Greeshma Chandra Shekar **C20084146**

CPSC-8430
Deep Learning
Homework 3
Extractive Question answering

GitHub link: https://github.com/Greeshma-C20084146/hw3.git
Email id: greeshc@g.clemson.edu

Introduction

This comprises posing questions about a document or a paragraph and identifying the responses like portions of text included inside actual document.

The Spoken-SQuAD, the first significant sizable spoken question answering dataset, is a dataset of spoken questions and their responses, will be used to fine-tune a BERT model. It is written in English.

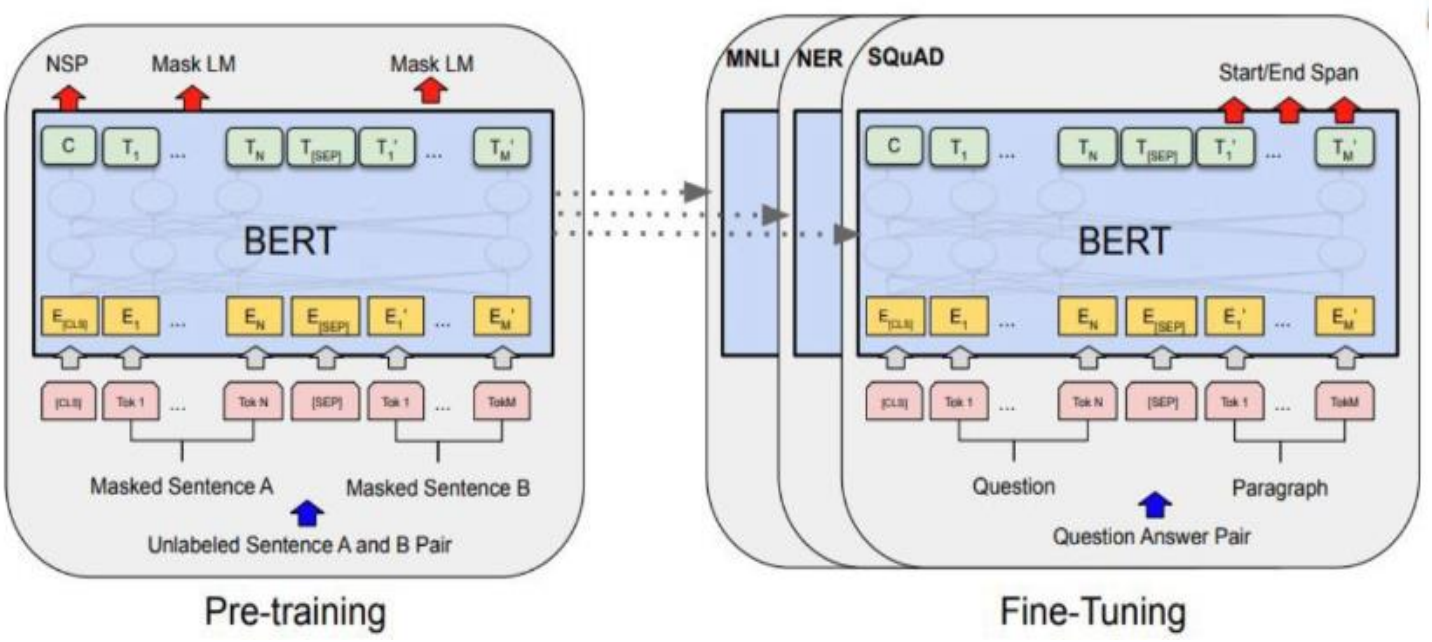
The Dataset Preparation

We get the dataset from their GitHub link, flatten it in Python, and then load it with the Hugging Face library.

```
DatasetDict({
    train: Dataset({
        features: ['id', 'title', 'context', 'question', 'answers'],
        num_rows: 37111
    })
    test: Dataset({
        features: ['id', 'title', 'context', 'question', 'answers'],
        num_rows: 15875
    })
})
```

Using both context and question fields is fairly straightforward. The start field of the answer comprises the index of first character for each and every answer in context, whereas the text field is mostly self-explanatory.

BERT, Bidirectional Encoder Representations from Transformers



Train the data preprocessing

Making use of tokenizer, we convert the input text into ids the model can comprehend. BERT model is employed for this.

Any unwanted spaces that might exist at the start or end of some questions can be removed in the dataset because they don't add anything.

The token’s initial and final positions inside context that relate to the answer to the question must be generated as labels.

Greeshma Chandra Shekar C20084146

The dataset gives the beginning character of context-answer. By adding length of the answer we can determine ending character of the context-answer.

Test data don't require any label generation.

The fine-tuning of the model

Each newly created feature is linked to the original sample it was extracted from using the dataset's ID column.

We set certain hyperparameters such as the total number of epochs model is trained, the learning rate, and some decay of weight as well as specify that the model has to be saved at the end of each and every epoch, forgo evaluation, and results has to be uploaded to the Model Hub.

- Mask the corresponding start and end logits to tokens which is outside of context.
- Start and end logits is then transformed into probabilities by using softmax.
- A score is then assigned to each start_token and end_token pair obtained by multiplying the corresponding two probabilities.
- Sought out pair with the highest score that produced the correct response. For instance, the one with a start_token which is lower than end_token.

We will examine the logit scores for the n_best start as well as end logits, except for the positions that produce:

- An answer not within context.
- An excessively lengthy answer.
- An answer of negative length.

We utilize the conventional AdamW, which is similar to Adam but has been updated to correct how weight decay is implemented. As soon as we have all the objects we need, including the model, train_dataloader, optimizer, and eval_dataloader, they are then sent to accelerator.prepare() method.

We can now analyze our model after the training is over. A tuple is returned from the Trainer's predict() method, with the first initial elements being model's predictions, in this case, a pair with the start logits and end logits. This is sent to compute_metrics() function.