Efficient Sentiment Analysis using BERT on Amazon Comments Dataset.

1.Samyak Jain

Dataset

The dataset used in the model is the data collected from Amazon. The data comprises of the comments from different customers with their sentiment values defined as 1 (for positive review) and 0 (for negative review). The name of the dataset used is amazon_cells_labelled.csv.

Preprocessing the data included making sure that the comments are in right datatype (i.e String) to be processed by the transformer as a string only and converting the unnecessary string values to appropriate numeric values for the sentiment. And thereon dividing the dataset into training and test data with test data as 20% of the dataset.

Methodology

Used bert-base-cased as the pre trained transformer model as this model works good on English cased text when you want to predict one of the two values from the dataset (Binary Classification) and the question asked is for sentiment analysis. So, the problem can be approached with this transformer. The Optimizer Algorithm used is Adam. The choice of framework is TensorFlow as in PyTorch the boilerplate code is more as compared to TensorFlow for BERT.

The activation function used in the model is Sigmoid activation function. The choice of this activation function is because of its ability to perform well on binary classification tasks, and since the problem is a sentiment analysis problem, I chose this activation function. It is differential and a monotonic function.

Hyperparameters are external parameters that helps to define the working of the algorithm. Epochs used in the model are 3, which define that the whole dataset has been gone through thrice. Increasing the number of epochs will improve the performance of the model but will take more time to train as the number of epochs increases.

Result

The model achieved an exceptional accuracy of 0.98 on the dataset with a loss value of 0.07. There is always room for improvement, we can increase the accuracy of the model by data augmentation, having more values and features to learn from. We can also increase the model's performance by increasing the number of epochs and fine tuning the hyperparameters. With the help of early stopping and other regularization techniques we can further increase the model's performance.

```
Epoch 1/3
77/77 [==============================] - 543s 7s/step - loss: 0.4332 - accuracy: 0.8182 -
Epoch 2/3
77/77 [==============================] - 489s 6s/step - loss: 0.1821 - accuracy: 0.9383 -
Epoch 3/3
77/77 [==============================] - 497s 6s/step - loss: 0.0693 - accuracy: 0.9821 -
```

```
loss: 0.4332 - accuracy: 0.8182 - val_loss: 0.2958 - val_accuracy: 0.8766

loss: 0.1821 - accuracy: 0.9383 - val_loss: 0.2877 - val_accuracy: 0.8831

loss: 0.0693 - accuracy: 0.9821 - val_loss: 0.4499 - val_accuracy: 0.8831
```

And in another attempt with 1 epoch we achieved 87% accuracy.

```
77/77 [==============================] - 478s 6s/step - loss: 0.4268 - accuracy: 0.8003 - val_loss: 0.3339
20/20 [==============================] - 33s 1s/step
Test Accuracy: 87.01%
```

```
p - loss: 0.4268 - accuracy: 0.8003 - val_loss: 0.3339 - val_accuracy: 0.8831
```