IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Fecha de publicación Junio 05, 2023, fecha de la última versión Junio 05, 2023

Digital Object Identifier Equipo2

Impacto del tamaño de secuencias de texto en el rendimiento de modelos de aprendizaje profundo para clasificación de sentimiento.

PIÑEIRO A.[1], YÉPEZ F.[2], GUERRA A.[3] y LUNA A. [4]
[1]Instituto Tecnológico y de Estudios Superiores de Monterrey. México, N.L 64849 (e-mail: A01705681@tec.mx)
[2]Instituto Tecnológico y de Estudios Superiores de Monterrey. México, N.L 64849 (e-mail: A01658002@tec.mx)
[3]Instituto Tecnológico y de Estudios Superiores de Monterrey. México, N.L 64849 (e-mail: A00828452@tec.mx)
[4]Instituto Tecnológico y de Estudios Superiores de Monterrey. México, N.L 64849 (e-mail: A01177358@tec.mx)

Este trabajo fue apoyado por el Tecnológico de Monterrey y los maestros del TC3002B.302.

ABSTRACT En este artículo, presentamos nuestra investigación acerca de cómo el tamaño máximo de entrada de una reseña en texto plano afecta el rendimiento de diferentes modelos de aprendizaje profundo en la clasificación de sentimiento. Se comparó el rendimiento de diferentes longitudes máximas de texto en una red neuronal profunda, una red neuronal convolucional y una LSTM. Para cada modelo y longitud de palabras máxima utilizada, realizamos la comparación de entrenar la primera capa de embedding y utilizar una pre-entrenada por Google conocida como Word2Vec con la finalidad de determinar qué efecto tiene esto en la clasificación de sentimiento.
Utilizamos una base de datos pública que contiene reseñas de películas en texto plano y la clasificación de cada una. Para probar con diferentes tamaños de reseña, creamos conjuntos de datos con diferente número de palabras permitidas, así como una matriz que traduce cada palabra encontrada en el corpus de los datos a su respectivo vector de Word2Vec.
Previo a entrenar los modelos, realizamos limpieza de los datos con la finalidad de que los modelos puedan utilizar la información más relevante de cada reseña para aprender a clasificar el sentimiento de las mismas. Seguidamente, entrenamos los modelos utilizando cada conjunto de datos generado tanto utilizando los vectores de Word2Vec como entrenando la capa de embedding.
Nuestra investigación revela que el tamaño de cada reseña de película tiene un impacto en la exactitud de los modelos de aprendizaje profundo así como la decisión de entrenar o utilizar vectores pre-entrenados para la capa de embedding.
Observamos que la red LSTM obtuvo los mejores resultados de clasificación de sentimiento alcanzando un 88.08% de exactitud. En general, fue beneficioso para todos los modelos probados incrementar la longitud de palabras por reseña. En el caso de las redes DNN y LSTM, fue mejor entrenar la capa de embedding mientras que en la red CNN fue mejor utilizar los vectores pre-entrenados de Word2Vec.
Esta investigación proporciona evidencia que muestra cómo el tamaño usado de cada reseña influye en la precisión de modelos de aprendizaje profundo. Los resultados son relevantes para profesionales en el campo de la inteligencia artificial que desean diseñar clasificadores de sentimiento a partir de texto plano y buscan determinar la mejor forma de ingresar sus datos a una capa de embedding en modelos de aprendizaje profundo.

INDEX TERMS Clasificación de sentimiento, tamaño de secuencias de texto, red neuronal, LSTM, CNN, DNN, aprendizaje profundo, inteligencia artificial, embedding.

I. INTRODUCCIÓN

LA comunicación en la web entre diferentes individuos ha estado presente desde los primeros días del Internet. Al comunicarse, se expresan emociones como felicidad, tristeza, enojo y muchas más. El poder analizar los sentimientos transmitidos en texto es algo que puede

ser de gran utilidad para muchas empresas al igual que para poder controlar y filtrar los mensajes que son compartidos en la web. El problema con esto es que la cantidad de textos es enorme y por ende es imposible pedir que una persona esté revisando cada mensaje. Para este tipo de problemas se ha creado el Aprendizaje Automático (Machine Learning) que es la parte de la inteligencia artificial que se centra principalmente en desarrollar sistemas que aprenden por sí mismos. Estos sistemas pueden llegar a ser automatizados a un nivel extraordinario lo cuál permite que se analicen enormes cantidades de texto en segundos.

El Análisis de Semántica (Sentiment Analysis) en línea consiste en evaluar opiniones y emociones en grandes volúmenes de datos de texto. La detección de estas emociones proporciona información valiosa que se puede utilizar en una variedad de cosas incluyendo la mejora de toma de decisiones empresariales, la satisfacción del cliente y la comprensión de los mercados. [7]

Los servicios de streaming compiten entre sí para poder brindar a sus usuarios la colección más completa y de mayor interés con la finalidad de que opten por su suscripción. Dado el costo de mantener video y audio disponible en todo momento a lo largo de varias regiones en los servidores de estos servicios, es necesario contar con un método inteligente que permita separar aquellas películas que los usuarios desean que estén disponibles en sus catálogos y las que ya no desean más. Para esta investigación se nos presentó una base de datos que contiene miles de comentarios de distintas películas que son clasificados en dos categorías: positivo y negativo. Nuestro objetivo en esta investigación es comparar diferentes modelos de inteligencia artificial, determinar el tamaño de textos ideal para maximizar la eficiencia de la capa de embedding y clasificar de mejor forma el sentimiento de reseñas de películas.

II. FUNDAMENTOS

Nuestro trabajo está inspirado en modelos de aprendizaje profundo por lo que es necesario entender el funcionamiento de cada uno.

A. LSTM

Long Short-Term Memory (LSTM) es un tipo de red neuronal recurrente que ha demostrado ser de alta eficiencia en el procesamiento de secuencia de datos y en la forma que maneja problemas de larga duración o dependencias a largo plazo. A diferencia de las redes neuronales recurrentes, las cuales en ciertos casos tienen dificultades para recordar información de relevancia a medida que el tiempo pasa, las LSTM están diseñadas específicamente para solucionar este problema.

La arquitectura de una LSTM se compone de unidades de memoria llamadas "celdas". Cada celda contiene una estructura interna que contiene las siguientes piezas: una puerta de olvido, una puerta de entrada, una puerta de salida y una memoria de largo plazo. Estas puertas permiten que la celda decida qué información retener, que información descartar y qué información emitir en cada paso de tiempo. [15]

Debido a esta estructura de puertas y memoria de largo plazo, las LSTM son capaces de capturar y recordar patrones de largo alcance en los datos secuenciales. Esto las hace de alta utilidad para tareas como el reconocimiento de voz, la traducción automática y el procesamiento del lenguaje natural, entre otros.

B. CNN

Una red neuronal convolucional (CNN) es un tipo de arquitectura diseñada para el procesamiento de datos con estructura espacial, como imágenes y señales de audio. A diferencia de las redes neuronales tradicionales, las CNN utilizan capas convolucionales que aplican filtros para poder extraer características locales de los datos de entrada.

Además de las capas convolucionales, las CNN suelen incorporar capas de agrupación, como el max pooling, para reducir la dimensionalidad de los mapas de características y mejorar la eficiencia computacional. También se pueden utilizar capas de normalización para mejorar la estabilidad del entrenamiento. Las CNN son altamente efectivas en tareas de visión por computadora, como reconocimiento facial, detección de objetos y clasificación de imágenes, ya que son capaces de aprender representaciones jerárquicas y complejas de manera automática. [16]

En nuestra investigación del estado de arte encontramos un articuló sobre los resultados que tuvo una red neuronal convolucional de siete capas con la finalidad de clasificar comentarios de películas como positivos o negativos. Los resultados obtenidos fueron altos con un 91% de exactitud. [10]

C. DNN

Las redes neuronales profundas (DNN) son algoritmos de aprendizaje profundo inspirados en el cerebro humano. Estas redes están compuestas por múltiples capas de neuronas artificiales interconectadas. Cada neurona combina las entradas de las capas anteriores utilizando pesos y una función de activación para generar una salida. Durante el proceso de alimentación de los datos a través de las distintas capas, las DNN pueden aprender patrones y características complejas mediante el ajuste de los pesos de las conexiones. Las DNN han mostrado ser altamente eficientes en tareas como visión por computadora, procesamiento de lenguaje natural y reconocimiento de voz, entre otros. Su capacidad para modelar relaciones no lineales y procesar grandes cantidades de datos las ha convertido en herramientas poderosas en diversas áreas. [18]

D. WORD EMBEDDING

Una red neuronal con word embeddings (incrustaciones de palabras) es un tipo de arquitectura que permite representar palabras en un espacio vectorial continuo. Estas representaciones vectoriales capturan el significado semántico y las relaciones entre palabras, lo cual facilita el procesamiento del lenguaje natural. Al entrenar la red neuronal con grandes cantidades de texto, los word embeddings aprenden a asignar vectores cercanos a palabras que tienen significados similares o que se usan en contextos similares. Esto permite realizar tareas como clasificación de texto, análisis de semántica y traducción automática de manera más eficiente y precisa. [18]

III. MÉTODO PROPUESTO

A. JUSTIFICACIÓN

El tamaño de las cadenas de texto puede tener un impacto significativo en la eficiencia computacional de los modelos de aprendizaje automático [17]. Si las cadenas son demasiado largas, aumenta la carga computacional necesaria para procesarlas y almacenarlas. Por otro lado, si las cadenas son demasiado cortas, se puede perder información importante. Una investigación sobre el tamaño ideal nos ayuda a determinar la longitud óptima que maximice la eficiencia computacional sin comprometer el rendimiento del modelo. En el artículo de "A deep neural network-based approach for sentiment analysis of movie reviews." escrito por Ullah, K., Rashad, A., Khan, M., Ghadi, Y., Aljuaid, H., y Nawaz, Z., en el 2022; los autores mencionan en las conclusiones que el tamaño de entrada fue uno de los parámetros clave para obtener mejores resultados. Para determinar el tamaño ideal decidimos comparar tres tamaños diferentes de cadenas de texto con dos tipos de embedding (uno propio y el Word2Vec de Google) y tres tipos de modelos (DNN, CNN, LSTM).

B. BASE DE DATOS

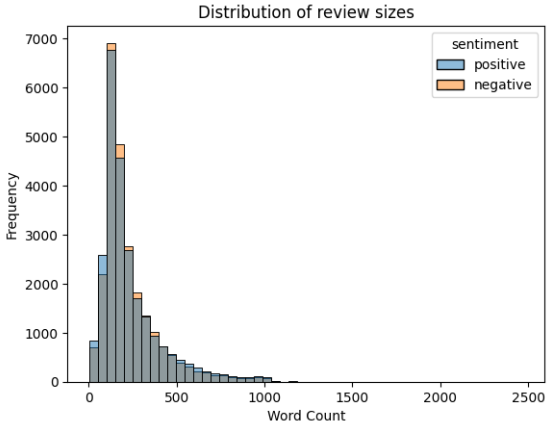
La elección de la base de datos es primordial para el experimento, pues representa la información con la cual entrenamos al modelo. Debe ser relevante, con información precisa y actualizada para que el modelo de inteligencia artificial tome decisiones significativas.

La base de datos seleccionada para el experimento es IMDB Dataset of 50K Movie Reviews, creada por la Universidad de Stanford. La cual cuenta con 50,000 comentarios de IMDB, 50% de ellos son positivos y el otro 50% son negativos; esto es muy importante ya que nos ayuda a evitar sesgo y tener mayor generalización al ser presentado con nuevos datos.

Realizamos un análisis de la base de datos con el cual concluimos que la base de datos cuenta con la información necesaria para realizar el experimento. En el análisis confirmamos que los datos están balanceados con 50.19% de comentarios positivos y 49.81% de datos negativos; la diferencia entre ambos es muy pequeña. Existen 418 datos duplicados en la base de datos, sin embargo, aún excluyéndose hay una gran cantidad de datos para entrenar y probar el modelo.

Algo muy relevante en el análisis de datos fue el número de palabras de los comentarios, ya que el tamaño de la entrada para el modelo es un factor determinante en los resultados del mismo. En el artículo "A deep neural network-based approach for sentiment analysis of movie reviews." escrito por Ullah, K., Rashad, A., Khan, M., Ghadi, Y., Aljuaid, H., & Nawaz, Z., en el 2022; se creó una red neuronal convolucional de 7 capas, la cual fue probada con la misma base de datos que usamos para el experimento. En dicho artículo los autores descubrieron que el tamaño de los comentarios para el input afecta, y los mejores resultados se obtuvieron con un tamaño de 1200 caracteres.

En el análisis de datos observamos que la distribución del tamaño de palabras tiene sesgo a la derecha; con un promedio de aproximadamente 230 palabras por comentario. Como se puede observar en la siguiente imagen, la distribución del tamaño de oración entre comentarios positivos y negativos está balanceada.



Esta imágen pertenece a los datos encontrados en la base de datos sin ningún tipo de pre-procesamiento.

C. EMBEDDING

Los algoritmos de aprendizaje automático pueden procesar datos textuales, pero requieren una representación numérica del texto para trabajar con él de manera efectiva. Para poder procesar el texto de nuestras reseñas utilizamos técnicas como tokenización y embedding.

La tokenización es el proceso de descomponer el texto en unidades más pequeñas llamadas tokens, que suelen ser palabras. A cada token se le asigna un identificador numérico único o un índice. Este paso permite que el texto sea procesado por modelos de aprendizaje automático, ya que la mayoría de los modelos requieren que las entradas sean numéricas.

El embedding de texto, es una forma de representar palabras u oraciones como vectores numéricos. Se asignan palabras a representaciones vectoriales, donde las palabras con significados o contextos similares están más cerca entre sí. Cuando se aplica embedding en oraciones se tienen como objetivo capturar el significado o la información semántica de una oración completa en una representación vectorial. Esta técnica permite que los modelos de aprendizaje automático aprendan relaciones y patrones en datos textuales. [18]

En nuestra investigación realizamos el embedding de nuestro dataset de manera manual y utilizando transfer learning con el modelo de Google Word2Vec. Antes de poder utilizar el Word2Vec o nuestra propia capa de embedding se realizo una tokenización de nuestro dataset. Para esto utilizamos la función de Tokenizer() de la librería de TensorFlow donde recibe nuestra lista de reseñas y nos retorna una matriz de vectores donde cada palabra de las reseñas recibe un numero único basado en la repetición que tiene.

Para realizar nuestro propio Embedding() también se utilizo la librería de TensorFlow. Esta función actúa como la primera capa de nuestros modelos y recibe el tamaño de vocabulario, tamaño de la dimensión de embedding y tamaño de las cadenas de texto. Del otro lado, para utilizar el embedding Word2Vec de Google solo se importa su API y se entrega la matriz de vectores que se creó anteriormente aacediendo a los vectores de cada palabra presente en el vocabulario de las reseñas. El retorno de estos dos métodos es una matriz donde cada palabra de nuestro vocabulario esta numéricamente mas cerca a palabras similares facilitando la generación de patrones para el resto de capas de nuestro modelo.

D. ARQUITECTURAS DE MODELOS

Comparamos 3 tipos de redes neuronales: CNN (Convolutional Neural Network), DNN (Deep Neural Network) y LSTM (Long Short-Term Memory), con la finalidad de determinar el tipo de arquitectura con mejores resultados para las diferentes configuraciones de embedding. Para las redes neuronales se uso la librería de Keras, que permite crear y entrenar modelos de redes neuronales rápida y sencillamente.

1) DNN

Para la red neuronal profunda, probamos con una arquitectura muy básica, con la finalidad de verificar si compite con redes más complejas en cuanto a resultados y tiempo.

La red cuenta con 5 capas:

1) Embedding: Mapea las palabras a vectores de números reales. Hiper parámetros:
   - input dim: Dimensión de Entrada
   - output dim: Dimensión de Salida
   - input length: Tamaño de las secuencias de input.
   - input length: Tamaño de las secuencias de input.
   - weights: Matriz pre-entrenada. (Se usa con embedding Word2Vec)
   - trainable: Booleano para configurar si la capa es entrenable o no. (Se usa con embedding Word2Vec)
2) Flatten: Convierte los datos en un vector unidimensional.
   - data format: Orden de las dimensiones en las entradas.
3) Dense: Capa que captura relaciones entre los datos y permite aprender representaciones más complejas.
   - units: Dimensionalidad del espacio de salida.
   - activation: Función de activación.
   - kernel initializer: Función de regularización aplicada al vector de sesgo.
4) Droput: Evita el sobreajuste, "apagando" algunas neuronas durante el entrenamiento.
   - rate: Valor entre 0 y 1. Fracción de las unidades del input a "apagar".
5) Dense: Capa de salida para realizar la clasificación binaria.
   - units: Dimensionalidad del espacio de salida.
   - activation: Función de activación.
   - kernel initializer: Función de regularización aplicada al vector de sesgo.

2) CNN

Para la red neuronal convolucional nos basamos en la arquitectura propuesta por el artículo de "A deep neural network-based approach for sentiment analysis of movie reviews." escrito por Ullah, K., Rashad, A., Khan, M., Ghadi, Y., Aljuaid, H., y Nawaz, Z., en el 2022. Estas capas densas permiten aprender representaciones más complejas y capturar relaciones no lineales en los datos de entrada.

La red cuenta con 7 capas:

1) Embedding: Mapea las palabras a vectores de números reales. Hiper parámetros:
   - input_dim: Dimensión de Entrada
   - output_dim: Dimensión de Salida
   - input_length: Tamaño de las secuencias de input.
   - input_length: Tamaño de las secuencias de input.
   - weights: Matriz pre-entrenada. (Se usa con embedding Word2Vec)
   - trainable: Booleano para configurar si la capa es entrenable o no. (Se usa con embedding Word2Vec)
2) Conv1d: Extrae características importantes de los datos de entrada y se crea un mapa de características usando un filtro o kernel. Una capa

convolucional extrae más características usando menores neuronas comparado a una capa densa.
- filters: Dimencionalidad del espacio de salida.
- kernel_size: Especifica la longitud de la ventana de convolución 1D.
- activation: Función de activación.

3) Conv1d: Resume las características seleccionadas en la capa anterior.
- filters: Dimencionalidad del espacio de salida.
- kernel_size: Especifica la longitud de la ventana de convolución 1D.
- activation: Función de activación.

4) GlobalMaxPooling1D: Reduce las dimensiones de las características y previene sobre ajuste. Con max-pooling el máximo de elementos se seleccionan del mapa de características
- data format: Orden de las dimensiones en las entradas.

5) Dense: Capa completamente conectada que captura relaciones entre los datos y permite aprender representaciones más complejas.
- units: Dimensionalidad del espacio de salida.
- activation: Función de activación.
- kernel_initializer: Función de regularización aplicada al vector de sesgo.

6) Droput: Evita el sobreajuste, "apagando" algunas neuronas durante el entrenamiento.
- rate: Valor entre 0 y 1. Fracción de las unidades del input a "apagar".

7) Dense: Capa de salida para realizar la clasificación binaria.
- units: Dimensionalidad del espacio de salida.
- activation: Función de activación.
- kernel_initializer: Función de regularización aplicada al vector de sesgo.

3) LSTM

Para la red neuronal de Memoria a Corto Plazo, probamos con una arquitectura muy básica, para verificar si LSTM mejora en gran manera los resultados y tiempo. Usamos una arquitectura similar a una red neuronal profunda, con la diferencia de una capa extra de LSTM. Las capas LSTM tienen unidades de memoria que pueden recordar información de secuencias anteriores y utilizarla para tomar decisiones en secuencias futuras. Esto es especialmente útil en el análisis de sentimientos, donde el contexto pasado puede afectar el sentimiento actual.

1) Embedding: Mapea las palabras a vectores de números reales. Hiper parámetros:
- input_dim: Dimensión de Entrada
- output_dim: Dimensión de Salida
- input_length: Tamaño de las secuencias de input.
- weights: Matriz pre-entrenada. (Se usa con embedding Word2Vec)
- trainable: Booleano para configurar si la capa es entrenable o no. (Se usa con embedding Word2Vec)

2) LSTM: Extrae características importantes de los datos de entrada y se crea un mapa de características usando un filtro o kernel. Una capa convolucional extrae más características usando menores neuronas comparado a una capa densa.
- units: Dimencionalidad del espacio de salida.
- dropout: Valor entre 0 y 1. Fracción de las unidades de la transformación lineal a "apagar".
- recurrent_dropout: Valor entre 0 y 1. Fracción de las unidades de la transformación lineal del estado recurrente a "apagar".

3) Dense: Capa completamente conectada que captura relaciones entre los datos y permite aprender representaciones más complejas.
- units: Dimensionalidad del espacio de salida.
- activation: Función de activación.
- kernel_initializer: Función de regularización aplicada al vector de sesgo.

4) Droput: Evita el sobreajuste, "apagando" algunas neuronas durante el entrenamiento.
- rate: Valor entre 0 y 1. Fracción de las unidades del input a "apagar".

5) Dense: Capa de salida para realizar la clasificación binaria.
- units: Dimensionalidad del espacio de salida.
- activation: Función de activación.
- kernel_initializer: Función de regularización aplicada al vector de sesgo.

IV. METODOLOGÍA

A. PREPROCESAMIENTO DE DATOS

La base de datos cuenta con información suficiente para las pruebas que realizamos. Sin embargo, fue necesario preparar los datos de mejor manera antes de comenzar a construir el modelo. Con el objetivo de que el entrenamiento fuera más rápido y mejorar la efectividad del modelo, los pasos del preprocesamiento de datos fueron los siguientes:

1) Eliminar información irrelevante, conocida como ruido (tags de HTML, símbolos).
2) Eliminar palabras irrelevantes que no aportan contexto ni sentido a clasificar sentimiento. (Palabras comunes como: 'the', 'a', 'in')
3) Eliminar puntuación debido a que no mejora el análisis de texto y simplifica el espacio de características.

4) Eliminar cadenas de texto que solo tienen un carácter y reemplazarlas con un espacio en blanco.
5) Eliminar múltiples espacios en blanco.
6) Convertir mayúsculas a minúsculas para mejorar la eficiencia de entrenamiento debido a que los algoritmos de ML no diferencian ambas letras

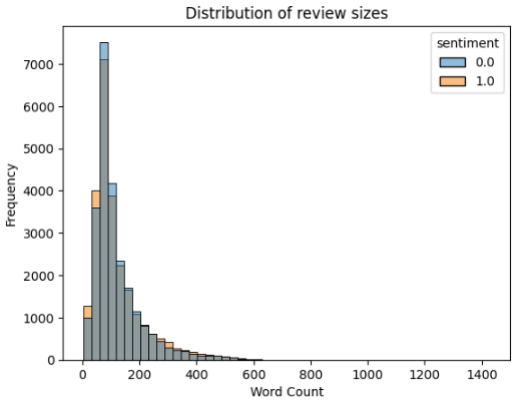
Después del procesamiento de datos, el número de palabras para cada reseña disminuye. Calculamos las nuevas medidas de tendencia central y las comparamos con las medidas antes del pre-procesamiento.

	review_size
count	49582.000000
mean	231.350167
std	171.542020
min	4.000000
25%	126.000000
50%	173.000000
75%	281.000000
max	2470.000000

FIGURE 1. Antes del pre procesamiento.

	review_size
count	49582.000000
mean	119.876850
std	90.066502
min	3.000000
25%	64.000000
50%	89.000000
75%	146.000000
max	1429.000000

FIGURE 2. Después del pre procesamiento.



La distribución del tamaño de reseñas queda bastante similar a la distribución anterior tan solo que el tamaño de las reseñas disminuyó notablemente al haber depurado la información existente en las reseñas y mantener solo aquellas palabras fundamentales para clasificar sentimiento.

B. HIPER PARÁMETROS PARA CADA MODELO

Para cada tipo de red neuronal usamos diferentes hiper parámetros. Los hiper parámetros de la capa de Embedding son los únicos que cambian para las pruebas; por lo que a continuación presentamos los valores de los hiper parámetros fijos en todas las otras capas en cada modelo. Los hiper parámetros usados para la capa de Embedding se encuentran en las secciones posteriores.

1) DNN

Elegimos los hiper parámetros de la red neuronal profunda usando un modelo de red neuronal muy sencillo.

Para la tasa de dropout nos basamos en el artículo de "Improving Neural Networks with Dropout" de Srivastava, N., en 2013. Este artículo menciona que es óptimo usar una tasa de dropout de entre 20% y 50%.

A continuación mostramos los valores para cada hiper parámetro:

1) Flatten: No recibe hiper parámetros.
2) Dense:
   - Neuronas: 8
   - Activación: 'relu'
   - Inicilizador de Kernel: HeUniform()
3) Droput:
   - Tasa: 0.4
4) Dense:
   - Neuronas: 1
   - Activación:'sigmoid'

2) CNN

Elegimos los hiper parámetros de la red neuronal convolucional basados en el artículo "A deep neural network-based approach for sentiment analysis of movie reviews." escrito por Ullah, K., Rashad, A., Khan, M., Ghadi, Y., Aljuaid, H., y Nawaz, Z., en el 2022.

A continuación mostramos los valores para cada hiper parámetro:

1) Conv1d:
   - Filtros: 128
   - Tamaño de Kernel: 3
   - Activación: 'relu'
2) Conv1d:
   - Filtros: 64
   - Tamaño de Kernel: 3
   - Activación: 'relu'
3) GlobalMaxPooling1D: No recibe hiper parámetros.
4) Dense:
   - Neuronas: 50
   - Activación: 'relu'
5) Droput:
   - Tasa: 0.3
6) Dense:
   - Neuronas: 1
   - Activación: 'sigmoid'

3) LSTM

Elegimos los hiper parámetros de la red neuronal profunda basados en el artículo "Comparative Study of Deep Learning-Based Sentiment Classification" escrito por S. Seo, C. Kim, H. Kim, K. Mo and P. Kang, en 2020. en el que usan un número de 128 celdas para LSTM.

A continuación mostramos los valores para cada hiper parámetro:

1) LSTM:
   - Celdas: 128
   - Dropout: 0.2
   - Dropout Recurrente: 0.2
2) Dense:
   - Neuronas: 64
   - Activación: 'relu'
   - Inicilizador de Kernel: HeUniform()
3) Droput:
   - Tasa: 0.25
4) Dense:
   - Neuronas: 1
   - Activación: 'sigmoid'

C. PARÁMETROS PARA EMBEDDING

1) Embedding Propio

1) Tamaño de Vocabulario:
   - input_dim: 22,175 equivalente al 10% de tokens de todo nuestro data set. Quisimos considerar la décima parte de las palabras más repetidas para nuestra experimentación.
2) Tamaño de Dimensiones:
   - output_dim: 300D debido a que es el mismo que utiliza el Word2Vec de Google y de esta forma se podrá realizar una comparación utilizando la misma dimensión de vectores.
3) Tamaño de las cadenas de texto:
   - input_length: Esta variable es la que probamos en nuestra investigación y hablaremos de esta en el siguiente inciso.

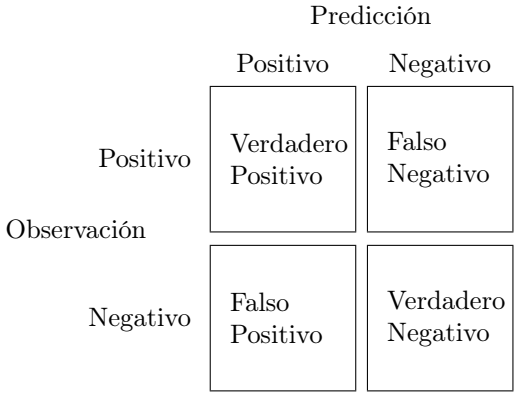
2) Embedding Word2Vec

1) Tamaño de Vocabulario:
   - input_dim: 22,175 equivalente al 10% de tokens de todo nuestro dataset.
2) Tamaño de Dimensiones:
   - output_dim: 300D, valor por default.
3) Tamaño de las cadenas de texto:
   - input_length: Esta variable es la que probamos en nuestra investigación y hablaremos de esta en el siguiente inciso.

D. MÉTRICAS DE EVALUACIÓN

Para evaluar la efectividad de nuestros modelos de Deep Learning, utilizaremos métricas comunes de clasificación binaria dado que en este caso contamos con reseñas positivas o negativas. La ventaja de realizar aprendizaje supervisado es que contamos con las etiquetas correctas y se las puede utilizar para evaluar el desempeño de las predicciones realizadas por el modelo.

En este sentido, utilizaremos una matriz de confusión para mostrar el número de clasificaciones correctas e incorrectas realizadas por cada modelo. La matriz de confusión se puede utilizar para calcular varias medidas estadísticas como la exactitud (accuracy), precisión, sensibilidad (recall) y F1-score. Una matriz de confusión se ve de la siguiente forma:



La exactitud o accuracy, muestra la proporción de registros clasificados correctamente, en este caso, clasificar una reseña como positiva o negativa según sea el caso. La precisión mide la proporción de registros clasificados correctamente de aquellos clasificados como positivos, un alto valor de precisión indica que se tiene una baja tasa de falsos positivos.

La sensibilidad o recall, muestra la proporción de registros correctamente clasificados como positivos, un alto valor de recall indica que se tiene una baja tasa de falsos negativos. Por último, el F1-score combina la precisión con el recall para conocer el rendimiento del modelo al clasificar correctamente los registros de ambas categorías. Calculamos dichas métricas durante las pruebas de cada modelo a través de la librería Keras con su módulo de métricas.

E. PRUEBAS

Se decidió utilizar tres versiones diferentes de nuestro dataset para alimentar a nuestras dos versiones de embedding y posteriormente a nuestros tres modelos de aprendizaje profundo. Las tres versiones del dataset varían en la cantidad máxima de palabras que tienen:

1) 89 palabras: la mediana del dataset.
2) 120 palabras: el promedio del dataset.
3) 145 palabras: el tercer cuartil del dataset.

El número máximo de palabras en las tres versiones de nuestros dataset fueron tomadas desde el lado derecho hacia el izquierdo, es decir, se tomaron las ultimas palabras de cada reseña. Decidimos eliminar las primeras palabras de las reseñas de películas ya que se obtienen múltiples beneficios al hacer esto. En primer lugar, permite filtrar información introductoria o contextual que no tiene relevancia al tratar de analizar el sentimiento principal de la reseña. Al hacer esto, el modelo

puede centrarse en el contenido principal y mejor la probabilidad de capturar el sentimiento general referente a la película. También, eliminar las primeras palabras permite que se mantenga la última parte de la reseña donde regularmente se expresa de manera más clara y directa el sentimiento más impactante o crítico del texto. Esta estrategia también puede ayudar a reducir el ruido y mejorar la precisión del análisis al eliminar información preliminar o descriptiva que no contribuye de forma significativa a la opinión general sobre la película. [19] Al final, eliminar las primeras palabras puede proporcionar una representación más precisa del sentimiento central de la reseña.

1) Tabla de pruebas

	Tamaño de reseñas	Tipo de Embedding	# Prueba
DNN	89	Embedding Propio	1
		Word2Vec	2
	120	Embedding Propio	3
		Word2Vec	4
	145	Embedding Propio	5
		Word2Vec	6
CNN	89	Embedding Propio	7
		Word2Vec	8
	120	Embedding Propio	9
		Word2Vec	10
	145	Embedding Propio	11
		Word2Vec	12
LSTM	89	Embedding Propio	13
		Word2Vec	14
	120	Embedding Propio	15
		Word2Vec	16
	145	Embedding Propio	17
		Word2Vec	18

F. ENTRENAMIENTO

Los algoritmo de optimización que usaremos para la compilación de los modelos son Adam y RMSProp. Ambos son ampliamente considerados como unos de los mejores algoritmos de optimización en el aprendizaje profundo. Adam tiene la capacidad de adaptar la tasa de aprendizaje de forma individual para cada parámetro del modelo, lo cual permite un ajuste fino durante el proceso de entrenamiento. Al combinar conceptos del descenso de gradiente estocástico y el método de momento adaptativo, Adam logra una convergencia más rápida y estable del modelo. Para nuestro modelo, es adecuado utilizar estos algoritmos de optimización para su entrenamiento debido a su eficiencia computacional que permite un procesamiento rápido. Adam proporciona adaptabilidad, mejora de la convergencia y manejo efectivo de gradientes dispersos o de gran magnitud, lo que lo convierte en una opción ventajosa para el análisis de semántica al capturar de manera más precisa los patrones semánticos y el sentimiento en los textos. [14]

Para nuestro modelo, optamos por una división de datos en una proporción de 80-20, donde el 80% de los datos se utilizaron para entrenar el modelo y el 20% restante se reservó para pruebas. Además, dentro de los datos de entrenamiento, se asignó el 15% para la validación del modelo con la finalidad de realizar early stopping y evitar sobre-ajustar el modelo a los datos de entrenamiento.

V. RESULTADOS

Presentamos los resultados obtenidos al evaluar el rendimiento de cada una de las pruebas definidas anteriormente con sus respectivos conjuntos de evaluación. Utilizamos cada métrica de evaluación para determinar el rendimiento de cada modelo entrenado.

Realizamos las 18 pruebas definidas en la tabla de pruebas. El objetivo era determinar qué modelo tuvo mejores resultados, con qué tipo de embedding y con qué longitud de reseñas.

Como mencionamos en el método propuesto, evaluamos las diferentes configuraciones obteniendo la matriz de confusión. A continuación, mostraremos los resultados obtenidos:

Como se puede ver en la gráfica cada longitud de reseña se probó para cada modelo tanto entrenando la capa de embedding como utilizando Word2Vec. Cada prueba calculó las métricas de evaluación.

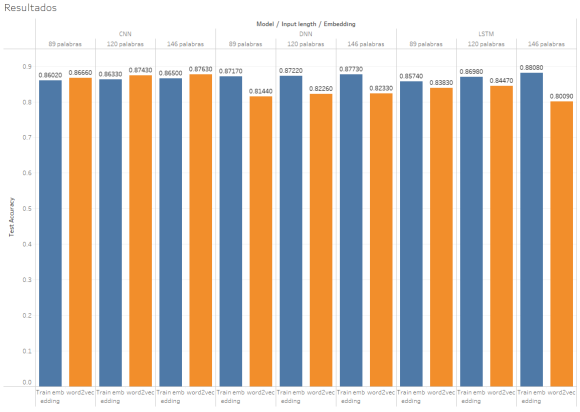
Obtuvimos que la mejor exactitud o accuracy alcanzado de todos los modelos probados fue de 88.08% en la red neuronal LSTM con la longitud de palabras más larga para cada reseña equivalente a 146 palabras que es el valor del tercer cuartil de la longitud total de todas las reseñas del dataset tras realizar el pre-procesamiento de los datos. En general se puede observar que la exactitud de todos los modelos es similar entre sí. Si el objetivo es clasificar correctamente la mayor parte de las reseñas de películas, el modelo mencionado será el más óptimo.

En cuanto a precisión del modelo, el mejor resultado obtenido fue nuevamente de la red LSTM con 92.54% para reseñas con longitud equivalente a la media de palabras por reseña. En otras palabras, este es el modelo con menor tasa de falsos positivos o registros negativos clasificados como positivos.

El modelo con mejor sensibilidad o recall fue la red LSTM obteniendo un valor de 96.19% para reseñas con longitud de 146 palabras utilizando Word2Vec en la capa de embedding. Si lo que se busca es clasificar correctamente reseñas positivas, este deberá ser el modelo seleccionado.

Para conocer el modelo que clasifica de forma correcta tanto reseñas positivas como negativas es necesario emplear el F-1score, resultando en que el valor más alto de 87.85% pertenece de igual forma a la red LSTM que utiliza 146 palabras por reseña.

Dependiendo del caso de uso, se podrá seleccionar un modelo diferente dependiendo de lo que se busca lograr por lo que cada una de estas métricas resulta útil. Habrán casos en los que se opta por un modelo que clasifica correctamente las películas que le ocasionan un sentimiento positivo a las personas para priorizar que se encuentren dentro del catálogo de un servicio de streaming mientras que si se opta por un modelo que prioriza categorizar correctamente sentimientos negativos, este podrá ser usado para determinar qué películas remover del catálogo del servicio de streaming.



Como se puede observar en la gráfica ampliada de accuracy, según incrementa el número de palabras utilizadas por reseña para el entrenamiento del modelo, se obtienen mejor resultados de clasificación de sentimiento. Esto puede deberse a que mientras cuenta con más palabras la red, esta puede tener más contexto para clasificar el sentimiento de la reseña. Es notorio que esto es un hecho tanto para cada modelo particular como para la forma de hacer embedding en la red.

El único caso para el que no funcionó incrementar el número de palabras por reseña fue en la red LSTM utilizando para la capa e embedding Word2Vec, el modelo pasó de tener 84.47% de accuracy a 80.09%, una reducción significante de exactitud.

1) Rendimiento DNN

A pesar de la simpleza de la red neuronal profunda con tan solo una capa con 8 neuronas, esta alcanzó resultados asombrosos favoreciendo a incrementar el número de palabras por reseña y entrenando su propia capa de embedding con el diccionario de palabras existente en la base de datos procesada.

El mejor resultado se obtuvo con reseñas ajustadas a 146 palabras de longitud entrenando su capa de embedding alcanzando un 87.73% de accuracy.

2) Rendimiento CNN

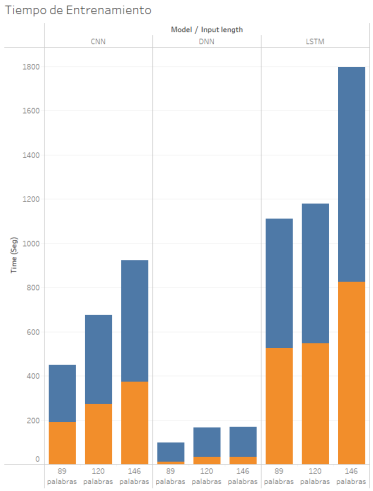
Los modelos obtenidos de la red neuronal convolucional fueron los únicos que mostraron mejora al utilizar los vectores pre-entrenados de Word2Vec. Para las tres longitudes de reseñas utilizadas, los modelos con mejor rendimiento fueron los que no entrenaron su capa de embedding y simplemente utilizaron la matriz de vectores como pesos de su capa.

Aún así, evidenciamos que a medida que aumentó el tamaño de la reseña, aumentó el nivel de accuracy alcanzado. El modelo con mayor exactitud fue el que se entrenó con Word2Vec como parte de su capa de embedding utilizando como entrada reseñas ajustadas a longitud de 146 palabras logrando un 87.63% de accuracy.

3) Rendimiento LSTM

Al comparar la red neuronal que utiliza LSTM con la red CNN y la red DNN, observamos que se obtienen mejores resultados de accuracy, siendo el máximo 88.08% como describimos anteriormente. En esta red también observamos que la exactitud aumenta a medida que aumenta el número de palabras al que se ajustó cada reseña de película.

Además de analizar los resultados de precisión, exactitud, sensibilidad y F1-score, es importante tomar en cuenta el tiempo de entrenamiento. Algunos modelos pueden tener resultados un poco mejores, pero si el tiempo de entrenamiento es mucho más grande, nos conviene elegir otro modelo por el costo computacional de entrenarlos. Es por ello que en la siguiente tabla se muestra el tiempo de entrenamiento para cada una de las pruebas, con la finalidad de analizar el costo beneficio de cada uno.



Los resultados en cuanto a tiempo nos muestran que DNN es muy eficiente, pues es el modelo con menor tiempo de entrenamiento, teniendo como tiempo máximo 2 minutos y 14 segundos. Mientras que LSTM es el modelo con mayor tiempo de entrenamiento, con un máximo de 16 minutos y 12 segundos. La diferencia entre ambos tiempos es muy grande; sin embargo, la diferencia de accuracy obtenido es de tan solo 0.36%. Es por esto que DNN también es una buena opción, pues ofrece buenos resultados en tiempos eficientes sin requerir gran poder computacional.

El modelo que más tomó tiempo de entrenar fue la red LSTM superando en gran medida a CNN y DNN. Aún así, el tiempo de entrenamiento sigue siendo relativamente corto ya que el que más se tardó tomó 30 minutos. Puede ser que para ciertas implementaciones el tiempo adicional no sea necesario ya que se obtiene resultados similares con un modelo DNN que tarda 10 veces menos en entrenar y es mucho más simple.

Presentamos el siguiente Jupyter Notebook con la finalidad de transparentar nuestra investigación y hacerla replicable: https://bit.ly/45PKnl2

VI. CONCLUSIÓN

Con las comparaciones que realizamos en el artículo obtuvimos información relevante acerca de cuáles son las mejores opciones para el análisis de sentimientos en reseñas de películas.

El tipo de red neuronal LSTM con el embedding entrenado en la primera capa, tuvo los mejores resultados para el análisis de sentimientos de reseñas de películas. Esto gracias a que LSTM captura y recuerda patrones de secuencias en el texto.

El modelo de red neuronal profunda (DNN) tuvo un rendimiento poco menor al de LSTM, y cuenta con la ventaja de que es mucho más rápido de entrenar y es más sencillo por lo que este sería una buena opción optar por este modelo cuando los recursos computacionales son limitados.

El modelo de red neuronal convolucional tuvo el rendimiento más bajo. Esto debido a que los modelos CNN son menos efectivos capturando relaciones a largo plazo en las secuencias.

En cuanto a los tipos de embedding, tanto el entrenado con el diccionario de tokens disponibles en las reseñas de películas como el Word2Vec fueron beneficiosos, sin embargo, entrenar nuestra propia capa de embedding resultó ser más beneficioso al realizar las pruebas en redes DNN y LSTM. El embedding Word2Vec tan solo tuvo mejores resultados para CNN.

Finalmente, comprobamos con este artículo que la longitud de entrada que usamos para los modelos si tiene efecto en el rendimiento del mismo. Para los 3 modelos y los 2 tipos de embedding resultó ser así ya que entre más grande fue la longitud de las reseñas, mejores fueron los resultados de accuracy obtenidos. Usar los datos del 3er Cuartil tuvo un buen rendimiento, aunque sería de utilidad realizar un análisis posterior en el que se pruebe usando el máximo de palabras como longitud de las reseñas.

Para realizar mejoras al modelo, sería de importancia tomar el modelo ganador (LSTM, con embedding entrenable y longitud de 146 palabras) , y realizar más pruebas ajustando hiperparámetros del modelo en lugar de mantenerlos constantes como se realizó en las pruebas. Podríamos probar con diferentes números de celdas para la capa de LSTM, diferentes números de neuronas para la capa densa y diferentes tasas de dropout. De igual forma se puede modificar hiperparámetros como la tasa de aprendizaje para llegar a tener mejores resultados de accuracy.

Se cumplió el objetivo de encontrar relación entre el tamaño de entrada de las secuencias de texto en el rendimiento de modelos de aprendizaje profundo así como determinar en qué casos y para qué modelos funciona entrenar con el diccionario de tokens la capa de embedding.

RECONOCIMIENTO

Nos gustaría expresar nuestro profundo agradecimiento a nuestros profesores de clase por su dedicación, conocimientos y apoyo a lo largo de este proceso de investigación. Su experiencia y orientación han sido fundamentales para nuestra comprensión y aplicación de los conceptos de inteligencia artificial y análisis de sentimientos. Además, su disposición para responder nuestras preguntas, brindar comentarios constructivos y fomentar nuestro crecimiento académico ha sido de gran utilidad.

REFERENCES

[1] Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal. Doi:10.1016/j.asej.2014.04.011. Recuperado de www.scopus.com el 10 de mayo de 2023.

[2] Jassim, M. A., Abd, D. H., & Omri, M. N. (2023). Machine learning-based new approach to films review. Social Network Analysis and Mining. Doi:10.1007/s13278-023-01042-7. Recuperado de www.scopus.com el 10 de mayo de 2023.
[3] Alayba, A. M., Palade, V., England, M., & Iqbal, R. (2018). A combined CNN and LSTM model for arabic sentiment analysis. Doi:10.1007/978-3-319-99740-7_12. Recuperado de www.scopus.com el 10 de mayo de 2023
[4] Kaur, G., & Sharma, A. (2023). A deep learning-based model using hybrid feature extraction approach for consumer sentiment analysis. Journal of Big Data, 10(1) doi:10.1186/s40537-022-00680-6. Recuperado de www.scopus.com el 13 de mayo de 2023.
[5] Kim, Y. (2014). Convolutional neural networks for sentence classification. Paper presented at the EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 1746-1751. doi:10.3115/v1/d14-1181. Recuperado de www.scopus.com el 13 de mayo de 2023.
[6] Liu, Y., Bi, J., & Fan, Z. (2017) Multi-class sentiment classification: The experimental comparisons of feature selection and machine learning algorithms. Expert Systems with Applications, 80, 323-339. doi:10.1016/j.eswa.2017.03.042. Recuperado de www.scopus.com el 13 de mayo de 2023.
[7] Benrouba, F., & Boudour, R. (2023). Emotional sentiment analysis of social media content for mental health safety. Social Network Analysis and Mining, 13(1) doi:10.1007/s13278-022-01000-9. Recuperado de www.scopus.com el 13 de mayo de 2023.
[8] Fan, X. (2023). Artificial intelligence technology-based semantic sentiment analysis on network public opinion texts. International Journal of Information Technologies and Systems Approach, 16(2) doi:10.4018/IJITSA.318447. Recuperado de www.scopus.com el 13 de mayo de 2023.
[9] Taherdoost, H., & Madanchian, M. (2023). Artificial intelligence and sentiment analysis: A review in competitive research. Computers, 12(2) doi:10.3390/computers12020037. Recuperado de www.scopus.com el 13 de mayo de 2023.
[10] Ullah, K., Rashad, A., Khan, M., Ghadi, Y., Aljuaid, H., & Nawaz, Z. (2022). A deep neural network-based approach for sentiment analysis of movie reviews. Doi:10.1155/2022/5217491. Recuperado de www.scopus.com el 13 de mayo de 2023.
[11] A. A. L. Cunha, M. C. Costa, and M. A. C. Pacheco, "Sentiment analysis of youtube video comments using deep neural networks," International Conference on Artificial Intelligence and Soft Computing, China, 2019.
[12] Srivastava, N. (2013). Improving neural networks with dropout. University of Toronto, 182(566), 7.
[13] S. Seo, C. Kim, H. Kim, K. Mo and P. Kang, "Comparative Study of Deep Learning-Based Sentiment Classification," in IEEE Access, vol. 8, pp. 6861-6875, 2020, doi: 10.1109/ACCESS.2019.2963426.
[14] Kingma, D.P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR).
[15] Song, F., Li, Y., Cheng, W., & Dong, L. (2023). An improved dynamic programming tracking-before-detection algorithm based on LSTM network. Eurasip Journal on Advances in Signal Processing, 2023(1) doi:10.1186/s13634-023-01020-3
[16] An, J. H., Wang, Z., & Joe, I. (2023). A CNN-based automatic vulnerability detection. Eurasip Journal on Wireless Communications and Networking, 2023(1) doi:10.1186/s13638-023-02255-2
[17] Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.
[18] Giri, S., & Banerjee, S. (2023). Performance analysis of annotation detection techniques for cyber-bullying messages using word-embedded deep neural networks. Social Network Analysis and Mining, 13(1) doi:10.1007/s13278-022-01023-2
[19] Tuarob, S., Satravisut, M., Sangtunchai, P., Nunthavanich, S., & Noraset, T. (2023). FALCoN: Detecting and classifying abusive language in social networks using context features and unlabeled data. Information Processing and Management, 60(4) doi:10.1016/j.ipm.2023.103381