que se cree un número entre 0 y 1. Para cada número en el estado de la celda C_{t-1}, donde 1 significa "mantener completamente" y 0, "olvidar completamente".

2. Posteriormente, la celda decide qué información debe almacenar en su memoria, para ello utiliza $\tilde{C}_t$ y i_t.

3. Luego, la unidad actualiza el estado antiguo C_{t-1} con el nuevo C_t, para ello computa la expresión de C_t.

4. Finalmente, se determina la salida h_t en función de una versión filtrada del estado de la celda.

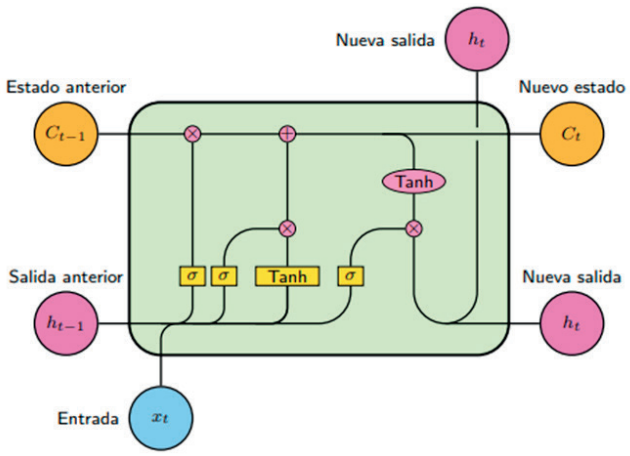
2.3.3. Aplicaciones

De acuerdo con Yu et al. (2019) aparte del análisis de sentimientos, gran parte de los resultados más emocionantes y prometedores alcanzados por las redes neuronales recurrentes se han logrado con las LSTM, resultados en tareas como reconocimiento de voz, modelado acústico, predicción de trayectorias, inserción de oraciones, análisis de correlación, entre otras.

III. MÉTODOS

Debido a la naturaleza del caso de investigación, para su desarrollo se tomó en consideración el proceso de análisis de sentimientos descrito por Chauhan et al. (2020) en su estudio. De acuerdo con el autor mencionado, dicho proceso puede ser visto como un conjunto de pasos de minería de datos que hacen uso de múltiples disciplinas como

Figura 2
Celda LSTM



la extracción de información, el procesamiento del lenguaje natural, las estadísticas y el aprendizaje automático. El proceso paso a paso del análisis de sentimientos se muestra en la Figura 3.

Como se puede observar en la Figura 3, el proceso está compuesto por fases como: recolección de datos, preprocesamiento de datos, identificación de sentimientos, selección y extracción de características, clasificación de sentimientos y reporte de polaridad. Las dos primeras fases se enfocan en obtener los datos que se procesarán esto mediante API's expuestas por las redes sociales o repositorios de datos y posteriormente limpiar dicha data removiendo todo el ruido posible, las siguientes tres fases involucran el procesamiento de los datos donde se encuentras las palabras o frases que revelan el sentimiento e identifica la objetividad o subjetividad de la opinión, se extraen las características y finalmente se clasifica el sentimiento. Adicionalmente, el autor menciona que en enfoques de análisis de sentimientos con aprendizaje profundo los pasos 3 y 4 los hace el modelo de red neuronal durante su fase de entrenamiento mientras el paso 5 lo realiza luego de haber sido entrenado y validado.

3.1. Recolección de datos

Para el desarrollo del presente estudio se tomó como fuente de datos un repositorio de datos proporcionado por la plataforma Kaggle (2016 USA Presidential Election Tweets-61M Rows, s. f.), el cual contiene 61 millones de tweets recopilados entre el 30 de agosto de 2016 al 28 de febrero del 2017, en el contexto de las elecciones presidenciales de los Estados Unidos de América del año 2016.

Dicho repositorio complementa la información propiamente del tweet con otros 17 metadatos asociados como el identificador del mismo, la fecha de creación, el identificador de la cuenta en donde se originó, entre otro; sin embargo, para esta investigación solo se consideraron los campos que se indican en la Tabla 1.

Figura 3
Proceso de Análisis de sentimientos

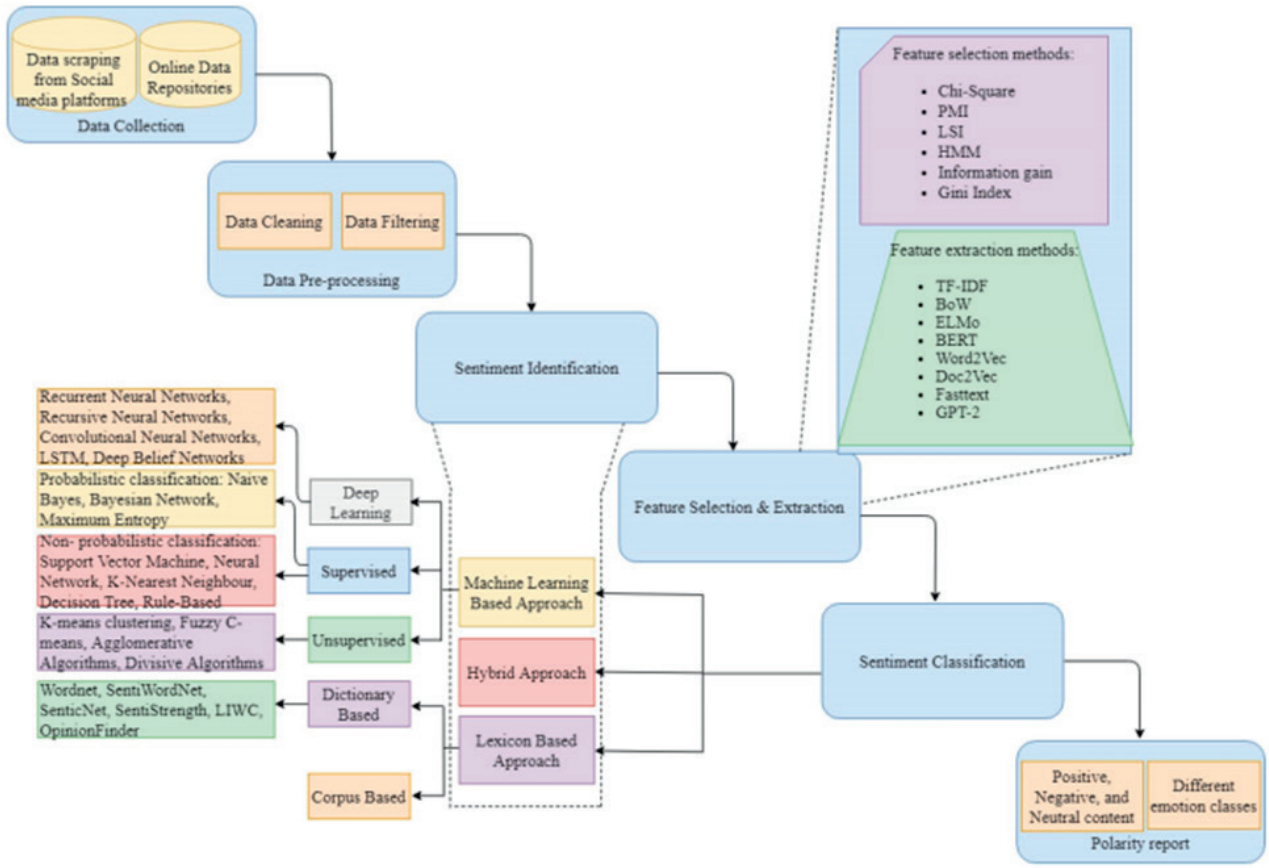


Tabla 1
Metadatos considerados

Metadato	Metadato en español	Descripción	Utilidad
candidate_id	Identificador del candidato	Valor numérico que representa al candidato a quien se hace referencia en el texto del tweet. En este caso para Clinton toma el valor de 1 y para Trump el valor de 2.	Facilita el conteo de tweets positivos y negativos de cada candidato y por ende el cálculo del PvTRatio.
tweet_text	Texto del tweet	Texto publicado en el tweet que representa la opinión de determinada persona respecto a las elecciones presidenciales.	Entrada que la red procesa para calcular su polaridad. Por fines del estudio se trabajó solo con polaridades positivas y negativas asignando valores 0 y 1 respectivamente.
polarity	Polaridad	Grado de positividad o negatividad que contiene el mensaje escrito en el texto del tweet. Oscila entre -1 y 1 que representa la máxima negatividad y positividad respectivamente.	Target o salida esperada que utiliza la red para realizar su aprendizaje.

3.2. Preprocesamiento

Como pasos previos al entrenamiento y en general a la utilización del dataset obtenido, se deben realizar ciertos filtros para reducir información que no será relevante para el análisis.

En primer lugar, partimos del hecho de que se ha removido toda información no relevante para el procesamiento del Dataset, entendiéndose con ello las columnas de datos que no son de utilidad; los retweets, pues estos no contienen ningún texto que refleje opinión y adicionalmente la información de candidatos de los cuales se tienen muy pocos tweets asociados, razón por la cual solo se consideraron Clinton (1) y Trump (2).

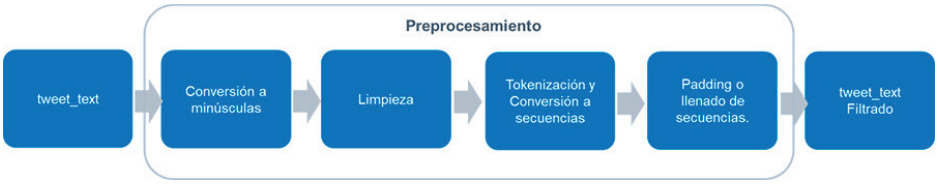
Una vez asegurado las condiciones mencionadas el preprocesamiento toma lugar. Su primera fase, la fase de limpieza, consta de 3 etapas tal y como lo describe la Figura 4.

Posteriormente el preprocesamiento cuenta con 2 fases adicionales, que se realizan para tener la entrada como se le requiere para el entrenamiento y validación, como tal, estos no forman parte de una depuración o filtrado, sino se trata de procesos encargados de la conversión de cada tweet en una secuencia de números, para esto se realiza la conversión respecto de un vocabulario, utilizando el Tokenizer de Keras, a este se le provee de un tamaño de vocabulario, esto se obtiene en un proceso previo que calcula la cantidad de palabras más usadas en los tweets (aquellas que superan las 4 apariciones son consideradas), y permite con esto transformar los tweets en secuencias de números. Adicional a este proceso se tiene uno que se encarga de completar los tweets a una longitud fija, esto para que todas las entradas tengan una longitud específica, para esto se definió como longitud máxima 150 y los tweets son rellenados a esta longitud.

La etapa de preprocesamiento dadas las consideraciones anteriores tendría como entrada finalmente, el conjunto de tweets del dataset, y tendría como salida una secuencia de números de longitud fija (150) que serviría como entrada del modelo para el análisis de sentimientos (Ver Figura 5).

Figura 4
Fase de Limpieza

Figura 5
Preprocesamiento



3.3. Técnica de Clasificación de Sentimiento

Posterior a la recolección y al procesamiento de datos, de acuerdo a la metodología descrita, las siguientes etapas corresponden a la extracción de características y la clasificación de los tweets en base a su polaridad, para poder llevar a cabo dichas etapas existe una variada gama de algoritmos de utilidad, tales como la máquina de soporte de vectores, el árbol de decisión, la regresión logística, las celdas LSTM, entre otros.

En base al estudio realizado por Ansari et al. (2020) de naturaleza similar al presente, donde se pretendía procesar una gran cantidad de tweets usando las técnicas de análisis de sentimientos con mejor desempeño y comparado las precisión de sus predicciones, destaca la arquitectura de red neuronal LSTM, pues de acuerdo con dicha investigación, la técnica en cuestión obtuvo la precisión más alta alcanzando un 75%. En consecuencias a los buenos resultados obtenidos, en el presente trabajo de investigación se utilizará como técnica de extracción de características y clasificación de sentimientos a la red neuronal recurrente de tipo LSTM.

3.3.1. Especificación del Modelo

Topología

El modelo definido para el análisis de sentimientos planteado consta de 4 capas aparte de la entrada, que debe ser una secuencia de números de una longitud definida de 50, esto por las limitaciones de Twitter y una revisión del dataset (preprocesamiento).

Así el modelo cuenta con las entradas y salidas indicadas en la Tabla 2.

Tabla 2

Entradas y salidas del modelo

Entrada	Secuencia de números que representa las palabras de tweet llenada a una longitud definida de 50, esto dadas las limitaciones de Twitter y el número de palabras que encontramos en la data.
Salida	Polaridad del tweet en un rango entre 0 y 1, donde el primero implica una negatividad del comentario y el segundo una positividad respecto del comentario.

Las capas planteadas para el modelo son las siguientes:

- Capa Embedding, donde buscamos reducir del vector de números obtenido del tokenizer como entrada a una dimensión que se utiliza como parámetro de esta capa, en este caso se realizó pruebas con un rango entre 10 y 800.
- Capa LSTM, es el Core del análisis de sentimientos, se encarga de evaluar las relaciones que existen entre las palabras del texto, se consideraron entre 1 y 10 unidades LSTM para las pruebas.
- Capa Densa, corresponde con un mecanismo de atención, hasta cierto punto añade mayor complejidad al modelo, pero permite un mejor análisis, donde se ha visto en otros estudios de PLN que se obtienen buenos resultados al incluirla.
- Capa de Salida, corresponde a la salida del modelo, que considera solo 1 neurona de salida dado que se trata de la polaridad que buscábamos obtener.

Función de Activación

La activación para cada capa se da de distinta manera, las funciones de activación utilizadas son:

Función sigmoide.

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

$$\sigma'(x) = \sigma(x)[1 - \sigma(x)]$$

Función tangente hiperbólico.

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

$$tanh'(x) = 1 - \tanh^{2}(x)$$

Función ReLU.

$$ReLU(x) = \max(0, x)$$

$$ReLU'(x) = \begin{Bmatrix} 0, x < 0 \\ 1, x > 0 \\ 0, x = 0 \end{Bmatrix}$$

Estas funciones son usadas por las diversas capas para calcular la activación, así tenemos:

- Capa LSTM, como se mencionó anteriormente, esta consta de unidades LSTM que internamente utilizan la función sigmoide y tangente hiperbólico para la activación neta y para las compuertas (input, forget, output). Para el cálculo de la entrada neta en la mayoría de los casos es la sumatoria del producto de los pesos con las entradas más el bias que es otro de los parámetros de cada parte de la LSTM.
- Capa Densa, al tratarse de una capa con todos los enlaces, su función de activación es definida como parte del modelo, y como generalmente se hace en problemas de

procesamiento de lenguaje natural se consideró una activación ReLU.

- Capa de Salida, dado que buscamos que la salida de la red sea la polaridad del tweet, es decir, cuan negativo o positivo es respecto del candidato; utilizamos una función sigmoide para determinar este valor en un rango de 0 a 1 donde corresponden al negativo y positivo máximo respectivamente.

Función de Pérdida

En el modelo planteado se ha optado por la entropía cruzada binaria como función de pérdida cuya expresión matemática se muestra a continuación:

$$H_q(q) = -\frac{1}{n}\sum_{i=1}^{n} y_i.\log\big(p(y_i)\big) + (1-y_i).\log\,(1-p(y_i))$$

Donde y es la clase y $p(y)$ es la probabilidad de que la predicción sea la clase y.

La función de entropía cruzada es ampliamente utilizada en tareas de clasificación, pero en este caso debido a que la clasificación es binaria (polaridades 0 o 1) se terminó aplicando la entropía cruzada binaria.

Función de Aprendizaje

Respecto a la función de aprendizaje u optimizador, se eligió el optimizador mini-batch de Adam (Adaptive Moment Estimation), cuya expresión matemática se muestra a continuación:

$$m = \beta_1.m + (1-\beta_1)\Delta W$$

$$v = \beta_2.v + (1-\beta_2)\Delta W^2$$

$$W = W - \frac{\propto.m}{\sqrt{v+\varepsilon}}$$

Donde m y v representan los 2 momentos, siendo m el que modela la media de los gradientes a lo largo del tiempo, mientras que v hace lo mismo con la varianza. β_1 es igual, en la mayoría de los casos a 0.9, mientras que β_2 casi siempre es fijado en 0.99.

La elección de dicho optimizador se justifica en que ha sido utilizado exitosamente en otros estudios de características similares al presente, como el de Bilal et al. (2018), u otros trabajos en el área de visión computacional y procesamiento de leguaje natural. Adicionalmente, de acuerdo con Ullah et al. (2020) Adam tiene una alta eficiencia computacional y demanda poca capacidad de memoria. Este autor también menciona que Adam acepta efectivamente la estimación del primer y segundo momento de gradiente para calcular el tamaño del ajuste e integra efectivamente las fortalezas de otros 2 buenos optimizadores: RMSProp Y AdaGrad.

Ventajas y Desventajas del método

Las LSTM frente a las redes neuronales recurrentes (RNN), permiten un mejor análisis del contexto de un texto, esto dado que consideran la secuencialidad de las palabras que lo conforman utilizando algo similar a la memoria. Sin embargo, esta secuencialidad es lineal, y se analiza de principio a fin del texto sin una posibilidad de interpretarla en un sentido inverso, si bien una LSTM podrá “predecir” palabras futuras, no considera las que se ubican después como una entrada directa, y esto, es un problema que se ejemplifica al utilizar modificadores.

El problema de las LSTM se abordó con las Bi-LSTM, que constan de dos capas que se ejecutan en ambos sentidos para considerar tanto el orden secuencial de inicio a fin como el de fin a inicio, sin embargo, si el costo computacional de las LSTM ya era alto, se hace aún mayor si utilizamos las Bi-LSTM. Es por esto, que en este estudio se decidió utilizar las LSTM frente a las Bi-LSTM, si bien las ultimas darían mejores resultados, los obtenidos por las LSTM ya son lo suficientemente representativos con 0.85 de precisión.

Respecto del desarrollo del modelo se tuvieron ciertas dificultades propias del tipo de estudio, la elección de la arquitectura del modelo (capas) se realizó siguiendo estudios similares, sin embargo, estos no detallaban la implementación de estas lo que nos dificultó la comprensión. Por otro lado, luego de la elección del modelo la elección de hiperparámetros a utilizar se tuvo una dificultad similar, dado que la variedad de estudios de este tipo y la cantidad de data con la que se tenía que probar hizo que fuera complicado ir cambiando los parámetros y viendo los resultados; bajo esto, se intentó utilizar un optimizador de estos hiperparámetros a modo de ayuda, se utilizó AxClient configurando el rango de variación de los hiperparámetros pero la ejecución de este para la cantidad de data que se estaba manejando tomó demasiado tiempo dificultando la obtención de resultados.

IV. EXPERIMENTOS NUMÉRICOS

4.1. Descripción del Sistema

Para el desarrollo de este trabajo de investigación, se utilizó el lenguaje Python con las librerías Pandas,

Tensorflow, Keras; donde, para el procesamiento de la data se utilizó Pandas, mientras que para el desarrollo del modelo de red neuronal planteado se utilizó Keras.

Para la validación del modelo planteado se utiliza un 20% del dataset filtrado para cada candidato, el restante se utiliza para el entrenamiento. La salida del modelo planteado es la polaridad esperada de cada tweet y en función de la positividad obtenida de esta se puede ver la probabilidad de ganar de un candidato que es la salida final esperada.

4.2. Requisitos Mínimos

Los experimentos computacionales del presente trabajo de investigación fueron realizados utilizando un cuaderno computacional en la nube proporcionado por la plataforma Kaggle, cuyas especificaciones son las siguientes:

- NVIDIA TESLA P100 GPU de 2 núcleos con 16GB RAM
- CPU de 4 núcleos con 16 GB de RAM
- TPU de 4 núcleos con 16 GB de RAM.
- Disco HDD de 19,6 GB.

4.3. Instancias de Prueba

Se utilizó para las pruebas el dataset proporcionado por la plataforma Kaggle (2016 USA Presidential Election Tweets-61M Rows, s. f.), que como se menciona en la etapa de recolección de datos, contiene data en el contexto de las elecciones presidenciales de los Estados Unidos de América del año 2016. De la información disponible se está utilizando 350000 registros para cada candidato considerado (Clinton y Trump) por lo que el conjunto de datos consta de 700000 registros, de estos, el 20% corresponde a data de validación y el 80% a data de entrenamiento, quedando con 560000 registros para entrenamiento y 140000 para la validación.

Como se referenció en la etapa de recolección de pruebas, esta data neta considerada para el estudio se consiguió luego de realizar varios filtrados, la data inicial contenía información de otros candidatos no relevantes para el estudio que fueron depurados por ello, así mismo, había valores pico que distorsionaban los resultados obtenidos por lo que también se retiraron de la data. Las polaridades luego de realizado este filtro son como se muestra en la Figura 6. Asimismo, se realizó un ajuste sobre las polaridades para hacerla más acorde al modelo, como se especificó en puntos anteriores.

4.4. Resultados Numéricos

El desarrollo del modelo tuvo varias pruebas, se empezó realizando ajustes manuales a los parámetros del mismo como la tasa de aprendizaje, número de neuronas en cada capa, tamaño del Embedding, tamaño del batch, número de épocas, etc., Luego de un ajuste manual que dio resultados visibles se optó por utilizar un optimizador de hiperparámetros ajustando los rangos de variación en función de los resultados obtenidos, finalmente obtuvimos los siguientes valores para los hiperparámetros de la red de la Tabla 3.

Tabla 3

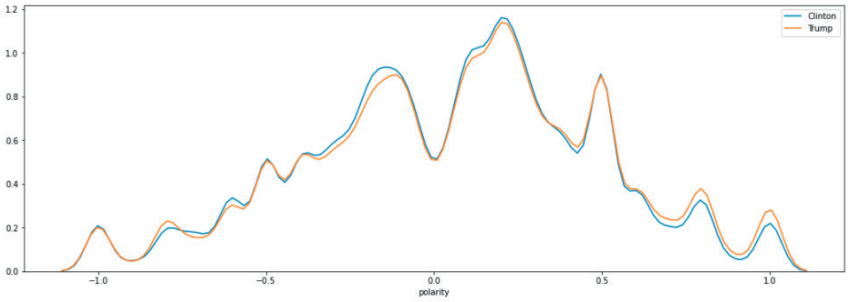
Hiperparámetros del modelo

Hiperparámetros	Valor
Tasa de aprendizaje	0.007169632435246209
Tasa de dropout en la capa de salida	0.04510561357026982
Unidades LSTM	10
Neuronas Densas	18
Número de Épocas	2
Tamaño del Batch	806
Tamaño del Embedding	173
Máximo número de palabras	50

Estos fueron con los que se consiguieron los mejores resultados, para determinar cuan buenos eran los resultados se utilizó la métrica de precisión binaria;

Figura 6

Polaridades filtradas



esta métrica de acuerdo a la página oficial de Keras (Team, s. f.) calcula la frecuencia con la que las predicciones coinciden con etiquetas binarias, dados los pasos especificados en la recolección de datos, se trata del caso planteado.

Finalmente, el resultado obtenido para la métrica especificada es el indicado en la Tabla 4.

Estos valores corresponden con los mejores resultados obtenidos resultado de optimizar con AxClient, mientras que los resultados intermedios obtenidos para llegar a estos son los indicados en la Tabla 5.

La evolución del performance frente al número de iteraciones es como se muestra en la Figura 7.

Tabla 4

Resultados obtenidos respecto a las métricas

Métrica	Valor
Precisión Binaria Entrenamiento	0.9855
Precisión Binaria Validación	0.9802

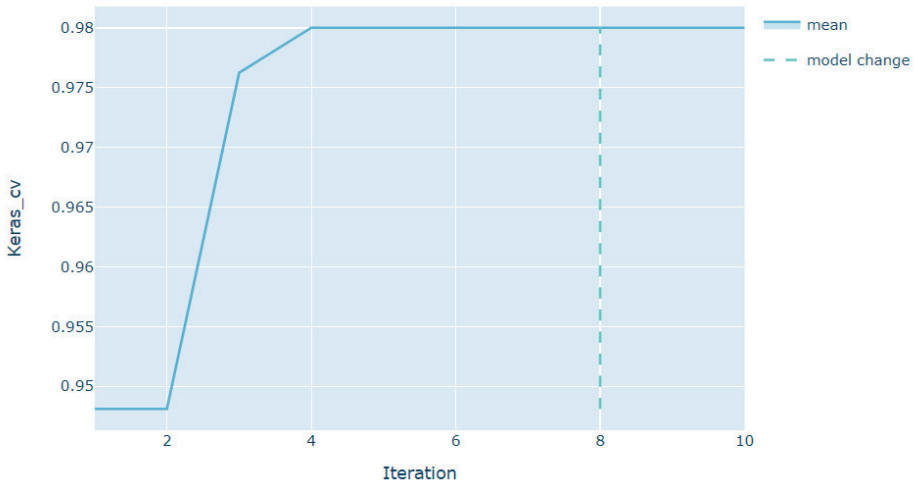
Tabla 5

Optimización de Hiperparámetros AxClient

keras_cv	trial_index	arm_name	learning_rate	dropout_rate	lstm_units	neurons_dense	num_epochs	batch_size	embedding_size
0.948100	2	2_0	0.022628	0.020568	5	10	1	894	441
0.563386	3	3_0	0.278257	0.015930	2	3	1	862	645
0.976257	4	4_0	0.006078	0.015087	8	9	6	848	323
0.980021	5	5_0	0.007170	0.045106	10	18	2	806	173
0.977114	6	6_0	0.000598	0.203325	10	20	7	812	329
0.977786	7	7_0	0.000850	0.085168	10	5	3	775	287
0.979464	8	8_0	0.001609	0.062056	10	13	5	817	262
0.977386	9	9_0	0.002921	0.043281	9	12	3	822	256
0.978686	10	10_0	0.001121	0.136968	10	15	3	785	250
0.978021	11	11_0	0.003018	0.066256	8	18	4	826	300

Figura 7

Performance vs Iteraciones



A modo de complemento, podemos observar también la evolución de la métrica precisión binaria (Figura 8) y del error (Figura 9) para el mejor modelo.

Figura 8

Gráfica de evolución de la precisión binaria

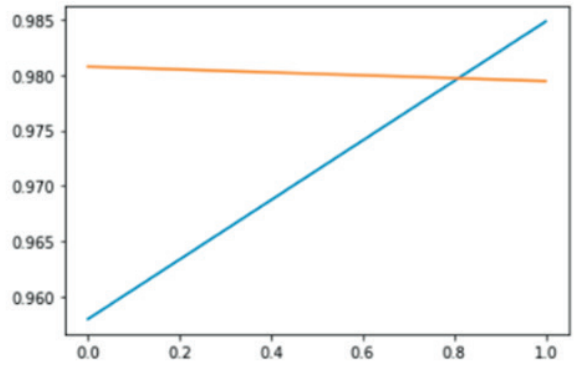
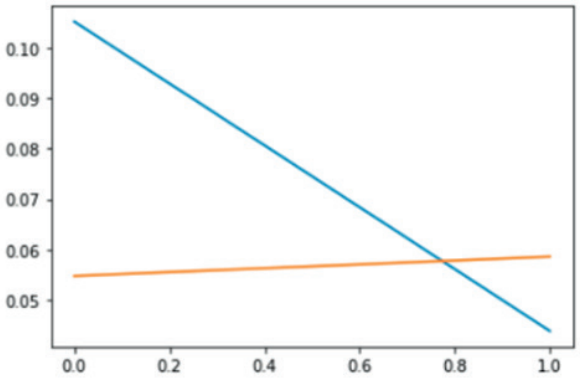


Figura 9

Gráfica de la evolución del error



V. ANÁLISIS DE RESULTADOS

Sobre el análisis de resultados primero resulta importante revisar la evolución de AxClient a través de las iteraciones, donde para ello la Figura 8 muestra precisamente esta evolución, podemos ver que durante las dos primeras iteraciones se mantuvo relativamente constante sin mejoras significativas, donde la mejora se inició a partir de la tercera iteración hasta la cuarta que llega hasta un límite de mejora, los incrementos a partir de la iteración 4 son poco significativos, es por esto que se toman estos parámetros para realizar un entrenamiento adicional y verificar los resultados dados por AxClient.

La gráfica de evolución de la precisión binaria muestra una mejora en el tiempo para la misma, sin embargo, la evolución de su valor para la validación se cruza con el del entrenamiento, esto implica que de limitar ligeramente el número de iteraciones podríamos obtener mejores resultados. Otro indicador de lo mismo es la gráfica de la evolución del error, esta nos muestra que en el tiempo el error ha ido disminuyendo, pero el de validación ligeramente aumentando hasta que se invierten, esto nos lleva a pensar que sería adecuado reducir ligeramente el número de iteraciones para obtener mejores resultados.

Para la validación de los resultados obtenidos, podemos utilizar la relación de positividad respecto del total para cada candidato, esto nos indicaría su preferencia respecto de la población y por tanto serían una aproximación de quien tendría más probabilidades de ser elegido. Para esto tenemos la siguiente fórmula.

$$PvTRatio = \frac{Positive}{Positive + Negative}$$

Donde:

$Positive$, corresponde al número de tweets con un sentimiento positivo hacia el candidato.

$Negative$, corresponde al número de tweets con un sentimiento negativo hacia el candidato.

Los resultados obtenidos siguiendo esta fórmula son los indicados en la Tabla 6.

Tabla 6

Ratio de ganancia por candidato predicho

Candidato	Positive	Negative	Ratio de ganancia
Trump	39820	30074	0.56972
Clinton	38850	31256	0.55416

Mientras que la aplicación de la fórmula sobre la información del dataset se obtiene lo indicado en la Tabla 7.

Tabla 7

Ratio de ganancia por candidato real

Candidato	Positive	Negative	Ratio de ganancia
Trump	40177	29717	0.57483
Clinton	39241	39865	0.55974

A partir de las tablas Tabla 6, Tabla 7 y de las preferencias de las elecciones de USA del 2016, podemos decir que nuestra aproximación es correcta.

Debemos considerar que los resultados obtenidos se han hecho con un dataset en inglés y

considerando la eliminación de caracteres especiales no reconocibles como caracteres especiales, emoticons, urls, etc., por lo que, la aplicación sobre un caso en otro idioma requeriría que la red sea entrenada con un dataset representativo de este tipo.

VI. CONCLUSIONES

En este punto planteamos algunas de las conclusiones a las que hemos llegado a través del desarrollo de este trabajo tanto en la investigación teórica como en la aplicación práctica de los conceptos adquiridos.

El análisis de sentimientos es un campo de estudio que está cobrando relevancia los últimos años y las técnicas que se utilizan para darle solución son variadas y de alta complejidad.

Las redes neuronales profundas son una de las técnicas que da buenos resultados comparada con otras en el análisis de sentimientos.

La definición de un modelo para análisis de sentimientos como el planteado en este trabajo requiere de la comprensión de las funciones de cada capa a utilizar y de qué maneras estas influyen en los resultados obtenidos, así como para parametrizar adecuadamente tanto las entradas como las salidas.

Conseguir los resultados esperados para un modelo de red neuronal puede ser un proceso largo y tedioso si no se cuenta con recursos computacionales adecuados.

El encontrar los hiperparámetros adecuados para un modelo de red neuronal toma bastante tiempo y esfuerzo, se puede tomar como referencia algunos optimizadores de estos para reducir esfuerzos.

Si bien los optimizadores como AxClient son un buen apoyo para el tuning de los hiperparámetros realizar pruebas adicionales para verificar que los parámetros sean correctos permite tener mayor comprensión de la funcionalidad que cumple cada capa del modelo.

Utilizar una red Bi-LSTM en lugar de una LSTM daría mayores posibilidades tanto para el análisis del contexto como para una interpretación en distintos idiomas, esto junto con la inclusión de algunas capas como una CNN luego de la Embedding podrían mejorar los resultados obtenidos. (Grover et al., 2017; Yang et al., 2020)

VII. REFERENCIAS BIBLIOGRÁFICAS

[1] P. Chauhan, N. Sharma, y G. Sikka, «The emergence of social media data and sentiment analysis in election prediction», *J. Ambient Intell. Humaniz. Comput.*, ago. 2020, doi: 10.1007/s12652-020-02423-y.

[2] Ansari, M. Z., Aziz, M. B., Siddiqui, M. O., Mehra, H., & Singh, K. P. (2020). Analysis of Political Sentiment Orientations on Twitter. *Procedia Computer Science*, *167*, 1821-1828. https://doi.org/10.1016/j.procs.2020.03.201

[3] Bovet, A., & Makse, H. A. (2019). Influence of fake news in Twitter during the 2016 US presidential election. *Nature Communications*, *10*(1), 7. https://doi.org/10.1038/s41467-018-07761-2

[4] Grover, P., Kar, A. K., Dwivedi, Y. K., & Janssen, M. (2017). The Untold Story of USA Presidential Elections in 2016—Insights from Twitter Analytics. En A. K. Kar, P. V. Ilavarasan, M. P. Gupta, Y. K. Dwivedi, M. Mäntymäki, M. Janssen, A. Simintiras, & S. Al-Sharhan (Eds.), *Digital Nations – Smart Cities, Innovation, and Sustainability* (pp. 339-350). Springer International Publishing. https://doi.org/10.1007/978-3-319-68557-1_30

[5] Bilal, M., Asif, S., Yousuf, S., & Afzal, U. (2018). 2018 Pakistan General Election: Understanding the Predictive Power of Social Media. *2018 12th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*, 1-6. https://doi.org/10.1109/MACS.2018.8628445

[6] Torres, J. (2019, septiembre 22). Redes Neuronales Recurrentes. *Jordi TORRES.AI*. https://torres.ai/redes-neuronales-recurrentes/

[7] Yang, S., Yu, X., & Zhou, Y. (2020). LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example. *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, 98-101. https://doi.org/10.1109/IWECAI50956.2020.00027

[8] Pérez Sanjuán, A. (2019). *Aplicación para el análisis de sentimientos y tendencias en redes sociales* [Proyecto/Trabajo fin de carrera/grado, Universitat Politècnica de València]. https://riunet.upv.es/handle/10251/126316

[9] Yu, Y., Si, X., Hu, C., & Zhang, J. (2019a). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, *31*(7), 1235-1270. https://doi.org/10.1162/neco_a_01199

Fuentes de financiamiento:

Propias.

Conflictos de interés:

Los autores declaran no tener conflictos de interés.