Applying QANet to SQuAD 2.0

Aleksander Dash, Andrew Zhang, Nolan Handali
Department of Computer Science
Stanford University
(adash, azhang97, nolanh)@stanford.edu

Abstract

In this paper, we build a question answering system for the Stanford Question Answering Dataset (SQuAD) 2.0. Our model is heavily based on the QANet model proposed by Yu et al[5] and adapted to work on the version 2.0 dataset. On the dev dataset, our model achieved an F1 score of 68.98 and an EM score of 65.79. On the test set, an ensemble of models achieved an F1 score of 67.44 and an EM score of 63.96.

1 Introduction

Machine question answering is a field that has gained significant popularity among researchers over the past few years, and a key dataset at the heart of the increase in popularity is the Stanford Question Answering Dataset (SQuAD)[2], which consists of over 100,000 questions crowdsourced from Wikipedia. State of the art machines managed to reach, and even surpass human level performance on this dataset about a year ago, and as a response, SQuAD 2.0 [1] was created, which combined the original questions with 50,000 more questions which were written adversarially by workers to be unanswerable, yet look similar to the original questions.

SQuAD consists of (context, question, answer) tuples, where the context is a passage from Wikipedia, and the the answer for a given question and context is either a span of the context, or 'no answer'. A span is simply an excerpt of text from the context, meaning that the answer can also be represented as two indices from the context.

Currently, cutting edge models use pretrained contextual embeddings (PCE) to achieve high performance, but for the purposes of this project, we focused on the non-PCE divison, and chose not to use BERT or ELMO.

In this paper, we describe an end-to-end deep learning model based on the work done by Yu et al [5] that aims to remove the need for recurrent networks, and use convolution and self-attention as the framework for encoding query and context. The relation between context and question is also learned by attention, similar to the techniques described by Seo et al in their work with bidirectional attention flow [3].

2 Related work

The baseline model that we are comparing our work to is the Bidectional Attention Flow model from the work done by Seo et al[3]. In this paper, the authors used both GloVe word embeddings and character embeddings that they passed through a layer of LSTM cells individually before merging them in a bidirectional attention flow layer (hence the name of the model), before passing the attention outputs through two more layers of LSTM cells and finally passing these outputs through a dense softmax layer and a LSTM softmax layer to get the start and end of the answer spans in the context for a particular question. Note this paper targets SQuAD 1, so there is no focus in this paper on determining if a question has no answer. This model performed particularly well for its time, as an

ensemble model based around BiDAF was able to outperform all previous approaches at the time the paper was published (back in 2016). The only major disadvantage of this model was that it was heavily reliant on RNNs, and therefore difficult to parallelize. This in turn meant that the model was slow to train and did not take advantage of the massive parallelism commonly available at the time.

One-and-a-half years later, a paper titled "QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension" was published, which also targeted the SQuAD v1 dataset. This model forms the basis for our work in this paper. The model aimed to achieve state-of-the-art performance on SQuAD v1 while simultaneously avoiding the achilles heel of the BiDAF model: namely, the structure of the model allows it to be massively trained in parallel. The model uses only convolutions and self-attention, which can all be computed in parallel for a single layer, and this led to better performance in part because the authors were able to train the model on more data than previous models, since their model was able to process training examples quicker. The authors estimated in their paper that QANet was 4.3 times faster than BiDAF to train, and 7.0 times faster to evaluate, which meant they had a massive advantage as they could use data augmentation techniques to increase the number of training examples available, and thereby process more training data than the BiDAF model. QANet achieved state-of-the-art performance when it was released, significantly outperforming the previous best model on the official leaderboard.

It is perhaps no surprise given the time of publication that the authors of the QANet paper decided one of the two major pillars of their model would be attention. Less than one year earlier, the hugely influential paper "Attention Is All You Need" was published, which showed the power of basing a model just around attention mechanisms [4]. That paper, of the time of writing, has over 1300 citations, and it is likely this paper was a major reason why the QANet authors decided to rely heavily on attention in their model. The QANet authors did not compare variations of their model with different types of attention (e.g. single-head vs. multi-head vs. convolutional multi-head attention), which is something we wanted to test while tuning their model to SQuAD 2.0.

3 Approach

The first thing we did was improve on the BiDAF model that was already implemented in the starter code for the default final project by adding character embeddings to it. We modified the embedding layer such that the characters were projected with the same dimensionality as the word embeddings were projected with, which is currently 50.

The final model that we implemented was based off of the model detailed in the QANet paper by Yu et al[5], which consists of five layers as shown in the figure below.

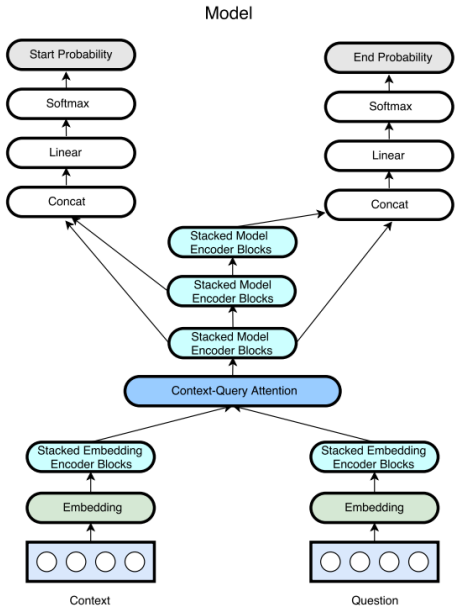

Figure 1: The overall QANet model, from the paper by Yu et al [5].

1. The **input embedding layer** consists of the GloVe embeddings of each word concatenated with the output of a convolutional layer that processes the embeddings of each character that makes up a word up to a maximum length of 16 characters per word. Then, we passed this concatenation through a convolutional projection to our model's hidden size, which we set to 96, and then finally passed through a highway network.
2. The input is then passed through the **embedding encoder layer**, where we first add positional encoding to the input since otherwise the convolutional structure of the network would lose track of that information. The rest of this block consists of 4 convolutional layers with depthwise separable convolutions (which they argue is more memory-efficient and generalizes better), 96 filters each and kernel size 7, followed by a self-attention layer with 8 heads, and finally a feed-forward layer. The self attention layer is adopted from Vaswani et al[4]. This structure is the encoding block shown in figure one.
3. A **context-query attention layer** then computes the attention score between each word in the question/query and each word in the context/paragraph as follows: first, a similarity matrix S is computed for every query-context pair and then normalised by applying the softmax function over each row to get $\overline{S}$ and over each column to get $\overline{\overline{S}}$. The similarity function for a query word q and a context word c is the trilinear function $f(q, c) = W_0[q, c, q \odot c]$ where W_0 is a trainable parameter. Finally, if C is the matrix of the encoded context, and Q is the matrix of the encoded query, the context-to-query attention A is computed as $A = \overline{S} \cdot Q^\top$, and the query-to-context attention B is computed as $B = \overline{S} \cdot \overline{\overline{S}}^\top \cdot C^\top$.
4. The context C and the attention matrices A and B are then fed through the **model encoder layer**. This is similar in structure to the embedding encoder layer: the model encoder layer consists of seven encoder blocks chained together, but each block has 2 convolutional layers instead of 4. This step is repeated three times to get three output matrices M_0, M_1, M_2, and the weights are shared between each repetition.
5. Finally, the matrices M_0, M_1, M_2 are fed through the **output layer**. This output layer consists of two parts, which predict the probability that each context position is the start or the end of an answer span respectively. They are both modeled as softmax functions with separate trainable weight matrices, and the softmax prediction for the start of an answer span uses matrices M_0 and M_1 while the softmax prediction for the end of an answer span uses matrices M_0 and M_2. Finally, the loss function is defined simply as the average cross-entropy between the predicted start/end of the answer span and the true start/end of the answer span for each training example in the dataset.

We were able to carry over the word/character embedding layer and the context-query attention layer from our modifications to the BiDAF model in the starter code, but we implemented the QANet encoding block layer as well as the QANet output layer ourselves. Both of these layers follow the same general structure as in the previous model description, but we were forced to change some values, namely the hidden dimensionality of the model and the number of heads in the self-attention layer from the values recommended in the QANet paper, in order to satisfy memory constraints. For the encoding block itself, we wrote each of the sub-components except for the feed-forward network ourselves, namely the positional encoder, the repeated depth-wise separable convolutions, and the multi-head self-attention network. We opted to keep the same solution as the starter code did for the BiDAF model for how to predict "no answer": We simply have a token prepended to the context paragraphs where if that token's prediction for the start and end of the span is higher than the prediction for any other token, we choose that option.

The major benefit of this model is that it avoids the use of any recurrent networks, theoretically allowing for faster training, as we no longer need to process the input sequentially. Additionally, with the heavy use of convolution and attention, the model should learn how to process local structures due to the use of convolutions, and global interactions from the use of self-attention.

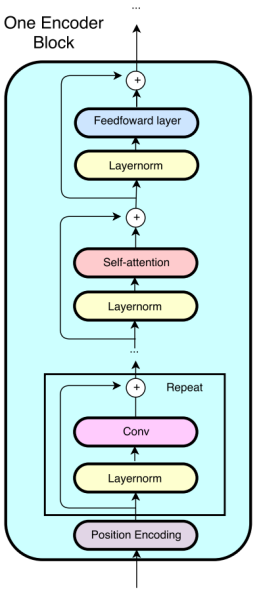

Figure 2: A visualisation of a QANet encoder block, from the paper by Yu et al [5].

4 Experiments

4.1 Dataset

The dataset that we are using is a slightly modified version of the SQuAD 2.0 dataset[1], where our training data is identical, but our dev and test sets are drawn from the official dev data. The data consists of (context, question, answer) triples, where each context is drawn from Wikipedia articles, and the answer is either "No answer", or a span from the context.

4.2 Evaluation method

For our evaluation, we use the same metrics as the official SQuAD leaderboard, which are Exact Match (EM) and F1 scores.

4.3 Implementation details

After adding the character embeddings to the baseline BiDAF model, we trained the model for 30 epochs with a learning rate of 0.001, and a dropout rate of 0.2. We used a batch size of 64. On Azure NC12, this took around 7 hours to train.

For our QANet model, we reduced the batch size to 32, as the model ran out of memory for larger batch sizes. Additionally, rather than using the Adagrad optimizer from the baseline BiDAF model, we switched to using an Adam optimizer, as that is what the original paper used. The learning rate that we used was a warm up scheme that had an inverse exponential increase from 0.0 to 0.001 in the first 1000 steps, and a constant rate after that. Additionally, we have a exponential moving average on all trainable variables with a decay rate of .9999. For regularization, we used L2 weight decay on the trainable variables with $\lambda = 3 * 10^{-7}$. Additionally, we had dropout layers on word and character embeddings with probability 0.1 for word embeddings, and 0.05 for character embeddings. We had dropout layers between every layer in the encoding block with probability 0.05.

We had issues with running out of memory on the GPU, so to combat this, we experimented with varying the number of parameters in the encoder block. For example, the first model replaced the multi-headed self-attention layer with a single head, as we wanted a base from which to compare experiments to and improve on. This is listed in the table below as QANet. Training time for this model took around 9 hours with a batch size of 32 on an Azure NC12 machine. Despite using half the batch size, QANet with character embeddings was able to train in almost the same time as the BiDAF model, showcasing a strength of this model, as its reliance on convolutions over RNNs enable it to be trained in parallel.

We also attempted to vary the type of attention computation in the middle of each encoder block. Our first attempt only used single-head self-attention as defined in [4], since we were having problems fitting our model in memory on the GPUs we trained on. After improving the efficiency of our model, we were able to implement and train the model using both multi-head self-attention and convolutional multi-head self-attention, though not without minor compromises. When we implemented multi-head self-attention with 2 and 3 heads, we had to reduce the number of encoder blocks in the model encoder layer from 7 to 6 and 4, respectively. We also experimented with convolutional multi-head self-attention with 3 heads and 5 blocks per layer, where the convolution would be over the words in each input query with kernel size 1, 3, and 5 respectively for each of the three heads. In the end, we found that vanilla multi-head self-attention with 2 heads and 6 blocks per layer in the model encoding stage was the best compromise, and resulted in the best performance of any of our single models (no ensembles).

In an effort to improve results, we built an ensemble model from a combination of the different experiments we tried. We combined the BiDAF + char embedding model, the base QANet model, and the 2-head and 3-head QANet models for our ensemble model. We choose these because we wanted varying types of models, rather than several of the same model with different random initialization, mostly due to a time constraints, as after experimenting with different models to see which gave the highest individual score, we had used up most of our time allotted for this project. In order to determine the best ensemble model, we tried both majority vote and a simple average, but found that a weighted average gave the best results. The weights that we found gave the best results were 0.16 for BiDAF, 0.2 for QANet, 0.384 for 2-head QANet, and 0.256 for 3-head QANet.

4.4 Results

Model	Dev set		Test set	
	EM	F1	EM	F1
Baseline BiDAF	57.05	60.62	56.298	59.920
BiDAF with Char Embeddings	61.39	64.84	–	–
QANet	62.914	66.467	–	–
QANet convolutional attention	62.527	66.227	–	–
QANet multi-head attention (2 heads)	64.24	67.67	–	–
QANet multi-head attention (3 heads)	63.74	67.04	–	–
BiDAF + QANet ensemble	**66.258**	**69.432**	**63.956**	**67.442**

Table 1: Comparison of model performances on dev and test set

Table one summarizes the the results we achieved from testing our various models. As of the time of writing, our ensemble scores place us at 5th place on the non-PCE leaderboard on Gradescope, where the top scorer has an EM of 65.224 and an F1 of 68.438. These results are what we expected, as they are around or slightly below the best non-PCE models on the official SQuAD2.0 leaderboard, which indicates our model performs on par with other approaches that do not use PCE.

5 Analysis

We now examine two examples of categories of incorrect answers given my our model, and theorize as to the causes of these problems.

5.1 Incorrect verb/direct object association

In the example shown below, we see that the model correctly identified that King Louis identified an edict, but incorrectly associated him with issuing the Edict of Nantes. This might be due to our model encoding layer not capturing enough of the global context, but rather relying too heavily on the local context, as the Edict of Nantes is closer in the sentence than the correct Edict the King issued.

- **Question:** Who had issued the Edict of Nantes?
- **Context:** Renewed religious warfare in the 1620s caused the political and military privileges of the Huguenots to be abolished following their defeat. They retained the religious provisions of the Edict of Nantes until the rule of Louis XIV, who progressively increased persecution of them until he issued the Edict of Fontainebleau (1685), which abolished all legal recognition of Protestantism in France, and forced the Huguenots to convert. While nearly three-quarters eventually were killed or submitted, roughly 500,000 Huguenots had fled France by the early 18th century[citation needed].
- **Answer:** N/A
- **Prediction:** Louis XIV

5.2 Incorrect Pronoun Association

In the example below, we see that the query is located almost word for word within the context, but the noun being referred to is "she". This she is actually referring to one of Triton's daughters, but the antecedent for this pronoun is over a sentence away. Additionally, since Triton is likely to not be included in the pretrained word vectors, and there is no context in the context that Triton is actually male. As a result, the model incorrectly associates Triton with the "She" of the sentence, coming to the incorrect answer. One thing that would help our model with this is to train on a more diverse dataset, as if there was ancient Greek myths, then even though we are using pretrained word vectors, the character embeddings that are learned over the course of training might contain enough information that Triton gets correctly identified as a male.

- **Question:** At what village did a Triton stop to rest on a sandy beach
- **Context:** The origin of the legendary figure is not fully known. The best-known legend, by Artur Oppman, is that long ago two of Triton's daughters set out on a journey through the depths of the oceans and seas. One of them decided to stay on the coast of Denmark and can be seen sitting at the entrance to the port of Copenhagen. The second mermaid reached the mouth of the Vistula River and plunged into its waters. She stopped to rest on a sandy beach by the village of Warszowa, where fishermen came to admire her beauty and listen to her beautiful voice. A greedy merchant also heard her songs; he followed the fishermen and captured the mermaid.
- **Answer:** N/A
- **Prediction:** Warszowa

6 Conclusion

In this project, we implemented an end-to-end deep learning model to perform reading comprehension problems on the SQuAD 2.0 dataset. We applied state-of-the-art ideas from SQuAD 1.0, namely, the ideas of removing the necessity for a recurrent structure, and relying solely on convolution and self attention from QANet, to this new dataset. Our ensemble model achieved 63.956 EM and 67.442 F1 scores on the hidden test set, which ranks fifth on the class leaderboard in the non-PCE division as of March 19th. One limitation we ran into was memory issues on the GPUs we were training on, which meant we were unable to train with large batch sizes or the full set of parameters that we wished to use. We noticed that when we changed from single-head to multi-head attention the model performed better, but as we had to reduce the number of encoder blocks when we increased the number of heads for self-attention, we found 6 encoder blocks and 2-head self-attention to be the best compromise. In the future, we could look for ways to improve the memory efficiency of our model, as well as try to train additional different models in order to create a better ensemble model.

References

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

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[3] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[5] 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. *CoRR*, abs/1804.09541, 2018.