

QA System with QANet

Stanford CS224N Default Project

Yiwen Liu

Department of Electrical Engineering
Stanford University
ywliu24@stanford.edu

Mengman Nie

Department of Electrical Engineering
Stanford University
mengman@stanford.edu

Abstract

In the past few years, the most successful QA models are primarily based on RNNs with attention. However, RNN encoders have several shortcomings, such as non-parallelism and difficulty in learning long dependencies. In this project, we use transformer as encoder in place of RNN, and want to find a model that can achieve both accurate and fast reading comprehension.

1 Key Information to include

- Mentor: Daniel Do

2 Introduction

Question answering system has always been an active field in the Natural Language Processing (NLP) researches. In the past few years, the most successful models are primarily based on Recurrent Neural Networks (RNNs) with attention. Though a lot of progress has been made, due to its sequential nature, RNN's operations are unparallelizable, which makes both training and inference slow. In addition, with linear interaction distance, RNNs have difficulty in learning long dependencies. This is a severe problem in QA system, since the context are usually long paragraphs.

Based on these problems, in this project, we implemented a QA model based on Transformer, hoping to achieve both accurate and fast reading comprehension. We focused on reading comprehension among all QA problems, which is to select a part of text from the given context to answer some certain question. Instead of LSTM, this model used convolution layers and self-attention to form encoders. Given a paragraph of context and a question, it will output the probability of each context word being the start or end of the answer. However, against our expectation, this model did not perform very well. The speed is low due to its large amount of parameters, and the accuracy cannot match that of BiDAF because of overfitting.

3 Related Work

The research about QA system has dated back to 1960s. In recent years, it has become increasingly popular in the NLP field. A great number of datasets, such as SQuAD [1], TriviaQA [2], WikiReading [3], etc. have been build, and a lot of end-to-end neural network models have been proposed to tackle this problem. In these years, RNN-based models, such as BiDAF [4], DCN [5], R-Net [6], etc., are predominant in this field. These models are mainly based on RNN with attentions. For example, BiDAF uses bidirectional-LSTM as encoders, with both Context-to-Query (C2Q) Attention and Query-to-Context (Q2C) Attention. DCN applies coattention, which involves a second-level attention computation. R-Net, in addition to C2Q attention layer, adds a self-attention layer.

Recently, attempts have been made to use attentions to fully replace RNNs. In [7], Transformer model has been proposed. It shows great performance in both accuracy and speed without using

RNNs. Therefore, in recent years, works have also been dedicated to QA models with Transformers in place of RNNs, such as BERT [8], QANet [9], etc.

4 Approach

4.1 Baseline

The baseline model is based on Bidirectional Attention Flow (BiDAF) [4]. The original BiDAF model includes trainable character-level embeddings in addition to the word-level embedding, which we exclude from our baseline.

4.2 Character-level Embeddings

In the baseline model, the word embeddings $x_w \in \mathbf{R}^{300}$ are assigned using pretrained GloVe. However, when encounter out-of-vocabulary (OOV) words, GloVe deals with them by simply assigning some random vector values. If not remedied, this random assignment would end up confusing the model.

Therefore, character-level embeddings is applied to handle OOV words. It uses a one-dimensional convolutional neural network (1D-CNN) to find numeric representation of words by looking at their character-level compositions. Specifically, the trainable character embedding is passed through a 1D conv layer and get $x_c \in \mathbf{R}^{200}$. Then, the concatenation $[x_w; x_c] \in \mathbf{R}^{500}$ is passed through a two-layer highway network [10] to obtain the final embedding. The process is shown in Figure 1.

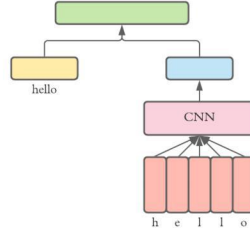


Figure 1: Character embedding

In addition to handling OOV words, character-level embeddings also allow us to condition on the internal structure of words. The original BiDAF model also includes this mechanism.

4.3 QANet

In this part, we implemented QANet as suggested in [9]. It is exclusively built upon convolutions and self-attentions instead of LSTM. As is shown in Figure 2, QANet consists of five major components: an embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer, and an output layer.

4.3.1 Embedding layer

The model used a concatenation of word embedding and character embedding for each word. The approach is the same as described in section 4.2. The final embedding output for each word is $x \in \mathbf{R}^{500}$.

4.3.2 Embedding encoder layer

Instead of the LSTM used in BiDAF, here the encoder layer is a stack of encoder blocks, which is of the form $[\text{conv} \times \# + \text{self-attention} + \text{feed-forward}]$, as shown in the right of Figure 2. For conv layer, we used depth-wise separable convolution¹. Each block contains 4 conv layers. For self-attention, we

¹https://github.com/seungjunlee96/Depthwise-Separable-Convolution_Pytorch

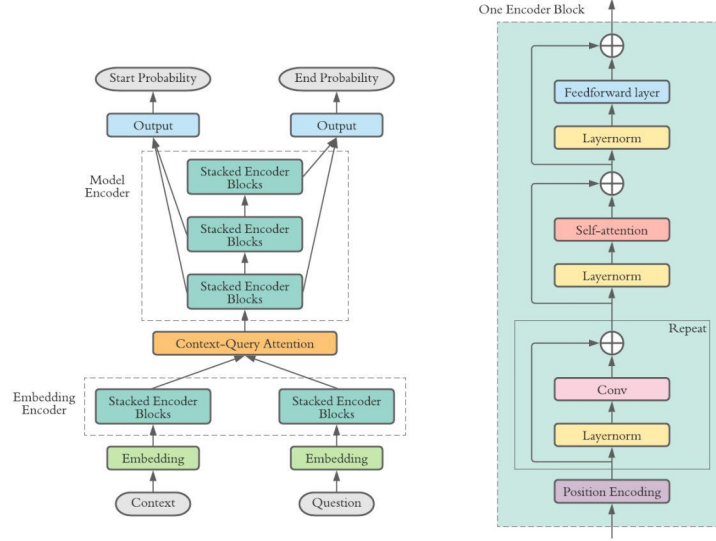


Figure 2: QANet architecture

used multi-headed self-attention² with 8 heads. Each of the conv, self-attention, and feed-forward layer in the block is wrapped in a residual block.

Context and question have separate encoders. In this layer, the encoder block number is 1. For both context and question encoder, the input vector is the output of embedding layers, with $d = 500$. The embedding is first encoded by a position encoder, which has output size $d = 128$, then enters the block. The output of this layer is also of size $d = 128$.

4.3.3 Context-query attention layer

This layer's implementation is similar to that of BiDAF. First it takes both encoded context and question as input and computes the similarities between each pair of words. More concretely, for context hidden state $c_i, i = 1, \dots, N$ and question hidden state $q_j, j = 1, \dots, M$, the similarity score is

$$S_{ij} = w_{sim}^T [c_i; q_j; c_i \circ q_j]$$

Then, for C2Q Attention, we take softmax of S through each row and use the distribution as weight to each q_j . More specifically, it is

$$\begin{aligned} \bar{S}_{i,:} &= softmax(S_{i,:}) \\ a_i &= \sum_{j=1}^M \bar{S}_{ij} q_j \end{aligned}$$

For Q2C Attention, we take softmax of S through each column to get $\bar{\bar{S}}$, then use the multiplication of $\bar{S}$ and $\bar{\bar{S}}$ as weight to each c_i . In equation it is as follows:

$$\begin{aligned} \bar{\bar{S}}_{:,j} &= softmax(S_{:,j}) \\ S' &= \bar{S} \bar{\bar{S}}^T \\ b_i &= \sum_{k=1}^N S'_{ik} c_k \end{aligned}$$

²<https://github.com/jadore801120/attention-is-all-you-need-pytorch>

Finally, the attention layer output is:

$$g_i = [c_i; a_i; c_i \circ a_i; c_i \circ b_i]$$

4.3.4 Model encoder layer

The model encoder is built from the same encoder blocks as the embedding encoder layer. In original QANet, it contains three stacked encoder blocks, each having 7 blocks, and the number of conv layers within the block is 2. The three stacked encoder blocks have outputs M_0 , M_1 , M_2 respectively, each contributes to different output layers.

In our model, we use only one stacked encoder block, with its output feeding into both start and end output layers.

4.3.5 Output layer

There are two separate output layers, one for start position and one for end position, both consisting of a linear layer and a softmax. One uses $[M_0, M_1]$ and outputs the probability of each position in the context being the start of the answer (p^s), the other uses $[M_1, M_2]$ and outputs the probability of each position in the context being the end of the answer (p^e). More specifically,

$$\begin{aligned} p^s &= \text{softmax}(W_1[M_0; M_1]) \\ p^e &= \text{softmax}(W_2[M_1; M_2]) \end{aligned}$$

Finally, the loss function is of the form

$$L = -\frac{1}{N} \sum_{i=1}^N [\log(p_{y_i^s}^s) + \log(p_{y_i^e}^e)]$$

where y_i^s and y_i^e are respectively the true start and end position of the i th example.

5 Experiments

5.1 Data

The dataset is part of SQuAD 2.0. There are 129941 examples in train set, 6078 examples in dev set and 5915 examples in test set. For each example, it is consisted of context, question and answer. The context is an excerpt from Wikipedia. The question is the question to be answered according to the context. The answer is a span from the context.

5.2 Evaluation method

Performance is measured via EM(exact match) and F1 score. For dev and test set, there are three answers provided for each answerable question therefore maximum F1 and EM scores across the three answers are chosen as the final evaluation result.

5.3 Experimental details

There are five model configurations besides the baseline BiDAF model. From the view of model architectures, it has BiDAF based models(BiDAF, BiDAF_c) and transformer based models(QANet, QANet_reduced, QANet_w, QANet_w_reduced). BiDAF_c adds the character level embeddings in addition to the baseline model(BiDAF). QANet is fully implemented according to the architecture in the original paper[9]. QANet_reduced simplifies the QANet in two ways. In embedding encoder, convolutional layers in residual block decrease from 4 to 2. Model encoder reduces from 3 encoder blocks to 1 encoder block and the number of convolutional layer inside it is reduced from 2 to 1. QANet_w gets rid of character level embeddings and position encoding from the QANet. QANet_w_reduced is the combination of previous two simplified versions of QANet.

For all the six models, the learning rate is all set to 0.001. Batch size for BiDAF, BiDAF_c, QANet_reduced and QANet_w_reduced is 64 and it is reduced to 32 for QANet and QANet_w due to the memory limitation. For BiDAF and BiDAF_c, it takes about 30 to 40 minutes to run a single epoch. And it takes about 20 to 30 minutes to run a single epoch for QANet_reduced and QANet_w_reduced. As for QANet and QANet_w, one epoch needs about 90 to 110 minutes.

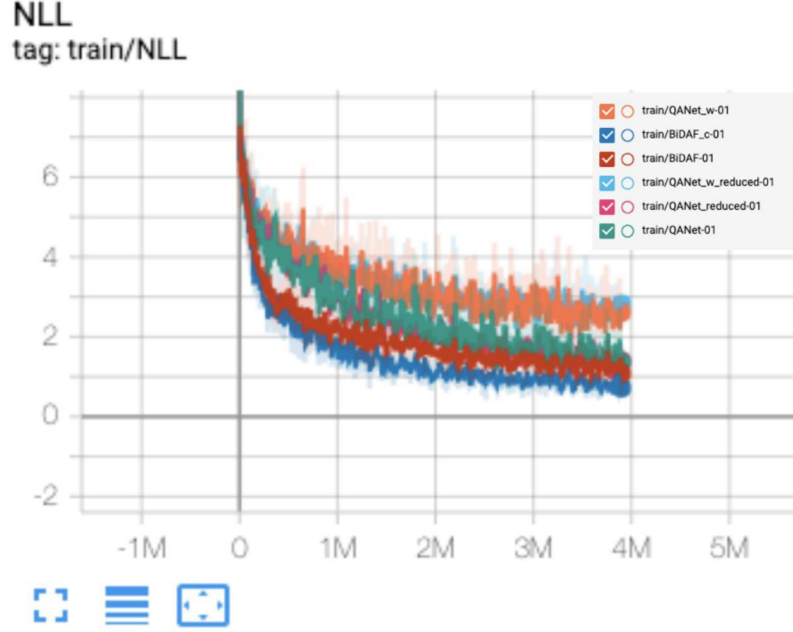


Figure 3: Training loss

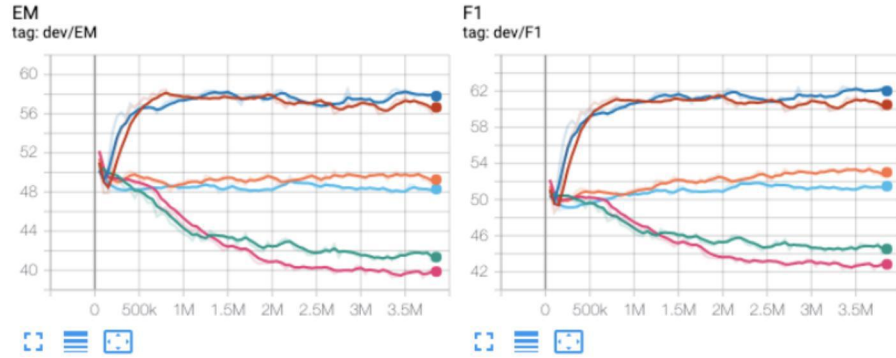


Figure 4: Dev loss, EM and F1

5.4 Results

Among the five models, the best performance comes from BiDAF_c. It achieves 55.976 EM and 60.324 F1 on the test leader board and 58.343 EM and 62.485 F1 on the validation leader board as is shown in table 1. For the four QANet based models, their performances are worse than the baseline model in both EM and F1 scores as is shown in table 2 and figure 4.

Besides, from figure 4, the EM and F1 score of QANet_w and QANet_w_reduced do not increase much though the training losses decrease. For QANet and QANet_w, the EM and F1 score decrease in the training and their training losses decrease as well as is shown in figure 3. This is not consistent with the description in the original QANet paper. One possible reason is that QANet is too complex and is overfitted to the training dataset.

Dataset	EM	F1
Validation set	58.343	62.485
Test set	55.976	60.324

Table 1: Best scores

6 Analysis

BiDAF_c outperforms the baseline model because character level embeddings take the internal structure of words into consider and handle OOV words. For QANet and QANet_w, they are overfitted to the training data possibly because the number of parameters is too large in their architectures, as is shown in the table 4. This is verified by the fact that QANet_reduced and QANet_w_reduced perform better than QANet and QANet_w.

According to table 3, the training speed of BiDAF is about 2x the speed of BiDAF_c. This is predictable because character level embedding applies one-dimensional convolutional neural network to every character in a word, which is time consuming. The training speed of BiDAF_c is about 2x the speed of QANet. In embedding layer, both models use word embedding and character embedding. However, QANet uses stacked encoder blocks instead of bidirectional RNNs in the embedding encoder layer and model encoder layer as is shown in table4. In the embedding encoder layer, one stacked encoder block is consisted of 4 conv layers, a self-attention layer and a feed-forward layer while the conv layers decrease to 2 in the three stacked encoder blocks of model encoder layer.

Model	EM	F1
BiDAF	58.23	61.85
BiDAF_c	58.34	62.49
QANet	50.48	50.80
QANet_reduced	52.14	52.19
QANet_w	49.97	53.64
QANet_w_reduced	49.61	52.29

Table 2: EM and F1 score

Model	speed (it/s)
BiDAF	80-90
BiDAF_c	42-48
QANet	22-27
QANet_reduced	38-45
QANet_w	45-60
QANet_w_reduced	95-120

Table 3: Training speed comparison

Model	BiDAF	QANet
Embedding layer	word embedding	word and character embeddings
Embedding encoder layer	one bidirectional LSTM	one stacked encoder block
Attention layer	context-query attention	context-query attention
Model encoder layer	two bidirectional LSTMs	three stacked encoder blocks

Table 4: Model architecture comparison

7 Conclusion

This project implements a QA system based on Transformer called QANet. It turns out that QANet does not outperform the baseline model(BiDAF) in either accuracy and speed. However, adding character level embedding to the baseline model achieves better F1 and EM score than baseline model though its speed is slower. Reduced versions of QANet are experimented with and their results are better than QANet. One possible reason for the unexpected results for QANet is that QANet is overfitted to the training data due to its complex architecture and too many parameters.

References

- [1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.
- [2] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.
- [3] Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. Wikireading: A novel large-scale language understanding task over wikipedia. *arXiv preprint arXiv:1608.03542*, 2016.
- [4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.
- [5] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.
- [6] W Wang, N Yang, F Wei, B Chang, and M Zhou. R-net: Machine reading comprehension with self-matching networks. *Natural Lang. Comput. Group, Microsoft Res. Asia, Beijing, China, Tech. Rep*, 5, 2017.
- [7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.
- [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [9] 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. *arXiv preprint arXiv:1804.09541*, 2018.
- [10] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.