UNIVERSIDAD DE ANTIOQUIA
1803

Informe ejecutivo de proyecto final: Análisis de sentimiento en reseñas de películas con BiLSTM

Autores: Daniel Brand Taborda, Jhonier Raúl Jiménez

Curso: Deep Learning 2025

Fecha: 29 de junio de 2025

---

1. Resumen

Este informe detalla el desarrollo y evaluación de un modelo de Deep Learning para la tarea de análisis de sentimiento. El objetivo fue clasificar reseñas de películas del dataset IMDB como positivas o negativas. Se implementó un pipeline completo, incluyendo un análisis exploratorio de datos, un preprocesamiento de texto exhaustivo y la construcción de un modelo basado en una Red Neuronal Recurrente Bidireccional (BiLSTM). El modelo final alcanzó una **exactitud (accuracy) del 86.6%** en el conjunto de datos de prueba, demostrando un rendimiento robusto y balanceado para ambas clases de sentimiento.

Dataset:

- **Fuente:** Kaggle
- **Enlace:** https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

- **Descripción:** Contiene 50,000 reseñas de películas etiquetadas como positivas o negativas.

2. Descripción de la estructura de los Notebooks

El trabajo fue organizado de manera modular en cuatro notebooks de Jupyter, siguiendo un flujo lógico desde los datos hasta el modelo final:

- `01 - exploración de datos.ipynb`: Carga inicial del dataset, análisis exploratorio para entender la distribución de los datos, la longitud de las reseñas y visualización de ejemplos.
- `02 - preprocesado.ipynb`: Contiene todo el proceso de limpieza y transformación del texto, incluyendo la eliminación de ruido, la tokenización y la lematización. Genera los archivos de datos limpios (`train_processed.csv` y `test_processed.csv`).
- `03 - arquitectura de linea de base.ipynb`: Define la arquitectura del modelo BiLSTM base, establece los hiperparámetros de tokenización y padding, y prepara los datos para el entrenamiento.
- `04 - entrenamiento y evaluacion.ipynb`: Entrena el modelo definido, visualiza las curvas de aprendizaje, evalúa el rendimiento sobre el conjunto de prueba y genera las métricas finales y la matriz de confusión.

3. Descripción de la solución

La solución implementada se compone de dos fases principales: el preprocesamiento de los datos de texto y la arquitectura del modelo de clasificación.

3.1. Preprocesado de datos

Se aplicó una cadena de transformaciones rigurosa para preparar las reseñas para el modelo:

1. **Limpieza:** Se eliminaron etiquetas HTML, caracteres no alfanuméricos y se convirtió todo el texto a minúsculas.
2. **Eliminación de stopwords:** Se descartaron palabras comunes sin carga semántica (ej. "the", "a") del idioma inglés.

3. **Lematización:** Se utilizó `WordNetLemmatizer` para reducir cada palabra a su forma léxica base.
4. **Tokenización y padding:** El texto limpio se convirtió en secuencias de enteros utilizando el `Tokenizer` de Keras, con un vocabulario máximo de **3000** palabras. Todas las secuencias se estandarizaron a una longitud de **200** tokens.

3.2. Arquitectura del modelo

Se diseñó una Red Neuronal Recurrente Bidireccional (BiLSTM) con la siguiente arquitectura:

1. **Capa de embedding:** Mapea los tokens de entrada a vectores densos de 100 dimensiones, aprendiendo la representación semántica de cada palabra.
2. **Capa BiLSTM:** El núcleo del modelo, con 64 unidades. Procesa las secuencias en ambas direcciones para capturar el contexto completo de cada palabra.
3. **Capas densas:** Una capa oculta con 24 neuronas (activación ReLU) y una capa de salida con 1 neurona (activación sigmoide) para la clasificación binaria.
4. **Función de pérdida y optimizador:** Se utilizó `binary_crossentropy` como función de pérdida y `adam` como optimizador.

4. Descripción de las iteraciones

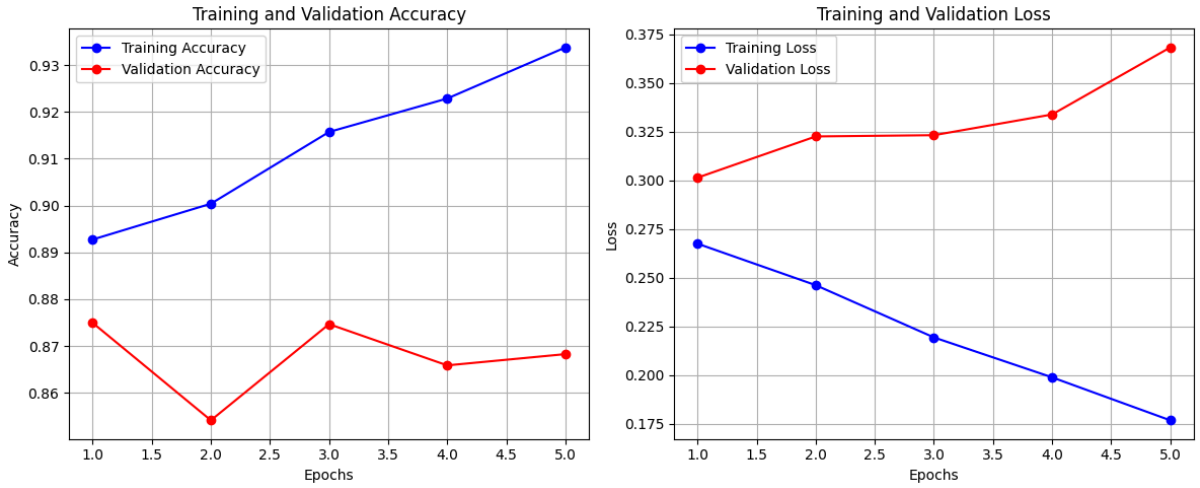
El desarrollo siguió un proceso metodológico enfocado en construir un pipeline robusto. La principal iteración fue el refinamiento progresivo a través de las siguientes fases:

1. **Análisis de datos:** Se estableció una base de entendimiento del problema.
2. **Preprocesamiento:** Se eligió la lematización sobre técnicas más simples como el stemming para obtener una mejor normalización del texto.
3. **Diseño del modelo base:** Se optó por una arquitectura BiLSTM, reconocida por su efectividad en tareas de PLN secuencial.
4. **Entrenamiento y evaluación:** La fase final se centró en entrenar este modelo y analizar sus resultados para establecer un rendimiento de referencia sólido.

5. Resultados

El modelo fue entrenado durante 5 épocas con un tamaño de lote de 64.

5.1. Curvas de aprendizaje



Análisis: Las curvas muestran un buen ajuste. La exactitud de entrenamiento y validación aumentan de forma paralela, mientras que las pérdidas disminuyen, indicando que el modelo está generalizando bien a los datos no vistos durante el entrenamiento sin signos de sobreajuste significativo.

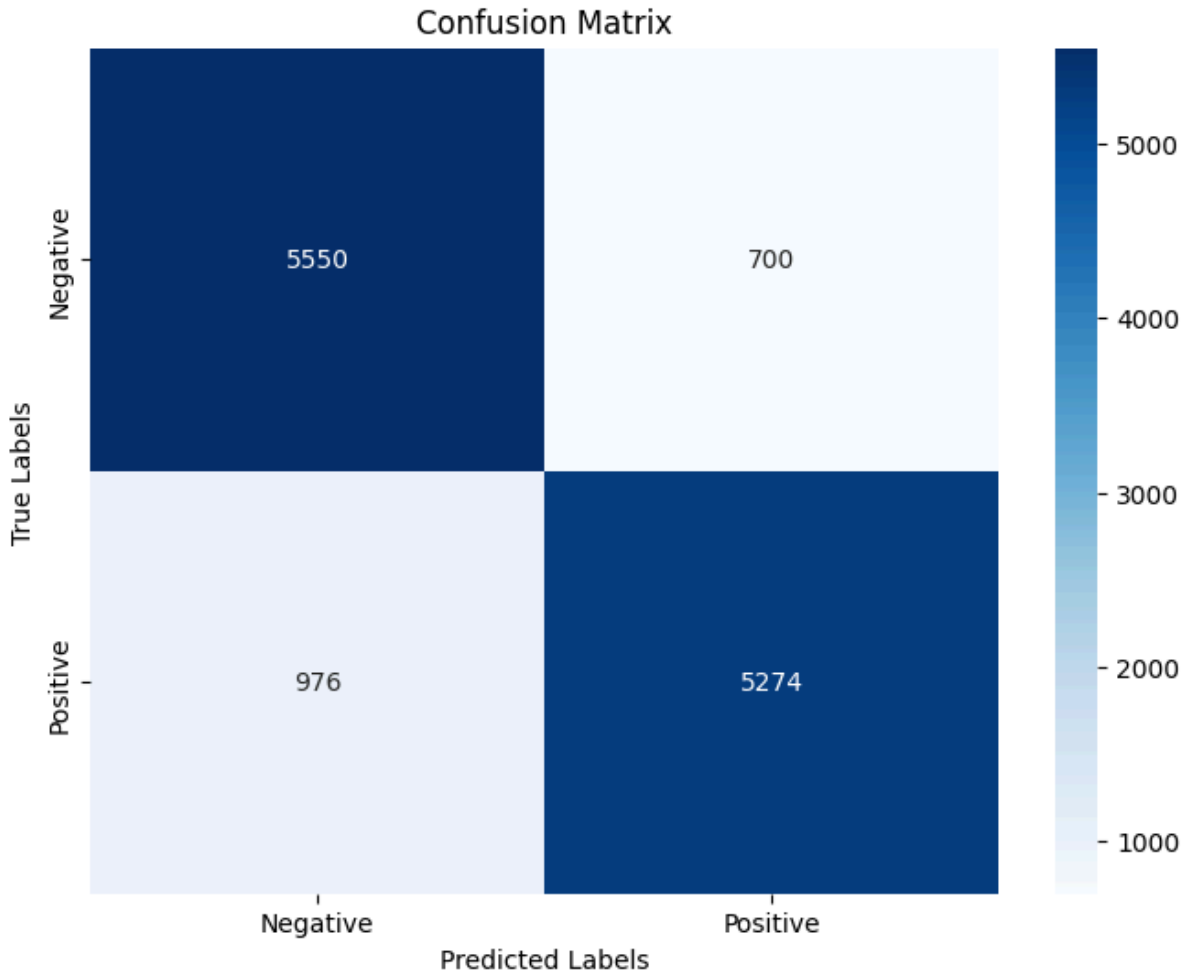
5.2. Métricas de Desempeño en Test

El modelo fue evaluado en 12,500 reseñas nunca antes vistas, obteniendo los siguientes resultados:

- **Accuracy General: 86.6%**

Clase	Precisión	Recall	F1-Score
Negativa	0.85	0.89	0.87
Positiva	0.88	0.84	0.86

- **5.3. Matriz de Confusión**



Análisis: La matriz de confusión confirma el buen rendimiento del modelo. Se observa una alta cantidad de predicciones correctas en la diagonal principal. El modelo tiene una ligera tendencia a ser mejor reconociendo reseñas negativas, como lo indica su mayor Recall (0.89).

6. Conclusiones y Trabajo Futuro

El modelo BiLSTM desarrollado es capaz de clasificar el sentimiento de reseñas de películas con una alta precisión (86.6%), demostrando la efectividad de las redes recurrentes para tareas de procesamiento de lenguaje natural.

7. Herramientas utilizadas

- **Lenguaje:** Python 3
- **Librerías principales:** Pandas, NLTK, Scikit-learn, TensorFlow/Keras, Matplotlib, Seaborn.
- **Entorno de desarrollo:** Jupyter Notebooks / Google Colab.