Cortx Writeup

Sean Pereira
Carnegie Institute of Technology
Carnegie Mellon University
Pittsburgh, PA 15213
sapereir@andrew.cmu.edu

Github/CortxHomeProblem

1 Introduction: Existing work and Literature Review

I based my model largely off of existing work done in the domain of Language Modeling and Question Answering. I list some of these projects I have taken from.

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- Question Answering on SQuAD with BERT
- Decoupled Weight Decay Regularization

2 Methods

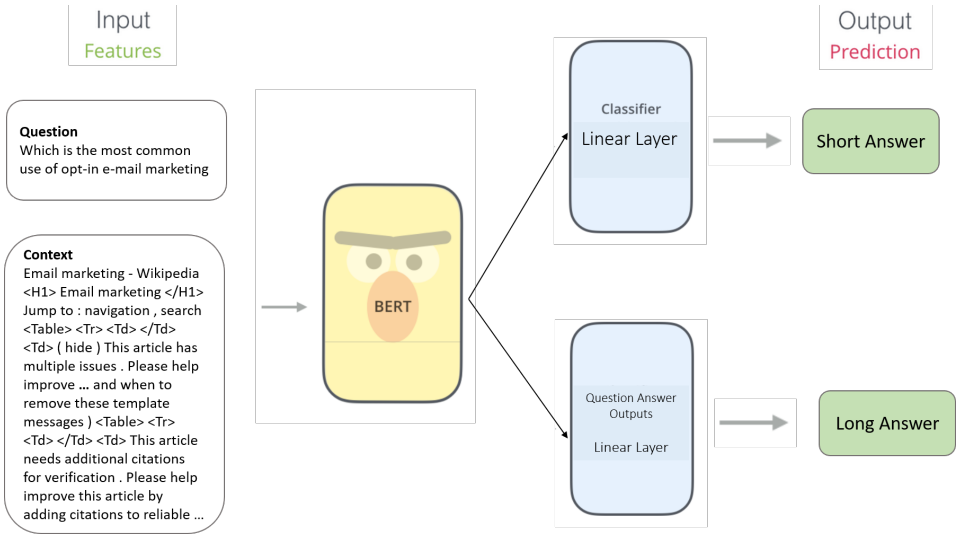
2.1 Data Pre-processing

The maximum allowed sequence that can be inputted into a tokenizer is of length 512. However, the document text or document html is much longer. Therefore, the document text was split into multiple examples further expanding the size of the training dataset. A few hyper-parameters I decided upon were chunksize of examples to convert, max sequence length for each example, max question length and document stride as the example may exist in between. The follow results demonstrate loading times based on the following:

- 1 Thread with functools.parital: 2829 s
- 2 Thread with functools.partial: 1076 s
- 4 Threads with functools.partial: 1011 s 8 Threads with functools.partial: 1025 s
- 1 Thread without functools: 2039 s

These results will differ from machine to machine.

2.2 Architecture



The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. There are two steps in developing framework:pre-training and fine-tuning. I used a pre-trained model unlabeled data over different and various pre-training tasks. For fine-tuning, the BERT model is first initialized with these parameters, and all of the parameters will fine-tuned using labeled data which was pre-processed from the downstream tasks.

To fine-tune such a model I utilize the BERT model in combination with two separate linear layers. These linear layers address the tasks specific to Question Answering. It takes the last hidden layer of BERT, feeds that into a linear layer to generate start and end positions over the input text sequence. Additionally, the following was done similarly where takes the last hidden layer of BERT, feeds that into a linear layer to generate classification of context with dropout. This task only involved answering long questions, but class answers were found to have a tighter constraint and were included in the loss function. Additionally, the class answers are usually either a Yes or No or unknown or long or short, which forces the model to learn under stricter requirements. This model used a sum of CrossEntropyLoss for the start position of the context, end position of the context and classification class of the context. For optimizing I used AdamW paired with get_linear_schedule_with_warmup. The intention behind using AdamW was that it demonstrates better generalization performance since each example in the wiki articles are unique. Additionally, in combination with a proper scheduler used to lower the learning rate in order to reduce the impact of deviating the model from learning on sudden new data set exposure (each unique example).

3 Limitations and Challenges

My current model faces several shortcomings that could have been addressed with additional computational power and more time. Other challenges faced which I've already addressed are also detailed here.

There are several things I would have tried differently with additional hardware or additional time:

- I would have utilized multiple existing libraries built into torch and apex to distribute training into multiple GPUs which I do have experience doing but didn't have the hardware capabilities to undertake. Additionally, A new technique that is currently gaining traction is 3D parallelism which is used primarily on extremely large models but can be used for memory and compute efficiency.
- I would have trained on various hyperparamters such as stride length, sequence length, question length, batch size, and different schedulers. I would have tried various loss functions such as one that penalizes more towards incorrect classification of labels.

- I would have trained on different output layers other than my existing two linear layers such as LSTMs, or CNNs as feature extractors. Additionally, I could have increase the number of linear layers for depth.
- I would have read more intensively into existing architectures from a multitude of companies and situated an effective medium solution that can leverage my hardware.
- Utilization of variable sequence lengths for each example. For example, I could have maintained an invariant where I sampled from a distribution where .95 times the sequence length was 384 and other times it was 512. This allows the model to generalize. Additionally, the same can be said for question length and stride.