Exploration of Attention and Transformers for Question and Answering

Stanford CS224N Default Project

Sing Kwan Ng
Department of Computer Science
Stanford University
singkwan@stanford.edu
Mentor: Chris Waites

Abstract

The project was intended to be an exploration of convBERT model without pre-training, but after training a base BERT model (encoder only Transformer) and achieving very low performance, the objective shifted towards trying to under-standing transformers and attention for Question Answering. Experiments on both hyperparameters and network architecture was done on the BERT model, with conclusion that this model will either overfit, or not converge. A hypothesis is suggested that without large corpus pretraining, simple self attention on a concatenated context and question has big difficiencies vs explicit cross attention to learn SQuAD. QAnet model was also trained for purposes of comparisons and analysis.

1 Introduction

The original project was intended to be implementation of the novel transformer based architecture with modified attention mechanism (convBERT[1]). The plan was to explore the performance and provide some analytical performance. However, due to issues with achieving performance on BERT model, the project has pivoted towards understanding the complexities of achieving performance in the BERT model, with analysis vs other proven attention based models.

2 Related Work

Various transformer based models have been used and have performed very well on Question and Answering benchmarks like SQuAD, while achieving better efficiency than RNN models (higher performance given same training time). Without pretraining, these include RNet, QAnet and TransformerXL, with QAnet achieving 82.2 EM score. With pretraining (on very large corpus on unsupervised task, and finetuned on question answering task) another level was achieved, with pretrained transformer models achieving even higher performance on SQuAD 2.0. For example ALBERT single model achieved 88.7 EM, spanBERT achieving 85.7 EM on SQuAD 2.0.

3 Approach

3.1 BiDAF model

First was to setup the environment and to test the baseline performance, and to gain insights on the training process. This model is an RNN based model, with attention Running the training and evaluation provided expected results.

3.2 BERT model

The BERT[2] model is an encoder only transformer model. One challenge was that BERT has always been used in a pre-trained on large corpus followed by fine tuning on specific use case, e.g. Questions and Answering. In this case, the approach is to design the network as it would have been for the Question and Answering fine tuning, but utilizing pre-trained word embeddings only.

Input Layer: Based on the BERT paper, the input uses a different structure (as seen in the diagram) than the baseline BiDAF. The preprocessing was modified to include [CLS] and [SEP] tokens, and appending context and question in a single input. Glove300 pretrained word embeddings were used. Position embeddings, is a numerical count of the position in the input. Segment embeddings is labelling the context and question as different segments (1 and 0) All the embeddings are summed up to become the input layer, with size (Batch size, Max sequence length, Embedding size)

Tranformer Layers: Each transfomer block is a multiheaded self attention as implemented in the original Attention is all you need paper[3]. This is connected to a stacked fully connected layer, with the output size set to match input size. This transformer block itself is also then stacked.

Output layers: For the fully connected output layer, 2 output architecture's were tested. The first one was referenced from huggingface's implementation of BERT, where the outputs of all final hidden layers of the transformer is connected to a fully connected layer, then to 2 final nodes, representing the start and end token. This is then split at the end. Another approach was to have 2 completely different fully connected layers after the transformer layers, that would end with a single node each predicting start and end tokens. This resembles that for QAnet output layers.

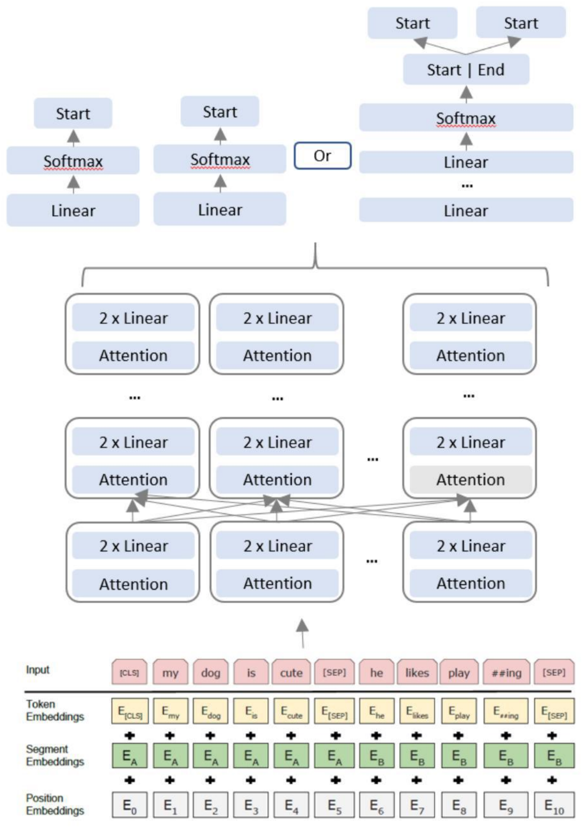

Figure 1: BERT like transformer architecture with 2 different output configurations

However, as described in the experiment section, the model's performance and ability to learn proved to be less than performant, with various variations tested. This lead to the exploration of QAnet as a way to compare and analyze the performance seen for this model.

3.3 QAnet model

This time, an existing code[4] for QAnet model was taken and adapted to current work flow due to time limitations. Permission was requested in Ed[1]. Data inputs was similar to baseline BiDAF model, where questions and context are fed in separately.

Input layer: Both word and Character embeddings are used. In this case, we are using Glove300 word and Character embeddings. The character embedding is fed through a convolution layer, to reduce dimension and extract key attributes. Both are concatenation $[wordemb; charemb]$, and finally padding is finally added to ensure consistent size of input into the network.

Embedding and Model Encoder blocks: It is a combination of [convolution-layer x No. of layers + self-attention-layer + feed-forward-layer]. The self attention here is the same multi-headed self attention as in previous BERT Model. Utilizes depthwise separable convolutions. The only difference between the embedding and model encoder block is the convolution layer number.

Context-query attention layer: The cross attention is calculated and then concatenated to the context and query hidden layers.

Output layer: The start and end outputs have their own heads. The start token is a linear layer applied on the concatenation of first 2 stacked encoder blocks. The end token is the same but applied on the 1st and last stacked encoder blocks.

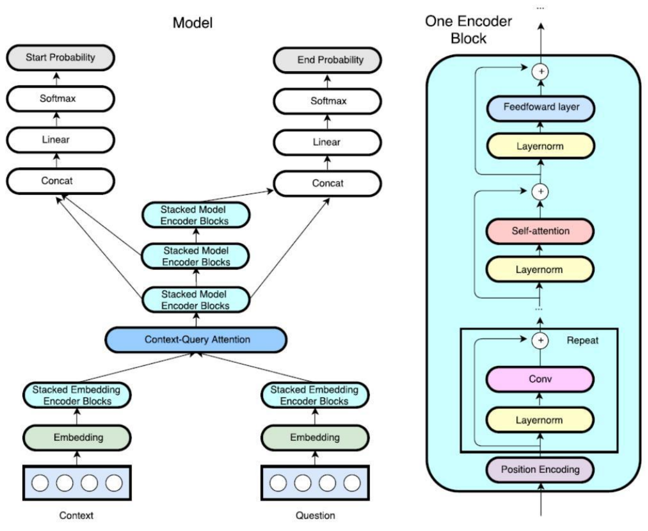

Figure 2: QAnet architecture

4 Experiments

4.1 Data

The dataset used was the provided SQuAD 2.0 data, with training set size of 129,900, and a dev sample size of 5891. This includes questions and context with samples that do not have answers.

[1]https://edstem.org/us/courses/3098/discussion/323474 ; It was made clear that I would explicitly call it out, and credit would be deducted for not implementing it without starter code

4.2 Evaluation method

The evaluation metrics used are the negative log likelihood losses (NLL), the EM and F1 scores (as described in earlier lectures).

4.3 Experimental details

For the BERT model, a lot of problems arose in achieving consistently declining training losses. Even then, the training loss decreased very slowly, without any apparent improvements on the dev set.

Setting	Value
Output network	Dual heads for START END
Transformer layers	8
Dim Transformer FC layer	300
No. of attention heads	12
Batch size	16
Learning Rate	1e-4 (Adam)

Table 1: Bert best settings

For the QAnet model, training stability was not an issue. Majority of settings are kept to the same settings used in the original paper.

Setting	Value
Dimensions	128
Embedding conv layers	4
Model encoding blocks	7
Model conv layers	2
Attention heads	8
Batch size	16
learning Rate	1e-3 (Adam)

Table 2: QAnet settings

4.4 Results

Early on, the main struggle was finding a combination of network architecture and hyper parameters to allow the model training loss to drop and converge. As different network infrastructure pieces and hyperparameters were tuned, the majority of combinations lead to no convergence. Finally, after removing l2 regularization and with specific combination of network architecture settings, the model converges, even achieving similar NLL loss in training as other models. However, the dev performance shows that the model is probably just overfitting, but adding back l2 regularization, the model training loss fails to converge again. The range of variations attempted is in table 3, but most of the time would fluctuate between not converging, or having slow reduction in training loss but not performing on dev set, as seen in figure 3

Experiments	Variations
Output Network	Single | Dual head
Freezing embeddings	1 / 0
Transformer FC layer dim	300, 450
Batch size	16, 32
Learning rate (Adam)	1e-2, 1e-3, 1e-4
l2 regularization	0, 0.01, 3e-7

Table 3: Experiments variations ran

The performance of the "best" BERT model did not perform well, achieving close to BiDAF and QAnet training loss, but not working on the dev set. The baseline BiDAR model and QAnet model

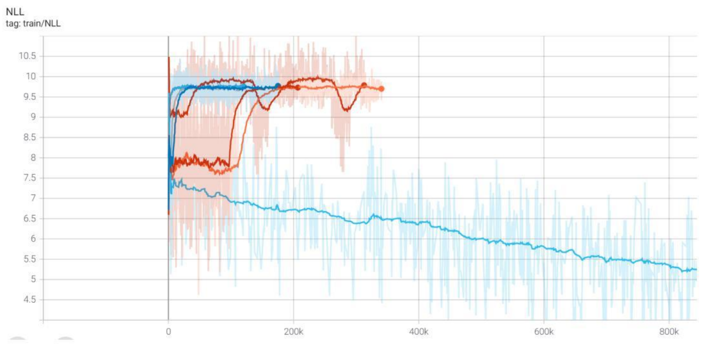

Figure 3: Light blue line at bottom is model that converges, the rest are various attempts with different configurations

however, achieved convergence quickly. Overall, comparing the 3 models in table 4, the worst model is the BERT model (without pretraining), that seems to overfit the data, but not learn effectively to perform on the dev set. The QAnet model beats the baseline BiDAF model but a pretty substantial margin, also converging faster.

Model	Test EM	Test F1	Eval EM	Eval F1
QAnet	61.3	64.6	64.2	57.4
BERT (no pretrain)	-	-	35.9	39.1
BiDAF (baseline)	-	-	56.5	59.8

Table 4: Overall model comparison

[1] BERT not submitted for test set evaluation as it does not even perform on dev set. BiDAF is the baseline model provided.

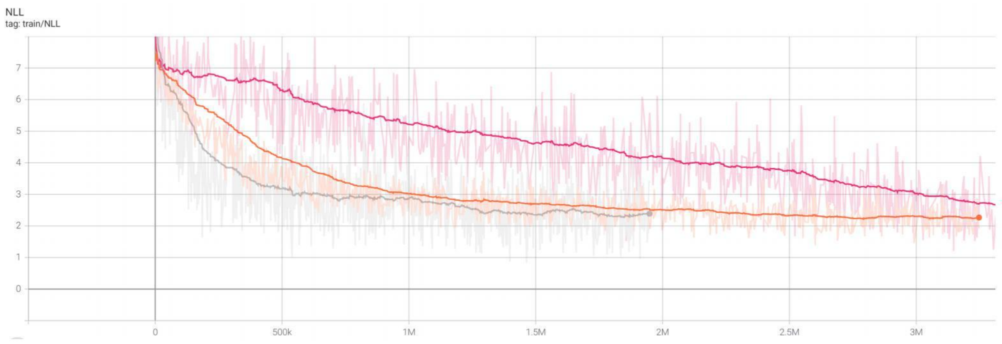

Figure 4: Models training loss comparisons; Pink: BERT, Orange: BiDAF, QAnet: Gray

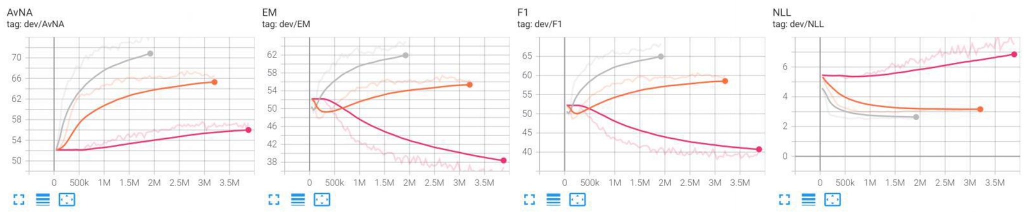

Figure 5: Models evaluation set comparisons; Pink: BERT, Orange: BiDAF, QAnet: Gray

5 Analysis

For the BERT model, it seems that it is unable to actually learn on the question and answering dataset, even with pretrained embeddings. The obvious hypothesis is that the architecture transformer encoder with self attention only isn't able to learn without pretraining, even if using pretrained word embeddings. The 2 pretraining task for BERT, are masked span prediction, and next sentence prediction, which allows it to retain large amount of information. The initial expectations was that even without pretraining, it would be able to achieve some performance. But based on the T5 paper[5], the degredation for using these networks without pretraining is high. This ranges from around 20 % for GLUE and super GLUE, but SQuAD was the task that showed the highest degredation in performance for lack of pretraining, as eeen in table 5

Benchmark	Metric	With pre-train	Without pre-train	percent diff.
GLUE	Avg. Scores	83.3	66.2	-20.4
Super GLUE	Avg. Scores	71.4	53.0	-25.7
SQuAD	EM	80.88	50.31	-37.8

Table 5: T5 pretraining vs no pretraining benchmark comparison

Another thing that could have improved the model is increasing parameter size. The network size tested here is substantially smaller than that of actual BERT, with the the biggest difference in the transformer feedforward linear layer dimension (3072 on BERT vs 300 here) and hidden dimensions for the attention layer is also smaller, at 768 on BERT vs 300 here. Increasing parameters was tested, with feedforward linear dimension increased from 300 to 450, but model performance did not improve. There is a chance that increasing it further (which was not done due to GPU size restrictions) would improve its ability to learn, but would likely have a very low upper bound.

The hypothesis of why BiDAF and QAnet works without pretraining, and able to achieve strong performance on SQuAD 2.0 is on the difference in usage of the attention mechanism. BiDAF uses a bidirectional attention mechanism, concatenating context to question, and question to context attention (together with context hidden states). QAnet uses self attention in the input encoder but separating question and context inputs in its own input layers. After that, an explicit cross attention between context and question is used. BERT utilizes self attention on a concatenated input of both question and answer, with a segment embedding added to the input to help identify the context and query. However, this mixes effectively self attention of context to itself (and question to itself) and cross attention (between context and question).

Last observation is that the data imbalance of no answers could be one of the reasons the BERT model struggled to converge. With about 50 % of the data being no answer, the model would at many times get stuck at predicting no answer for all, which would still provide a baseline performance, like a local minima that the model gets stuck at.

6 Conclusion

The BERT model will either overfit, or not converge if trained on SQuAD data without pretraining on large corpus. This is some what aligned with the findings in T5. A hypothesis is suggested that without large corpus pretraining, simple self attention on a concatenated context and question has big difficiencies vs explicit cross attention to learn SQuAD. QAnet and BiDAF explicit question and query cross attention mechanisms allow it to learn SQuAD effectively.

References

[1] Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Convbert: Improving bert with span-based dynamic convolution, 2021.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[4] Bang Liu. qanet-github. `https://github.com/BangLiu/QANet-PyTorch`, 2017. [Online; accessed 12-March-2021].

[5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.