Universidad del Valle de Guatemala
Facultad de ingeniería

Data Science
Laboratorio 4

Andrés de la Roca
Jun Woo Lee

Guatemala, 2023

Índice

Exploración y procesamiento de datos de datos

Luego de una exploración del conjunto de datos realizado por medio de una investigación se descubrió lo siguiente:

El conjunto de datos es específicamente para tareas de procesamiento de lenguaje natural y clasificación de texto. Este conjunto de datos contiene críticas de películas recopiladas de la base de datos del sitio IMDB. En total contiene 50,000 reseñas de películas, divididas en 25,000 registros de entrenamiento y 25,000. Cada conjunto se divide de manera equitativa entre críticas positivas y negativas.

Las reseñas están en un formato de secuencia de números enteros, donde cada número representa una palabra única, las palabras más comunes se asignan números bajos.

Las reseñas de este conjunto de datos pueden tener diferentes longitudes de palabra, lo que en esencia significa que las secuencias de números representativas de las palabras también variarán en su longitud.

Del preprocesamiento se puede decir lo siguiente; Primero, las reseñas de películas en texto sin procesar se tokenizan, lo que significa que se dividen en palabras individuales. Luego, se asigna un número entero único a cada palabra en función de su frecuencia en el conjunto de datos, donde las palabras más comunes tienen números más bajos. Esto crea un vocabulario que se utiliza para convertir las reseñas en secuencias de números enteros.

Además de la tokenización y la asignación de números, se aplica un proceso llamado "padding" para asegurarse de que todas las secuencias de números tengan la misma longitud. Dado que las reseñas pueden variar en longitud, se rellenan las secuencias más cortas con ceros en la parte delantera o trasera para igualar la longitud máxima deseada.

El conjunto de datos se divide luego en conjuntos de entrenamiento y prueba, con etiquetas de clasificación asignadas a cada reseña para indicar si es positiva o negativa. Con todas estas transformaciones, el conjunto de datos se convierte en una matriz de números enteros que representa las reseñas de películas y sus etiquetas de clasificación.

Este proceso de preprocesamiento es crucial para que los modelos de aprendizaje automático, como las redes neuronales, puedan trabajar con datos de texto y realizar análisis de sentimiento. Los datos preprocesados se utilizan luego como entrada para entrenar y evaluar los modelos de análisis de sentimiento.

Construcción del modelo

El objetivo de esta práctica es entrenar el modelo para que se pueda predecir si una reseña de película es positiva o negativa basándose en el texto de la reseña. Para este modelo se utilizo una red neuronal secuencial diseñada para tareas de procesamiento de lenguaje natural, sus componentes son los siguientes:

En primer lugar, se utiliza una capa de Embedding, que tiene como objetivo representar palabras en un espacio vectorial de 128 dimensiones. La entrada de esta capa es un

vocabulario de 50,000 palabras, lo que significa que puede manejar una amplia gama de palabras. Esta representación vectorial permite que la red neuronal capture las relaciones semánticas entre las palabras y aprenda patrones en el texto.

A continuación, se emplea una capa Bidirectional LSTM (Long Short-Term Memory) con 64 unidades. Esta capa procesa secuencialmente el texto en ambas direcciones (izquierda a derecha y derecha a izquierda), lo que le permite capturar tanto el contexto anterior como el posterior de cada palabra en la secuencia. La opción "return_sequences=True" indica que esta capa produce una secuencia de salidas en lugar de una única salida, lo que es importante para mantener información detallada en cada paso de tiempo.

Luego, se utiliza una capa GlobalMaxPool1D para reducir la dimensionalidad de la secuencia de salidas de la capa LSTM y retener las características más relevantes. Esto ayuda a evitar el sobreajuste y a acelerar el proceso de entrenamiento.

Después de la capa de reducción de dimensionalidad, se agrega una capa Dense (totalmente conectada) con 30 unidades y función de activación ReLU. Esta capa realiza operaciones lineales y no lineales para aprender patrones más complejos en los datos.

Se incluye una capa de Dropout con una tasa del 5% para prevenir el sobreajuste al desactivar aleatoriamente el 5% de las neuronas durante el entrenamiento.

Finalmente, la última capa es una capa Dense con una única unidad y función de activación sigmoide. Esta capa produce una única salida binaria, lo que la hace adecuada para tareas de clasificación binaria, cómo determinar si un comentario es positivo o negativo. La activación sigmoide comprime la salida a un valor entre 0 y 1, que se interpreta como la probabilidad de pertenecer a una de las dos clases (por ejemplo, positivo o negativo). En resumen, este modelo combina capas de procesamiento de texto, LSTM bidireccional y capas de redes neuronales densas para abordar tareas de clasificación de texto.

Entrenamiento del modelo

El entrenamiento de este modelo se realiza compilando con parámetros para configurar la función de pérdida, el optimizador y las métricas de evaluación. En este caso, se utiliza la función de pérdida 'binary_crossentropy', que es común en tareas de clasificación binaria y mide la discrepancia entre las predicciones del modelo y las etiquetas reales. El optimizador 'rmsprop' se encarga de ajustar los pesos de la red durante el entrenamiento para minimizar la pérdida. Además, se define la métrica de 'accuracy' (exactitud) para evaluar el rendimiento del modelo en términos de la proporción de predicciones correctas.

Una vez configurado, el modelo se entrena con el método fit. Los datos de entrenamiento y las etiquetas correspondientes se utilizan para ajustar los pesos del modelo durante el proceso de entrenamiento. El parámetro batch_size determina el tamaño de los lotes de datos que se utilizan en cada iteración del entrenamiento, y el modelo se entrena durante 5 épocas (epochs).

Además, se especifica una división del 20% de los datos de entrenamiento para su uso como conjunto de validación utilizando validation_split. También se proporciona explícitamente un conjunto de validación separado para los datos y etiquetas utilizando, lo que permite evaluar el rendimiento del modelo en datos no vistos durante el entrenamiento.

En resumen, este código configura y entrena el modelo para una tarea de clasificación binaria utilizando una función de pérdida específica, un optimizador y métricas de evaluación. El modelo se entrena a lo largo de varias épocas, con lotes de datos y una división para la validación. El rendimiento del modelo se evalúa en función de la exactitud en el conjunto de validación

Resultados

10 épocas

Archivo de referencia	Nuestro modelo
782/782 - 14s - loss: 1.2683 - accuracy: 0.7986 - 14s/epoch - Pérdida de la Prueba: 1.2682605981826782 Exactitud de la Prueba (Test accuracy): 0.7986000180244446	Loss: 0.5552350282669067 Accuracy: 0.8598799705505371

5 épocas

Archivo de referencia	Nuestro modelo
782/782 - 14s - loss: 1.2683 - accuracy: 0.7986 - 14s/epoch - Pérdida de la Prueba: 1.2682605981826782 Exactitud de la Prueba (Test accuracy): 0.7986000180244446	Loss: 0.30403923988342285 Accuracy: 0.8712000250816345

En comparación con el modelo anterior, hemos logrado avances significativos en términos de rendimiento. El modelo anterior obtuvo una pérdida de 1.2683 y una precisión del 79.86%. Mientras que en el nuevo modelo, al entrenar durante 10 épocas, se obtuvo una pérdida de 0.5552 y una precisión de 85.99%. Una mejora mucho mejor ya pasando los 80% de precisión. Sin embargo, se notó que después de la 5ª época, la tasa de aprendizaje ya no mostraba mejoras significativas, sino que en realidad empezaba a empeorar. Por lo que, se decidió aplicar early stopping y callback por si acaso en las 5 etapas antes de que terminara viera un empeoramiento en una época para evitar el overfitting. Al hacerlo, el modelo logró una pérdida de 0.3040 y una precisión del 87.12%, llegando a tener un rendimiento mejor que el de 10 épocas. Esta decisión de limitar el entrenamiento a 5 épocas no solo mejoró la eficiencia del modelo, sino que también previno el overfitting y garantizó una mejor generalización de datos. Estas mejoras que se pueden ver en nuestro modelo sobre el de referencia es debido a que el nuevo modelo incorpora una serie de mojarras que lo deja capturar relaciones más complejas en los datos como la capa LSTM se cambió de unidireccional de 128 unidades a bidireccional de 64 unidades. Lo que permite que el modelo tenga información sobre pasos de tiempos anteriores y futuros. Adicionalmente se incorporó una nueva capa de GlobalMaxPooliD que permite que el modelo reduzca la dimensionalidad y capture las características más importantes de las secuencias. Finalmente, también se agregó el dropout y la capa densa de activación relu que ayudó a que el modelo pueda

aprender representaciones más complejas y también ayuda a regularizar para que evite que el modelo depende mucho de una neurona específica.

Conclusiones

- La inclusión de una capa de Dropout con una tasa del 5% ayuda a prevenir el sobreajuste del modelo durante el entrenamiento, lo que es especialmente importante en NLP debido a la alta dimensionalidad de los datos.
- La utilización de LSTM bidireccionales permite que el modelo capture tanto el contexto anterior como el posterior de cada palabra en la secuencia, lo que es beneficioso en tareas de NLP donde el contexto es crucial.
- El uso de una capa de Embedding permite al modelo representar palabras en un espacio vectorial, lo que facilita la captura de relaciones semánticas y patrones en el texto.
- La implementación de técnicas como early stopping y la incorporación de capas adicionales como LSTM bidireccional y GlobalMaxPool1D permitió al nuevo modelo superar significativamente al anterior, alcanzando una precisión del 7.12%.
- La combinación de regularización mediante dropout y la adición de capas densas con activación ReLU demostró ser esencial para prevenir el sobreajuste y garantizar que el modelo generalizara eficientemente a datos no vistos.

Referencias

Keras (2023) IMDB movie review sentiment classification dataset. Extraído de keras.io

Van Burskirk, A. (2023) NLP Models: The Importance of Parameters. Extraído de wordbot.io

Verhaar, P. (2020) Language Models: Battle of the Parameters. Extraído de Medium

Janetzky, P. (2021) Natural Language Processing: From one-hot vectors to billion parameter models. Extraído de Medium