SQuAD 2.0 Question Answering with Reformer

Stanford CS224N Default Project

Haiyuan Mei
Department of Computer Science
Stanford University
hmei0411@stanford.edu

Abstract

In this course project, I aim to explorer the Reformer [1] based QANet model [2], using pytorch for the question-answering tasks defined in SQuAD2.0 [3]; I will train three models: the baseline BiDAF [4] model, the QANet model which applies the original Transformer [5] model, and the Reformer based QANet. The project will be non-PCE based, confining the study on the Reformer model and the original Transformer [5] only.

1 Key Information to include

External collaborators: No; Mentor: NA, Sharing project: No

2 Approach

2.1 BiDAF baseline

The BiDAF [4] model is described in detail in default final project handout. The default course project code for BiDAF model(https://github.com/minggg/squad) is used to train the first baseline model.

2.2 Transformer and QANet

Transformer is first introduced in Attention Is All You Need [5]. It is the foundation of almost all of today's state of the art language models, including but not limited to BERT, XLNet, ALBERT, etc. I give a brief description of how the so called 'Scaled Dot-Product Attention' and 'Multiple Heads Attention' are calculated. Suppose the word embedding matrix is denoted as $E \in \mathbf{R}^{d \times n}$, where d is the word vector dimention and n is the size of vocabulary. I first make 3 linear transformations from E to $K \in \mathbf{R}^{d_k \times n}$, $Q \in \mathbf{R}^{d_k \times n}$, $V \in \mathbf{R}^{d_v \times n}$: $K = W_K E$, $V = W_V E$, $Q = W_Q E$; The 'Scaled Dot-Product Attention' and 'Multiple Heads Attention' are then built upon the three matrices K, Q, V:

$$\begin{aligned} &\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \\ &\text{MultiHead}(Q, K, V) = \text{Concat}\left(\text{head}_1, \text{head}_2, ..., \text{head}_h\right) W^O \\ &\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{1}$$

Where $W_i^Q \in \mathbf{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbf{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbf{R}^{d_{model} \times d_v}$, and $W^O \in \mathbf{R}^{hd_v \times d_{model}}$. and the output of the multi head attention is a sequence of d_{model} word vectors.

The QANet is very similar to the default baseline BiDAF model, except that its key component the stacked embedding encoders and stacked model encoders changed from RNN attention transformer attention; and the transformer used in QANet has an extra type of sublayer, the convolutional sublayer which starts with a layer norm module and a convolutional module. QANet code is adapted from QANet-pytorch for study purpose.

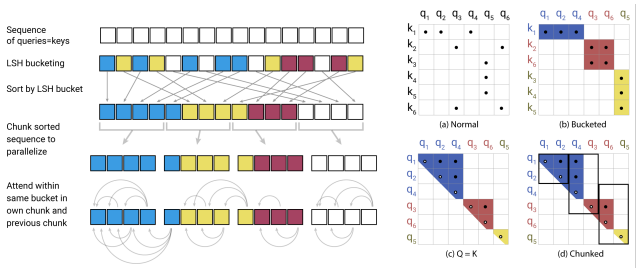

Figure 1: LSH Attention: hash-bucketing, sorting, chunking steps and the resulting causal attentions

2.3 QANet with Reformer

The idea to combine Reformer and QANet structure is original, in order to quick prototype the course project training, the Reformer model is adapted from reformer-pytorch. Reformer(the efficient transformer [1]) is designed to handle 1 million words in context windows, while using only 16GB of memory. It combines two techniques: locality-sensitive hashing to reduce the sequence-length complexity as well as reversible residual layers to reduce storage requirements. I briefly describe the LSH attention here.

As can be seen from transformer Scaled Dot-Product Attention, the term QK^T will have both computational and memory complex $O(l^2)$ which makes it very hard to even fine tune a model on a single GPU. The idea of hashing attention is to only consider a small subset of closest keys for the L elements in Q, which can be achieved by locality-sensitive hashing (LSH). Figure 1 illustrates the whole process: first traverse the l elements and calculate a hash $h(x)$ for each of them which takes $O(l)$ complexity; then sort the elements by the hash value which takes $O(l \log l)$ complexity. The overall before chunking is $O(l \log l)$. I leave the detailed introduction of LSH attention and reversible residual layer to the final report.

3 Experiments

3.1 Data

I used SQuAD V2.0 (Stanford Question Answering Dataset) [3] to train all three models. Each input contains a context paragraph sequence and a question sequence; and the output is the answer to the question, denoted by a pair of indexes indicating the starting and ending index of the span of text in the context paragraph. There are roughly half of the SQuAD examples are no-answer, in order to support this scenario, each context paragraph will be inserted a starting word(index 0) as the first word, and the index 0 will be used to output an unanswered question. The data is devided into 3 portions: train (129,941 examples), dev (6078 examples) and test (5915 examples).

3.2 Evaluation method

Same as the default course model, in this project I will use three metrics to to evaluate each of the models: Exact Match (EM), F1 and AvNA, as described below:

- **Exact Match (EM)** Measures whether the prediction is correct: $\text{EM} = \begin{cases} 1, & \text{correct.} \\ 0, & \text{otherwise.} \end{cases}$
- **F1**: he harmonic average of precision and recall: $\text{F1} = 2 \cdot \frac{precision - recall}{precision + recall}$
- **AvNA (Answer vs. No Answer)**: measures the classification accuracy of your model when only considering its answer (any span predicted) vs. no-answer predictions.

3.3 Experimental details

I first train two baseline models: the default BiDAF, with batch size 64, hidden size 100, 1 layer rnn input encoder and 2 layers model encoder; and the adapted QANet model, with batch size 32, hidden

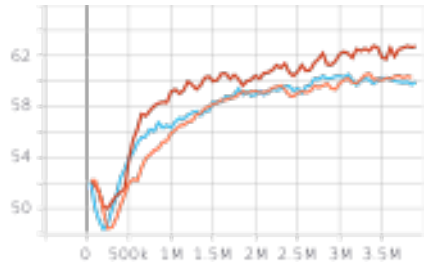

Figure 2: F1 values. Light orange: baseline, dark orange: QANet with transformer, blue: QANet with Reformer

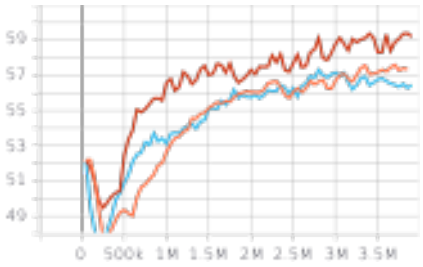

Figure 3: EM values. Light orange: baseline, dark orange: QANet with transformer, blue: QANet with Reformer

size 100, a single layer transformer input encoder sublayer with 4 cnn modules, and a 7 layer model encoder transformer sublayers each with 2 cnn modules. Both trained 30 epochs. I then start training the QANet+Reformer model, with batch size 100, hidden size 100, a single layer reformer input encoder sublayer, and 2 double layer reformer model encoder sublayers for 30 epochs.

3.4 Results

The initial results shows that Reformer based QANet can only roughly achieve the baseline model. However it takes about 15 hours to train, compared to only 4 hours for baseline BiDAF, 9 hours for transformer based QANet. Obviously it needs more work to make it comparable with QANet.

4 Future work

In the rest of the project I will be focusing on improving the Reformer [1] based QANet model [2] to generate equivalent results as the Transformer [5] based QANet. Currently it only achieves the same evaluation results as BiDAF, and it's a lot slower compared to QANet. Future explorations will be to use Reformer in pretrained models such as BERT [6], ALBET [7] and XLNet [8], etc.

References

[1] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.

[2] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension, 2018.

[3] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018.

[4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2016.

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

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[7] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.

[8] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.