UNIVERSIDAD NACIONAL DE SAN MARTIN

RCSI
https://revistas.unsm.edu.pe/index.php/rcsi
e-ISSN: 2709-992X

Artículo original
Original article
Ene-Jun, 2023

Análisis de sentimientos en Twitter: Un estudio comparativo

Sentiment analysis in Twitter: A comparative study

Lovera, Fernando Andres[1*]
Cardinale, Yudith[1]

[1]Department of Computer Science, Universidad Simón Bolívar, Caracas, Venezuela

Recibido: 4 Ago. 2022 | **Aceptado:** 16 Oct. 2022 | **Publicado:** 20 Ene. 2023
Autor de correspondencia*: flovera@usb.ve
Cómo citar este artículo: Lovera, F. A. & Cardinale, Y. (2023). Análisis de sentimientos en Twitter: Un estudio comparativo. *Revista Científica de Sistemas e Informática, 3*(1), e418. https://doi.org/10.51252/rcsi.v3i1.418

RESUMEN

El análisis de sentimientos ayuda a determinar la percepción de usuarios en diferentes aspectos de la vida cotidiana, como preferencias de productos en el mercado, nivel de confianza de los usuarios en ambientes de trabajo, o preferencias políticas. La idea es predecir tendencias o preferencias basados en sentimientos. En este artículo evaluamos las técnicas más comunes usadas para este tipo de análisis, considerando técnicas de aprendizaje de máquina y aprendizaje de máquina profundo. Nuestra contribución principal se basa en una propuesta de una estrategia metodológica que abarca las fases de preprocesamiento de datos, construcción de modelos predictivos y su evaluación. De los resultados, el mejor modelo clásico fue SVM, con 78% de precisión, y 79% de métrica F1 (F1 score). Para los modelos de Deep Learning, con mejores resultados fueron los modelos clásicos. El modelo con mejor desempeño fue el de Deep Learning Long Short Term Memory (LSTM), alcanzando un 88% de precisión y 89% de métrica F1. El peor de los modelos de Deep Learning fue el CNN, con 77% de precisión como de métrica F1. Concluyendo que, el algoritmo Long Short Term Memory (LSTM) demostró ser el mejor rendimiento, alcanzando hasta un 89% de precisión.

Palabras clave: análisis de sentimiento; aprendizaje; clasificación; twitter

ABSTRACT

Sentiment analysis helps to determine the perception of users in different aspects of daily life, such as product preferences in the market, level of user confidence in work environments, or political preferences. The idea is to predict trends or preferences based on feelings. In this article we evaluate the most common techniques used for this type of analysis, considering machine learning and deep machine learning techniques. Our main contribution is based on a proposal for a methodological strategy that covers the phases of data preprocessing, construction of predictive models and their evaluation. From the results, the best classical model was SVM, with 78% accuracy, and 79% F1 metric (F1 score). For the Deep Learning models, the classical models had the best results. The model with the best performance was the Deep Learning Long Short Term Memory (LSTM), reaching 88% accuracy and 89% F1 metric. The worst of the Deep Learning models was the CNN, with 77% accuracy as an F1 metric. Concluding that the Long Short Term Memory (LSTM) algorithm proved to be the best performance, reaching up to 89% accuracy.

Keywords: sentiment analysis; learning; classification; twitter

1. INTRODUCCIÓN

Los usuarios de Twitter hacen uso de la plataforma para expresar desde opiniones hasta emociones sobre cualquier tópico. Los modelos de clasificación inteligentes han demostrado su capacidad de predicción de sentimientos en textos, para determinar la percepción de los usuarios sobre aspectos de la vida cotidiana (Mostafa, 2013), como pueden ser: compras de productos en el mercado o incluso gustos políticos. La información extraída por análisis de sentimientos se puede usar como conocimiento para análisis posteriores. En general, se quiere predecir los resultados de preferencias o tendencias de un tópico particular a partir del sentimiento (Li et al., 2022).

Surge entonces la pregunta sobre cuál es la mejor técnica de análisis de sentimientos en textos. En este artículo evaluamos las técnicas usadas más comunes para detectar sentimientos en textos de poca longitud, específicamente en tuits, cuya longitud es de 280 caracteres. El objetivo es evaluar técnicas tanto de modelos inteligentes de Machine Learning, como Regresión Logística, Naive Bayes y Support Vector Machine (SVM), como de Deep Learning, como Convolutional Neural Network (CNN), Long Short Term Memory (LSTM) y Bidirectional Long Short Term Memory (Bi-LSTM). El conjunto de datos (dataset) utilizado, para la evaluación de tales técnicas es el de Sentiment140 que contiene 1,600,000 tuits en Ingles etiquetados como positivo, negativo y neutral, así como metadatos que describen cada Tuit.

Para efectos del análisis de sentimientos, sólo es necesario el contenido del texto del Tuit junto con su etiqueta (sin embargo, en este trabajo no se considera la etiqueta neutral). Para realizar el análisis de sentimientos correctamente, es necesario realizar una limpieza del texto, que incluye desde eliminación de caracteres no alfanuméricos (esto deja a los emoticones de lado, pero no afectara los resultados, ya que la forma en que fue etiquetado el dataset toma en cuenta los emoticones) hasta corrección de errores ortográficos. También es importante realizar una exploración de datos, que ayude a visualizar estadísticas referentes a la distribución del dataset. Dichas estadísticas se utilizan para conocer aspectos relevantes del dataset (como balance de datos) y poder entrenar adecuadamente los modelos inteligentes. Así, adicionalmente en este artículo, proponemos un enfoque metodológico que incluye las fases de preprocesamiento de datos (basado en Natural Language Processing – NLP), construcción de modelos inteligentes de predicción y evaluación comparativa para identificar cuál de los modelos presenta mayor precisión para predecir el sentimiento en textos tuits.

2. MATERIALES Y MÉTODOS

Esta sección trata sobre la metodología que fue empleada en nuestro estudio. Planteamos una sección en la que exponemos una estrategia metodológica general para realizar análisis de sentimientos y luego realizamos nuestro estudio comparativo basados en esta metodología. Esta sección consta de dos subsecciones: Estrategia metodológica y Estudio comparativo.

2.1. Estrategia metodológica

Para realizar el estudio comparativo de diferentes técnicas de aprendizaje, proponemos una estrategia metodológica que abarca desde la fase de extracción del dataset hasta la evaluación comparativa. En la Figura 1, se muestra el esquema general de nuestra estrategia metodológica, sus fases y las actividades consideradas en cada una. En las primeras dos fases se utilizan técnicas de NLP para normalizar el texto. Esta normalización sirve para tener una representación regular del texto, que será una entrada adecuada a los algoritmos inteligentes para asegurar un proceso de aprendizaje correcto. A continuación, se detalla cada fase de la estrategia.

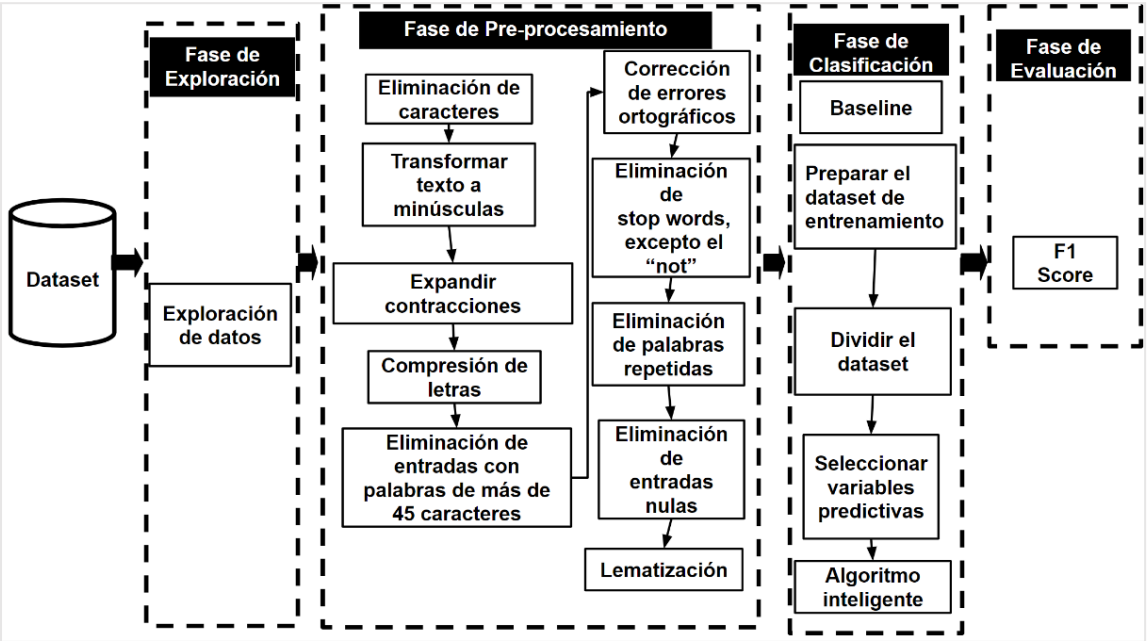

Figura 1. *Esquema general del enfoque metodológico propuesto*

Fase 1: Fase de Exploración de datos

El conjunto de datos (dataset) es la entrada que será procesada en cada fase del enfoque metodológico propuesto. La exploración de datos representa la fase inicial y consiste en elaborar un conjunto de medidas y gráficos que dan información sobre el dataset a utilizar. Es una de las etapas que mayor tiempo requiere. En ella se intenta visualizar las características de distribución del dataset para verificar la heterogeneidad y balance de los datos.

En particular, para el caso de análisis de sentimientos en tuits, una gráfica muy importante es la distribución de tuits positivos y negativos. Es importante que haya una cantidad similar de tuits etiquetados como positivos y negativos, de manera que, al entrenar los algoritmos de aprendizaje, sean capaces de aprender a diferenciar entre las dos clases. Entre las actividades que se desarrollan en esta fase, se encuentran: cálculo de frecuencia de bigramas, frecuencia del número de palabras positivas en tuits marcados como positivos, frecuencia del número de palabras negativas en tuits marcados como negativos, frecuencia del número de palabras negativas en tuits marcados como positivos, frecuencia del número de palabras positivas en tuits marcados como negativos. Una técnica para saber si un dataset esta balanceado, es utilizar el conteo de las clases de predicción, siguiendo el principio de entropía de Wu et al. (2011). En un dataset de n instancias, si se tienen k clases de tamaño c_i, la entropía puede calcularse como se muestra en la siguiente ecuación.

$$\mathbf{E}=-\sum_{i=1}^{k}\frac{c_i}{n}\log\frac{c_i}{n}$$

La ecuación anterior es equivalente a log(k) todas las clases son equilibradas, aproximadas al mismo tamaño n/k, o 0 cuando hay una única clase. Por lo que la entropía tiende a 0 cuando el dataset no está balanceado. Entonces, se puede usar la siguiente ecuación para determinar el balance, retornando valores cercanos a 0, si se trata de un dataset no balanceado, o cercanos a 1 si el dataset es balanceado.

$$\text{Balance} = \frac{E}{\log k}$$

Fase 2: Fase de Preprocesamiento

Luego de asegurarnos que el dataset es apropiado, hay que realizar la fase de preprocesamiento. Resulta de gran importancia la extracción y descubrimiento del conocimiento escondido en texto no estructurado (Kao & Poteet, 2007). Es necesaria la identificación temprana de ruido y la limpieza adecuada de datos, para que los modelos inteligentes puedan aprender a detectar sentimiento.

Teniendo en cuenta que el texto del Tuit necesita estar en un formato adecuado, la fase de preprocesamiento considera las siguientes acciones:

Eliminación de caracteres: Estos son caracteres que no ayudan a detectar sentimiento, tales como: (i) caracteres HTML: se busca eliminar del texto del Tuit, elementos que mencionen otros usuarios, es decir, que contengan arroba, por ejemplo @leomessi; (ii) URLs y hashtags: también es necesario eliminarlos para reducir el ruido; y (iii) caracteres no alfanuméricos como: ! ´ ” · $ % & / ( ) = * ¡ ... Ninguno de estos caracteres es de interés para el análisis de sentimientos. Se pueden utilizar expresiones regulares para remover del Tuit cualquiera de estos elementos innecesarios.

Transformar texto a minúsculas: Se recomienda transformar a minúscula todo caracter para facilitar el proceso de preprocesamiento, dada la sensibilidad a mayúsculas y minúsculas que pueden presentar las técnicas de análisis de texto. Además, de esta manera se mantiene la uniformidad del texto (Baby et al., 2017).

Expandir contracciones: Las contracciones de palabras, normales para el idioma, como don’t, wouldn’t, deben ser transformadas a do not y would not, respectivamente.

Compresión de letras: En este caso se tratan casos de palabras incorrectas con letras repetidas más de dos veces, como “pleaseeeeeee” o “yeeeees”, cuyos caracteres son comprimidos, formando please y yes, respectivamente. En este caso, palabras como apple, fool o letter, quedarían escritas correctamente.

Eliminación de entradas con palabras de más de 45 caracteres: Dado que en inglés el número máximo de letras por palabra es 45.

Corrección de errores ortográficos: Es recomendable detectar y corregir errores ortográficos en el texto de los tuits. Una forma de realizar esta corrección es usando algoritmos basados en la distancia de Levenshtein (métrica de distancia entre dos secuencias de caracteres, palabras) (Yujian & Bo, 2007), que, además, encuentra permutaciones de una palabra.

Eliminación de stop words excepto el “not”: Palabras tales como a, the, this.

Eliminación de palabras repetidas: Para eliminar la cantidad de palabras innecesarias a procesar, se eliminan las repeticiones en secuencia de una palabra. Por ejemplo, de Professor great great great, solo interesa: Professor y great.

Eliminación de entradas nulas: Por la cantidad de filtros anteriores, es probable que hayan quedado tuits sin texto. Este subconjunto de tuits no es de interés, por lo tanto, no se deben considerar en el dataset.

Lematización: La importancia de la lematización es llevar las palabras a la forma básica o de diccionario (conocido como lema) para así eliminar terminaciones de inflexión (Plisson et al., 2004), por ejemplo, el lema de la palabra studies sería study.

Fase 3: Fase de clasificación

Una vez que se ha limpiado el conjunto de datos, es necesario dividirlo en un conjunto de entrenamiento y uno de pruebas, asegurando evitar que se solapen. En la fase de clasificación se seleccionan los modelos a evaluar y se preparan los datos para cada modelo. Comprende las siguientes actividades:

Baseline: Proponer un modelo ingenuo. Para tener un punto de referencia, es recomendable proponer un modelo ingenuo de análisis de sentimientos, que servirá como comparación para los otros algoritmos; se trata de una línea base cuyo propósito es identificar el peor comportamiento que puede lograr un modelo y compararlo con el comportamiento de modelos con algoritmos más avanzados.

Preparar el dataset de entrenamiento: Transformación de texto en n-gramas. Los datos de entrada son descritos por texto del Tuit y una etiqueta. El texto en lenguaje natural no es de utilidad para algoritmos que generen modelos inteligentes. Se debe entonces, encontrar una forma apropiada de representar el texto. Una forma ingenua de hacerlo es utilizando el método de bolsa de palabras (bag of words en inglés) para crear una matriz de frecuencia de términos (Term Frequency, TF en inglés). La frecuencia de un término (TF), se refiere a la cuenta cruda de ocurrencia de un término dentro de un documento. En la siguiente ecuación, se muestra cómo se calcula la frecuencia del elemento d en un documento de tamaño n.

En nuestro caso el documento sería cada Tuit. Se trata entonces, de la frecuencia de aparición de un término, en comparación al tamaño del documento.

$$t(d)=\frac{f_d}{\sum_{d=1}^{n} f_d}$$

No obstante, es conocido que usando esta metodología se llega a dos problemas: (i) una matriz dispersa, en la que abunda la cantidad de 0s; y (ii) una cantidad exponencial de columnas o mejor conocida como atributos (features).

Para solucionar la inmensa cantidad de columnas, es posible utilizar n-gramas (secuencia de n palabras). Sin embargo, se mantiene el problema de la cantidad de 0s en la matriz de entrenamiento. Por ello, se puede utilizar una forma de conteo más realista y usada en el medio académico, que combina TF con la frecuencia inversa en un documento (Inverse Document Frequency – IDF), resultando la estrategia TF–IDF.

IDF permite determinar cuanta información provee una palabra; es decir, que tan común o rara es la palabra en todos los documentos. Es necesario tomar esta medida porque como es conocido, existen palabras que pueden ser frecuentes en los tuits mas no son de gran importancia, es decir, no arrojan información, por ejemplo, la palabra the. La siguiente ecuación, muestra el cálculo de IDF, donde N es el número total de documentos del conjunto de datos y $|df_t|$ representa la cantidad de apariciones de t en los documentos.

$$\text{tf-idf}_{t,d}=\left(1+\log t f_{t,d}\right)\text{-}\log\frac{N}{df_t}$$

Finalmente, la siguiente ecuación muestra cómo se combina el TF e IDF para producir el conteo TF-IDF, que refleja la importancia de una palabra en un documento. A partir de esta ecuación se genera la matriz de entrenamiento.

$$\text{tf-idf}_{t,d}=\left(1+\log t f_{t,d}\right)\cdot\log\frac{N}{df_t}$$

Dividir el dataset: Para tener un conjunto de datos que valide al modelo de aprendizaje, se deben separar los datos de entrenamiento de los de prueba. Una división típica es destinar el 70% del dataset para el entrenamiento, mientas que el 30% restante será un subconjunto independiente destinado a las pruebas. A pesar de que la división del conjunto puede llegar a ser arbitraria, lo recomendable es que el subconjunto de entrenamiento sea mayor al subconjunto de pruebas, ya que de esta manera se mejora el modelo de clasificación y la estimación del error será más precisa (Dobbin & Simon, 2011).

Seleccionar variables predictivas: Es conocido que cuando el dataset no tiene suficientes features, el modelo es propenso a hacer underfit (el modelo no puede captar adecuadamente la estructura de los datos) y cuando el dataset tiene muchos features es fácil que el modelo termine en overfit (problema típico en estadística en donde se aprende demasiado bien del dataset, pero no se llega a una función que pueda generalizar) (Grus, 2015). Por lo tanto, los features usados para entrenar los modelos de Machine Learning tienen una gran influencia en el desempeño del modelo.

En Machine Learning y estadística, la selección de features también se conoce como selección de variable o selección de subconjunto de variables. Este es el proceso de seleccionar (manual o automáticamente) un conjunto relevante de variables (features) para ser usado en la construcción del modelo inteligente. La selección de features es un paso importante porque simplifica el modelo, haciéndolo más fácil de interpretar por los investigadores, acorta tiempos de entrenamiento y evita el overfitting. Existen varios algoritmos para realizar esta selección, entre ellos: information gain (IG), variance and entropy, useful features, independent/non-redundant, Hill Climbing, select k best.

Algoritmo inteligent: Dependiendo del algoritmo a implementar, se deben afinar ciertos parámetros (por ejemplo, si es una red neural, probablemente se deberá especificar la cantidad optima de neuronas). Sin embargo, es importante destacar que no todos los algoritmos tienen la flexibilidad de hiperparámetros, por ejemplo, el algoritmo de Regresión Logística simple no consta de hiperparámetros.

Fase 4: Fase de evaluación

Para esta fase, se debe encontrar una métrica apropiada para medir el desempeño de cada algoritmo y comparar que tan eficiente fue cada uno en la tarea a resolver.

F1 score: Dependiendo del problema a resolver, puede ser que, para un problema específico, sea más conveniente utilizar la métrica de error cuadrático medio o la métrica de exactitud. Las métricas que se eligen para evaluar los algoritmos de Machine Learning son muy importantes. De manera general, existen muchas métricas a considerar cuando se habla de un algoritmo de Machine Learning y no todas son relevantes para un problema en particular.

La matriz de confusión representa un mecanismo a considerar en los problemas de clasificación, dado que permite visualizar el desempeño de un algoritmo que se emplea en aprendizaje supervisado. Existen varias métricas que se pueden apreciar en esta matriz, por ello su importancia.

La medida de precisión dice, cuando un modelo predice una clase como positiva, que tan seguido es correcta esta predicción. Es una medida útil cuando el costo de los falsos positivos es alto. Para comparar la eficiencia en términos de precisión de los algoritmos implementados en la fase de Clasificación, se puede utilizar la métrica de F1, que expresa la relación entre precisión y exhaustividad (recall). F1 se utiliza cuando los falsos positivos y los falsos negativos son vitales. La medida de exhaustividad recall es útil cuando el costo de los falsos negativos es alto.

La exactitud (accuracy) es una medida que se toma cuando es importante saber que tan seguido el modelo hace predicciones correctas. Se usa cuando los positivos verdaderos y los negativos verdaderos son importantes.

2.2 Estudio comparativo

Siguiendo la estrategia metodológica propuesta, realizamos el estudio comparativo de diferentes técnicas de aprendizaje. Todo el código fuente se puede encontrar en nuestro repositorio el cual es público, el mismo incluye el análisis completo del dataset, además de la implementación de todos los algoritmos.

Fase 1: Fase de evaluación

Exploración de datos

Para extraer la información sobre la distribución de los datos, utilizamos la técnica sencilla de conteo. La Figura 2, refleja que la cantidad de tuits en Sentiment140 con sentimiento positivo y sentimiento negativo es balanceada. La Figura 3, muestra la cantidad de palabras por Tuit, siendo seis el tamaño más frecuente de los tuits en Semtiment140.

Otros datos que se extrajeron del dataset durante el proceso de análisis de los datos, son los siguientes: (i) la cantidad de palabras positivas y negativas por Tuit esta entre 1 y 4, lo que demuestra una vez más el balanceo del conjunto de datos; (ii) hay 10,786,068 palabras en total (cantidad de unigramas), con 98,164 palabras únicas; (iii) la menor cantidad de palabras que hay en un Tuit es 1; (iv) la mayor cantidad de palabras que hay en un Tuit es 26; y (v) hay 9,196,821 bigramas en total, siendo únicos 3,245,405.

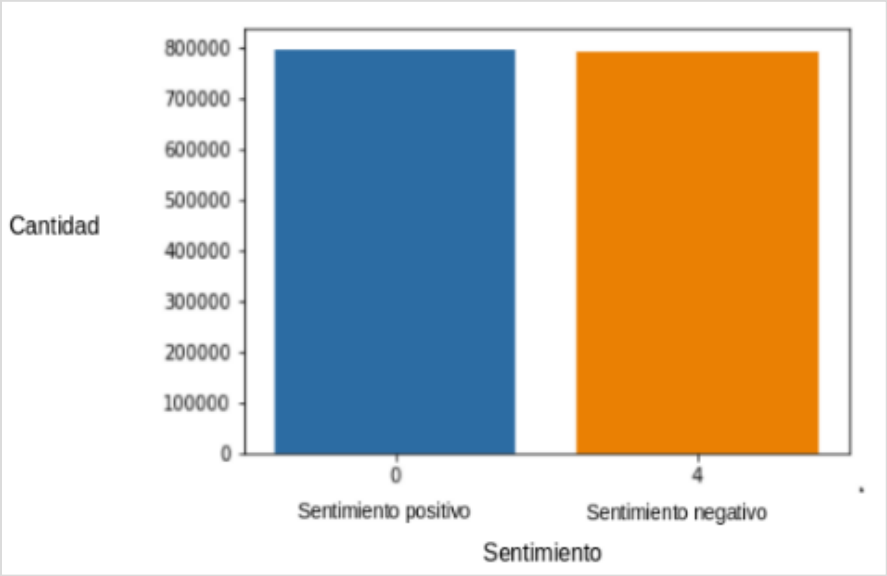

Figura 2. *Distribución de tuits*

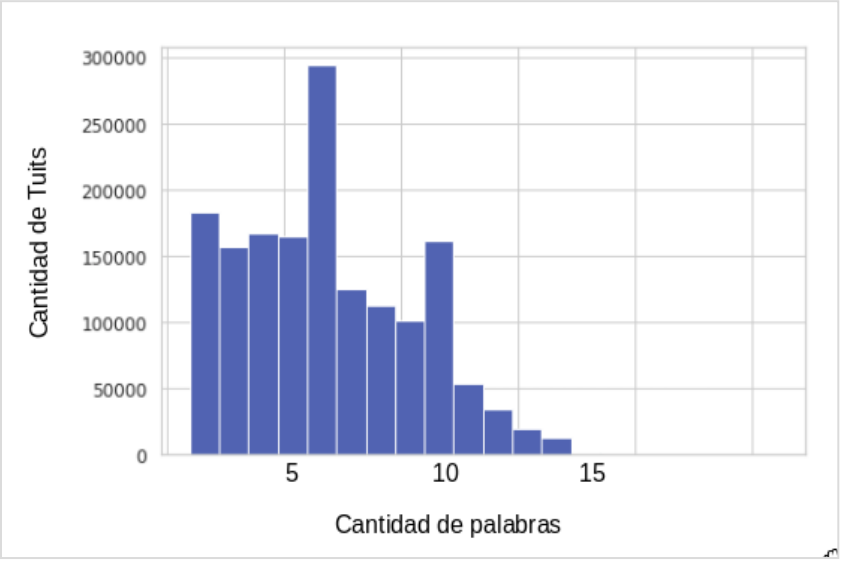

Figura 3. *Cantidad de palabras*

Fase 2: Fase de preprocesamiento

Eliminación de caracteres: Se logró a través de las siguientes expresiones regulares:

$$<.*?> |\&([a-z0-9]+|\#[0-9]\{1,6\}|\#x[0-9a-f]\{1,6\})$$

Transformación texto a minúsculas: Se utilizo la función predefinida. lower () de Python, que transforma todos los caracteres de un texto dado, a minúsculas. Es un builtin que se carga automáticamente cuando se carga el intérprete de Python, o cuando se inicia alguna sesión o ambiente interactivo.

Expandir contracciones: En este caso, se utilizó un diccionario con contracciones en inglés, que contiene pares (key, value), donde key es la contracción y value es su respectiva expansión. Las contracciones key

Compresión de letras: Para esta actividad, se implementó una función que detecta letras continúas repetidas más de dos veces y elimina las letras repetidas.

Eliminación de entradas con palabras de más de 45 caracteres: Luego del paso anterior, se aplica otra función desarrollada para eliminar las palabras cuya longitud fuera mayor a 45 caracteres.

Corrección de errores ortográficos: En este paso se usó un corrector ortográfico, basado en el algoritmo de distancia de Levenshtein y teorema de Bayes.

Eliminación de stop words excepto el "not": En la librería Natural Language Toolkit (NLTK) (conjunto de librerías que ofrece herramientas para manipular texto en lenguaje natural) existe una funcionalidad que permite conocer los stop words de un idioma. Se uso esta funcionalidad para eliminar de los tuits los stop words de Ingles. Fue pertinente no eliminar el stop word not.

Eliminación de palabras repetidas: Se realizó a través de otra función desarrollada para filtrar palabras repetidas consecutivas. Por ejemplo, el Tuit: sentiment sentiment analysis is fun, se convierte en: sentiment analysis is fun.

Eliminación de entradas nulas: Se realizo una búsqueda lineal sobre todas las entradas del dataset, representadas con un dataframe (estructura de datos bidimensional), para identificar y eliminar aquellas cuyo texto quedo vacío (columna Tuit de la Tabla 1).

Tabla 1.

Ejemplos de tuits

Sentimiento	Query	Tuit
Positivo	Jquery	dcostalis: Jquery es mi nuevo amigo
Negativo	Examen	jvciOus: Estudiando para Examen de Historia ugh.

Lematización: Para la lematización, se utilizó de nuevo la librería NLTK, específicamente la funcionalidad WordNetLemmatizer ().

Fase 3: Fase de clasificación

Para este estudio comparativo, se consideraron los algoritmos de Regresión Logística, Naive Bayes, como método probabilístico y SVM, que es un modelo más avanzado; y modelos de Deep Learning, que han demostrado ser eficientes para estos problemas. A continuación, describimos los pasos realizados en la fase de clasificación:

Baseline: Propuesta de modelo ingenuo: Baseline. Es la implementación más simple, que somete cada entrada del conjunto de datos a una evaluación de proporción. Para realizar dicha evaluación se utiliza el conjunto de palabras positivas y negativas de dos diccionarios en inglés ya preestablecidos, propuesto en Hu & Liu (2004). En el diccionario de palabras positivas, se encuentran palabras como: accesible, catchy, celebration, fertile. Mientras que, en el diccionario de palabras negativas, se encuentran palabras como: abominable, aggressive, bomb, damaged. El modelo ingenuo consiste en contar cuantas de las palabras del Tuit se encuentran en la lista de palabras positivas y cuantas en la lista de palabras negativas. Luego, el sentimiento del Tuit depende de cual conteo sea mayor. Para ello se implementó una función que cuenta la cantidad de palabras positivas y palabras negativas, y etiqueta la polaridad dependiendo de cual cantidad sea mayor. Estos resultados se utilizan como una base sobre la cual comparar la fiabilidad de los subsiguientes modelos (Liu, 2010).

Preparar el dataset de entrenamiento: La preparación del dataset, se basó en el uso de n-gramas y TF-IDF. Para esto, se utilizaron las funciones CountVectorizer y TfidfVectorizer del paquete sklearn.feature_extraction.text, que mapean colecciones de texto a matrices de conteo de tokens o features TF-IDF, respectivamente. Cualquiera de estas funciones puede utilizarse para hacer el conteo de frecuencias de unigramas, bigramas y trigramas.

Dividir el dataset: Se dividió el dataset en 70% para el conjunto de entrenamiento y 30% para el conjunto de pruebas, usando la función train_test_split de la librería Sklearn.

Seleccionar variables predictivas: Como consecuencia de la división del texto en n-gramas, para n entre 1 y 3, tenemos una matriz con abundantes columnas de n-gramas, pero no todas son relevantes para la clasificación. Existen varias formas de escoger que variables son las requeridas para entrenar un buen modelo predictivo. En este estudio se seleccionó SelectKBest de Sklearn.

Algoritmo inteligente. Algoritmos de Machine Learning clásicos:

Regresión Logística (Logistic Regression en inglés): Se trata de una técnica estadística utilizada para predecir la probabilidad de respuesta binaria basada en una o más variables independientes, dicha probabilidad se estima utilizando la función logística o función Sigmoide (ecuación 1.1) y la ecuación de regresión logística (ecuación 1.2).

$$1.1. \quad p=\frac{1}{1+e^{-y}}$$

$$1.2. \quad y=\beta_0+\beta_1 X_1+\ldots+\beta_n X_n$$

Los βi son llamados coeficientes beta (Lunt, 2015) y reflejan la influencia o importancia de las variables de predicción. Aplicando Sigmoide a la regresión logística se obtiene la ecuación 1.3.

$$1.3. \quad p=\frac{1}{1+e^{-\beta_0+\beta_1 X_1+\ldots+\beta_n X_n}}$$

Según el autor, Lunt (2015), los modelos basados en regresión logística construyen predictores de un resultado distribuido normalmente, y son basados en la suposición de que los datos son linealmente separables. Para la implementación en código se instancia el modelo LogisticRegression () con los features seleccionados en el paso de selección de variables predictivas.

Naive Bayes: Este clasificador pertenece a un conjunto de clasificadores probabilísticos basados en la suposición de que todos los eventos son independientes de una variable (Xu, 2018), esto quiere decir que el efecto del valor de un predictor (B) en una clase (A) es independiente de los valores de otros predictores, esta suposición se llama independencia condicional de clase. Descomponiendo la ecuación 1.4, es conocido que P(A|B) es la probabilidad a posteriori, en donde se expresa la probabilidad de A, dada la evidencia B, al contrario de la verosimilitud (también conocida como Likelihood).

$$1.4. \quad P(A|B)=\frac{P(B|A)*P(A)}{P(B)}$$

La verosimilitud viene dada por P(B|A), y mide la bondad de ajuste del modelo. La misma se forma a partir de la distribución de probabilidad conjunta de la muestra.

Basados en la ecuación 1.4, se puede traducir al contexto del análisis de sentimientos en tuits, como lo muestra la ecuación 1.5, donde x representa el con-junto de prueba generado (datos de entrada) y S el sentimiento.

$$1.5. \quad P(S|x)=\frac{P(x|S)*P(S)}{P(x)}$$

Se usa la ecuación 1.6 para realizar las predicciones, siendo y la predicción, b una relación entre la cantidad de tuits de entrenamiento con etiquetado positivo y negativo y m una relación entre los tuits de entrenamiento con etiquetado negativo y positivo.

$$1.6. \quad y = mx + b$$

Para la implementación en código de este algoritmo se procede a entrenar el modelo de Naive Bayes mediante el cálculo de los coeficientes de la ecuación 1.6, considerando los features seleccionados en el paso de selección de variables predictivas.

Support Vector Machine (SVM) - Máquina de vectores: El SVM, es un modelo de aprendizaje supervisado especializado en clasificación y análisis de regresión. Los datos por clasificar se representan con un vector n-dimensional de características con las que se localiza cada entrada del dataset como un punto en un

espacio. El algoritmo determina un hiperplano que separa de la mejor manera posible los puntos etiquetados en el conjunto de entrenamiento.

Una vez determinado, se puede predecir una nueva entrada ubicando su representación en el espacio y determinando de qué lado del hiperplano se localizó. Dado un conjunto de entrenamiento, (X1, Y1) ... (Xn, Yn), Xi ∈ Rn, el hiperplano se define como $wx^{\intercal} + b = 0$, donde w es el vector de pesos, x representa la entrada o input, y b representa el sesgo o bias. w y b deben satisfacer las siguientes desigualdades, $wx^{\intercal} i + b \geqslant 1$ si yi = 1 y $wx^{\intercal} i + b \leq -1$ si yi = −1. El objetivo de este algoritmo es encontrar los valores de w y b que separen de la mejor manera posible el conjunto de datos.

Se procede a entrenar el modelo de SVM, que presenta las siguientes ventajas: (i) efectividad en espacios vectoriales con varias dimensiones; (ii) usa un subconjunto de puntos en el espacio, por lo que su entrenamiento no es tan pesado; y (iii) existe la posibilidad de especificar funciones de Kernel. A continuación, mostramos los algoritmos de Aprendizaje Profundo (Deep Learning en inglés).

Redes Neuronales Convolucionales (Convolutional Neural Network o CNN en inglés): Las CNN son redes neuronales, específicamente son versiones regularizadas de un perceptrón multicapas (i.e., redes de neuronas completamente conectadas). Estas redes toman ventaja de patrones jerárquicos en los datos y los ensamblan en patrones complejos, compuestos de patrones más pequeños y simples. La convolución lineal de datos se representa como una función lineal que dadas dos funciones f y g, produce una tercera que expresa como la forma de una es modificada por la otra. En particular, para el caso de análisis de texto, estos filtros consisten en filtros que determinan que atributos (features) son realmente importantes para una instancia.

Debido a que este algoritmo contiene capas que realizan convoluciones de los datos (filtros para obtener los features más importantes), su implementación se basa en colocar una capa embebida (embedding layer en inglés), para lo cual primero se crea la matriz de pesos de la siguiente manera:

1. Previamente se tokenizan los datos; luego con cada token se calcula el índice que lo representa, usando la función word_index. Es decir, se obtiene un diccionario de palabras y un numero entero asignado a esa palabra, lo cual se entiende como su índice.
2. Para tener buenos valores de embeddings podemos cargarlos de palabras pre entrenadas basadas en datos de entrenamiento mucho más grandes. La base de datos GloVe (diccionario embedding_index) contiene embeddings de palabras pre entrenadas. Primero, se colocan los embeddings de las palabras en un diccionario donde las claves son las palabras y los valores de los embeddings de palabras. Luego, con los embeddings GloVe asignados a un diccionario (word_index), se busca la representación vectorial de cada palabra del dataset.
3. A partir de word index y embeddings (generados en los dos pasos anteriores), se construye la matriz embedded_matrix: se revisa cada palabra para ver si está en embeded_index, y si lo está se coloca en la matriz en la columna correspondiente a la posición de la palabra en word index.
4. Luego, se utiliza keras.layers.Embedding para cargarlo en una capa de la CNN.

Después de crear la matriz de pesos, se construye red CNN:

1. Convertir una oración tokenizada a una matriz de oración, donde cada fila representa una palabra como vector.
2. Varios filtros que mapean diferentes features en el mismo sector pueden complementarse, mientras que filtros en diferentes sectores contribuyen a concentrarse en secciones pequeñas de los textos. Por lo que se usan cinco filtros diferentes de tamaño 2, 3, 4, 5 con un total de 50, 50, 50, 30, 20 matrices filtros (función de activación ReLU) por cada nivel convolucional.

3. Extracción de los features independientes más significativos de su sector en el texto (n-gramas más importantes), usando max-pooling para cada salida de las capas de convolución.
4. Combinación de los valores máximos de cada capa en un solo vector y procesado por capa oculta completamente conectado.
5. Predicción final al pasar por la función Sigmoide lo que ha mapeado la capa exterior.

Redes de gran memoria de corto plazo (Long Short Term Memory o LSTM en inglés): Las redes neurales de LSTM contienen unidades especiales de memoria en sus capas intermedias (Sak, Senior, & Beaufays). Dichos bloques de memoria contienen células de memoria que almacenan un estado temporal de la red además de puertas que controlan el flujo de información. La puerta de entrada controla el flujo de activaciones en la célula de memoria, mientras que la puerta de salida controla el flujo de salida de la célula de activación en el resto de la red. Finalmente, se tiene una puerta de olvido, para calcular la cantidad de datos que se debe mantener.

Para este modelo se utiliza un algoritmo basado en redes neurales recurrentes (RNN) que contiene capas que se retroalimentan. Todas las redes RNN, tienen bucles en su capa recurrente que les permite mantener información en memoria. Sin embargo, existen problemas que requieren aprender también dependencias a larga distancia. Las redes RNN no pueden lograr aprender estas dependencias, debido a que el gradiente de la función decae de forma exponencial con el tiempo (se conoce como fuga del gradiente). Las redes recurrentes LSTM intentan resolver este problema, incluyendo en sus celdas unidades de memoria. Se utilizan tres compuertas que permiten saber cuándo la información entra en la memoria, cuando sale y cuando se olvida, solucionando así el aprendizaje de dependencias a largo plazo. Para su implementación se coloca la misma capa embebida desarrollada para CNN:

1. Tokenizar las oraciones de la misma manera que en CNN.
2. Crear una capa de inclusión o embedding layer en el modelo secuencial.
3. Eliminación de feature maps unidimensionales, colocando una capa de dropout y promoviendo la independencia de features.

Para los hiperparámetros de la capa intermedia de esta red, se toma en consideración:

1. vocab_size: Se refiere a la cantidad de palabras usadas en el diccionario. Escogemos el mismo número de entradas y salidas en que tenemos tokens distintos.
2. max length: Expresa la longitud máxima de una oración (Tuit). Los tuits que sobrepasen este tamaño, son excluidos del entrenamiento, por lo que este parámetro debe ser lo suficientemente grande para que pueda incluir los tuits de entrenamiento. Como optimizador se utiliza Adam, un algoritmo para optimización que se basa en el gradiente de primer orden de funciones estocásticas.

Redes bidireccionales de gran memoria de corto plazo (Bidireccional Long Short Term Memory o Bi-LSTM en Ingles): Este método de aprendizaje entrena una red neuronal para utilizar un contexto de secuencia simétrica de muy largo alcance utilizando una combinación de elementos de procesamiento no lineal y bucles de retroalimentación lineal para almacenar el contexto de largo alcance. Este modelo mejora el rendimiento del modelo para problemas de clasificación de secuencias al entrenar dos capas de LSTM en la secuencia de entrada, la primera en dicha secuencia y la segunda en una copia invertida de la secuencia de entrada. Esto puede proporcionar un con-texto adicional a la red, resultado en un aprendizaje más rápido e incluso más completo.

Los hiperparámetros se mantienen como los de LSTM. Lo que varía en este ca-so es la forma bidireccional de la red. Para su implementación se procede de la siguiente forma:

1. Crear un diccionario word2vect con embeddings GloVe w2v (Mikolov et al., 2013).

2. Tokenizar las oraciones de la misma manera que en CNN (lista de tokens).
3. Vectorizar los tokens únicos con el Tokenizer de keras.
4. Calcular la matriz de pesos (embedded_matrix). Se revisa que cada palabra tiene un vector no vacío con w2v, y se coloca en la matriz en la columna correspondiente a su índice en el vector token_list.
5. Crear el modelo con el contenedor bidireccional de keras y la matriz de pesos embedded_matrix.

Fase 4: Fase de evaluación

F1 score: Para la comparación del comportamiento de los modelos, se usó la métrica F1, con la implementación disponible en la librería de Sklearn de Python llamada F1 score.

3. RESULTADOS Y DISCUSIÓN

Como se aprecia en la Tabla 2 el mejor modelo clásico fue SVM, con 78% de precisión, y 79% de métrica F1 (F1 score). Para los modelos de Deep Learning, se obtuvo un mejor resultado que en los modelos clásicos, lo cual tiene sentido debido al proceso de entrenamiento por el que pasa el modelo. El modelo con mejor desempeño fue el modelo de Deep Learning Long Short Term Memory (LSTM), alcanzando un 88% de precisión y 89% de métrica F1. Como se esperaba, el peor de los modelos de Deep Learning fue el CNN. Podemos apreciar las bondades de los modelos en la Tabla 2, a pesar de ser modelos diferentes, todos se comportan de forma similar tanto para la detección de sentimientos.

Tabla 2.

Resultados del F1 score, precisión y exhaustividad

Modelo	F1 score	Precisión	Exhaustividad
Baseline	0,743	0,685	0,813
Regresión logística	0,788	0,780	0,797
Naive Bayes	0,772	0,781	0,763
SVM	0,792	0,781	0,803
CNN	0,773	0,773	0,773
LSTM	0,891	0,881	0,871
Bi-LSTM	0,860	0,851	0,856

4. DISCUSIÓN

Existe mucha investigación realizada en el tema de análisis de sentimientos tales como los estudios de Li et al. (2022), Cambria (2016), Liang et al. (2022), De Albornoz et al. (2011), Mirtalaie et al. (2018), Chu & Roy (2017), Oliveira et al. (2017). Se han publicado estudios muy completos que revisan los avances y los retos del análisis de sentimientos (Lin et al., 2020; Kaur et al., 2017; Abirami & Gayathri, 2017; Hussein, 2018; Chaturvedi et al., 2018), el uso de Deep Learning para hacer análisis de sentimientos (Zhang et al., 2018), diferentes técnicas de análisis de sentimientos en Twitter (Kharde & Sonawane, 2016; Giachanou & Crestani, 2016), análisis de sentimientos en diferentes idiomas (Al-Ayyoub et al., 2019), entre otros. La mayoría de estos trabajos describen cualitativamente las diferentes técnicas para realizar análisis de sentimientos en textos.

Breck & Cardie (2017) investigaron la ejecución de varias técnicas de aprendizaje de máquina, como Naive Bayes, Máximum Entropía y SVM, en el dominio de opiniones sobre películas. A partir de su análisis, obtuvieron 82,9% de exactitud en su modelo, utilizando SVMs con unigramas. Un aporte de nuestro trabajo es que consideramos bigramas y trigramas, además de unigramas. Además, exploramos modelos más complejos, como aquellos relacionados con redes neurales recurrentes.

En Ribeiro et al. (2016), se estudiaron 24 métodos de análisis de sentimientos con 18 conjuntos de datos diferentes. Los resultados destacan la medida en que el rendimiento de predicción de estos métodos varía considerablemente entre los conjuntos de datos. En Desai & Mehta (2016) se evalúaron cuatro métodos de aprendizaje supervisado: Naive Bayes, Entropía Máxima, SVM y Random Forest. Los autores concluyen que Naive Bayes es el algoritmo más simple para implementar y de mejor compresión en comparación con SVM y Entropía Máxima. En Go et al. (2009) se estudiaron algoritmos de clasificación de Machine Learning sobre un dataset. Los autores llegan a la conclusión de que el uso de algoritmos de aprendizaje es una forma efectiva para el análisis de sentimientos en Twitter. Todos estos trabajos enfatizan la importancia de realizar experimentos para comparar diferentes métodos de análisis de sentimientos antes de elegir uno como solución. Sin embargo, no proponen o especifican un enfoque para realizar dicha comparación. En este trabajo, proponemos una estrategia metodológica como guía.

Con respecto a las estrategias de preprocesamiento, en Angiani et al. (2016) se especificaron diferentes filtros para limpieza de los datos originales (en este caso tuits) para determinar cuál combinación de filtros produce mejores resultados en el aprendizaje. Además, se utilizan técnicas de stemming para evitar variaciones de palabras como greatly, greatest y greater, representando a todas estas como great. Así se decrementa la entropía y se aumenta la relevancia del concepto great. De forma similar, en Jianqiang & Xiaolin (2017) se concluye que los resultados experimentales indican que la eliminación de URLs, de palabras de detención (stop words) y de números afectan mínimamente el rendimiento de los clasificadores. En nuestro estudio comparativo, usamos técnicas de NLP para implementar los filtros y asegurar un correcto preprocesamiento del dataset.

Entendiendo que el análisis de sentimientos se ha empezado a definir en términos de Deep Learning, se suele tener redes neurales con más de dos capas intermedias en las que resulta indispensable ajustar sus hiperparámetros. Según los autores de Martínez Cámara et al. (2019) y Liu (2010), actualmente los esfuerzos de investigación se centran en mejorar la codificación de la información contextual subyacente en una secuencia de texto. Son esas redes neuronales con una mayor capacidad de representación las que cada vez más aumentan su complejidad, lo que significa que tienen más hiperparámetros que deben definirse. Inspirados en los resultados de esa investigación, en nuestro trabajo entonamos los hiperparámetros de los modelos de Deep Learning para lograr modelos eficientes.

En esta investigación nos centramos únicamente en la comparación de dos modelos de Deep Learning, LSTM y Bi-LSTM, ya que nos interesa estudiar el comportamiento de las celdas LSTM y cómo éstas agregan información a largo plazo de una forma secuencial, por ello su popularidad. Sin embargo, hay otros modelos basados en RNNs como los Transformadores, Transformer Learner (TL) en inglés (Tabinda Kokab et al., 2022). Los TL conforman una extensión de las RNN, y pueden realizar muchos cómputos en paralelo, mientras que los modelos LSTM y Bi-LSTM deben computar secuencialmente. Entonces, la única ventaja intuitiva que esperaríamos con modelos TL es poder entrenar modelos del lenguaje en unos pocos días en lugar de meses. Esperamos en trabajos futuros comparar resultados con otros modelos como TLs, e incluso con Grafos del Conocimiento (Lovera et al., 2021), Knowledge Graphs en inglés, entre otros. Estos otros métodos también ofrecen comportamientos de aprendizaje alternos y pueden potencialmente ser beneficiosos para el análisis de texto, en particular de sentimientos.

5. CONCLUSIONES

La sección de resultados demuestra mayor precisión para algoritmos que son capaces de manejar más información del texto. En particular, se nota mejoría cuando el modelo es capaz de entender dependencias a larga distancia en el Tuit, como es el caso de LSTM y Bi-LSTM. Así, aunque los mensajes de Twitter tienen características únicas, los algoritmos de Deep Learning han mostrado ser asertivos para determinar el sentimiento expresado. El modelo con peor desempeño según la Tabla 2 es el basado en el algoritmo de

Naive Bayes, esto se debe a la fuerte suposición de independencia del algoritmo. Es claro que existe dependencia entre las palabras en texto, por lo que la suposición de Naive Bayes es frecuentemente violada, esto explica su desempeño para esta tarea.

Aun cuando realizar análisis de sentimientos en textos muy cortos (que por lo general no cumplen con las debidas estructuras gramaticales y reglas sintácticas del lenguaje) representa un gran reto, en general, los resultados demuestran que los modelos basados en Deep Learning combinados con técnicas de NLP resultan ser adecuados en este contexto.

Además, concluimos que nuestra contribución principal, la estrategia metodológica planteada, resulta ser la adecuada para el estudio de textos cortos, ya que cubre los tratamientos necesarios para su transformación y entendimiento. Dicha estrategia termina ofreciendo una plataforma de ejecución a los algoritmos inteligentes. Esta plataforma resulta ser lo suficiente genérica y robusta lo que facilita a que los algoritmos puedan aprender potencialmente diferentes tareas.

FINANCIAMIENTO

Ninguno.

CONFLICTO DE INTERESES

No existe ningún tipo de conflicto de interés relacionado con la materia del trabajo.

CONTRIBUCIÓN DE LOS AUTORES

Conceptualización, curación de datos, análisis formal, investigación, metodología, supervisión, validación, redacción - borrador original, redacción - revisión y edición: Lovera, F. A. & Cardinale, Y. C.

REFERENCIAS BIBLIOGRÁFICAS

Abirami, A. M., & Gayathri, V. (2017). A survey on sentiment analysis methods and approach. *2016 Eighth International Conference on Advanced Computing (ICoAC)*, 72–76. https://doi.org/10.1109/ICoAC.2017.7951748

Al-Ayyoub, M., Khamaiseh, A. A., Jararweh, Y., & Al-Kabi, M. N. (2019). A comprehensive survey of arabic sentiment analysis. *Information Processing & Management*, *56*(2), 320–342. https://doi.org/10.1016/j.ipm.2018.07.006

Angiani, G., Ferrari, L., Fontanini, T., Fornacciari, P., Iotti, E., Magliani, F., & Manicardi, S. (2016). A comparison between preprocessing techniques for sentiment analysis in Twitter. *CEUR Workshop Proceedings*, *1748*, 1–11. https://ceur-ws.org/Vol-1748/paper-06.pdf

Baby, C. J., Khan, F. A., & Swathi, J. N. (2017). Home automation using IoT and a chatbot using natural language processing. *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 1–6. https://doi.org/10.1109/IPACT.2017.8245185

Breck, E., & Cardie, C. (2017). Opinion Mining and Sentiment Analysis. In *The Oxford Handbook of Computational Linguistics 2nd edition*. Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199573691.013.43

Cambria, E. (2016). Affective Computing and Sentiment Analysis. *IEEE Intelligent Systems*, *31*(2), 102–107. https://doi.org/10.1109/MIS.2016.31

Chaturvedi, I., Cambria, E., Welsch, R. E., & Herrera, F. (2018). Distinguishing between facts and opinions

for sentiment analysis: Survey and challenges. *Information Fusion*, *44*, 65–77. https://doi.org/10.1016/j.inffus.2017.12.006

Chu, E., & Roy, D. (2017). Audio-Visual Sentiment Analysis for Learning Emotional Arcs in Movies. *2017 IEEE International Conference on Data Mining (ICDM)*, 829–834. https://doi.org/10.1109/ICDM.2017.100

De Albornoz, J. C., Plaza, L., Gervás, P., & Díaz, A. (2011). A Joint Model of Feature Mining and Sentiment Analysis for Product Review Rating. In *Advances in Information Retrieval* (pp. 55–66). https://doi.org/10.1007/978-3-642-20161-5_8

Desai, M., & Mehta, M. A. (2016). Techniques for sentiment analysis of Twitter data: A comprehensive survey. *2016 International Conference on Computing, Communication and Automation (ICCCA)*, 149–154. https://doi.org/10.1109/CCAA.2016.7813707

Dobbin, K. K., & Simon, R. M. (2011). Optimally splitting cases for training and testing high dimensional classifiers. *BMC Medical Genomics*, *4*(1), 31. https://doi.org/10.1186/1755-8794-4-31

Giachanou, A., & Crestani, F. (2016). Like It or Not: A Survey of Twitter Sentiment Analysis Methods. *ACM Computing Surveys*, *49*(2), 1–41. https://doi.org/10.1145/2938640

Go, A., Bhayani, R., & Huang, L. (2009). Twitter Sentiment Classification using Distant Supervision. *Processing*, 1–6. https://www-cs-faculty.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf

Grus, J. (2015). *Data Sciemce from Scratch: first principles with python* (1st ed.). O'Reilly Media.

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '04*, 168. https://doi.org/10.1145/1014052.1014073

Hussein, D. M. E.-D. M. (2018). A survey on sentiment analysis challenges. *Journal of King Saud University - Engineering Sciences*, *30*(4), 330–338. https://doi.org/10.1016/j.jksues.2016.04.002

Jianqiang, Z., & Xiaolin, G. (2017). Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis. *IEEE Access*, *5*, 2870–2879. https://doi.org/10.1109/ACCESS.2017.2672677

Kao, A., & Poteet, S. R. (2007). *Natural Language Processing and Text Mining* (1st ed.). Springer London. https://doi.org/10.1007/978-1-84628-754-1

Kaur, H., Mangat, V., & Nidhi. (2017). A survey of sentiment analysis techniques. *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 921–925. https://doi.org/10.1109/I-SMAC.2017.8058315

Kharde, V. A., & Sonawane, S. S. (2016). Sentiment Analysis of Twitter Data: A Survey of Techniques. *International Journal of Computer Applications*, *139*(11), 5–15. https://doi.org/10.5120/ijca2016908625

Li, W., Shao, W., Ji, S., & Cambria, E. (2022). BiERU: Bidirectional emotional recurrent unit for conversational sentiment analysis. *Neurocomputing*, *467*, 73–82. https://doi.org/10.1016/j.neucom.2021.09.057

Liang, B., Su, H., Gui, L., Cambria, E., & Xu, R. (2022). Aspect-based sentiment analysis via affective knowledge enhanced graph convolutional networks. *Knowledge-Based Systems*, *235*, 107643. https://doi.org/10.1016/j.knosys.2021.107643

Lin, P., Luo, X., & Fan, Y. (2020). A Survey of Sentiment Analysis Based on Deep Learning. *International*

Journal of Computer and Information Engineering, *14*(12), 473–485. https://publications.waset.org/10011630/a-survey-of-sentiment-analysis-based-on-deep-learning

Liu, B. (2010). Sentiment analysis and subjectivity. *Handbook of Natural Language Processing, Second Edition*, *2*, 627–666. https://www.cs.uic.edu/~liub/FBS/NLP-handbook-sentiment-analysis.pdf

Lovera, F. A., Cardinale, Y. C., & Homsi, M. N. (2021). Sentiment Analysis in Twitter Based on Knowledge Graph and Deep Learning Classification. *Electronics*, *10*(22), 2739. https://doi.org/10.3390/electronics10222739

Lunt, M. (2015). Introduction to statistical modelling: linear regression. *Rheumatology*, *54*(7), 1137–1140. https://doi.org/10.1093/rheumatology/ket146

Martínez Cámara, E., Rodríguez Barroso, N., Moya, A. R., Fernández, J. A., Romero, E., & Herrera, F. (2019). *Deep Learning Hyper-parameter Tuning for Sentiment Analysis in Twitter based on Evolutionary Algorithms*. 255–264. https://doi.org/10.15439/2019F183

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations ofwords and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 1–10. https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Mirtalaie, M. A., Hussain, O. K., Chang, E., & Hussain, F. K. (2018). Sentiment Analysis of Specific Product's Features Using Product Tree for Application in New Product Development. In *Advances in Intelligent Networking and Collaborative Systems* (8th ed., pp. 82–95). Springer. https://doi.org/10.1007/978-3-319-65636-6_8

Mostafa, M. M. (2013). More than words: Social networks' text mining for consumer brand sentiments. *Expert Systems with Applications*, *40*(10), 4241–4251. https://doi.org/10.1016/j.eswa.2013.01.019

Oliveira, D. J. S., Bermejo, P. H. de S., & dos Santos, P. A. (2017). Can social media reveal the preferences of voters? A comparison between sentiment analysis and traditional opinion polls. *Journal of Information Technology & Politics*, *14*(1), 34–45. https://doi.org/10.1080/19331681.2016.1214094

Plisson, J., Lavrac, N., & Mladenić, D. D. (2004). A rule based approach to word lemmatization. *Proceedings of the 7th International Multiconference Information Society (IS'04)*, 83–86. http://eprints.pascal-network.org/archive/00000715/

Ribeiro, F. N., Araújo, M., Gonçalves, P., André Gonçalves, M., & Benevenuto, F. (2016). SentiBench - a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ Data Science*, *5*(1), 23. https://doi.org/10.1140/epjds/s13688-016-0085-1

Tabinda Kokab, S., Asghar, S., & Naz, S. (2022). Transformer-based deep learning models for the sentiment analysis of social media data. *Array*, *14*, 100157. https://doi.org/10.1016/j.array.2022.100157

Wu, Y., Zhang, Q., Huang, X., & Wu, L. (2011). Structural opinion mining for graph-based sentiment representation. *Empirical Methods in Natural Language Processing*, 1332–1341. https://aclanthology.org/D11-1123.pdf

Xu, S. (2018). Bayesian Naïve Bayes classifiers to text classification. *Journal of Information Science*, *44*(1), 48–59. https://doi.org/10.1177/0165551516677946

Yujian, L., & Bo, L. (2007). A Normalized Levenshtein Distance Metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 1091–1095. https://doi.org/10.1109/TPAMI.2007.1078

Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *WIREs Data Mining and Knowledge Discovery*, *8*(4). https://doi.org/10.1002/widm.1253