Reporte técnico proyecto práctico redes neuronales

Integrantes:
Carlos Daniel Corrales Arango - 2122878
Jose Manuel Palma Oquendo - 2125182

Universidad del Valle
Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
2025 - I

Resumen

Este proyecto realizado explora el uso de redes neuronales para la clasificación de “Reseñas reales de negocios calificadas desde 1 a 5 estrellas”. En la primera parte, se implementó un Perceptrón Multicapa (MLP) utilizando TensorFlow/Keras, descubriendo las formas que existen para tratar los datos mediante tokenización y vectorización, seguido por la implementación de hiperparametros con Keras Tuner. En la segunda parte, se avanzó hacia arquitecturas más avanzadas de redes neuronales, tales como las Redes Recurrentes (RNN), LSTM y GRU. Esto con el objetivo de profundizar más dentro de las estrategias utilizadas para tener un rendimiento óptimo en tareas de datos secuenciales. En esta entrega se aplicaron técnicas modernas como regularización, normalización, y early stopping para mejorar el rendimiento de los modelos. Cada arquitectura fue evaluada usando métricas como precisión, matriz de confusión, curvas AUC y curvas de entrenamiento/validación. En conjunto, el trabajo demuestra cómo las redes neuronales modernas pueden capturar patrones complejos en texto y mejorar el rendimiento en tareas de clasificación multiclase.

Tabla de contenido

Introducción

El análisis de texto se ha convertido en una herramienta de suma importancia para comprender grandes volúmenes de datos no estructurados. El caso de este proyecto son las reseñas que los usuarios dejan sobre los negocios, las cuales contienen una puntuación de 1 a 5 estrellas. Automatizar la interpretación de estas reseñas mediante algoritmos de aprendizaje profundo como las redes neuronales es una tarea valiosa y desafiante para cualquier empresa, ya que permite clasificar o predecir comportamientos del consumidor.

El presente proyecto busca desarrollar modelos que, a partir de una reseña, prediga de forma automática la calificación otorgada por el usuario en una escala de 1 a 5 estrellas. Para esto, se utiliza un conjunto de datos obtenidos desde la plataforma Kaggle, compuesto por 10.000 muestras, las cuales plantean un entorno ideal para aplicar y comparar diversas arquitecturas de redes neuronales.

El objetivo general del proyecto es construir modelos de redes neuronales capaces de capturar patrones semánticos dentro de las secuencias y así poder asignar una etiqueta de calificación correspondiente. Para cumplir con este objetivo, el trabajo se dividió en dos etapas principales: en la primera entrega se implementaron modelos basados en perceptrón multicapa (MLP), utilizando una configuración manual de la red neuronal y otro modelo basado en hiperparametros gracias a las herramientas que ofrece Keras Tuner para encontrar la mejor configuración posible dentro de un conjunto finito de combinaciones. En la segunda entrega se implementaron arquitecturas recurrentes más sofisticadas como RNN simples, LSTM (Long Short-Term Memory) y GRU (Gated Recurrent Units), todas en versiones bidireccionales. En ambas fases, se aplicaron técnicas de preprocesamiento como la tokenización, vectorización, y se añadieron embeddings pre entrenados como GloVe en la segunda fase con el objetivo de enriquecer la representación semántica de las palabras.

A lo largo del proyecto se empleó la librería Keras de TensorFlow para el diseño, entrenamiento y evaluación de los modelos. Se realizó una comparación del rendimiento de cada arquitectura usando métricas como precisión, matriz de confusión y curvas de entrenamiento. Los resultados muestran una mejora significativa en el desempeño al utilizar modelos de redes neuronales recurrentes frente al MLP, especialmente en cuanto a la capacidad de capturar dependencias contextuales en los textos.

Este informe presenta el proceso completo de desarrollo, las decisiones técnicas tomadas en cada etapa, y un análisis comparativo que evidencia el valor de utilizar redes neuronales especializadas para el tratamiento de lenguaje natural en tareas de clasificación.

Contenido

Preparación del dataset

Primeramente, se analizó el conjunto de datos dado para identificar la estructura general y validar su correcta lectura e interpretación. Se visualizaron las primeras filas, se revisaron los tipos de datos y se verificaron las columnas disponibles utilizando funciones estándar de pandas, librería que ofrece el lenguaje de programación Python.

Para comprender mejor las características del dataset, se utilizó una función de resumen que permitió visualizar el tipo de dato de cada columna, el número de valores únicos y algunos ejemplos representativos por cada columna.

Entrenamiento

El proceso de entrenamiento comienza con la definición de las variables de entrada y salida. En este caso, la variable X corresponde al texto de las reseñas, mientras que la variable Y representa la cantidad de estrellas asignadas que van del 1 al 5.

A continuación, se dividió el conjunto de datos en dos subconjuntos: 80% para entrenamiento y 20% para el test.

Dado que el modelo debe realizar una clasificación multiclase, fue necesario convertir las etiquetas al formato One-Hot Encoding. Esto transforma cada valor de estrella en un vector binario de longitud 5, donde solo la posición correspondiente a la clase está activa (valor 1), y las demás son 0. Este formato es requerido por Keras cuando se utiliza la función de pérdida categorical_crossentropy, adecuada para problemas de clasificación multiclase.

Tokenización

Se encontró que la longitud promedio de las reseñas en promedio tienen una longitud de 113 palabras. Además, el 50% de las reseñas tienen 101 palabras o menos. Sabiendo esa información sumamente relevante, se definió entonces un vocabulario limitado a las 10.000 palabras más frecuentes, además, se estableció una longitud máxima de 130 palabras por reseña, todo lo anterior en base al análisis estadístico de la distribución de palabras por texto.

Finalmente, se inicializó y entrenó un tokenizador sobre el conjunto de entrenamiento, el cual se encargó de convertir el texto a secuencias numéricas.

Vectorización

Luego del entrenamiento del tokenizador, se transformaron las reseñas de texto en secuencias numéricas, donde cada número representa una palabra según su posición en el vocabulario aprendido. Este proceso permitió traducir el lenguaje natural a un formato interpretable por la red neuronal. Posteriormente, se aplicó padding a todas las secuencias para estandarizar su longitud a 130 palabras, tal como se definió previamente, lo cual es fundamental para asegurar una entrada consistente a los modelos de redes neuronales.

Embeddings GloVe

Para enriquecer la representación semántica del texto, se integraron vectores de palabras preentrenados GloVe (Global Vectors for Word Representation), específicamente la versión de 100 dimensiones. GloVe proporciona una forma de capturar similitudes semánticas entre palabras basándose en su coocurrencia en grandes corpus de textos, lo cual resulta especialmente útil en tareas de procesamiento de lenguaje natural.

Se construyó una matriz de embeddings de tamaño 10.000 x 100, correspondiente a las 10.000 palabras más frecuentes del vocabulario utilizado, donde cada fila representa el vector GloVe de una palabra.

Esta capa de embeddings sirvió como entrada a las redes recurrentes, proporcionando una representación densa y contextualizada del texto que permite a las redes capturar relaciones semánticas más profundas entre las palabras.

Red neuronal con arquitectura MLP de configuración manual

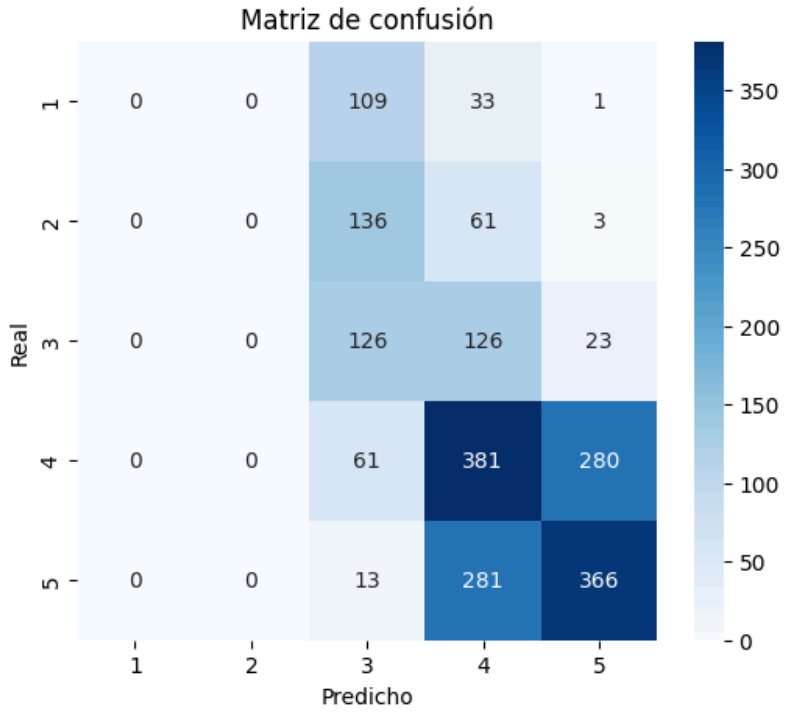
Se construyó una red neuronal con arquitectura Multilayer Perceptron (MLP) de configuración manual. Un MLP es un tipo de red neuronal profunda compuesta por múltiples capas totalmente conectadas (densas), donde cada neurona en una capa está conectada a todas las neuronas de la siguiente. Este tipo de red es especialmente útil para tareas de clasificación y regresión cuando los datos ya han sido transformados a un formato numérico fijo.

Primero, se inició con una capa de Embedding que transforma cada palabra en un vector denso de 100 dimensiones. La red cuenta con dos capas ocultas densas, la primera con 64 neuronas y la segunda con 32, ambas con activación ReLU y regularización L2 para prevenir el sobreajuste. Se incorporaron capas Dropout del 40% y 30% respectivamente para mejorar la generalización. La salida consta de 5 neuronas con una función de activación softmax, adecuadas para este problema de clasificación multiclase. El modelo fue compilado con el optimizador AdamW, la función de pérdida categorical_crossentropy y la métrica de accuracy. Para evitar sobreentrenamiento, se utilizó early stopping con paciencia de 5 épocas y por último, el modelo fue entrenado durante un máximo de 20 épocas.

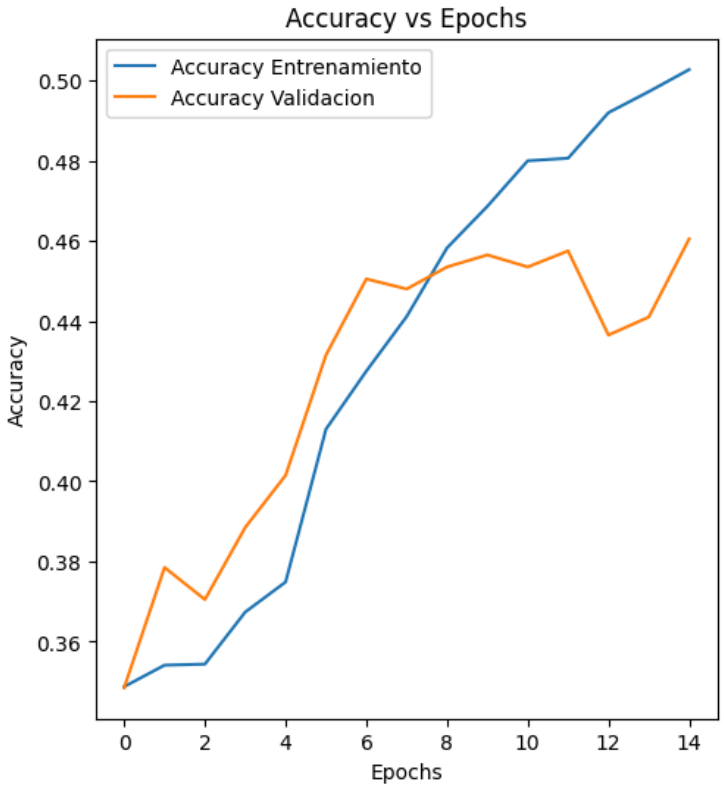
Métricas

A continuación, se presentan las métricas de evaluación y visualizaciones obtenidas tras el entrenamiento del modelo, las cuales permiten analizar su desempeño. Se incluyen la matriz de confusión, la evolución del accuracy y de la función de pérdida a lo largo de las épocas.

La matriz de confusión presentada muestra la relación entre los valores reales y los predichos por el modelo. En la tabla, los valores reales se encuentran representados en las filas, mientras que las predicciones corresponden a las columnas. Se puede observar que el modelo no logra clasificar correctamente ninguna instancia de las clases 1 y 2, de hecho, la mayoría de los ejemplos de estas dos clases se clasifican como clase 3. Aunque se evidencia un buen rendimiento en las clases 4 y 5, con altos valores en la diagonal, también se puede ver confusiones en estas clases. Esto puede ser un indicador de un desbalance en el conjunto de datos, o una limitación en la capacidad del modelo para generalizar sobre todo el conjunto de clases.



En la gráfica de Accuracy vs Epochs se observa que tanto la precisión de entrenamiento como la de validación aumentan durante las primeras épocas, lo cual indica que el modelo logra aprender progresivamente. Pero, a partir de la época 6, ambas curvas comienzan a estabilizarse y, aunque la precisión de entrenamiento sigue mejorando, la precisión de validación se estanca en torno al 46%. Esto sugiere que el modelo alcanza un punto donde no generaliza de forma efectiva, probablemente debido a un sobreajuste, el cual podría mejorarse mediante técnicas como regularización, ajuste de hiperparámetros o aumento del conjunto de datos.



La gráfica de Loss vs Epochs muestra una disminución de la pérdida en el conjunto de entrenamiento y en el de validación durante las primeras épocas, lo que indica un aprendizaje efectivo del modelo. Sin embargo, a partir de la época 6, la pérdida de validación se estabiliza e incluso muestra un ligero aumento hacia las últimas épocas, mientras que la pérdida de entrenamiento continúa disminuyendo. Esto sugiere el inicio de un sobreajuste, en el cual el modelo comienza a ajustarse demasiado a los datos de entrenamiento, perdiendo capacidad de generalización. Esta tendencia refuerza lo observado en la gráfica anterior y apunta a la necesidad de aplicar estrategias de control como early stopping o regularización más agresiva para mejorar el rendimiento en datos no vistos.

Red neuronal con arquitectura MLP con hiperparametros

En la primera entrega del proyecto, se implementó un enfoque de optimización de hiperparametros utilizando la librería Keras Tuner, con el objetivo de identificar la mejor configuración posible para la red neuronal MLP que se estaba trabajando en esta primera fase. Esta herramienta automatiza el proceso de búsqueda de combinaciones de hiperparametros que maximicen el desempeño del modelo.

Se utilizó la estrategia de búsqueda aleatoria (RandomSearch), la cual explora de forma eficiente el espacio de hiperparámetros definidos. Entre los parámetros ajustados se encuentran:

- Dimensión del embedding (embedding_dim): valores entre 50 y 200.
- Número de unidades en las capas densas (units_layer1, units_layer2, units_layer3).
- Funciones de activación (relu y tanh).
- Tasa de regularización L2.
- Tasa de Dropout por cada capa de la red.
- Uso opcional de una tercera capa oculta.
- Optimizador (adam o adamW).

Cada configuración de hiperparámetros fue evaluada durante un número limitado de épocas para medir su desempeño preliminar mediante la métrica de exactitud en validación y así poder sacar la mejor configuración posible.

Tras completar el proceso de búsqueda con Keras Tuner, se seleccionó el modelo con el mejor desempeño en el conjunto de validación, modelo el cual fue el siguiente:

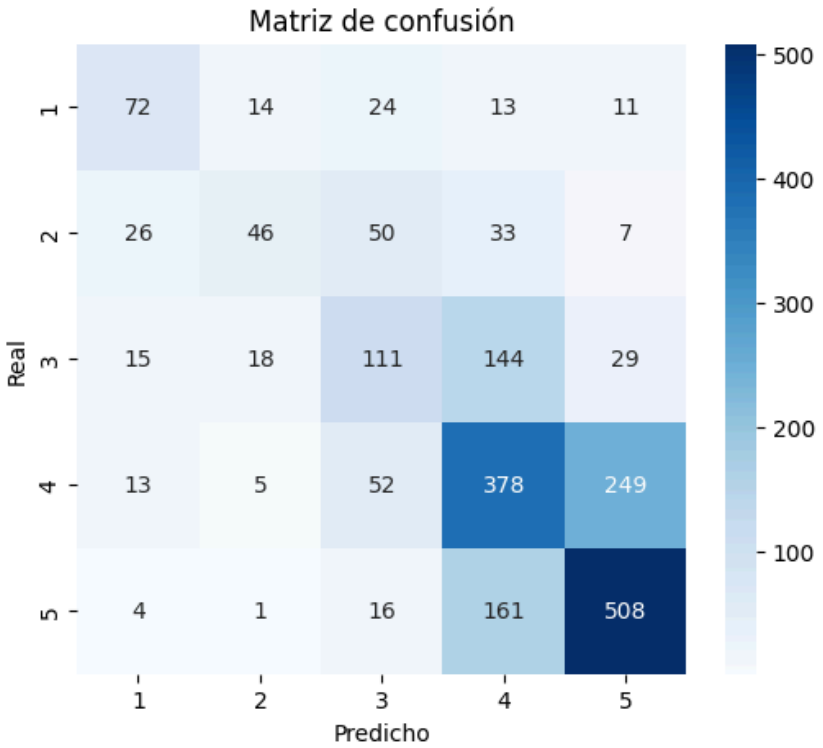
- Dimensión de la capa de embedding: 130
- Primera capa densa: 32 neuronas, activación tanh, Dropout de 0.1
- Segunda capa densa: 16 neuronas, activación tanh, sin Dropout
- Tercera capa densa (habilitada): 9 neuronas, activación relu, Dropout de 0.4
- Regularización L2: 0.01
- Optimizador: adam

Esta configuración permitió al modelo alcanzar un equilibrio entre complejidad y generalización, logrando un mejor desempeño respecto a configuraciones manuales o arbitrarias.

Métricas

Una vez entrenado el modelo con la configuración óptima determinada por Keras Tuner, se procedió a evaluar su rendimiento utilizando diversas métricas y visualizaciones.

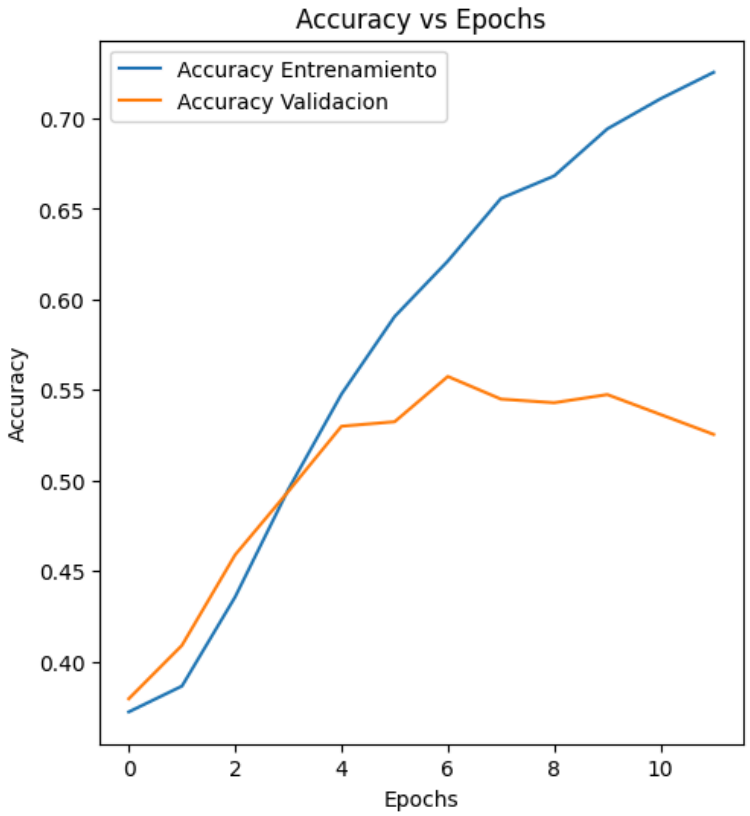
Primero, se generó una matriz de confusión que permite visualizar el desempeño del modelo en términos de clasificación correcta e incorrecta por clase (estrellas de 1 a 5).



Como se puede observar, el modelo funciona bien para distinguir reseñas muy positivas (clase 5). Sin embargo, tiene dificultades en las clases intermedias y negativas, especialmente en diferenciar entre clases vecinas (como 4 y 5 o 2 y 3), lo cual es normal que pasara ya que los datos cuentan con un desbalance claro existiendo más reseñas de 4 y 5 estrellas que de la 1 a la 3. Como opinión del grupo de trabajo se podría considerar usar enfoques adicionales como:

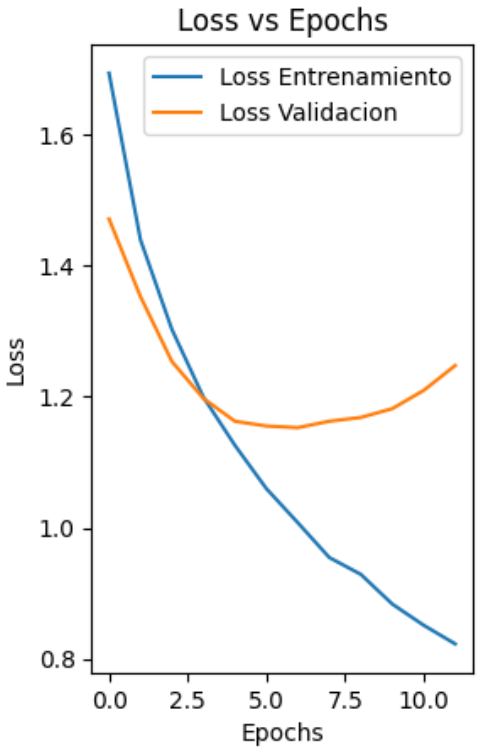
- Técnicas de balanceo de clases.
- Pérdidas personalizadas para clasificación ordinal.

Se graficó también la evolución de la precisión tanto en entrenamiento como en validación a lo largo de las épocas, lo cual permite observar si hubo sobreajuste o subajuste.



La gráfica de Accuracy vs Epochs muestra que, si bien el modelo mejora de forma constante su rendimiento en el conjunto de entrenamiento, alcanzando una precisión superior al 70 %, la precisión en validación se estabiliza cerca del 55 % a partir de la quinta época y posteriormente comienza a decrecer ligeramente. Esto indica un claro caso de sobreajuste (overfitting), donde el modelo comienza a aprender patrones específicos del conjunto de entrenamiento que no generalizan bien a nuevos datos. A pesar del uso de regularización y dropout, el modelo aún necesita ajustes adicionales para mejorar su capacidad de generalización, como una mejor selección de hiperparámetros o un mayor volumen de datos balanceados para entrenamiento.

Se acompañó por último la métrica de precisión con una gráfica de pérdida (loss), útil para diagnosticar estabilidad del entrenamiento y convergencia.



La gráfica de Loss vs Epochs refuerza la presencia de sobreajuste observada anteriormente. Aunque la pérdida en el conjunto de entrenamiento disminuye de manera constante hasta estabilizarse en valores cercanos a 0.8, la pérdida de validación se reduce inicialmente, pero a partir de la quinta época deja de mejorar y comienza a aumentar gradualmente. Este comportamiento podría decirse que el modelo empieza a memorizar los datos de entrenamiento en lugar de aprender patrones generalizables. A pesar del uso de técnicas como dropout, regularización L2 o early stopping, sería recomendable considerar una reducción en la complejidad del modelo para evitar este deterioro en el rendimiento de validación.

Red neuronal con arquitectura RNN sin memoria

Se construyó una red neuronal RNN sin memoria, se diseñó con el fin de procesar secuencias de texto y realizar clasificación multiclase. Esta red inicia con una capa de Embedding pre entrenada con Glove, la cual, convierte cada palabra en un vector denso de 100 dimensiones, permitiendo así captar relaciones semánticas y contextuales.

Posteriormente, se incorporó una capa bidireccional simpleRNN, encargada de procesar la secuencia tanto hacia adelante como hacia atrás, permitiendo capturar información contextual a partir de ambos extremos de la oración. Por otro lado, se utilizó inicialización ortogonal de pesos, sesgos en cero y regularización mediante recurrent dropout.

Este modelo cuenta con una capa de LayerNormalization para estabilizar la activación entre capas y acelerar la convergencia. La red continúa con una capa densa de 64 neuronas con

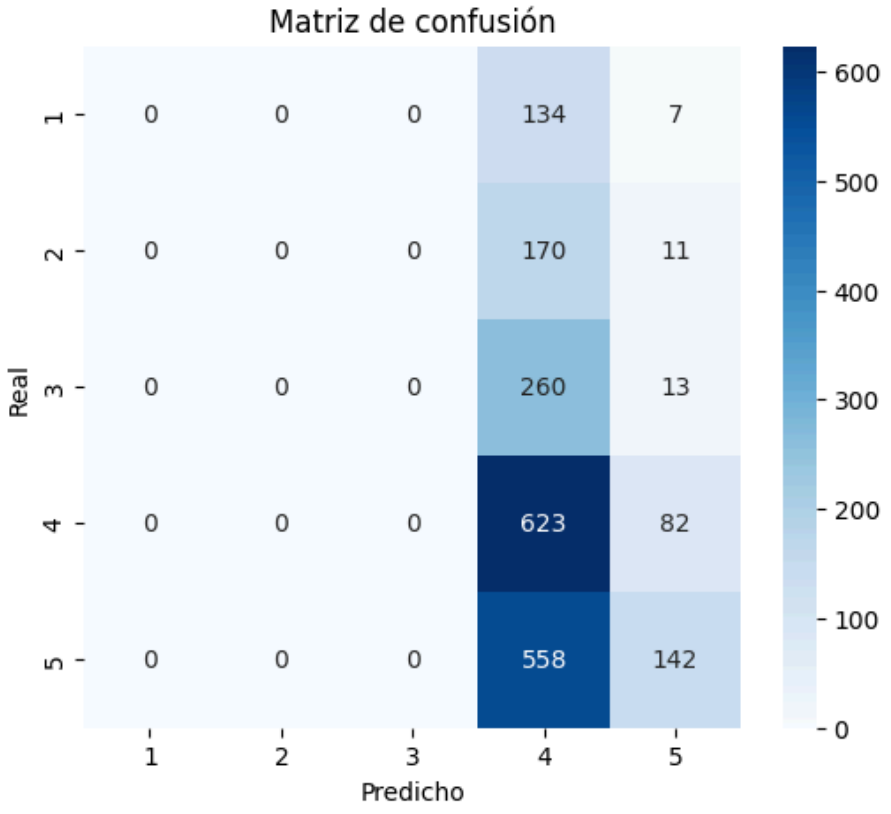
activación ReLU, regularización L2 y una capa Dropout del 30% para mejorar la generalización del modelo.

La capa de salida está compuesta por 5 neuronas con activación softmax, lo cual permite la clasificación en cinco categorías correspondientes a las valoraciones de 1 a 5 estrellas.

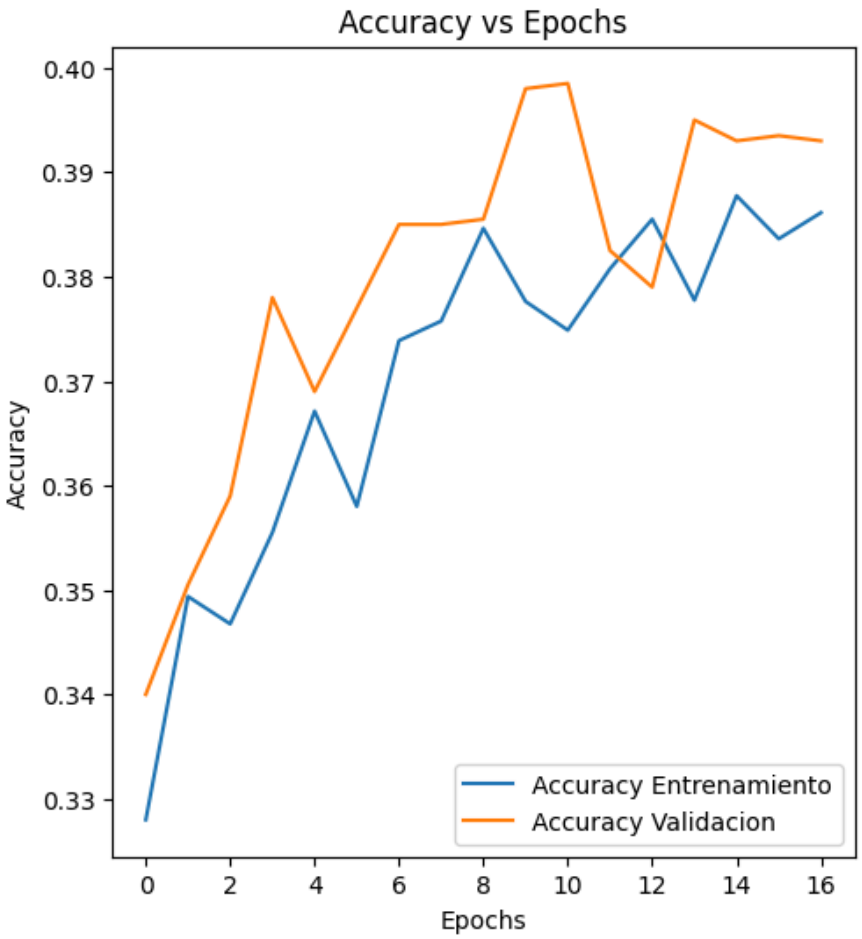
El modelo fue compilado con el optimizador Adam, utilizando la función de pérdida categorical_crossentropy y la métrica de accuracy. Para evitar el sobreentrenamiento, se implementó EarlyStopping con una paciencia de 5 épocas, restaurando los mejores pesos del modelo.

Métricas

En la matriz de confusión se puede observar que para las categorías 1,2 y 3, el modelo no logró clasificar correctamente ningún ejemplo. Por otro lado, se observa que en la categoría 4 el modelo clasifica una gran proporción de los ejemplos de forma correcta, pero para la categoría 5 aunque logra clasificar una buena cantidad de ejemplos correctamente, la cantidad de ejemplos clasificados erróneamente es más grande. Este comportamiento sugiere que el modelo está sobreajustado a las clases más frecuentes pues se puede observar que las 3 primeras clases las clasifica como clase 4 y 5 o que carece de representaciones efectivas para las clases menos representadas, lo que limita su capacidad de generalización.



En la gráfica de accuracy se observa que la brecha entre el accuracy de entrenamiento y el de validación es pequeña, lo cual posiblemente nos podría indicar que el modelo arroja buenos resultados, sin embargo, como se observó previamente en la matriz de confusión este modelo no logra clasificar correctamente varios ejemplos.



Red neuronal con arquitectura LSTM

Se diseñó una red neuronal basada en una arquitectura LSTM bidireccional. Esta red también inicia con una capa de Embedding preentrenada con GloVe.

A continuación, se añadió una capa bidireccional LSTM, capaz de capturar información contextual desde el pasado y el futuro dentro de la secuencia. La capa al igual que en la RNN sin memoria, utiliza una inicialización ortogonal de pesos y un recurrent_dropout de para reducir el riesgo de sobreajuste al modelar secuencias largas.

Posteriormente, se incorporó una capa de Layer Normalization, que mejora la estabilidad del entrenamiento y acelera la convergencia al normalizar la activación dentro de cada instancia.

La red continúa con una capa densa con función de activación ReLU, inicialización glorot_uniform y regularización L2, seguida de una capa Dropout para fomentar la generalización del modelo.

La capa de salida cuenta con activación softmax, adecuada para la clasificación multiclase de reseñas entre 1 y 5 estrellas.

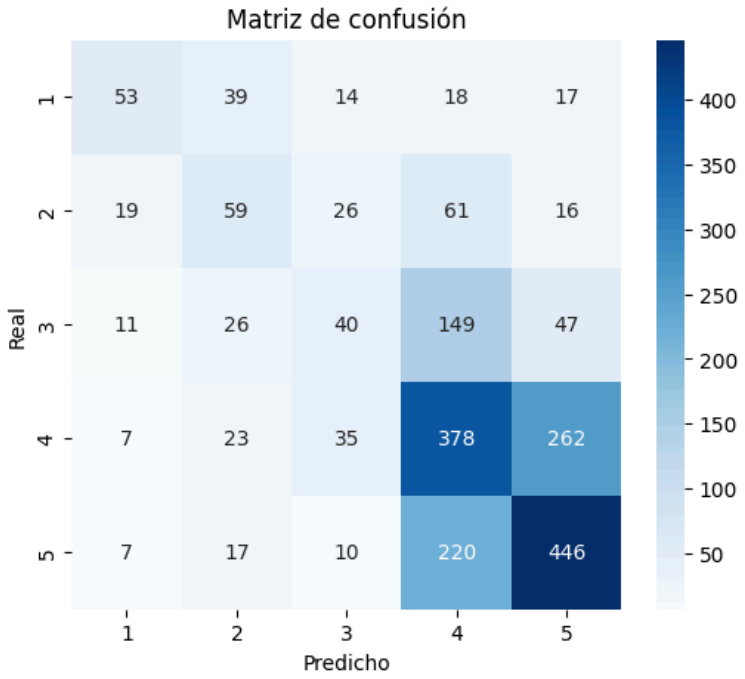
El modelo fue compilado con el optimizador Adam. Además, se utilizó categorical_crossentropy como función de pérdida y accuracy como métrica de evaluación.

Para prevenir el sobreentrenamiento, se empleó EarlyStopping con paciencia de 5 épocas, lo que permite detener el entrenamiento si no se observa mejora en la pérdida de validación. Para ilustrar el desempeño del modelo

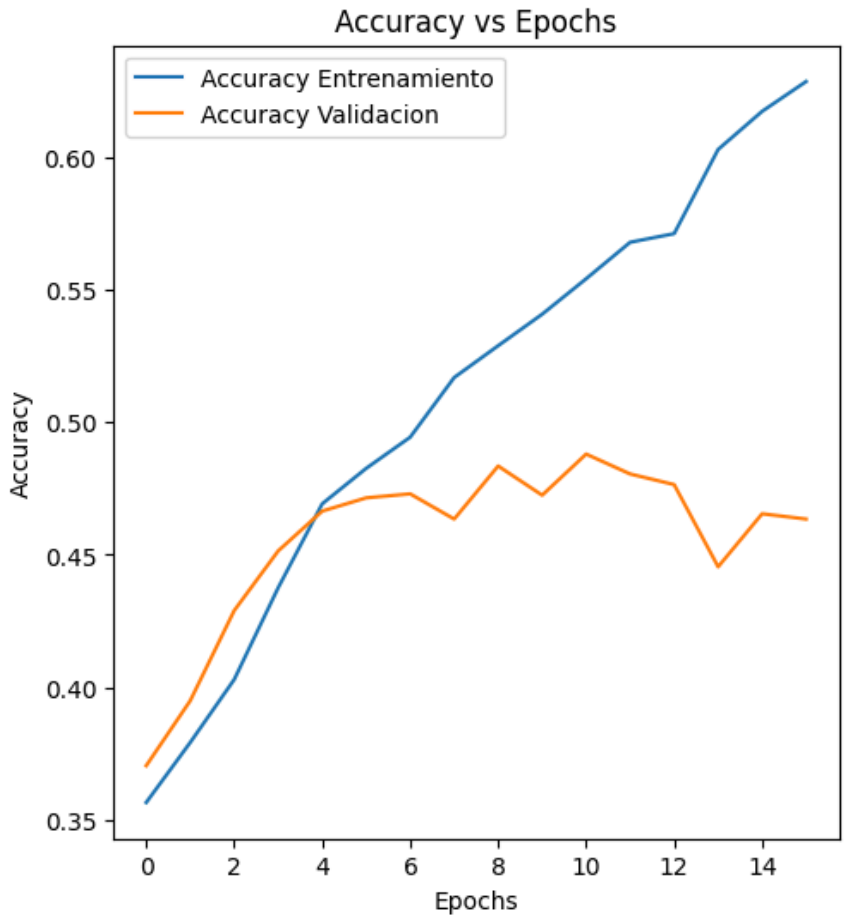
Métricas

Para evaluar el desempeño del modelo se emplearon las siguientes métricas:

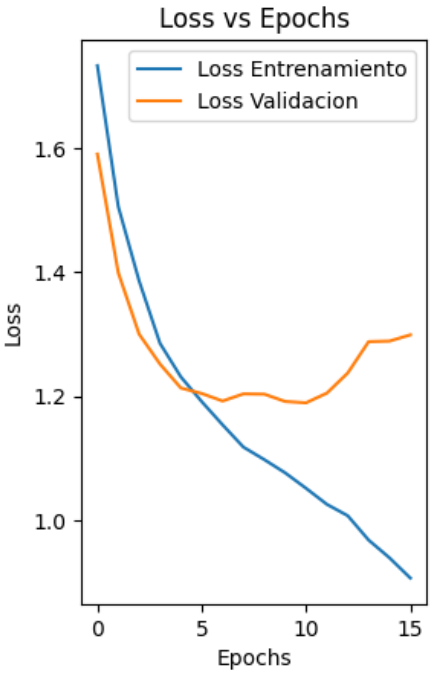
La matriz de confusión, en la cual se puede observar como el modelo logra un buen desempeño a la hora de clasificar las reseñas en las categorías 4 y 5, sin embargo, se evidencia que para estas categorías se presenta confusión pues hay un número considerable de reseñas clasificadas incorrectamente. Por otro lado, para las categorías 1,2 y 3 se evidencia que el modelo no logra un desempeño eficiente pues la gran mayoría de ejemplos fueron clasificados de forma incorrecta. Lo anterior, puede sugerir que hay una fuerte similitud semántica en las categorías 4 y 5 lo cual hace que la mayoría de casos clasificados correctamente estén allí. Otra posible causa de la confusión que presenta el modelo es que el número de datos no es muy grande.



En la grafica de accuracy podemos ver un claro sobreajuste del modelo.



En la siguiente gráfica se puede confirmar el sobreajuste, se evidencia como el modelo se adapta a los datos de entrenamiento lo cual le impide generalizar.

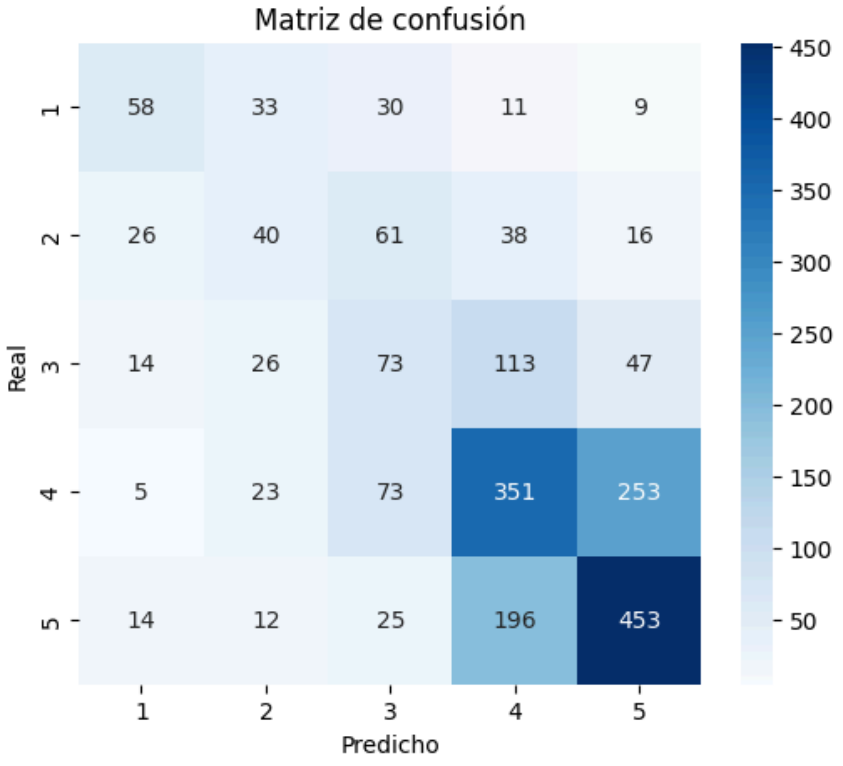


Red neuronal con arquitectura GRU

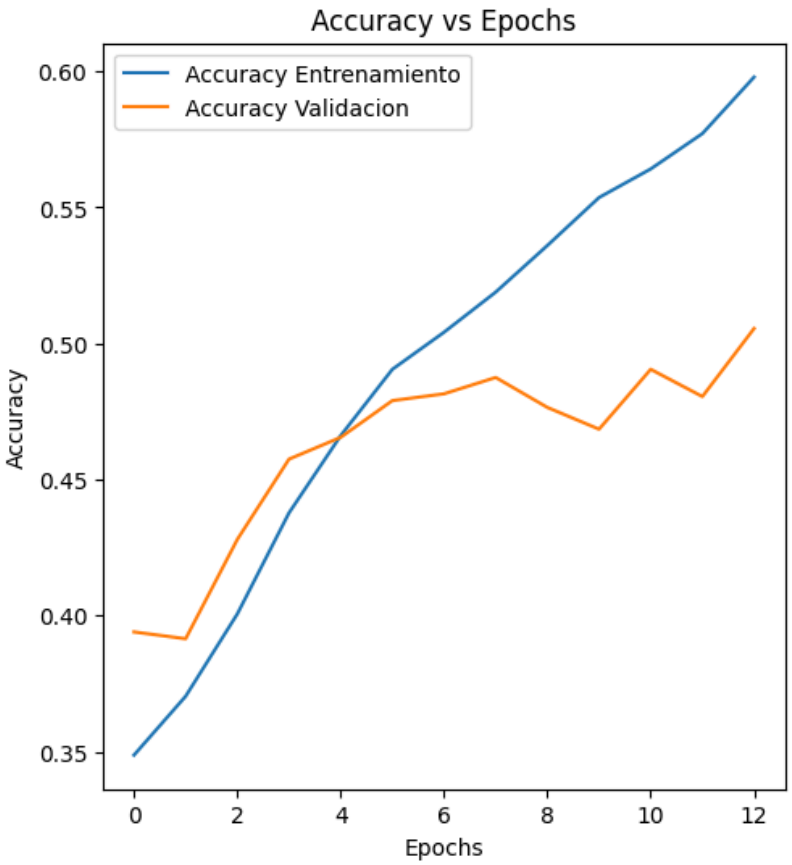
En esta parte se implementó una red neuronal basada en la arquitectura GRU, que es una variante más ligera y eficiente de las LSTM. Al igual que en los modelos anteriores, se utilizó vectores de embeddings GloVe. La red cuenta con una capa GRU bidireccional. También se incorporó normalización de capa (LayerNormalization) para estabilizar el entrenamiento, seguida por una capa densa con activación ReLU, combinada con regularización L2 y Dropout. La salida es una capa softmax de 5 neuronas, una por cada posible calificación de estrellas (de 1 a 5). El modelo fue entrenado con el optimizador Adam, usando una tasa de aprendizaje baja para asegurar un aprendizaje más estable. Además, se aplicó early stopping para detener el entrenamiento automáticamente si el rendimiento en los datos de validación dejaba de mejorar.

Métricas

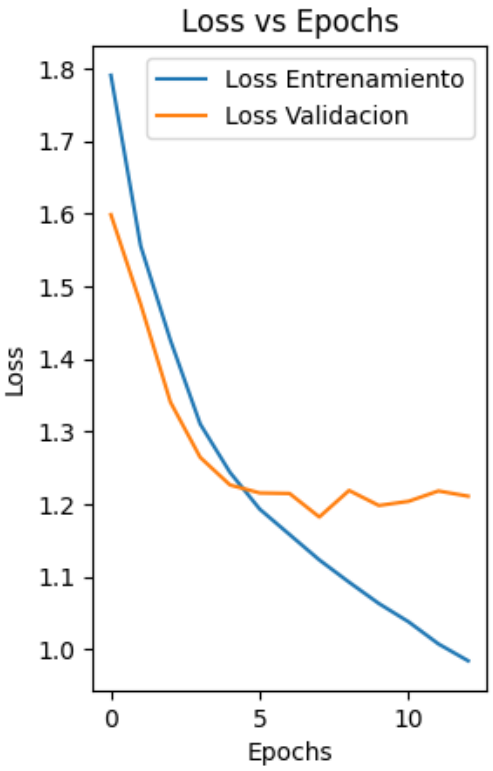
En la matriz de confusión se evidencia cómo al igual que en LSTM, la red tiende a clasificar de mejor manera las categorías de 4 y 5 estrellas, y para las otras categorías clasifica los ejemplos en varias categorías siendo la mayoría de ellas incorrectas.



En la siguiente gráfica podemos observar un sobreajuste. Se evidencia como el accuracy de validación llega hasta 0.50, de hecho, previamente a partir del epoch 6 empieza a fluctuar.



En esta gráfica se puede observar como el loss de validación tiende a aumentar levemente a partir del epoch 5.



Conclusiones

A lo largo del trabajo se trabajó con diversas arquitecturas de redes neuronales aplicadas a la clasificación de reseñas de usuarios, incluyendo redes MLP, redes recurrentes (RNN, GRU, LSTM) y técnicas avanzadas como el uso de embeddings preentrenados (GloVe), normalización por capas e inicialización de pesos personalizada.

En la primera fase, aunque se lograron mejoras mediante la optimización de hiperparámetros con Keras Tuner, las redes MLP presentaron limitaciones en su capacidad para capturar la estructura secuencial del texto, mostrando un rendimiento moderado y cierta tendencia al sobreajuste. Posteriormente, las redes recurrentes mostraron un comportamiento superior en términos de capacidad de modelado y precisión, logrando capturar mejor las dependencias contextuales del lenguaje natural, aunque también presentaron señales de sobreajuste en etapas "avanzadas" del entrenamiento.

La incorporación de GloVe y técnicas de regularización en la segunda entrega permitió un salto en rendimiento y generalización frente a los modelos anteriores, lo cual resalta la importancia de utilizar representaciones semánticas ricas y buenas prácticas de entrenamiento en problemas de procesamiento de lenguaje natural.

Los resultados sugieren que hay espacio para seguir mejorando, ya sea con arquitecturas más robustas como Transformers o estrategias más avanzadas de ajuste fino, además de tener un mejor tratado de los datos para que estén mejor balanceados. En conjunto, este trabajo permitió comparar enfoques y resaltar la evolución del rendimiento al integrar componentes más sofisticados, sentando una base sólida para futuras investigaciones o aplicaciones prácticas.

Bibliografía

- Keras Documentation
- GloVe Documentation
- Keras Tuner Documentation
- Tensor Flow documentation