**Sentiment Analysis using BERT**

GitHub Link: https://github.com/Abiben100/Assignment\_2\_LLM

1. **Introduction:**

In recent years Large Language Models have revolutionized the field of Natural Language Processing enabling AI to understand human like text with great accuracy, some of the groundbreaking models are Chat GPT, Gemini AI, Perplexity etc., among these, BERT introduced by Delvin et al. (2019) **[1]** have set a new variety of downstream NLP tasks such as Classification, Q&A and sentiment analysis.

For this project, I fine-tuned a pre-trained BERT model on the IMDB movie reviews dataset from Hugging face to classify reviews as either positive or negative. The goal was to use BERT’s strong language understanding and adapt it to this specific task. I handled data preprocessing, made small tweaks to the model, and trained it on labelled reviews. I then evaluated its performance using accuracy, F1 score, and a confusion matrix to see how well it did.

1. **Methodology:**
   1. **Dataset:**

I used the IMDB Movie Reviews Dataset, which has 50,000 labelled reviews (half positive, half negative). I loaded it from a CSV file, labelled the reviews as 1 (positive) or 0 (negative), and split the data 80/20 for training and testing.

* 1. **Preprocessing and Tokenization:**

I used the BERT tokenizer from Hugging Face (bert-base-uncased) to convert the reviews into input IDs and attention masks. Each review was limited to 256 tokens, with padding and truncation applied. I also built a custom PyTorch Dataset class to handle the data during training and evaluation.

* 1. **Model Architecture:**

The model was based on a pre-trained BERT. Most of it was frozen to save training time, except for the final encoder layer and the pooler. I added a simple classification head with a dropout layer and a linear layer to predict positive or negative sentiment.

* 1. **Training Setup:**

I trained the model using the AdamW optimizer (learning rate = 2e-5) and CrossEntropyLoss. Training ran for 3 epochs with a batch size of 16, and I made sure to use a GPU when available. After each epoch, I evaluated the model on the test set to check performance.

* 1. **Evaluation Metrics:**

I measured the model's performance using accuracy, F1 score, and a confusion matrix to spot any misclassifications. I also added a custom function to test the model on new sentences for real-time sentiment prediction.

1. **Results and Metrics:**
   1. **Training and Evaluation Metrics:**

The model was trained for 3 epochs with batch size of 16 with a test accuracy of 0.9062, F1 score of 0.9086 which is very high and have less losses.

* 1. **Confusion matrix:**

**A diagram of a confusion matrix

AI-generated content may be incorrect.**

Figure 1: confusion matrix

**True Negatives : 4397** - correctly predicted negative review.

**False Positives : 564** - reviews predicted as positive but negative.

**False Negatives : 374** - reviews predicted as negative but positive.

**True Positives : 4665** - correctly predicted positive reviews.

Looking at the results (figure 1), the model did a pretty good job overall, correctly predicting 9,062 reviews as either positive or negative. It made a few more mistakes by predicting negative reviews as positive (564 times) than the other way around (374 times). When we break it down, about 89% of the reviews the model labelled as positive were positive, which shows good precision. It also managed to catch around 93% of all the real positive reviews, meaning the recall is strong too. The F1 score, which balances precision and recall, was about 0.91, confirming that the model performs well. The slightly higher number of false positives suggests the model tends to be a bit optimistic when predicting positive sentiment, but overall, it’s pretty accurate.

1. **Training Loss and Training Accuracy curve:**

**A graph with a line

AI-generated content may be incorrect.**

Figure 2: Loss curve

To see how well the model was learning, I plotted the training and test loss over each epoch. The training loss went down consistently, which showed that the model was learning from the data. The test loss also decreased alongside it, which is a good sign - it means the model was generalizing well to new data and not overfitting.

1. **References:**
2. Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp.4171-4186. <https://doi.org/10.18653/v1/N19-1423>