BIDAF Model Optimization and QANet Model Re-implementation on SQuAD2.0

Yulian Zhou
Department of Computer Science
Stanford University
zhouyl@stanford.edu

Abstract

In this paper, we explored two end-to-end models for the question-answering task on SQuAD2.0 dataset: BIDAF (Bi-Directional Attention Flow) network, and QAnet, a Q&A model fully based on convolution and self-attention without recurrent. Two major goals were accomplished: Firstly, we improved the baseline BIDAF model by introducing character embedding, and linear ReLU activation in the fusion function. Secondly, we reimplemented the QANet model. The BIDAF/QANet single model achieved 63.85/61.94 EM and 67.24/65.47 F1 score on the development set respectively. The ensemble model achieved **EM = 66.644** and **F1 score = 69.7** on the development set, and **EM = 65.224** and **F1 score = 68.438** on the test set.

1 Introduction

Reading comprehension is the ability to process text and understand its meaning, and it is one of the fundamental skills for human-beings. Recent developments in deep learning techniques demonstrate the power of end-to-end neural network systems in many question-answering tasks. One example is the SQuAD (Stanford Question Answering Dataset)[9] challenging. SQuAD is a well-structured question answering dataset based on Wikipedia articles. It is composed of contexts, related questions, and ground truth answers. The questions are derived from short context documents by crowd-workers, and the answers are span-based. Two metrics are used to evaluate models: Exact Match (EM), and F1 score, which measures the weighted average of precision and recall at the token level. In SQuAD2.0, there are over 100,000 questions, and approximately half of them are unanswerable, and those unanswerable questions impose another barrier for machine to recognize whether they know the answer or not in reading comprehension[8].

The concept of "attention" in neural networks has a long history, particularly in image recognition. But only recently have attention mechanisms gain great popularity among recurrent neural networks architecture, which allows models to learn alignments between different modalities, and establish connections among different network layers. Multiple attention mechanisms have been developed and successfully applied in deep learning models of the question answering system, such as bi-directional attention[10], co-attention[12], and self-attention[11].

In this work, we have experimented with the classic BIDAF network with the focus on exploring the self-attention mechanisms. To improve the baseline model: we combined the regular GloVe word embedding with a convolutional character embedding; we tried different similarity and fusion functions on the bi-directional attention layer, and showed that an additional linear layer with ReLU activation in the fusion function boosted the model performance; and we replaced the bi-LSTM in the recurrent neural network layer with a bi-GRU, and increased the hidden size from 100 to 128. Moreover, we also re-implemented the QANet model. QANet achieved state-of-art performance on SQuAD1.1, and a single model gives EM = 82.471 and F1 = 89.306 (compared to BIDAF single model with EM = 67.974, F1 = 77.323), yet its performance on SQuAD2.0 is largely uninvestigated.

Here we compared the performance of the two models on SQuAD2.0 dataset, and showed that my improved BIDAF model gave similar performance as my QANet re-implementation, while BIDAF gave smoother training curve, and took less training time on Azure NV6 machine.

2 Related Work

In 2017, Seo et al.[10] proposed a bi-directional flow network (BIDAF), which emphasizes the notion of context-to-query and query-to-context attention in both directions. There are three major improvements in the attention mechanisms: 1. The attention layer is not constricted into a fixed size. Instead, it is computed for every time step along with the representation from previous layers, which reduces the information loss by early summarization. 2. Attention is memory-less, which means the attention at current time step is independent from attention at the previous time step. It focuses on query and context interaction, and abolishes any carryover of previous incorrect information. 3. The attention flows are bidirectional, from query to context and from context to query, which provide complimentary information, and enable a tight connection between the context and query. The bi-directional attention mechanisms greatly boost the performance of many question-answering tasks, where there is high lexical overlap between query and context.

Self-attention is developed to relate different positions of a single sequence to compute its internal representation. It empowers the recurrent network with stronger capability to memorize and to discover relations among tokens[1]. Self-attention has been successfully applied in a variety of tasks, such as reading comprehension, sentiment analysis, machine translation, and so on. In 2017, Vaswani et al.[11] introduced the Transformer model, which relies solely on self-attention without recurrence or convolutions. The Transformer not only shortens the training time significantly, but also achieves better scores than previous models in many tasks. In 2018, Yu et al. [13] proposed a new architecture – QANet, which replaces RNNs with exclusively convolution and self-attention in regular Q&A models. The models perform 3x to 13x faster in training and 4x to 9x faster in inference on the SQuAD dataset, while show equivalent accuracy compared to recurrent models like BIDAF.

3 Model

Figure 1 illustrates the BIDAF model and the QANet model, which share many similar components. Specifically, they both contain a word + character embedding layer, followed by a highway network layer; they use the same bi-directional attention mechanism; and the output layers are linearly transformated followed by softmax. They use different encoding mechanisms – BIDAF uses recurrent in the embedding encoding and modeling layer, while QANet uses encoder blocks composed of convolution and self-attention layers.

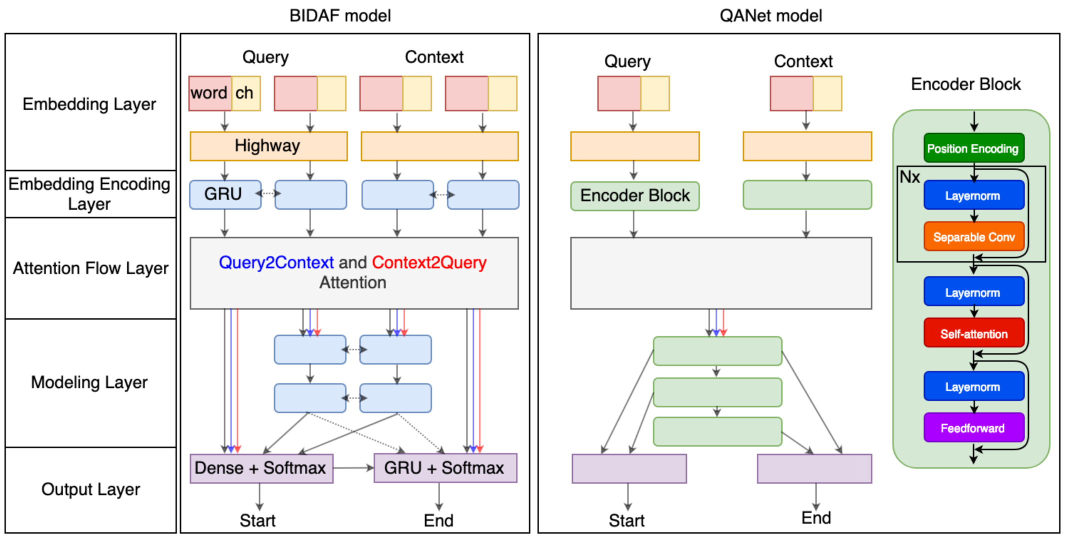

Figure 1: Architectures of BIDAF (left) and QANet (right) model

1. Embedding Layer
Word embedding: Pre-trained GloVe word vector $\boldsymbol{w} \in \mathbb{R}^{300}$ is used with fixed value during training.
Character embedding: Each character is initialized as a random vector $\boldsymbol{x} \in \mathbb{R}^{p}$ (BIDAF $p = 64$, QANet $p = 200$), and each word is truncated or padded up to 16 characters, where its character embedding $\boldsymbol{x} \in \mathbb{R}^{p \times 16}$. The character embeddings are passed through a convolutional network with bias and without padding, whose kernel size $k = 5$ and filter size $f = 200$. The output is activated with ReLU and then row max-pooled to represent word embedding $\boldsymbol{c} \in \mathbb{R}^{f}$.
Word embedding is concatenated with character embedding for each word. In BIDAF model, it is linearly projected to $\boldsymbol{x} \in \mathbb{R}^{d}$ and passed through a two-layer highway network. In QANet model, it is passed through a highway network, and then projected to $\boldsymbol{x} \in \mathbb{R}^{d}$.

2. Embedding Encoding Layer
BIDAF: The model uses a bidirectional Long Short-Term Memory (LSTM)[5] or Gated Recurrent Unit (GRU)[3], and the forward and backward outputs are concatenated for each word $\boldsymbol{x} \in \mathbb{R}^{2d}$.
QANet: The model uses one encoder block, composed of 4 separable convolution layers[2], 1 self-attention layer and 1 feed-forward layer[11], placed in residual blocks with layer norm.
Given the high lexical similarity between context and query, the same encoder block was shared between query and context in both models.

2a. Positional encoding Before the inputs are fed through the encoder, positional encodings are added to the word embeddings according to their positions in the input sequence. We use the sine and cosine functions of different frequencies $PE_{(pos,2i)} = sin(pos/10000^{2i/d})$ and $PE_{(pos,2i+1)} = cos(pos/10000^{2i/d})$ as in the Transformer paper[11].

2b. Residual block with layer norm

$$\boldsymbol{x} = \boldsymbol{x} + f(\text{norm}(\boldsymbol{x}))$$

2c. Depthwise and separable convolution

$$\text{DepthwiseSeparableConv}(\boldsymbol{x}) = \text{ReLU}(\text{PointwiseConv}(\text{DepthwiseConv}(\boldsymbol{x}))$$

Where DepthwiseConv is a Conv1d layer with group_size = input_size, padding = kernal_size / 2 (to keep output_size = input_size), and PointwiseConv is a Conv1d layer with kernel_size = 1.

2d. Multi-head self-attention

$$\begin{aligned} \text{ScaledDotProductAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{T}}{\sqrt{d_k}}\boldsymbol{V}) \\ \text{MultiHeadAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)\boldsymbol{W}^{O} \\ \text{where head}_i &= \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^{Q}, \boldsymbol{K}\boldsymbol{W}_i^{K}, \boldsymbol{V}\boldsymbol{W}_i^{V}) \end{aligned}$$

In the model, input $\boldsymbol{x} \in \mathbb{R}^{l \times d}$ is linearly projected to query $\boldsymbol{q} = \boldsymbol{W}^{Q}(\boldsymbol{x})$, key $\boldsymbol{k} = \boldsymbol{W}^{K}(\boldsymbol{x})$ and value $\boldsymbol{v} = \boldsymbol{W}^{V}(\boldsymbol{x})$, and transformed into $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v} \in \mathbb{R}^{h \times l \times d_k}$, where the $d = h \times d_k$. To compute the query-key interaction, transposed query $\boldsymbol{q} \in \mathbb{R}^{h \times l \times d_k}$ is multiplied with the transposed key $\boldsymbol{k} \in \mathbb{R}^{h \times d_k \times l}$ to obtain the score matrix $\boldsymbol{W}_{\text{weight}} \in \mathbb{R}^{h \times l \times l}$. Afterwards, the sentence mask is used to set the scores of padded sequence as -10^{9}, and softmax and dropout are also applied sequentially. The attention weight matrix is further attended to the transposed value $\boldsymbol{v} \in \mathbb{R}^{h \times l \times d_k}$ to get the attention output $\boldsymbol{x} \in \mathbb{R}^{h \times l \times d_k}$, which is reshaped back to $\boldsymbol{x} \in \mathbb{R}^{l \times d}$, and passed through another linear layer $\boldsymbol{W}^{O}$ to generate the final output.

2f. Feed-forward

$$\text{FFN}(\boldsymbol{x}) = \text{ReLU}(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2$$

Note: In this section, I adapted the skeleton code from the TRANSFORMER PYTORCH re-implementation, and DEPTHWISE SEPARABLE CONVOLUTION implementation, and I modified some details in each layer. For example, in the Transformer paper, the layer norm is applied to the layer output, instead I applied layer norm to the input. I removed the bias in layers immediate after layer norm, since layer norm already includes bias. I also added a ReLU layer to the depthwise separable convolution output according to the original paper[2]. I assembled layer implementations to produce the encoder block.

3. Attention Flow Layer
For context input $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N \in \mathbb{R}^{2d}$ and query input $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_M \in \mathbb{R}^{2d}$, the similarity matrix $\boldsymbol{S} \in \mathbb{R}^{N \times M}$ is calculated as $\boldsymbol{S}_{ij} = \boldsymbol{w}_{\text{sim}}^T[c_i; q_j; c_i \circ q_j] \in \mathbb{R}$. The Context-to-Query attention $\boldsymbol{a}_i = \sum_{j=1}^{M} \bar{\boldsymbol{S}}_{i,j}\boldsymbol{q}_j \in \mathbb{R}^{2d}$, where $\bar{\boldsymbol{S}}_{i,:} = \text{softmax}(\boldsymbol{S}_{i,:}) \in \mathbb{R}^M$, and the Query-to-Context attention $\boldsymbol{b}_i = \sum_{j=1}^{N} \boldsymbol{S'}_{i,j}\boldsymbol{c}_i \in \mathbb{R}^{2d}$, where $\boldsymbol{S'} = \bar{\boldsymbol{S}}\bar{\bar{\boldsymbol{S}}}^T \in \mathbb{R}^{N \times N}$, and $\bar{\bar{\boldsymbol{S}}}_{:,j} = \text{softmax}(\bar{\boldsymbol{S}}_{:,j}) \in \mathbb{R}^N$. The attention output is expressed as $\boldsymbol{g}_i = [\boldsymbol{c}_i; \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{b}_i] \in \mathbb{R}^{8d}$. For QANet, the same bidirectional attention mechanism is applied, and the only difference is that the input is $\boldsymbol{x} \in \mathbb{R}^d$, and the output is $\boldsymbol{x} \in \mathbb{R}^{4d}$.
For BIDAF only (All codes implemented by myself): There are several variations on the similarity function α and the fusion function β in the bidirectional attention layer. The variations of α and β include:

$$\begin{aligned}
\alpha(\boldsymbol{c}, \boldsymbol{q}) &= \boldsymbol{c}^T\boldsymbol{q} \\
\alpha(\boldsymbol{c}, \boldsymbol{q}) &= \boldsymbol{W}^T[\boldsymbol{c}; \boldsymbol{q}] \\
\alpha(\boldsymbol{c}, \boldsymbol{q}) &= \boldsymbol{c}^T\boldsymbol{W}\boldsymbol{q} \\
\beta(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{c'}) &= \text{ReLU}(\boldsymbol{W}_{\text{mlp}}[\boldsymbol{c}; \boldsymbol{a}; \boldsymbol{c} \circ \boldsymbol{a}; \boldsymbol{c} \circ \boldsymbol{c'}] + \boldsymbol{b}_{\text{mlp}}) \\
\beta(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{c'}) &= \text{ReLU}(\boldsymbol{W}_{\text{lin}}[\boldsymbol{c}; \boldsymbol{a}; \boldsymbol{c} \circ \boldsymbol{a}; \boldsymbol{c} \circ \boldsymbol{c'}] + \boldsymbol{b}_{\text{lin}})
\end{aligned}$$

Where mlp is short for multiple layer perceptron (feed-forward), and was tested in the BiDAF paper. We instead used a linear ReLU layer in the experiment.

4. Modeling Layer
BIDAF: The model uses two bidirectional LSTMs or GRUs, and the forward and backward outputs are concatenated to produce the modeling output $\boldsymbol{x} \in \mathbb{R}^{2d}$.
QANet: The attention output $\boldsymbol{x} \in \mathbb{R}^{4d}$ is linearly projected into $\boldsymbol{x} \in \mathbb{R}^d$ first. The model uses 3 identical modeling blocks, each of which contains 7 encoder blocks, and each encoder block is composed of 2 separable convolution layers, 1 self-attention layer and 1 feed-forward layer. The input vector is passed through the modeling block to produce $\boldsymbol{m}_1$, $\boldsymbol{m}_1$ is passed through the same block to produce $\boldsymbol{m}_2$, and $\boldsymbol{m}_2$ is passed through the same block to produce $\boldsymbol{m}_3$. (I used individual layer implementations to produce the modeling block)

5. Output Layer
BIDAF: It generates the probability distribution of the start and end indices throughout the context. Let $\boldsymbol{G} \in \mathbb{R}^{8d \times N}$ represents the attention output, and $\boldsymbol{M} \in \mathbb{R}^{2d \times N}$ represents the modeling output, $\boldsymbol{p}_{\text{start}} = \text{softmax}(\boldsymbol{W}_{\text{start}}[\boldsymbol{G}; \boldsymbol{M}]) \in \mathbb{R}^N$. $\boldsymbol{M}$ is passed through another two-layer bi-directional LSTMs/GRUs to obtain $\boldsymbol{M'} \in \mathbb{R}^{2d \times N}$, and $\boldsymbol{p}_{\text{end}} = \text{softmax}(\boldsymbol{W}_{\text{end}}[\boldsymbol{G}; \boldsymbol{M'}]) \in \mathbb{R}^N$.
QANet: Similarly, let $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3 \in \mathbb{R}^{d \times N}$ as the modeling outputs, $\boldsymbol{p}_{\text{start}} = \text{softmax}(\boldsymbol{W}_{\text{start}}[\boldsymbol{M}_1; \boldsymbol{M}_2]) \in \mathbb{R}^N$, and $\boldsymbol{p}_{\text{end}} = \text{softmax}(\boldsymbol{W}_{\text{end}}[\boldsymbol{M}_1; \boldsymbol{M}_3]) \in \mathbb{R}^N$.

4 Experiments

4.1 Dataset

Figure 2 shows the context, query and answer length distribution in the Train and Development (Dev.) set. We noticed that in the Train set, there is approximately 67% questions that are answerable, while in the Dev. set, there is only 48% questions that are answerable.

4.2 Evaluation metrics

Exact Match (EM) and F1 score are used to evaluate model accuracy. EM measures whether the answer span matches exactly with the ground truth answer. F1 scores is computed as the harmonic mean of precision and recall, where precision is calculated as the number of correct words divided by the length of the predicted answer, and recall is calculated as the number of correct words divided by the length of the ground truth answer. For evaluation, the predicted answer is measured against 3 human answers for each question, and the highest score among the three is recorded.

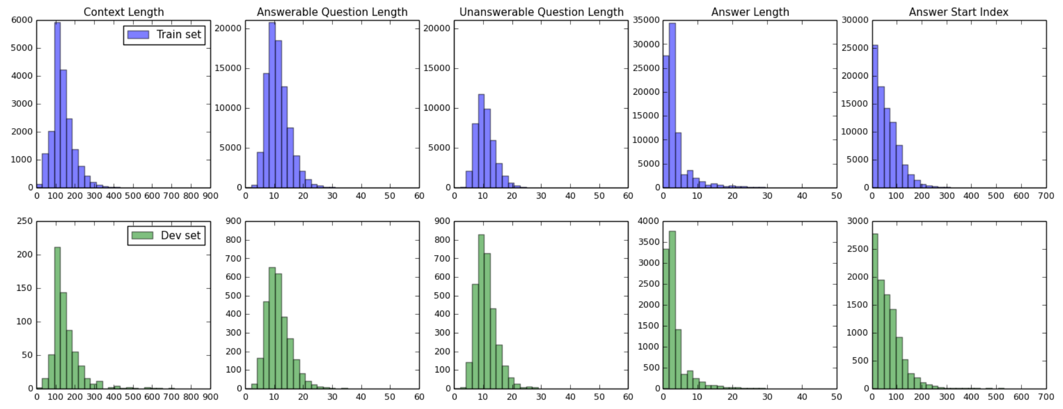

Figure 2: Data Statistics on Train and Dev. Set

4.3 Baseline model improvement and exploration

The following experiments were run on Azure NV6, and metrics were reported on the Dev. set. All the settings were default as in the starter code unless mentioned: optimizer=Adadelta, learning_rate = 0.5, weight_decay = 0, ema_decay = 0.999, batch_size = 64, drop_prob = 0.2, hidden_size = 100, max_context_len = 400, max_question_len = 60, word_vector_size = 300, char_vector_size = 64, word_len = 16, max_answer_len = 30, max_answer_span = 15.
We introduced character convolutional embedding, altered the fusion function β to include a linear ReLU activation, and switched the recurrent layer of bi-LSTM to bi-GRU and increased the hidden_size from 100 to 128. All these modifications improved the BIDAF model performance (Table 1).

Table 1: Performance comparison of different BIDAF models

Model #	**Based on**	**Change Made**	**EM**	**F1**
0	baseline	N.A.	56.49	59.88
1	0	+CharCNN	59.94	63.12
2	1	+Linear ReLU	61.60	65.17
3	2	LSTM–>GRU with increased hidden_size	63.85	67.24

The changes that failed to improve the models include: 1. Variations of the similarity functions α (runs were terminated early before 1M steps); 2. Replacing the recurrent layer in the embedding encoder with self-attention layers, or convolution + self-attention layers (similar as that in the QANet) (highest EM = 60.63, F1 score = 63.91); 3. Adding convolution + self-attention layers in parallel with the recurrent layer in the embedding encoder, and use concatenated output as attention layer input (highest EM = 63.43, F1 score = 66.69); 4. Adding additional self-attention layers in the modeling layer in various ways (highest EM = 60.96, F1 score = 64.36). 5. Pointer rewiring. For example, we appended an 'OOV' (out-of-vocabulary) at the end of each sentence, and tried to maximize $\boldsymbol{p}_{start}(0)$ and $\boldsymbol{p}_{end}$(seq_len). It didn't perform well probably due to the fact that the sequence lengths varied among different contexts. We also appended two 'OOV's to the beginning of each sequence, and tried to maximize $\boldsymbol{p}_{start}(0)$ and $\boldsymbol{p}_{end}(1)$, which didn't improve the model performance either (highest EM = 62.26, F1 score = 65.98).

4.4 QANet re-implementation

All the settings were as previously mentioned except: optimizer=Adam, learning_rate = 0.001 (first 1k steps with log warm up), weight_decay = 3×10^{-7}, ema_decay = 0.9999, drop_prob = 0.1, char_vector_size = 200. Additionally, stochastic layer dropout[6] was adopted: for an encoder block with N layers, each layer was dropped out with probability d/N, where d was the depth of the given layer.
We noticed that the learning curve in the QANet model was not as smooth as that in the BIDAF model, and the BIDAF model converged faster with less overfit measured by dev/NLL. It is probably because

Table 2: Performance comparison of QANet models with different settings

Model #	hidden_size	head_num	batch_size	EM	F1
4	96	3	24	60.78	64.36
5	96	4	20	61.94	65.47
6	128	8	8	59.62	63.47

there are still some unidentified bugs in my re-implementation, and the self-attention mechanisms make the QANet model less robust than recurrent models.

Moreover, the QANet model requires more memory for the multi-head attention, and we initially ran the model on the Azure NV12. However, with hidden_size = 96, head_num = 4, batch_size = 20 * 2, the model over-fitted on the Dev. set very quickly, and we obtained EM and F1 score $\sim$2 points lower than the same model on NV6, which is probably due to the simultaneous update on the model parameters with multiple cores[4]. Therefore, we used NV6 for the all the experiments.

Though convolution and self-attention layers used by QANet model are better suited for GPU parallel computing than recurrent layers (left-to-right or right-to-left) in the BIDAF model, we noticed that the QANet model was much slower than the BIDAF model (**model 5** took twice the time than **model 3** for the same step). We think there are three factors that may account for this phenomenon: 1. There are more layers in the QANet model; 2. The batch_size are smaller due to the memory limit; 3. The contexts and queries are generally short in length (tens or hundreds), while for sequences that are much longer, the benefit of parallel computing may be pronounced.

4.5 Results

Finally, we assembled the BIDAF model (**model 3**) and the QANet mode (**model 5**), and the ensemble model achieved **EM = 66.64** and **F1 score = 69.70** on the Dev. set, and **EM = 65.224** and **F1 score = 68.438** on the Test set (**None-PCE**, one submission made, and achieved first place on the Test Leaderboard at the time of submission on March 17, 2019).

5 Analysis

5.1 Error analysis

Figure 3 shows the error decomposition of BIDAF **model 3**. We noticed that there are $\sim$27% of errors derived from Answer vs. No Answer prediction (AvNA). Interestingly, there is about 2-fold higher error rate of predicting unanswerable questions as answerable than predicting answerable questions as unanswerable. It is probably caused by the discrepancy in the ratio of answerable vs. unanswerable questions in the Train vs. Dev set (Figure 2). In the case of "Error in Match", we found that there are more positional error (prediction is totally off position) than boundary error (prediction shows overlap), which indicates that the pointer mechanism is effective in predicting answer spans, and future work could be done to improve the prediction metrics with advanced pointer mechanisms, such as using attention in the pointer generator[7].

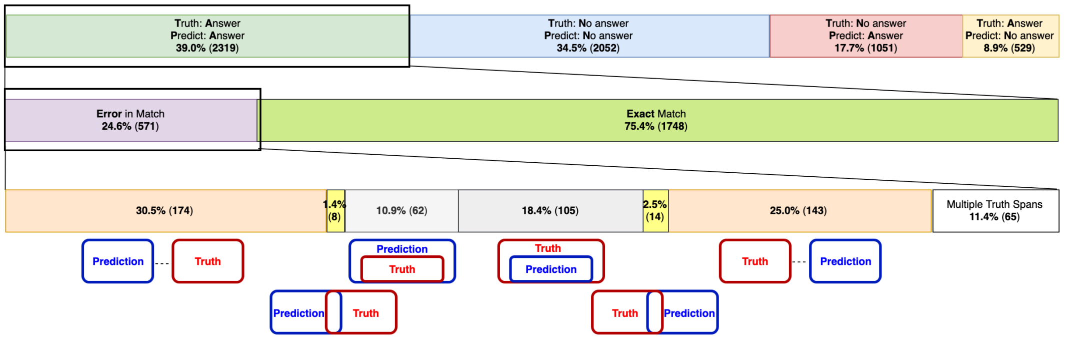

Figure 3: Types of Prediction Errors

Figure 4 shows the context length, question length, and answer length and start index distribution with AvNA error. The blue dashed lines represent the Gaussian simulation on context and question length in AvNA cases, and green dashed lines are over the entire Dev. set. There was no obvious difference identified in both cases, or vs. the whole Dev. set.

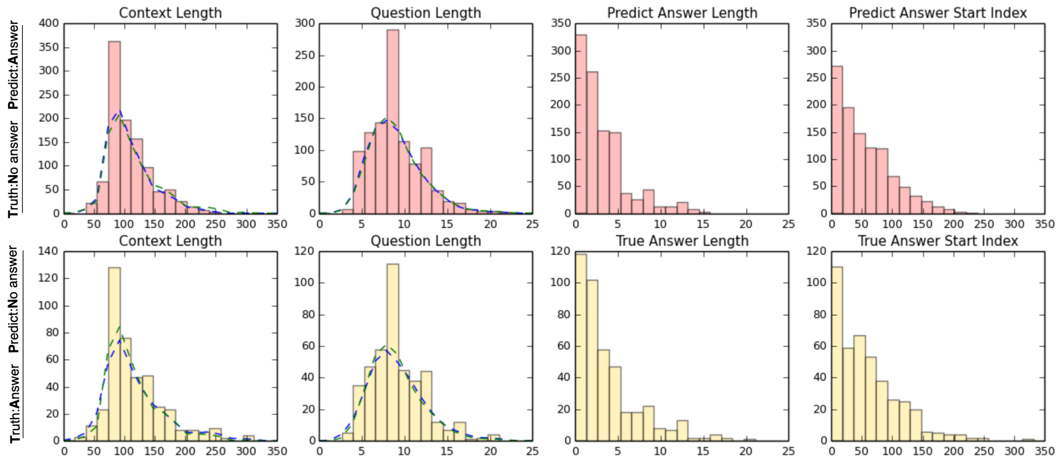

Figure 4: Context, Question, and Answer Distribution with AvNA

We also did error analysis with QANet **model 5** (data not shown), and found very similar error pattern as BIDAF **model 3**, though it showed slightly higher probability of predicting questions as answerable. For example, there were 1191/2388 cases that QANet model predicted unanswerable/answerable questions as answerable (compared to 1051/2319 cases in BIDAF).

5.2 Model comparison

Finally, we compared the performance of BIDAF and QANet models on different types of questions in the Dev. set (Table 3). Overall, the BIDAF model outperformed QANet model in most of the question types. Both models performed the best in answering "When", since the answers may involve some numbers. They performed worst in answering "Other", possibly because those questions are less structural, such as "Purpose of Telnet", or there are some typo in the question, like "WHen did ARPNET and SITA become operational". As expected, neither models performed well in "How" and "Why" questions, given that those question may require some "logic reasoning" other than query-context matching. Interestingly, QANet model shows similar or even better performance in both types of questions, indicating that self-attention may be effective in "logic reasoning".

Moreover, for individual category, we didn't observe significant performance difference when the interrogative words are placed in the start of the sentence (first letter in uppercase) or inside of the sentence (first letter in lowercase) (data not shown).

Table 3: Query Types and Performance Comparison of BIDAF and QANet model

		BIDAF			**QANet**		
Type	**#**	**EM**	**F1**	**AvNA**	**EM**	**F1**	**AvNA**
Overall	5951	67.24	63.85	73.45	65.49	61.96	72.26
What	3625	64.22	67.17	73.35	61.71	65.37	72.47
Who	688	63.52	66.24	71.37	62.50	64.65	69.62
How	569	58.88	64.90	71.35	58.88	63.61	69.60
When	451	72.06	73.55	78.71	72.06	72.78	76.72
Where	253	59.68	64.71	73.91	56.52	61.14	70.75
Which	214	69.16	72.71	78.97	67.29	71.09	78.97
Why	86	54.65	63.76	74.42	59.30	68.19	76.74
Other	65	44.62	54.38	61.54	33.85	41.94	58.46

6 Conclusion

In this paper, we performed two main tasks: Firstly, we experimented with the baseline BIDAF model, and improved its performance by modifying its embedding layer, encoding layer and attention layer. Secondly, we re-implemented the QANet model, which achieved similar though slightly lower performance, compared to the BIDAF model. These experiments demonstrated the power of attention mechanisms – bi-directional attention and self-attention – in deep learning models in the question-answering systems. Future work such as introducing contextual embedding, as well as attention in the output pointer layer, may further promote the model performance.

Acknowledgments

The author would like to thank the CS224N course instructors and course assistants for implementing the baseline model and providing guidance in project development, and Microsoft Azure for GPU service and support.

References

[1] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[2] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[4] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[7] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

[8] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

[9] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[10] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[12] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[13] 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.