# All SQuAD Needs is Self-Attention: Char-QANet

Stanford CS224N Default Project

**Luis Alcaraz**
Department of Computer Science
Stanford University
alcaraz@stanford.edu

**Troy Lawrence**
Department of Electrical Engineering
Stanford University
troylaw@stanford.edu

## Abstract

Question answering (QA) is a key problem within natural language processing (NLP), and initial solutions to this problem, such as using LSTMs, perform decently well but still not nearly at the level of human benchmarks. (Rajpurkar & Jia et al. '18)[1] Therefore, state of the art models have investigated the use of self-attention, a key feature of the Transformer model. In this project, we will use self-attention building off QANet and apply it to the SQuAD 2.0 dataset. (Wei Yu1 & Dohan et al. '18)[2] In the process of building a robust model to improve F1 and EM scores, we have explored the addition of custom character embeddings as well as introducing coattention for the encoder layer on top of a Bidirectional Attention Flow (BiDAF) architecture. Each customization of the baseline BiDAF system provided us provided certain improvements, but in the end the introduction of QANet [2] with its version of character embeddings performed the best in terms of F1 and EM scores.

## 1 Introduction

Much of natural language processing can be understood in light of a question-answering paradigm: a query is presented and an answer has to be found using some document or context. Devices such as Alexa and programs such as Google's page ranking exist precisely as a result of humankind's necessity for Question-Answering models in our lives. Fundamentally this process is how we go from states of ignorance to knowledge, and as such, capturing this method in a computational system that can process information faster than our minds can is one of the greatest achievements of our time.

The introduction of the Stanford Question Answering Dataset (SQuAD) in 2016 with its 100,000+ human-labelled question, context, and answer associations [3] provided the groundwork for much subsequent experimentation and successes in this field. In this paper we use the second version of SQuAD which introduced 50,000 non-answerable questions among its corpus.

In the original SQuAD paper, the authors' best models produced an F1 score of 51%. However, this achievement pales when compared to the average humans ability to achieve an 86.8% F1 score in a Question Answering context. Our research is dedicated to building and improving upon different models over the SQuAD 2.0 to produce higher F1 scores in the hopes of approaching and potentially one day surpassing human performance. In this process we have taken and tweaked the working Bidirectional Attention Flow (BiDAF) model provided to us. We first added character embeddings to the already computed word embeddings. We then explored the use and interchangeability of a coattention encoder. Lastly, we overhauled the BiDAF architecture by implementing a QANet as described in (Yu et al., 2018) and experimenting with its performance with the introduction of our previous character embeddings and coattention layer.

## 2 Related Work

A key concept used in question answering is the idea of a sliding window. The way this works is that given a certain question and a document or context within which the answer to the question can be found, the answer can be found within a set of contiguous words or sentences. The best way to find the beginning and ending indices for this window has been the work of many researchers for well over a decade.

In more recent times, Richardson et al. provided a new multiple choice dataset MCTest [4] which Wang et al. used in 2015 to propose a statistical model with syntax and semantics to bound this window to achieve better results in machine comprehension [5]. Slightly before them, however, Kim et al. showed the advantages of using convolutional neural networks (CNN) for sentence classification[6]. Soon neural networks dominated the area.

Our research picks up from the use of recurrent neural networks (RNN) in the Bidirectional Attention Flow model proposed in (Seo et al., 2016) [7]. RNNs are able to incorporate temporal information in a way CNNs cannot and thus work well with machine translation and prediction. The model from this paper was provided to us as a baseline to work from and concepts from this paper (such as character embeddings) were further used in our project to improve our performance. Our implementation of coattention used the discussion found in Xiong et al.'s paper on the topic [8]. All these methods have succeeded on obtaining F1 scores well beyond Rajpurkar et al.'s scores from the introduction of the SQuAD. However, a new approach to the topic arose when transformers entered the discussion with the paper "Attention is all you need" [9]. The transformer allowed for levels of parallelism not afforded by the RNNs through its introduction of self-attention, where each embedding can compute its attention independent of the computationally temporal position of its neighbors. Wei Yu et al. used this framework as the inspiration for their QANet[2], which inspiration we also use significantly in our research.

## 3 Approach

### 3.1 Baseline

Our baseline model is based on a Bidirectional Attention Flow (BiDAF) model as discussed in [7]. However, unlike in the paper our baseline only utilizes word-level embeddings, rather than a concatenation of both word and character level embeddings. This model uses encoder layers that consist of bidirectional LSTMs to create temporal dependencies between timestamps of the embedding layer's output, an attention layer, modeling layer that refines the sequence vectors after attention through two bidirectional LSTMs, and an output layer that produces a vector of probabilities for postions: $\mathbf{p}_{start}$, $\mathbf{p}_{end}$.

### 3.2 Character Embeddings

The first addition to the baseline model was introducing character-level embeddings. Its form was based off the Seo et al. paper [7] and input from students and staff on the Ed platform. After dropout, we used a 1-D convolutional neural net to be more faithful to the paper, and this meant squeezing the batch_size and seq_len dimensions together. For the convolution we used a kernel size of five and had the input channel take in the embedding size and channel out the hidden size. We maxpooled the resulting vector over the entire width and, after a reshape, concatenated it to the word embeddings. The only departure from the paper was using a linear projection on the concatenated vector of word and character embeddings to keep the hidden_size to 100 to avoid complications down the line. We finished by submitting this embedding to a Highway Encoder.

### 3.3 Coattention Encoder

Coattention is a form of attention that attends to the both the question and document/context simultaneously [8]. As stated in (Lu et al., 2016)[10], coattention improves performance by not just looking at "where to look" (visual attention) but also at "what words to listen to" (question attention).

An affinity matrix (L) is first computed by multiplying the question (Q) and context (C) matrices output by the previous contextual embed layer in our BiDAF architecture. Next with linearization, attention weights for both the document and question are found from L. Through further multiplication and concatenation and as described below, from these weights we produce a matrix U.

$$\text{L} = \text{C}^T Q \in \mathbf{R}^{(m+1)\mathbf{x}(n+1)}$$
$$\text{A}^Q = softmax(L) \in \mathbf{R}^{(m+1)\mathbf{x}(n+1)}; \quad A^D = softmax(L) \in \mathbf{R}^{(n+1)\mathbf{x}(m+1)}$$
$$\text{B}^Q = CA^Q \in \mathbf{R}^{l\mathbf{x}(n+1)}; \quad B^C = [Q; B^Q]A^C \in \mathbf{R}^{2l\mathbf{x}(m+1)}$$
$$\text{U} = [\text{C};\text{B}^C] \in \mathbf{R}^{2l\mathbf{x}m}$$

Matrix U is later fed into the bidirectional LSTMs in the decoder module.

### 3.4 QANet Overview

IOur current approach builds off of the provided baseline by infusing it with the architecture presented in the paper QANet [2], which architecture utilizes the key features presented in Attention Is All You Need (e.g., self-attention) [9]. These features allow the algorithm to learn deeper connections between words that are far apart from each other, which is a reason why algorithms that use self-attention outperform RNN/LSTM models. We got inspiration from the following Pytorch implementation: PyTorch QANet Implementation. However, we realized the limitations of QANet, specifically memory, when implementing the model, and had to work around such limitations (those limitations and work-arounds will thoroughly be discussed later).

### 3.5 QANet Character Embeddings

QANet also makes use of character embeddings. In our first approach to character embedding with QANet, we followed the layout expressed in the paper introducing QANet (Yu et al., 2018) [2] This approach diverged from the one proposed above [8] by initializing the character weights instead of drawing them from a provided pretrained file and assigning the same size to both the in_channels and out_channels as a result of dimension necessity.

### 3.6 Encoder Blocks

Our implementation consists of two Encoder blocks that fuse into model encoder blocks as presented in QANet. This is a slight diversion from the Transformer model, given no decoder is used (although similar to the bidirectional LSMT model from the baseline). The reasoning behind this is motivated by question and answering, where putting either the context or the question through the decoder wouldn't allow the algorithm to understand what its looking for. A single encoder block consists of an embedded input that is first passed through a series of convolutional layers. Following the QANet architecture where the convolutional layer is repeatable, we ran the embeddings through 4 layers of convolutions for the embedding encoder blocks and two layers for the modal encoding layers. At each convolutional layer, we had a kernel of 7, and a hidden size of 128, per the QANet paper. The reasoning behind CNNs is their ability to capture local structure of text. The embedded input is then passed through a self-attention layer, where the model is trained to learn global structures of the text. It does this by using the multi-head attention, where a single head exhibits:

$$\text{MultiHead(Q,K,V)} = \text{Concat}(\text{head}_1, ..., head_h)W^O$$
$$\text{where head}_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

(Vaswani et al. 2017) [9] In our model we employ 8 heads, with the block finally ending in a feed-forward layer.

### 3.7 Context-Query Attention

In this block we utilized the attention mechanism presented in the baseline - BIDAF Attention. It utilizes the context ($C$) and query ($Q$) encodings. [2] This attention is calculated by computing the similarities between each pair of context and query words. This generates a similarity matrix, which is then transformed through row-wise softmax operations, rendering matrix $S$. Context-to-query is

then computed:

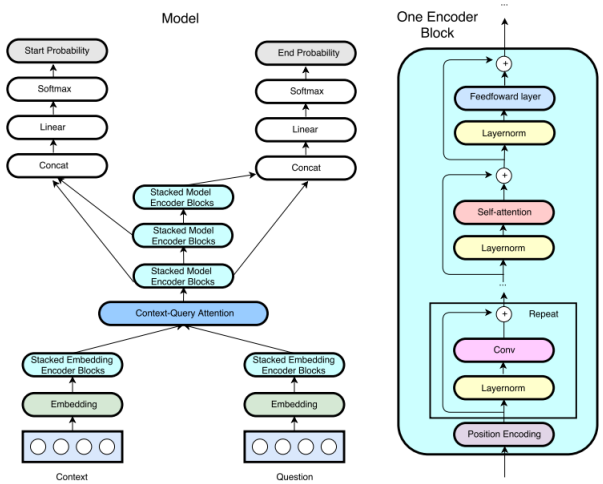
$$a_i = \sum_{i,:} S_{i,j} q_j$$

### 3.8 Output Layer

This layer, being task-specific, uses the model encoding block's output to predict the probability of each position of being the start or end of the answer span. These probabilities can be expressed as follows:

$$p^1 = softmax(W_1[M_0; M_1]), p_2 = softmax(W_2[M_0; M_2])$$

$W_1$ and $W_2$ are trained variable and $M_0$, $M_1$, $M_2$ are model encoder outputs.




## 4 Experiments

### 4.1 Data

The data we are currently using is from the provided, Stanford Question Answering Dataset. Specifically, this is the second version of SQuAD, where it now extends the original dataset with unanswerable questions. It has been pre-split into three sets: train (129941 examples), dev (6078 examples), and test (5915 examples).

### 4.2 Evaluation method

Our evaluation method consists of F1, exact match (EM), and Answer vs. No Answer (AvNA). F1 is the harmonic mean between recall and precision, calculated with the formula 2×prediction×recall/(precision+recall). EM is a binary metric that evaluates whether the model predicted correctly, answer or no answer (as opposed to AvNA). However, we directly optimize over the loss (NLL) as presented in the baseline model.

### 4.3 Experimental details

Our first adaptation to the baseline BiDAF model given to us was the implementation of character embeddings. Two main choices had to be made when implementing these embeddings. The first was the size of the convolutional neural network to use. Using a 2-D cnn would allow the program to convolve not only over each word but over each sentence, which would allow for more learning. The other benefit of this approach was not having to rearrange the 4-dimensional character weight embedding grabbed from the pretrained file. The F1 and EM scores that resulted from this implementation hovered around the mid 50s range. Looking into using the 1-D cnn approach,

we received better results with F1 at 61.5 and EM at 58. This followed the original paper more closely [7], and preserved the seq_len nicely, which nullified the need for detailed reshaping after convolution and maxpooling. The second question was whether we used this character embedding model or the one presented in the QANet paper [2]. The differences (as explained in the Approach section) were not much, but using the embeddings from the pretrained provided file proved more accurate, and we went with this approach.

The next implementation was the substitution of the BiDAF attention layer for coattention encoding. Coattention slightly edged out the BiDAF attention layer in EM. The computation of this attention was carried out as in [8] without conceptual modifications. When we implemented QANet, however, we learned quickly that trying to incorporate coattention into the model did not cause better performance. The self-attention layer of the QANet served its purpose better within that framework.

For QANet, we experimented with different hyperparameters, although trying to stick closely to the papers purposed hyperparameters. What first became an issue was the batch size. Although QANet was originally run with a batch size of 64, the computational limitations of using 1 GPU (in comparison to the 4 that were used in QANet) immediately showed. Therefore, we reduced our batch size to 16, given we only had access to 1 GPU. Furthermore, at first we utilized the optimizer and scheduler used by the baseline BIDAF model, Adadelta. However, with this optimizer and scheduler, loss would exponentially increase. Therefore, we used the proposed optimizer and scheduler mechanism presented in the QANet paper: Adam with learning rate set to 0.001, Gammas of $\gamma_1 = .8$ and $\gamma_2 = .999$. Furthermore, we set hidden size to 128 with a dropout rate of .01.

### 4.4 Results

Below we have a graph that visualizes our experiments along with a graph that presents the dev scores. Our test set scores came out to an EM score of 59.425 and an F1 score of 62.785.

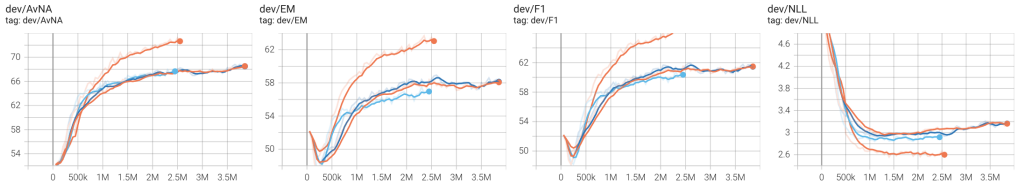



Figure 2: Dev Set Results: QANet (Orange) Coattention (Dark Blue) Character Level Baseline (Red) Baseline (Light Blue)

| Dev Results | | | |
|---|---|---|---|
| | Dev AvNA | Dev EM | Dev F1 |
| Baseline | 68.5 | 57.6 | 60.9 |
| Baseline+Char-Level | 68.3 | 58.0 | 61.5 |
| Coattention | 68.8 | 58.1 | 61.4 |
| QANet | **72.6** | **63.0** | **66.4** |

The biggest learning from the above results is the vast improvement QANet brings in comparison to the baseline. Even within the first million iterations, the rate of change from QANet is significantly better than that of the other three models which remain relatively the same. After a couple million iterations, QANet does not seem to plateau, but the other three have reached similar peaks. We do see improvement of the character embeddings and coattention as compared to the baseline (which in the graph does not show the iterations past 2.5M). As these were produced by relatively smaller changes to the codebase, the improvements were also small as expected, but such changes are not insignificant. To avoid clutter, we did not add the graph where coattention was used with the QANet, nor did we include the graph of the QANet using the non-pretrained character embeddings (described in section 3.5) as their results were less accurate.

## 5 Analysis

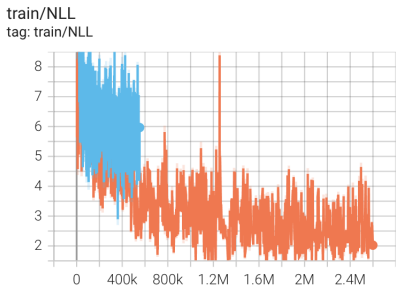
In terms of character embedding, the improvement of the model was expected to be not much more than slight since adding more information to the embeddings in the form of character contexts

should reasonably produce greater learning. Similarly, the difference in results between the character embeddings whose original weights were pulled from the provided file and those of the weights initialized on their own was expected.

We were surprised to find that the addition of coattention did not produce any noteworthy improvements over the baseline with the character embeddings. Upon closer inspection, it is apparent that the BiDAF attention provided to us is itself more similar to coattention than not, both calculating a similarity or affinity matrix and both utilizing softmax and matrix multiplication at similar moments. Although the coattention method may appear to be more sophisticated, within the provided BiDAF framework, it made sense that it produced results on a similar par to the original BiDAFAttention layer.

Similarly to how coattention did not improve upon the BiDAF model based on its fit within the overall architecture, we reason that the QANet's use of self-attention and context-query attention more seamlessly fits within the model than coattention. We experimented using the coattention layer in place of each of these two other attention layers. Since the context-query layer we used was the provided BiDAF attention layer, we did not predict much improvement here, given our previous results. And given the very purpose of self-attention and its use within the stacked nature of the encoding blocks for QANet, we also did not expect coattention to perform well in its place either. We left coattention to the side as a helpful project in better understanding the QANet structure and the benefits of self-attention and multi-head attention.

However, the QANet's implementation was very particular, where the hyperparameters presented in the paper had been optimized. For example with experimented with using Adadelta and the scheduler from the baseline. However, training appeared to stagnate early on (as seen below).




Above, we can see how the proper implementation of QANet (orange) learns while with an Adadellta optimizer, the first 4 epochs are stagnant (light blue). We believe this is due to Adam's bias correction towards moments, leading to a faster convergence. Furthermore, it is important to note the limitations of QANet, specifically its memory and computational complexity. In order to achieve results remotely close to the paper, one would require multiple GPU's to be able to run a full, 64 batch size model. Due to this limitation, we reduced our batch size to 16 and number of epochs to 20. This was in an effort to not burn through credits, or possible get our virtual machine deallocated. However, as one can see in the results section, QANet had not plateaued, and learning could have continued (which would have improved our test set results). Lastly, we found it very interesting too see such volatile NLL was in comparison to the baseline, indicating the CNN layers and Self-attention layers made the model a bit unpredictable (which is why hyperparameter optimization is key).

## 6 Conclusion

In the end what SQuAD needed was self-attention, a self-attention within a QANet framework. Other models may surpass our QANet implementation or improve upon its, but the main learning from our results was that the QANet improves significantly upon a Bidirection Attention Flow model. Our addition of character embeddings and coattention also caused our model to learn better when added in their appropriate places at the appropriate times, but they merely nudged our scores in the right direction. We were limited in our work by computational resources as iterations of models took

upwards of ten hours to run through 25-30 epochs. The time constraint was compounded by the limitations placed on us by Azure, as our accounts were both disabled at various times on account of insufficient credits. Through it all, we have seen how critical Question Answering is within an NLP frame of reference, how others before have gone about tackling this problem, and how our implementation of QANet has significantly improved upon the baseline BiDAF provided us.

## References


[1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[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. *arXiv preprint arXiv:1804.09541*, 2018.

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

[4] Matthew Richardson, Christopher Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

[5] Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Machine comprehension with syntax, frames, and semantics. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2.

[6] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.588*.

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

[8] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2017.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[10] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *arXiv preprint arXiv:1606.00061*, 2016.