CPSC 8430- DEEP LEARNING

Homework3 Report

Name: Hemanth Reddy Dasari

Email: hdasari@clemson.edu

Introduction:

Google developed BERT (Bidirectional Encoder Representations from Transformers), which is a pre-trained language model introduced in 2018. It is a deep neural network that can figure out how the words in a sentence or paragraph relate to one another in context. BERT's main technological advancement is that it is pre-trained on a sizable corpus of text using an unsupervised learning method, which means that it does not need any labeled input to learn.

With its remarkable performance on various natural language processing tasks such as named entity recognition, text classification, sentiment analysis, and question answering, BERT has become a significant breakthrough in the field. Since its creation in 2018, BERT has been widely adopted by both industry and academia, making it an essential component of contemporary natural language processing techniques.

Dataset:

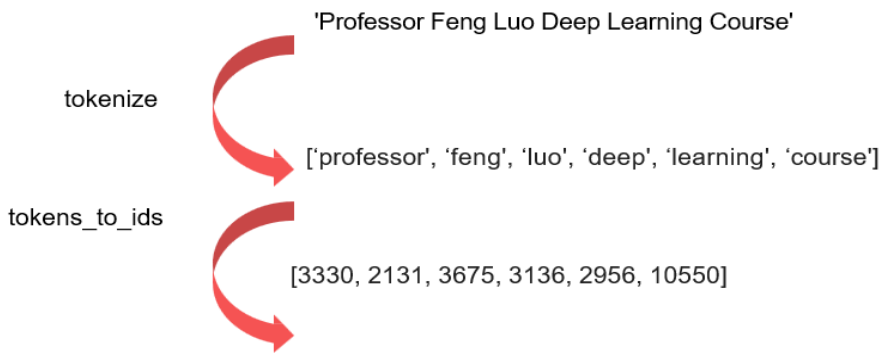
The Spoken-SQuAD dataset, a modified version of the SQuAD dataset that contains speech-to-text data for a variety of topics, is the focus of this report as we examine the efficacy of fine-tuning a pre-trained BERT model for the purpose of question-answering. As part of our research, we examine various hyperparameters and training approaches to determine how they affect the effectiveness of the model.

We added two different levels of white noise to the audio files of the testing set in order to assess the machine's comprehension capacity in situations with poorer audio quality. This resulted in various word mistake rates (WERs).

Tokenization:

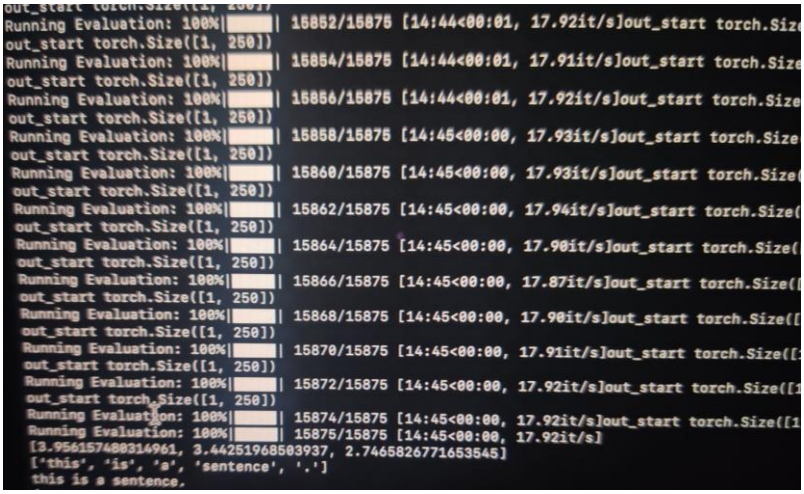
Tokenization is the act of breaking a written text into smaller pieces called tokens, which serve as the fundamental building blocks for tasks involving natural language processing, including named entity recognition, sentiment analysis, and machine translation.

Eg:



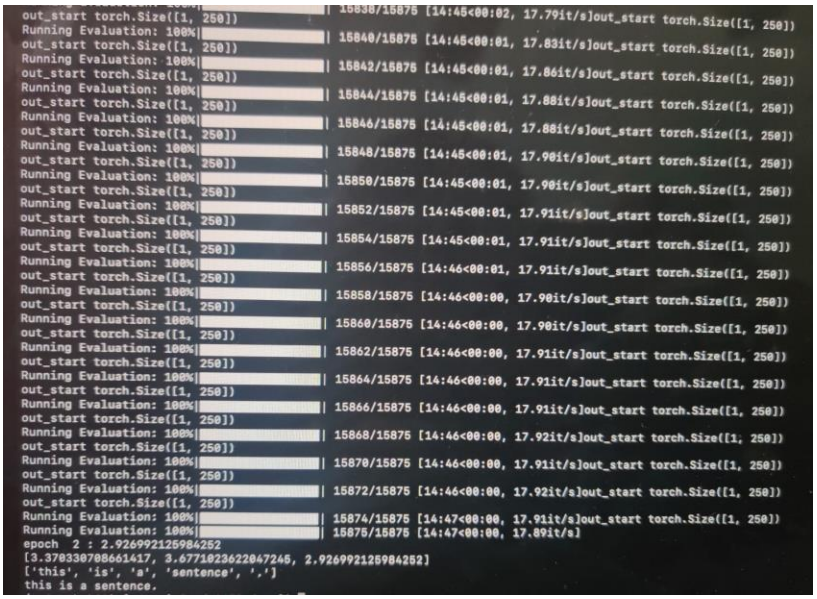
Base Model Implementation:

We apply a pre-trained base model from the hugging face for Question and Answer challenge to this task.



Implementing linear scheduler:

We used a linear scheduler for fine-tuning, where the learning rate decreases linearly with each epoch until it almost reaches 0 by the end of training.



Doc Stride and Preprocessing: