Deep Learning Homework 3- Report

Name -Anjan Kumar Depuru

CPSC 8430-Deep Learning

Github link-https://github.com/Anjandepuru/DEEP-LEARNING-HW-3

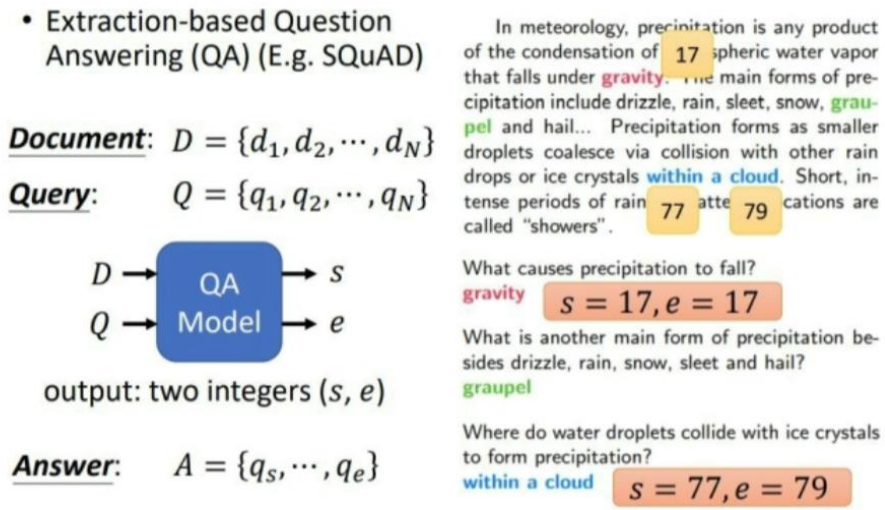
INTRODUCTION:

The main objective of the Question-Answering Model is the computer that can respond to inquiries when given some context, and occasionally when no context is provided. They can choose one choice from a list of available options, extract response phrases from paragraphs, generate answer paraphrases, and more. The true game-changer in NLP has been self-attention, also known as Transformer. Here, I'll talk about one such Transformer architecture variant dubbed BERT, give a quick rundown of its design, and describe how it works when answering questions.

Natural language processing has advanced significantly because of the SQuAD dataset, which has aided academics in creating models that are highly accurate at performing challenging language tasks like question answering.

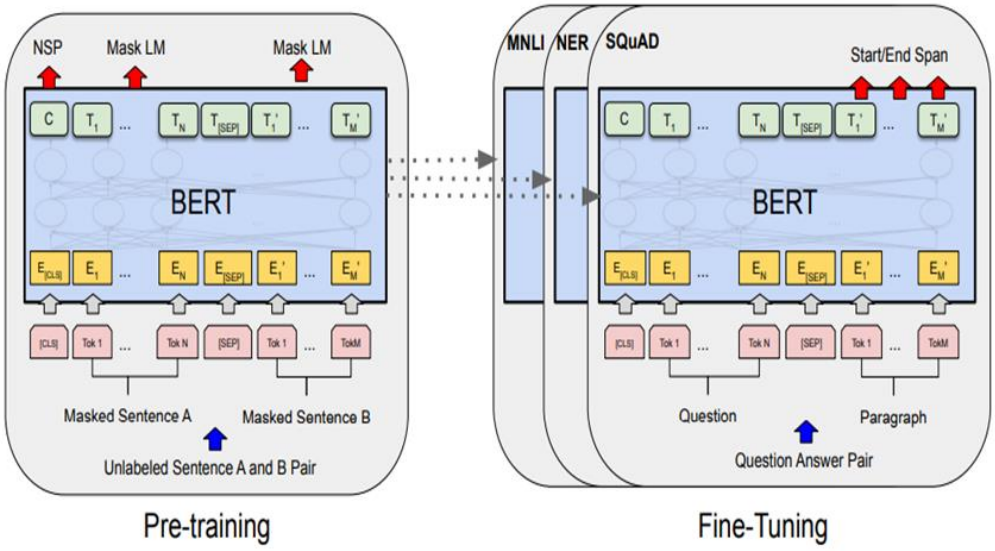
BERT employs Transformer encoder blocks, which utilize Multi-Headed Self Attention to capture contextual relationships among words or sub-words in text. To overcome the limitation of unidirectional language modeling, BERT adopts a pre-training objective called masked language model (MLM). During pre-training, MLM randomly masks certain tokens in the input, and the goal is to predict the masked word by leveraging the surrounding context.

Extractive Question answering is a model that can extract an answer from a given context, which can be in various forms such as a provided text, a table, or even HTML. This problem can be addressed using models based on BERT or similar architectures. The below picture gives the model of Extractive based question answering.

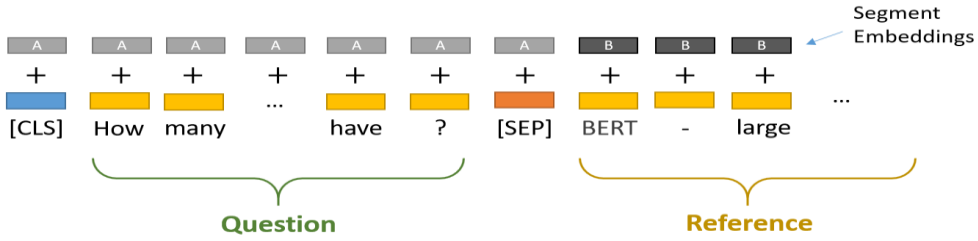


Model used:

BERT is a language model that is pre-trained to analyze both the left and right context at the same time in all of its layers. This unique feature allows BERT to be used for various natural languages processing tasks such as question answering and language inference, with minimal modifications to its architecture. Essentially, only one output layer needs to be added to leverage the power of BERT for these tasks.



The Figure mentioned earlier shows the pre-training and fine-tuning processes of BERT. Both processes use the same architectures, except for the output layers. To initialize models for different downstream tasks, the same pre-trained model parameters are used. During fine-tuning, all variables are updated. In addition, each input example now starts with a special [CLS] symbol and there is a separator token (SEP) that can be used to separate questions and answers.



Question: How many parameters does BERT-large have?

Reference Text: BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

The special [SEP] token is used to indicate a separation between the two pieces of text in the figure above.

To differentiate between the question and the context, BERT uses "Segment Embeddings," which are two embeddings learned during the pre-training process. These embeddings are added to the token embeddings and fed to the input layer, with one embedding for the question and another for the context.

Requirements:

We will be using the following Requirements.

1. Python
2. Pytorch
3. Hugging face Transformers
4. GPU
5. Tokenizer

DATASET:

The Stanford Question Answering Dataset (SQuAD) is a benchmark dataset for developing and evaluating question-answering systems. It consists of a collection of more than 100,000 question-answer pairs that are created by crowdsourcing from Wikipedia articles. The goal is to develop a machine-learning model that can read the passage and answer the corresponding question correctly.

To create spoken versions of the SQuAD dataset, we utilized a two-step process. First, we utilized Google's text-to-speech system to convert the written content of the SQuAD dataset into spoken language. Then, we employed CMU Sphinx to accurately transcribe the spoken text into written form. These spoken and transcribed versions of the SQuAD training and development sets were used to create the training and testing sets for Spoken SQuAD.

The evaluation set of this dataset contains 5,351 question-answer pairs, while the training set contains 37,111 question-answer pairs with a word error rate of 22.77%. In order to test the model's ability to comprehend in real-life situations with lower audio quality, we added two levels of white noise to the audio files of the testing set, resulting in different word error rates.

The SQuAD dataset has played an important role in advancing the field of natural language processing, and it has helped researchers develop models that can perform complex language tasks like question answering with high accuracy.

Approach:

It reads data from two JSON files that contain questions, context, and answers for training and validation. It then preprocesses the data, tokenizes the input and creates a BERT model, and optimizes it using AdamW. The script also includes code to schedule the learning rate and performs

training and validation on the BERT model. Finally, it calculates the accuracy of the predictions and saves the model.

It will be processing data from two JSON files, 'spoken_train-v1.1.json' and 'spoken_test-v1.1.json', which presumably contain training and testing data for the model.

The code then creates histograms of the length distributions for the context and question texts and sets the maximum sequence length and model path for the BERT model.

EXECUTION:

The get_data function in the code is then defined, which reads the data from the JSON file using the file path as an input and returns the number of questions, the number of questions that are both possible and impossible, the contexts, the questions, and the responses.

The training and validation data are then loaded from the corresponding JSON files using the get_data method. The answer_end is added to the responses in the data after they have been converted to lowercase.

Then, the maximum length of the context is set to 250, and the BERT model path is set to "Bert-base-uncased". The BertTokenizerFast is used to tokenize the questions and contexts, and the maximum length is set to 250, truncation is set to True, and padding is set to True.

In each epoch, the code prints the train accuracy and loss and evaluates the model using the WER (word error rate) metric. The WER scores are stored in a list wer_list.

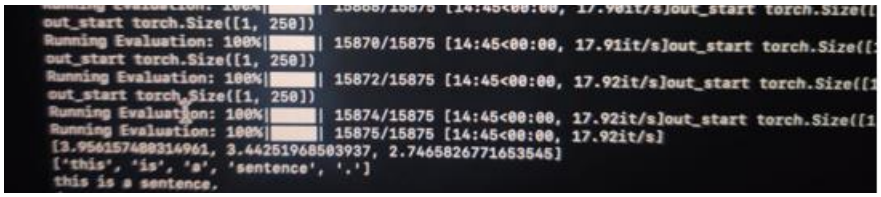
Therefore, the accuracy metric being used in this code is the WER score. The lower the WER score, the better the model's performance.

Tokenization:

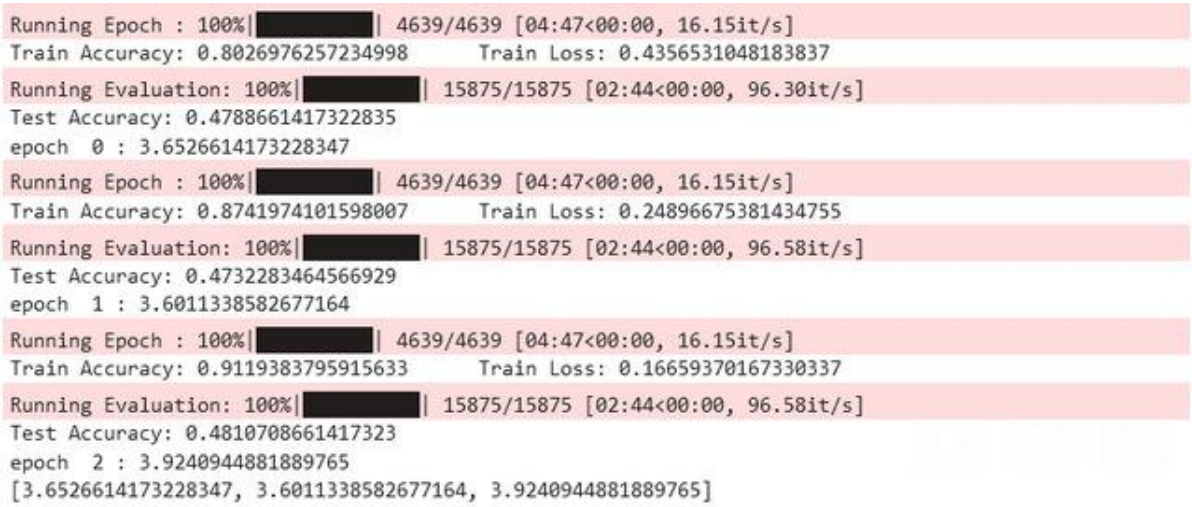
Tokenization is the process of breaking down a text into smaller units known as tokens, which are typically individual words or sub-texts. The purpose of tokenization is to make it easier for a machine to analyze and process a large amount of text by dividing it into manageable segments. Tokenization is an important step in many natural language processing tasks, such as sentiment analysis, language translation, and question answering.

Result:

For this task, we will implement a Question and Answer system using a pre-trained base model from Hugging Face. We will use a pre-trained model to answer questions based on a given context.



When using query and Answer models, it's crucial to pre-process the input text to make sure the model can comprehend the query and provide an appropriate response. One significant method is "doc stride," which is the process of dividing the incoming document (context) into smaller chunks and producing answers for each chunk independently.



From the above figure Train- accuracy and Train- loss is in the following pre-processing model. In each epoch, the code prints the train accuracy and loss and evaluates the model using the WER (word error rate) metric. The WER scores are stored in a list wer_list.

The values for the final text are given below screenshot. Therefore, the distribution of question length is an important factor to consider when evaluating and optimizing a Question Answering system. The distribution of question length in a BERT model depends on the maximum sequence length used during tokenization and the length distribution of the input questions in the dataset.

Tokenization is the process of splitting a sentence into separate words or tokens and storing them in a list or array. In this particular example, the tokenization process resulted in a set of tokens consisting of the words "this", "is", "a", "sentence", and a period ".

Token IDs are numeric representations of tokens in a sentence that machine learning models can process. These IDs are generated by a pre-trained tokenizer that assigns a unique number to each token in its vocabulary. In the below screenshot, the token IDs are [1049, 96, 8, 28292345], which correspond to the specific tokens in the sentence after being processed by the tokenizer.

```
[4.189417322834646, 4.934740157480315, 3.7332913385826774]
['this', 'is', 'a', 'sentence', '.']
this is a sentence.
```