Question-Answering Using Modified QANet

Stanford CS224N Default Project IID Track
Mentor: Sarthak Kanodia

Xiluo He
Department of Computer Science
Stanford University
xiluohe@stanford.edu

Zach Witzel
Department of Computer Science
Stanford University
zachwitz@stanford.edu

Abstract

Question-answering (QA) systems aim to understand a natural language passage and answer questions about the passage. In this project, we build a QA system for the Stanford Question Answering (SQuAD) 2.0 dataset. Our goal is to hand-implement and tune deep learning language models to achieve strong performance on this task. Our primary contribution is our exploration of various modifications to the QANet architecture. Most notably, we show that the coattention mechanism can not be a substitute for the context-query layer. Using our implementation of QANet, we obtained a performance of F1/EM of 69.104/65.754 for the dev set and F1/EM of 65.992/62.637 on the test set.

1 Introduction

The QA task is a natural language processing problem where the system aims to generate an in-passage answer to a question about a given document or passage. Specifically, it outputs a span of text from the passage that answers the question or outputs no answer if the answer is not in the passage. QA systems allow us to retrieve information efficiently and are used in applications such as virtual assistants and medical information retrieval systems.

There are many challenges in creating a high performing QA system. For instance, it includes the same problems as the reading comprehension task. It requires reasoning over long spans of text since information from throughout the passage (both local and far away) may be used. The model also needs to be able to encapsulate different word meanings based on the context. Additionally, question answering needs to learn question structures and understand the passage-question relationship for questions that do not use the same vocabulary as the passage.

Performance on SQuAD is commonly used to measure effectiveness of QA systems. This dataset contains passages with questions and correct answers. SQuAD 2.0 introduces unanswerable questions so the model has to also learn if the question is answerable from the passage.

Question: Why was Tesla returned to Gospic?
Context paragraph: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke). During that year, Tesla taught a large class of students in his old school, Higher Real Gymnasium, in Gospic.
Answer: not having a residence permit

Figure 1: Example of a SQuAD <Question, Context, Answer> Triple

In this work, we explore the performance of two deep learning model architectures that performed well on SQuAD 1.1: Bidirectional Attention Flow (BiDAF) and QANet. BiDAF uses recurrent neural networks (RNNs) and introduces the idea of bidirectional attention while QANet uses convolutional neural networks (CNNs) and self-attention. [1, 2] Our contributions are:

1. We reimplement the QANet architecture from scratch.
2. We explore modifications to BiDAF and QANet.
3. We evaluate both of these system's performances on SQuAD 2.0 and analyze their limitations.

We improved the BiDAF architecture by exploring different character embeddings. Interestingly, we found that 1D CNNs create better embeddings than 2D CNNs. We also experimented on QANet's original architecture, replacing the context-query attention with coattention. [3] Even though coattention does capture context-to-query attention and query-to-context attention, we found that it does not work well with the QANet architecture.

2 Related Work

Attention-based techniques aim to ascribe meaning to words based on their context. In addition to better comprehension of the passage itself, QA systems use attention in order for the question to learn the context of the passage and vice versa. BiDAF introduced the idea of bidirectional attention where the attention flows from the question to the context and from the context to the question. [1] This allows QA models to represent the passage knowing what the question will be.

However, RNN-based models, such as BiDAF, are slow during training due to the recurrent layers present in these models. QANet is a transformer-based model that aims to address this limitation by using CNNs, capturing context using self-attention and positional embeddings instead. [2] With recurrent layers forcing the output to be computed sequentially, forward computations can be parallelized and training speed can increase by 3 to 13 times faster in training than other state of the art models without sacrificing much accuracy. Its fast training speed allowed it to be trained on larger models and more data to further increase performance. The paper uses the context-query bidirectional attention in the BiDAF paper to capture contextual information between the passage and the question.

There are recent context-query attention models as proposed in the Dynamic Coattention Network paper. [3] Coattention fuses codependent representations of the question and the passage in order to only focus on the relevant parts of both. We will incorporate this coattention model into the QANet architecture. Since the original QANet and Dynamic Coattention Network papers do not include implementations by the authors, we believe that our replication will be a useful resource for further work and modifications on the QANet architecture.

3 Approach

3.1 BiDAF

We use the BiDAF model prodvided by the CS224N staff as the baseline for our project. This model uses a hidden size of 100 and consists of 5 layers as described in the BiDAF paper: embedding layer, encoder layer, attention layer, modeling layer, and output layer. [1] Our baseline model is trained using only word embeddings.

3.1.1 Character Embedding

We improve on the baseline model by incorporating a learnable 200D character-level embedding to the embedding layer to improve morphology and for the model to better learn words not in the vocabulary. The initial character embedding is padded or truncated to a length of 16 characters per word and resize each embedding to the hidden size by passing through a convolution filter. It is then max-pooled to give a fixed size embedding. The character embedding is concatenated with a pretrained 300D GloVE word embedding. [4] The character embedding layer has a dropout of 0.2 while the word embedding layer has a dropout of 0.1. The concatenated embedding is passed through a projection layer and a highway network. [5] This is done for both the context and the question.

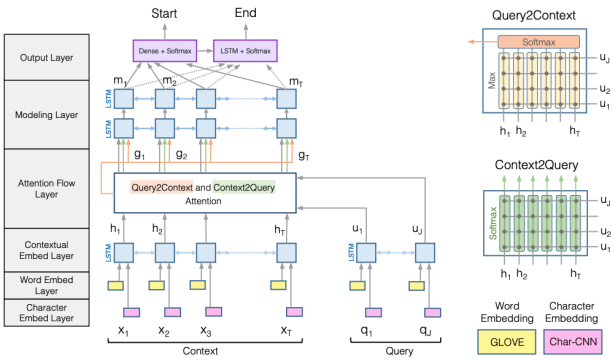

Figure 2: BiDAF Model Architecture

For the convolution filter, we experimented with both 1 dimensional and 2 dimensional convolutions. The input is given as *(batch size, embed size, sequence length, max word length)* which requires a 2D convolution. However, the 2D convolution will convolve around each sentence whereas 1D convolutions will convolve over each word, which might allow the model to learn more from the structure of the input. [6] We collapse the last batch size and sequence length dimensions but reshape the input in such a way that sequence length is recoverable for the 1D convolutions.

3.2 QANet

We implement QANet from scratch (aside from positional encodings and bidirectional flow attention). The QANet model has 5 layers and uses a hidden size of 128. [2] Outside of the model architecture, we employ a learning rate warm-up period to slowly increase our learning rate from 0 to 0.001 in the first 1000 steps. We used the Adam optimizer with $\beta = (0.8, 0.999)$, weight decay = 3e-7, and eps = 1e-7. The layers are implemented as follows:

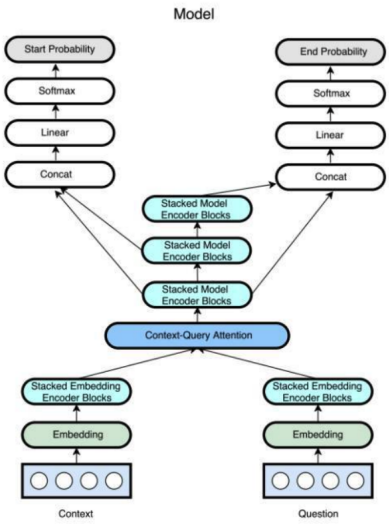

Figure 3: (a) The QANet Architecture

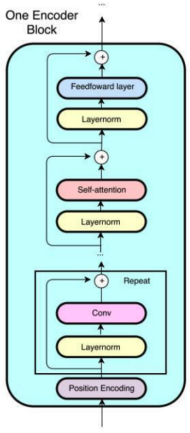

Figure 2: (b) QANet Encoder Block

Input Embedding Layer This layer consists of word-level and character-level embeddings. We used our implementation from the BiDAF model with 1D convolutions.

Embedding Encoder Layer The main component of QANet is the encoder block. It consists of a sinusoidal positional input encoding, convolutional layers, a self-attention layer, and a feed-forward layer. Each layer's output is normalized before going into the next layer to stabilize the hidden state dynamics and reduce training time. [7]. We also used a residual connection to allow

for better gradient flow through the network. Positional encoding are used in lieu of recurrence to inject information about the relative position of the tokens in the sequence. [8]. We refered to (https://github.com/hackiey/QAnet-pytorch/) to implement positional encodings. Depth-wise separable convolutions are used for the convolution layers to increase training speed. This layer consists of a depthwise convolution followed by a pointwise convolution which reduces the number of parameters in a convolution. However, our network was large enough so it increased efficiency without significantly reducing accuracy. [9] Self-attention was necessary to attend every word in the sequence to every other word without the use of RNNs.

The embedding encoder layer uses one encoder block to process the embeddings. The block contains 8 multi-headed attention heads and 4 convolutions with a kernel size of 7.

Context-Query Attention Layer QANet uses the same context-query attention mechanism as BiDAF. [1] From the encoded query and context, we create a similarity matrix to obtain the attentions and attended vectors from the context to the query and the query to the context. The only difference in implementation is that in QANet, the output of this layer is resized through a 1D convolution with kernel size 1.

Model Encoder Layer The model encoder layer is the core of the model and uses the encoder blocks described in the Embedding Encoder Layer section. We used 3 stacks of 7 encoder blocks. They had 2 convolutions per block and a kernel size of 5. Weights are shared between the between each block stack.

Output Layer The output layer predicts the probability of the start and end of the answer span and are modeled as:

$$p^{start} = softmax(W_1[M_0; M_1])$$
$$p^{end} = softmax(W_2[M_0; M_2])$$

where W_0 and W_1 are learnable parameters.

3.2.1 Modifications

After implementing the QANet architecture, we explored different model variations.

Number of Model Encoder Blocks After our initial runs, we wanted to explore the effect of changing the number of encoder blocks in the model encoder layer on runtime and accuracy. To do this, we changed the number of blocks and the batch size to fit more training examples per step. We tested this with 5 encoding blocks and 10 encoding blocks. Ultimately, we stuck to a set batch size because of hardware limitations.

Coattention Instead of using bidirectional attention in the context-query attention layer, we explore using coattention which attends to the query and context simulatenously. [3] From the context embedding D and the non-linear projection of the query embedding $Q = tanh(W^(Q)Q' + b^(Q))$, we calculate an affinity matrix:

$$L = D^T Q$$

The query-to-context attention is calculated:

$$A^Q = softmax(L)$$
$$C^Q = DA^Q$$

The context-to-query attention is likewise calculated. Then, we can compute the summaries of the previous attention contexts in light of each word. These two steps can be done in parallel to get a co-dependent representation of the question and document:

$$A^D = softmax(L^T)$$
$$C^D = [Q; C^Q]A^D$$

Then, we use a bidirectional LSTM to fuse temporal information to the coattention:

$$u_t = Bi - LSTM(u_{t-1}, u_{t+1}, [d_t; c_t^D])$$
$$U = [u_1, ..., u_m]$$

We substitute this coattention mechanism directly with the bidirectional flow attention.

4 Experiments

4.1 Data

We used the SQuAD 2.0 dataset for training and testing our model. The SQuAD 2.0 dataset contains 100,000 questions about sections of Wikipedia articles and an additional 50,000 unanswerable questions designed to appear similarly to answerable ones. [10]

4.2 Evaluation method

We evaluated our models using both an exact match (EM) and F1 score of the predicted answer compared to the true answer. The exact match requires a complete match between the predicted and labeled answer, so even if the prediction is functionally correct, such as by answering "Chicago, Illinois" compared to a label of "Chicago," the answer would be counted as wrong. The F1 score is the harmonic mean of the precision and the recall of the provided answers, which makes results more forgiving as it doesn't require complete equivalence.

4.3 Experimental details

Table 1: Max Dev EM and F1 Score by Model Type

Model	Learning Rate	Batch Size	Training Time
Baseline	0.5	64	3h 14m
BiDAF with 1D Character Embedding	0.5	64	3h 49m
BiDAF with 2D Character Embedding	0.5	64	3h 55m
QANet 5 Layers	0.001	16	8h 45m
QANet 7 Layers	0.001	16	11h 42m
QANet 10 Layers	0.001	16	12h 32m
QANet Coattention	0.25	16	2h 59m

For the QANet models with 7 and 10 layers, we ran into issues with the memory limits on Azure. This resulted in the training runs failing partway through, and thus we had to retrain those models several times before they worked. The memory issues were also why we had to go down to a 16 batch size for all QANet models compared to the 64 batch size described in the original paper and which we used for the BiDAF models.

All QANet models took significantly longer than BiDAF models to train. This makes sense as the model architecture is significantly more complicated. Additionally, the larger the number of encoding block layers in a QANet model, the longer it took to train.

The learning rate for every model besides the QANet with coattention was just picked to be the same as what was described in their respective implementation papers. For the QANet model using coattention, we tried using lower learning rates, but anything below 0.2 wouldn't really train; the graph was essentially stagnant. As such, we increased the learning rate until we got a meaningful result. However, the coattention version still significantly underperformed the other QANet models, and so we cut its training time short to save Azure credits, explaining the extremely short training time in the table.

4.4 Results

On the IID SQuAD test leaderboard, we were able to achieve an F1 score of 65.922 and an EM score of 62.637, which at the time of writing has us in 23rd place. From our readings of QANet's successful results in previous question answering challenges, we were not surprised by how well it performed.

When we were implementing the character embeddings, we tried both 1D and 2D convolutional character embeddings. The original paper described the embedding using a 1D convolution, but we thought there was a possibility that a 2D convolution could encapture more information and thus perform better. We were incorrect in our thought, and the 2D convolution performed worse than the 1D did, so we went with the 1D from then on.

Table 2: Max Dev EM and F1 Score by Model Type

MODEL	EM	F1
BASELINE	57.42	60.82
BIDAF WITH 1D CHARACTER EMBEDDING	62.34	65.82
QANET 5 LAYERS	63.84	67.34
QANet 7 Layers	**65.64**	**68.89**
QANET 10 LAYERS	63.65	66.83
QANET COATTENTION	51.10	53.22

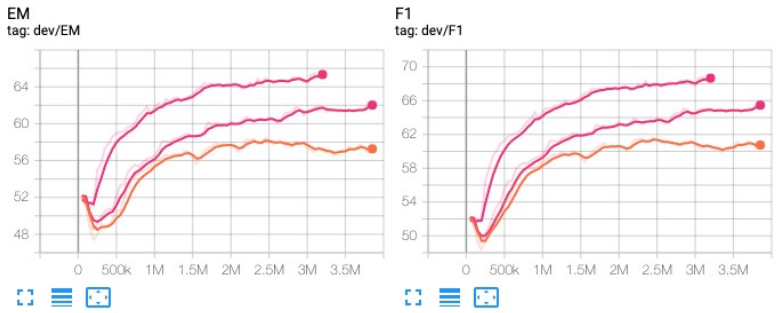

Figure 4: BiDAF (w/ Character Embedding) vs QANet Performance (bottom = baseline BiDAF, middle = character embedding BiDAF, top = QANet)

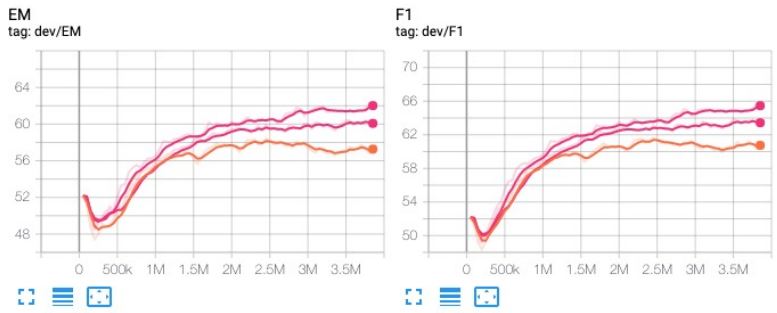

Figure 5: 1D vs 2D Convolutional Character Embedding vs Baseline (bottom = baseline, middle = 2D Convolution, top = 1D Convolution

However, what did surprise us was that when we tested QANet with different numbers of encoding block layers, the results went down for both an increase and a decrease in the number of layers. We first tried training with 5 layers and saw an increase when we went to 7, the number of encoding block layers in the original architecture, but were confused when it went back down again after we moved up to 10. This means that the increase in testing time as described in the previous section makes increasing the layers beyond 7 not worth it at all because it both takes longer to train and gives worse results. In hindsight, however, we realized that the original paper's model was tested with different numbers of layers and was already chosen to be optimal, so our results matching theirs help confirm that our implementation was correct.

We also tried replacing the the self-attention in the encoding block of QANet with coattention. This performed significantly worse than even the baseline implementation. As such, we cut it's training early to save compute time.

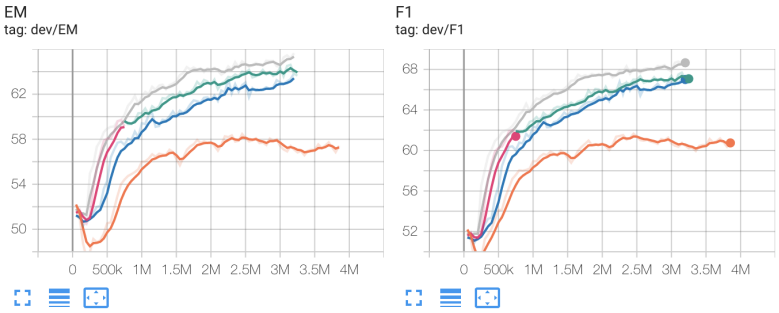

Figure 6: QANet with 5 vs 7 vs 10 Encoding Blocks vs Baseline
(orange = baseline, blue = 5 layers, green = 10 layers, gray = 7 layers)

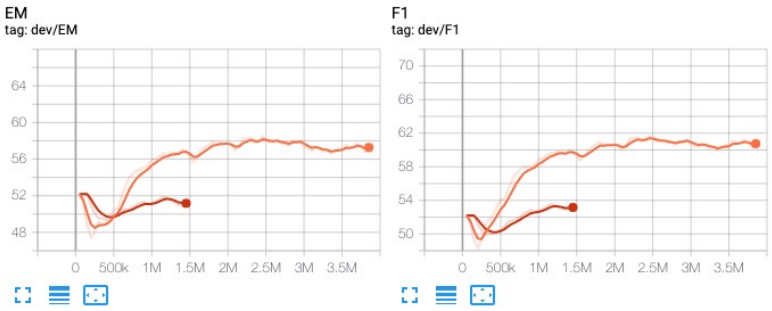

Figure 7: Coattention QANet vs Baseline
(red = Co-Attention QANet, orange = baseline BiDAF)

5 Analysis

Aside from just looking at the F1 and EM scores, we also looked into how the model was actually answering the questions themselves. From qualitative inspection, the 7-Layer QANet performs especially well when the answer is N/A. This is not terribly surprising though because it is the most common result and, due to the relative frequency of unanswerable questions, the model is pretty incentivized to be more cautious on when it thinks it should give an answer. When it does answer however, it will also frequently give more verbose answers than the labels do, such as "$4.093 million available for disbursement" instead of just "$4.093 million". This is very much expected because we selected for the model with the highest F1 to do well on the leaderboard, and the F1 calculation places a larger importance on including the right answer over giving a concise result.

Question: What are Harvard's Pell grant reserves?
Context: Harvard has the largest university endowment in the world. As of September 2011[update], it had nearly regained the loss suffered during the 2008 recession. It was worth $32 billion in 2011, up from $28 billion in September 2010 and $26 billion in 2009. It suffered about 30% loss in 2008-09. In December 2008, Harvard announced that its endowment had lost 22% (approximately $8 billion) from July to October 2008, necessitating budget cuts. Later reports suggest the loss was actually more than double that figure, a reduction of nearly 50% of its endowment in the first four months alone. Forbes in March 2009 estimated the loss to be in the range of $12 billion. One of the most visible results of Harvard's attempt to re-balance its budget was their halting of construction of the $1.2 billion Allston Science Complex that had been scheduled to be completed by 2011, resulting in protests from local residents. As of 2012[update], Harvard University had a total financial aid reserve of $159 million for students, and a Pell Grant reserve of $4.093 million available for disbursement.
Answer: $4.093 million
Prediction: $4.093 million available for disbursement

Figure 8: Context, Question, Answer, and Prediction for 7-Layer QANet

6 Conclusion

Overall, we learned a lot about QANet, how to design model architecture, and how to effectively interpret and implement model descriptions in scientific papers. We are able to successfully implement and create modifications for the QANet architecture. In doing so, we were able to achieve an increase in F1 and EM each by over 8 points compared to the baseline and placed us in 23rd on the leaderboard. We tried to implement many different changes to the QANet architecture, but none of which were more successful than our implementation of QANet as described in the original paper. Additionally, we were primarily limited on time, Azure credits, and memory. We could only train so many QANet modifications because QANet trains extremely slowly, using up our Azure credits and taking a full day to see results. Additionally, we had to do some runs multiple times and lower the batch size because we would run into a memory issue on Azure partway through. For future work, we would like to try testing parameterized positional encodings, letting the model learn the encoding parameters instead of requiring sinusoidal positional encodings. We hoped this would allow for a more flexible architecture, but did not have the time nor credits to test it.

References

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

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

[3] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016.

[4] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[5] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[6] Yoon Kim, Yacine Jernite, David A. Sontag, and Alexander M. Rush. Character-aware neural language models. *CoRR*, abs/1508.06615, 2015.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv e-prints*, page arXiv:1607.06450, July 2016.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[9] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

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