Project Report: Review Sentiment Classification

Introduction:

This project focuses on sentiment classification of product reviews. The goal is to classify reviews as either positive or negative. To achieve this, we utilized a dataset from Kaggle containing Amazon reviews (https://www.kaggle.com/datasets/kritanjalijain/amazon-reviews). Given the nature of the dataset and the need for efficiency, we made specific choices in data preprocessing and model architecture.

Dataset:

We used a dataset of Amazon product reviews obtained from Kaggle (https://www.kaggle.com/datasets/kritanjalijain/amazon-reviews). The dataset includes a large number of reviews, but to save time and resources, we initially trained our model solely on the test portion of the dataset. This allowed us to rapidly prototype and experiment with various model architectures and preprocessing techniques.

Data Preprocessing:

Our data preprocessing involved the following steps:

#### 1. Tokenization:

We created a tokenizer with a vocabulary size of 30,000 words to represent the text data efficiently.

#### 2. Sequence Length:

We set a maximum sequence length of 150 words for each review. Sequences longer than this were truncated (“post”), and shorter ones were padded (“post”) with zeros.

Model Architecture:

For our model architecture, we experimented with different designs to balance performance and efficiency. Ultimately, we settled on the following architecture:

```python

model\_review\_clf = tf.keras.Sequential([

# Embedding Layer

tf.keras.layers.Embedding(input\_dim=30000, output\_dim=8, input\_length=150),

# Convolutional Layers

tf.keras.layers.Conv1D(16, 3, activation="relu"),

tf.keras.layers.Conv1D(16, 3, activation="relu"),

tf.keras.layers.BatchNormalization(),

tf.keras.layers.Conv1D(16, 3, activation="relu"),

tf.keras.layers.Conv1D(16, 3, activation="relu"),

tf.keras.layers.BatchNormalization(),

tf.keras.layers.Conv1D(32, 3, activation="relu"),

tf.keras.layers.Conv1D(32, 3, activation="relu"),

tf.keras.layers.BatchNormalization(),

tf.keras.layers.Conv1D(32, 3, activation="relu"),

tf.keras.layers.Conv1D(32, 3, activation="relu"),

tf.keras.layers.BatchNormalization(),

# Bidirectional LSTM Layers

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16, return\_sequences=True)),

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),

# Dense Layers

tf.keras.layers.Dense(units=256, activation="relu"),

tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(units=256, activation="relu"),

tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(units=32, activation="relu"),

tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(units=1, activation="sigmoid")

])

```

Training Strategy:

To mitigate overfitting, we experimented with different strategies. Initially, we observed overfitting when training solely on the test data due to its limited size compared to complexity of data. To address this, we tried to hypertune our model and experiment with different architectures. We trained the model on all the test data of the Amazon data without a validation split more, providing more diverse training examples.

Subsequently, when training on the provided dataset, which is relatively small, we used a larger validation split of 0.5 to prevent overfitting. We fine-tuned the model to minimize overfitting by adjusting dropout rates and batch normalization parameters.

Finally, to achieve the best performance and generalize well to the provided dataset, we trained the model on the entire provided data.

Conclusion:

In this project, we successfully developed a sentiment classification model for product reviews. We leveraged a Kaggle dataset for initial model development, tuned model architecture for efficiency, and addressed overfitting concerns through various training strategies. The final model was trained on the provided dataset, achieving a balance between accuracy and computational efficiency. This project showcases the importance of efficient model design and adaptation to different data scenarios.