Question Answering with Self-Attention and Residuals

Luyao Hou
Stanford University
Stanford, CA 94305
luyaoh@stanford.edu

Abstract

Current end-to-end machine question answering models often use recurrent neural networks (RNNs) to encode sequential information within texts. However, due to their sequential nature, RNNs, especially when layered together, can be difficult to train and scale up. We propose a new question answering architecture that combines RNNs with self-attention and residual connections to speed up learning as well as improve the model performance. On the SQuAD2.0 dataset, our model achieves 60.90 F1 score after 1.5M training steps, which is better than our baseline BiDAF model that achieves a 59.92 F1 score after 3.9M training steps.

1 Introduction

Machine question answering has been an area that attracts a large amount of interests and work. Over the past few years, a number of models have been built that show significant performance improvement on question answering tasks. One such example is the BiDAF model (Seo et al., 2016) that uses both RNNs and context-query attention to achieve 59.92 F1 score on the SQuAD2.0 dataset. However, due to the sequential nature of RNNs, models that rely heavily on RNNs can be difficult to train and scale up especially when multiple RNN layers are used together. In the field of NLP where training examples are scarce compared to other fields, models that converge faster and learn more quickly would make more efficient use of training examples and thus potential for more performance improvements.

Recent development of self-attention and transformers (Vaswani et al., 2017) have suggested models that have both high parellelism and high performance. In this paper, we propose a model that learns more quickly and achieves higher performance by adding recurrency-free encoder blocks on top of RNNs. We then combine the output of RNNs and the encoder blocks with learnable residual connections. The resulting architecture can be found in Figure 1.

The main motivation behind our design is that 1) Given the large number of work that shows the success of RNNs in question answering, we want to use RNNs to encode sequential data. 2) Instead of adding more RNNs to the model to improve performance, we use stacked recurrency-free encoders because its parallelism has better training behaviour than RNNs. 3) We finall build residual connections within our model so that there are shortcuts from input to output and the model can thus learn faster.

2 Related Work

There have been large amounts of work showing great results on NLP tasks with or without RNNs. On one hand, models like BiDAF (Seo et al., 2016) uses RNNs to encode context and query embeddings

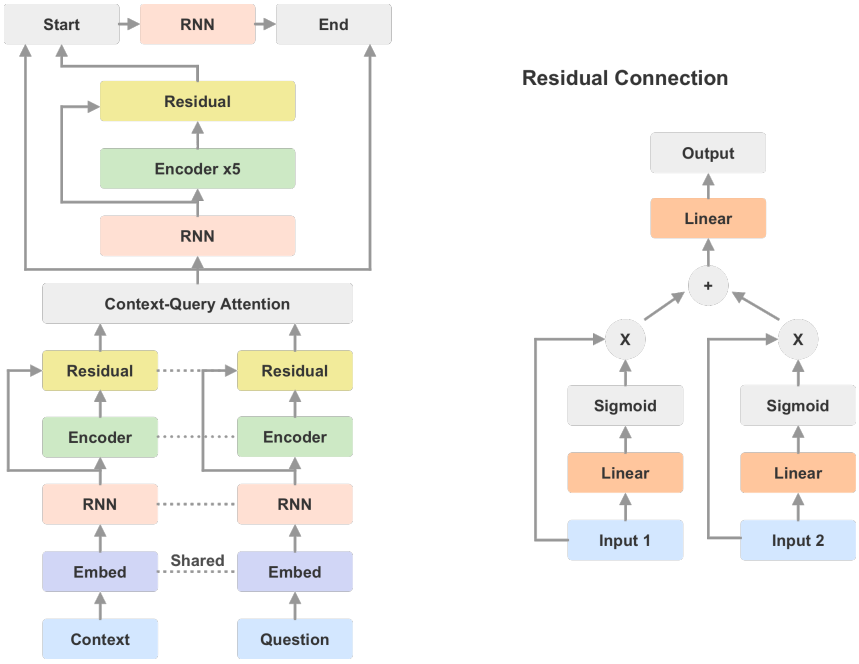

Figure 1: Model architecture

as well as the output of context-query attention. The r-net by Microsoft (2017)[1] shows a similar idea while also adding self-attention on top of RNN layers. On the other hand, the introduction of transformers (Vaswani et al., 2017), which uses multi-head self-attention and position encoding to replace RNNs, shows that recurrency-free models can achieve high performance on tasks like machine translation. The QANet paper (Yu et al., 2018) then builds on top of self-attention and uses both convolution and self-attention to build a recurrency-free question answering model. Our idea is thus to combine both RNN and self-attention and use residual connections to expediate the training of a deep network.

3 Approach

In this section, we first formulate the question answering problem as proposed by the SQuAD2.0 dataset (Rajpurkar et al., 2018). We then discuss our baseline model and the structure of our new model.

3.1 Question Answering and SQuAD2.0

SQuAD2.0 is a reading comprehension dataset consisting of questions proposed by crowdworkers on a set of Wikipedia articles. For each problem, the context is a piece of texts from a Wikipedia article and there is either no answer to the question or the answer must be a span of texts from the context.

Formally, given a context paragraph with n words, $C = \{c_1, c_2, ..., c_n\}$, and a question with m words, $Q = \{q_1, ..., q_m\}$, the output is either no answer or a span $S = \{s_i, s_{i+1}, ..., s_{i+j}\}$ from the context C.

3.2 Baseline

We use the BiDAF model by Seo et al. (2016) as our baseline model.

[1] https://www.microsoft.com/en-us/research/wp- content/uploads/2017/05/r- net.pdf

3.3 The Model

At a high level, our model has in total 5 layers: 1) the context and question embedding layer 2) the embedding encoder layer 3) the context-query attention layer 4) the model encoder layer and 5) the output layer. We use both RNN and self-attention at the embedding encoder and model encoder layers. As proposed by QANet, our self-attention layer is encapsulated inside a encoder block, which also has convolution layers at its input and feed foward layers at its output. We then use learnable residual connections to combine the output of RNN and encoder blocks.

1. Input Embedding Layer The model builds embedding for each word by combining its word level and character level embeddings. The word level embeddings come from the 300 dimensional pretrained GloVe word vector and are fixed during training. Unknown words are mapped to a special <unk> token. Each character is mapped to a 64 dimensional pre-initialized embedding that are trainable. For each word w of lengh l, we obtain its character embedding $e_c \in R^{l \times 64}$, which is then passed into a depthwise separable convolutional layer followed by a maxpool layer to obtain $x_c = MaxPool(Conv(e_c)) \in R^{200}$. We then concatenate the word level embedding x_w and character level embedding x_c to produce $[x_w; x_c] \in R^{500}$, which is then passed into a linear layer followed by two highway network to output $x \in R^h$ where $h = 100$.

2. Embedding Encoder Layer The first part of the this layer is a bidirectional single layer RNN whose output is of size $2h$. The output is then fed into an encoder block that is similar to the one proposed in QANet (YU et al., 2018). In our model, each encoder block has two depthwise separable convolution layers followed by a 2-head self-attention layer and then a feed forward layer. Each convolution layer has kernel size 7. We apply layer normalization before each of these layers. Each operation (layer normalization plus convolution/self-attention/feed-forward) is placed inside a learnable residual connection as shown in Figure 1. Let a be the input to an operation f, then the output $o = res(a, f(a))$ is calculated as follows.

$$
\begin{aligned}
g_a &= \sigma(ff_1(a)) \\
g_o &= \sigma(ff_2(f(a))) \\
o &= ff_3(g_a \odot a + g_o \odot f(a))
\end{aligned}
$$

Here σ is the sigmoid operation and ff_1, ff_2, ff_3 are feed forward layers. Each residual connection inside an encoder block has individual weights which are not shared. We add a sinusoid position encoder at bottom of each encoder block and the entire encoder block is then placed within another residual connection. If we denote the encoder block as enc, the input to this layer as X, the residual connection as res, then the output of this layer E is computed by.

$$
\begin{aligned}
X' &= rnn(X) \\
E &= res(X', enc(X'))
\end{aligned}
$$

3. Context-Query Attention Layer We use the same context-query attention layer proposed by QANet to produce two similarity matrices A and B.

4. Model Encoder Layer The input to this layer at each position has size $8h$ and can be denoted by $[c, a, c \odot a, c \odot b]$ where a, b are corresponding rows of A and B. The input is first fed into a 2-layer bidirectional RNN and then, similar to the embedding encoder layer, the output from RNN is then fed into encoder blocks. In this layer we use 5 encoder blocks stacked together and each encoder block is placed within a learnable residual connection. Each encoder block has the same parameters as the encoders in the embedding encoder layer. The output of this layer at each position has size $2h$.

5. Output Layer We use the same output layer as proposed by the BiDAF model.

Loss Function We prepend each context paragraph with a start-of-paragraph token at index 0 and the model predicts no answer by predicting that both the start and end of the answer span are at index 0. The loss function is defined as the negative log likelihood of predicted distributions indexed by gold start and end indices, averaged over all examples.

$$
L = -\frac{1}{N} \sum_{1}^{N} (-log(p_{start}(i) - log(p_{end}(j))))
$$

where N is batch size and i, j are gold start and end indices respectively.

Inference The model predicts start and end of the span, i and j, by maximizing $p_{start}(i) \times p_{end}(j)$ under the constraint that $i \leq j$ and $j - i + 1 \leq L_{max} = 15$. As discussed above, predicting $i = j = 0$ will generate a no-answer output.

3.4 My Contributions and Code

I built three major models for this project. The first one is a full QANet implementatin based on the paper, but the evaluation results of my QANet are lower than the baseline model. I then built a larger BiDAF model by basically adding more RNN layers and some feed forward layers in between. That model has worse evaluation results than the baseline as well. The last major model I built is the one I discussed in this paper. More evaluation results will be discussed in later sections.

I have also tried a number variants of each of these three major models and the model described here is the one with best results. The only layers I used from starter code are the context-query attention layer and the BiDAF output layer.

4 Experiments

4.1 Data

As discussed, we train the baseline and our model using the SQuAD2.0 dataset. In particular, there are 129,941 examples in the training set, 6078 examples in the dev set and 5921 examples in the test set.

4.2 Evaluation Metrics

We evaluate our model using both exact match (EM) and F1 scores. When a question has no answer, both EM and F1 are 1 if our model predicts no answer and 0 otherwise. When a model has answer, the maximum EM and F1 score based across three human provided answers are used. The final EM and F1 scores are calculated as the average across all examples.

4.3 Training Details

When training the model, we use L2 weight decay with $\lambda = 3 \times 10^{-7}$ and dropout between layers for regularization. The batch size is 32. We use a learing rate warmup schedule with an inverse exponential increase to 0.0006 within the first 1000 batches. The learning rate then stays constant at 0.0006 until an exponential decay to 0.0003 starting from the 8000^{th} batch and ending at the 62500^{th} batch.

We use an Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$. The model reaches maximum dev F1 score and minimum dev loss after around 1.5M batches before it starts overfitting to the training data.

4.4 Results

Table 1 compares the EM and F1 score of our model with that of other models. We also compare the number of samples it takes for each model to reach the reported F1 score. BiDAF-Large is a model we built by adding more RNN layers and feedforward layers in between to the baseline BiDAF. We trained the baseline and BiDAF-Large with the same hyperparameters. In particular, we used Adadelta with learning rate 0.5, rho 0.9 and $\epsilon = 10^{-6}$. Our Model-Small is a variation of our model where we reduced the number of encoder blocks in the model encoder layer from 5 to 3. The results for our model are obtained from **non-PCE leaderboard**.

Figure 2 shows the dev F1 score versus training step plot. BiDAF-Self-Attention is a model we built by adding character embedding and only self-attention layers to the baseline model without residual connections. The result shows that our model converges to a higher F1 score much more quickly

	Dev EM	Dev F1	Test EM	Test F1	Training Steps
BiDAF Baseline	56.43	60.01	56.30	59.92	∼3.9M
BiDAF-Large	54.55	57.94	-	-	∼3.0M
Our Model-Small	58.19	61.35	-	-	∼1.5M
Our Model	**59.22**	**62.54**	**57.36**	**60.90**	∼**1.5M**

Table 1: EM and F1 scores

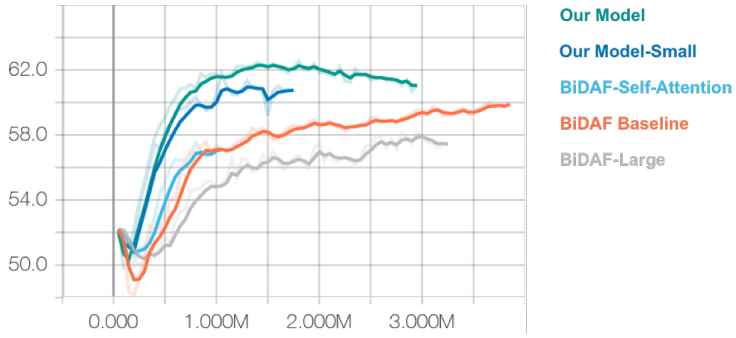

Figure 2: F1 score versus training steps.

than the baseline does. The addition of encoder blocks could contribute to the higher performance as Our Model-Small and BiDAF-Self-Attention both both obtain a lower F1 score. The residual connections also seem to expediate learning process as the BiDAF-Self-Attention model, which does not have residual connections, experiences a similar convergence rate as the baseline model. It is also interesting to observe that the addition of more RNN layers in the BiDAF-L model produces worse results than the baseline, which may suggest that stacked RNN layers can be harder to train and needs finer tuing of hyperparameters.

Even though my model shows improvements over the baseline both in performance and learning speed, the results are actually worse than what I expected. Given the high performance of a single QANet model, I would expect my model to achieve higher EM and F1 scores. I have written different tests and checked my QANet module implementation multiple times, so I think the problem is more likely to be caused by non-optimal hyperparameters. The limitations of the residual connections themselves may also contribute to the result because ultimately the residual connection is only a weighted linear combination of its input elements. I think one problem I had during the project is that I tried to switch and change the models themselves too often. I should have probably spent more time tuining a particular model rather than trying various models but only one or two set of hyperparameters for each one.

5 Analysis

5.1 Context and Question Lengths

In this section we explore the influence of context and question lengths to the correctness of our model and the baseline. In particular, for each length of context and question, we count the number of correctly and incorrectly predicted answers based on exact match from the dev set. We then remove any length that has 20 or less samples.

Figure 3 and Figure 4 show the percent of incorrect predictions, based on exact match, against different context and question lengths of our model and the baseline. It is quite surprising that both models are insensitive to the context and question lengths even though there are slightly more incorrect predictions at both ends of the question length. The use of bidirectional RNNs and context-query attention may help explain the models' abilities to process long inputs. Another possible explanation

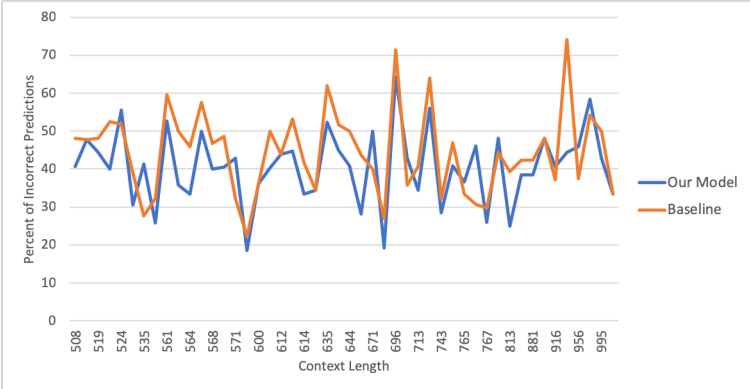

Figure 3: Percent of incorrect predictions vs context length

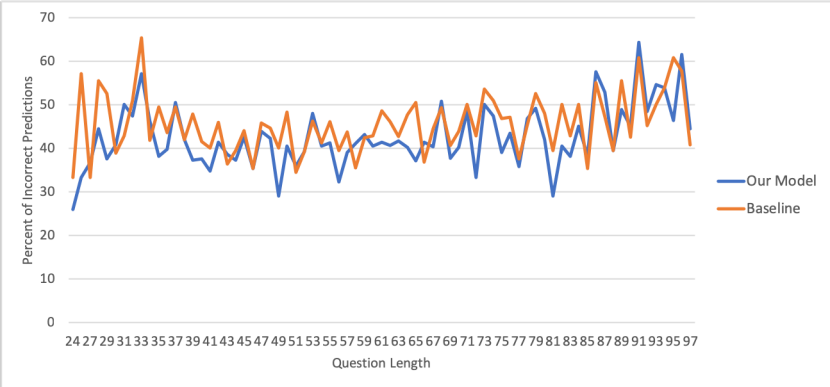

Figure 4: Percent of incorrect predictions vs context length

is that the EM scores of both models are not high to start with and are binary in its nature. Future work could explore how the context and questions lengths influence F1 scores of the predictions.

5.2 End of Span Predictions

In this section we explore the percent of answerable questions that our model and the baseline predict the correct starting position but incorrect ending position. Table 2 summarizes the result.

It can be observed that our model has a slightly lower percent of error probably because the introduction of encoder blocks into the model help enhance the representation of context-query attention encoding before it passes through the RNN in the output layer. However, the numbers are relatively close between our model and the baseline because our model uses the same output layer as the baseline BiDAF model. Future work could explore the effectiveness of different output architectures.

	Percent
BiDAF Baseline	24.40%
Our Model	**23.26%**

Table 2: Percent of answerable questions with correct start and incorrect end position predictions

6 Conclusion

We propose a machine question answering model that combines the benefits of RNN and self-attention using residual connections. The resulting model achieves higher EM and F1 scores than our baseline BiDAF model and also learns much more quickly than the baseline. We also test some variations of our model and the baseline, which suggests that residual connections could help expediate learning and the encoder blocks could bring performance improvements to the model.

We also explore the sentivity of our model to question nd context lengths as well as the percent of answerable questions that our model predicts the correct start position but incorrect end position. Both the baseline and our model are relatively insensitive to context and question lengths probably because of bidirectional RNNs and context-query attentions. The percent of answerable questions that the models predict correct start but incorrect end positions suggests a potential to work on the output layer of the model.

During this project, we have tried variations of different models. However, looking back, we think we should have explored more hyperparameter tuning for each model rather than changing the model constantly. Given the high performance of single QANet models, our model actually does not achieve our expectation in terms of the metrics. The early overfitting of our model suggests that further hyperparameter tuning might produce better results. Future work could explore the weights each residual connection assigns to its inputs and the types of features that RNNs and encoder blocks focus on in similar models.

References

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

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

[3] 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.

[4] 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.