SQuAD: To QANet and Beyond

Stanford CS224N Default Project

Darkhan Baimyrza
Department of Computer Science
Stanford University
dark@stanford.edu

Abstract

One of the important topics in NLP domain is machine reading comprehension through automated question answering. In this project we research and implement from scratch a question answering system based on QANet [1] neural network model. We compare the performance of QANet neural network architecture to one of the previous recurrent neural network, in particular a baseline based on BiDAF [2] architecture. Then we experiment by modifying components of QANet model in a novel way and observe the impact of architectural modifications to model's performance on SQuAD v2.0 [3] dataset.

1 Introduction

Machine reading comprehension has been one of the exciting research areas in NLP for obvious reasons. Over the past few years significant progress has been made in one of the ways we measure machine reading comprehension - automated question answering (Q&A). Until recent, most leading neural network architectures employed recurrent neural network (RNN) approach in their models. With the latest introduction of Transformer [4] that solely concentrates on just self-attention, we investigate how machine reading comprehension Q&A systems can not only be achieved without RNNs in their neural network architecture, but also achieve better results with just self-attention mechanism. The main research paper that inspired this project is a feed-forward type architecture - QANet [1], where it has no RNN componenets, and instead borrows ideas from Transformer.

In our project we first train RNN-based baseline inspired by BiDAF [2] model, then implement from scratch and train QANet model to show advantages of the latter neural network architecture over the former model. Further we investigate how simple modifications to QANet model affects its results and show quantitative details of those impacts in comparison with a baseline.

2 Related Work

Due to how human natural language is structured, specifically sequential nature of sentences, almost all previous neural network models employed RNNs to process input, and used an attention component to deal with interactions of words in long input of texts. Our baseline is based on a similar architecture, in particular BiDAF [2] architecture. The weakness of RNN-based models, however, is that they are often slow for both training and inference because of their recurrent nature, especially processing long input of texts. QANet [1] model, compared to BiDAF model, approaches Q&A task in a complete novel way, which is the main research of this project.

Our motivation in researching and implementing QANet model is based on its architecture of approaching machine reading comprehension task in a non-traditional way, by going away from RNNs and adapting ideas from Transformer [4]. We find Transformer model to be a very valuable architecture to study as it gave birth to a lot of other state-of-the-art models like BERT, GPT-3, etc.. Although QANet's use of Transformer is slightly modified in its Encoder Blocks, we still find QANet to be a perfect candidate for exploration into Transformer building blocks.

In this project paper, we mainly concentrate on the detailed architecture of QANet neural network, and describe our experimentations with the model.

3 Approach

First, we will formulate the reading comprehension problem as follows. We are given a context paragraph C with n words, $C = \{c_1, c_2, \ldots, c_n\}$, and a query sentence Q with m words, $Q = \{q_1, q_2, \ldots, q_m\}$, the objective of the model is to find a span of context words $S = \{c_i, c_{i+1}, \ldots, c_{i+j}\}$ from C, where S is the answer to Q. Since roughly half of SQuAD v2.0 [3] dataset's context paragraphs are unanswerable, we prepend a special **O**ut **o**f **V**ocabulary (OOV) token c_{OOV} to $C = \{c_{OOV}, c_1, c_2, \ldots, c_n\}$. In this case if C has no answer, its ground truth answer span is $S = \{c_{OOV}\}$. For further discussions, we will denote x to be a word for any $x \in C, Q$.

3.1 Baseline

As a baseline for this project we use the default initial model based on BiDAF [2] architecture without character embeddings. More details of this BiDAF model and architecture can be found in Default Final Project handout [5].

3.2 QANet Architecture

We implement a slightly modified QANet model based on the research paper "QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension" [1]. The main difference of this model compared to previous neural networks is that QANet does not use RNNs at all, and instead use convolutions and self-attentions in its feed-forward type architecture. QANet borrows the ideas from Transformer [4] where it exclusively used convolutions and self-attention as building components in its Encoder Blocks, as shown in Figure 1. Our implementation of QANet is made of five main layers.

3.2.1 Input Embedding Layer

We follow the main paper and adopt the same techniques to obtain embeddings for word x. In particular, we use pre-trained GloVe word embedding vectors e_w of dimension $d_1 = 300$ that are fixed during training, and character embeddings where each character represented as a trainable vector of dimension $d_2 = 64$. The length of each word is either truncated or padded to 16. We apply convolution to character embeddings to get a matrix of size $\mathbb{R}^{d_2 \times 16}$ then take the maximum of each row to get character embeddings e_c of size $\mathbb{R}^{d_2}$. Finally, for given word x we concatenate to get embedding $[e_w; e_c] \in \mathbb{R}^{d_1+d_2}$. After this, following the paper, we apply two-layer highway network [6] on top of this representation. The output dimension of this layer is of size $d_1 + d_2$ for each individual words. For simplicity of further discussion, we will denote x to be a word embedding vector of size $\mathbb{R}^{d_1+d_2}$ as described above, not word $x \notin C, Q$.

3.2.2 Embedding Encoder Layer

The input dimension of this layer is of size $d_1 + d_2$ for each individual words, which is immediately projected to a hidden size $h = 96$. This layer consists of just one Encoder Block as shown in the right side of Figure 1, which is the main novelty of QANet architecture. Similar to the Transformer, QANet's Encoder Block contains four sublayers with residual connections in the following order: Position Encoding Sublayer, Convolution Sublayer, Self-Attention Sublayer, and Feed-Forward Sublayer, with exception of Convolution Sublayer that is not present in Transformer encoder block and this Sublayer can be repeated and stacked multiple times within this Encoder Block. In this model layer Encoder Block has 4 repeated Convolution Sublayers.

The implementation of all Sublayers (Position Encoding, Self-Attention, Feed-Forward) are the same as in the Transformer, where we use multi-head attention with 8 heads in all Encoder Blocks of QANet model.

We use Depthwise Separable [7] convolutions in Convolution Sublayer with kernel size 7.

The output dimension of this layer is h for each words.

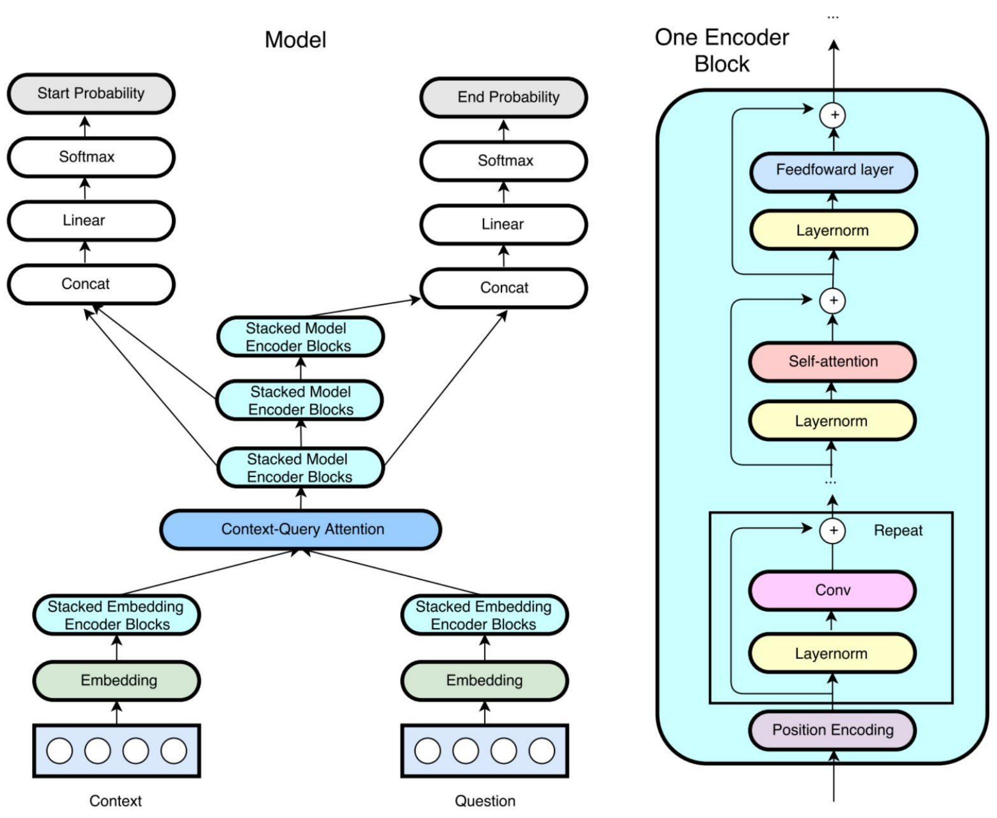


Figure 1: A high-level overview of the QANet architecture (left) with several Encoder Blocks. Encoder Block (right) is used throughout the model.

3.2.3 Context-Query Attention Layer

The input dimension of this layer is of size h for each context words C and query words Q. This is the layer where the fusion between context and query takes place. Context-to-Query and Query-to-Context attentions are calculated in this layer. Since the paper follows the same technique as BiDAF, we have reused our baseline's implementation of this layer in our QANet model, where more details can be found in Default Final Project handout [5].

The output of this layer is the concatenation $[c; a; c \odot a, c \odot b] \in \mathbb{R}^{4h}$, where $c \in \mathbb{R}^h$ context word embeddings, $a \in \mathbb{R}^h$ Context-to-Query attention, $b \in \mathbb{R}^h$ Query-to-Context attention, and $\odot$ is Hadamard product.

Due to the speed up technique we have employed for our QANet model, we immediately project (resize) the output dimension size from $4h$ to h.

3.2.4 Model Encoder Layer

The input dimension of this layer is of size h. In this layer we use the stack of 7 Encoder Blocks as described in Embedding Encoder Layer, with a few modifications. The number of Convolution Sublayer is 2 in each Encoder Blocks.

This stack of Encoder blocks are reused 3 times, in other words, the weights of this stack are shared. After each stack pass-through we calculate outputs of model encoders $M0, M1, M2$ as shown in Figure 1, from bottom to top.

The output dimension of this layer is of size h for each $M0, M1, M2$.

3.2.5 Output Layer

The input dimension of this layer is of size h. For this layer we follow the paper to predict the probability of each position in C being the start or end of an answer span. More specifically, the probability of starting and ending position are modeled as:

$$p^s = softmax(W_1[M0; M1]), \;\; p^e = softmax(W_2[M0; M2])$$

where W_1 and W_2 are two trainable variables and $M0, M1, M2$ are outputs from the previous layer. Finally, the objective function is defined as the negative sum of the log probabilities of the predicted distributions indexed by true start and end indices, averaged over all training examples:

$$L(\theta) = -\frac{1}{N}\sum_{i}^{N}[\log(p^s_{y^s_i}) + \log(p^e_{y^e_i})]$$

where y^s_i and y^e_i are respectively true starting and ending position of answer span for example i, and θ contains all training variables.

4 Experiments

4.1 Data

We use the official SQuAD v2.0 [3] dataset that contains (context, query, answer) triples. The whole dataset is split up into train set (129,900 examples), dev set (5951 examples), test set. Using the default project starter code set up, we trained our model on train set of the SQuAD v2.0, and evaluated on dev set. More details about the SQuAD v2.0 dataset can be found in Default Final Project handout [5].

4.2 Evaluation method

We employed standard evaluation methods of Q&A tasks in NLP, in particular after training our QANet [1] model on SQuAD v2.0 dataset, we obtained F1 score (harmonic mean of precision and recall) and EM score (Exact Match) evaluation metrics on dev and test set of SQuAD v2.0. More details about F1/EM evaluation scores and their calculation formulas can be found in Default Final Project handout.

4.3 Experimental details

We have followed the paper and started with the default hyperparameter configurations for training. Although we have experimented with different hyperparameters, after all experimentations the default hyperparameter configuration set up delivered the best results.

We trained our QANet model using two types of standard regularizations, L2 weight decay with parameter $\lambda = 3 \times 10^{-7}$, and dropout on word/character embeddings and between layers, where word/character dropout rates were 0.1/0.05 respectively, and dropout rate of 0.1 between every two layers.

We chose the hidden size of our model to be 96, and batch size set to 32, training steps are default 50K of the project starter code set up. Following the paper, we used ADAM optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$, and learning rate of 0.001. We used exponential moving average that was applied to all trainable variables with a decay rate 0.9999.

Due to memory requirements of our model, we started training on the Azure's NC6 VM with NVIDIA Tesla K80 GPU with 12GB, and had to upgrade to Azure's NC6_v2 VM with NVIDIA Tesla P100 GPU with 16GB.

During our implementation of QANet from scratch, we have experimented with different model modifications, namely: a model with different hidden sizes $h = 128$, $h = 64$; a model without character embeddings; a model with less Encoder Blocks (1, 3, 5) in the stack of Model Encoder Layer (out final version has a stack of 7); a model where Context-Query Attention Layer's output was not projected (reduced) to dimension h but kept at $4h$; a model with default project's optimizer Adadelta with default parameters. All of the above modifications to QANet did not result in any

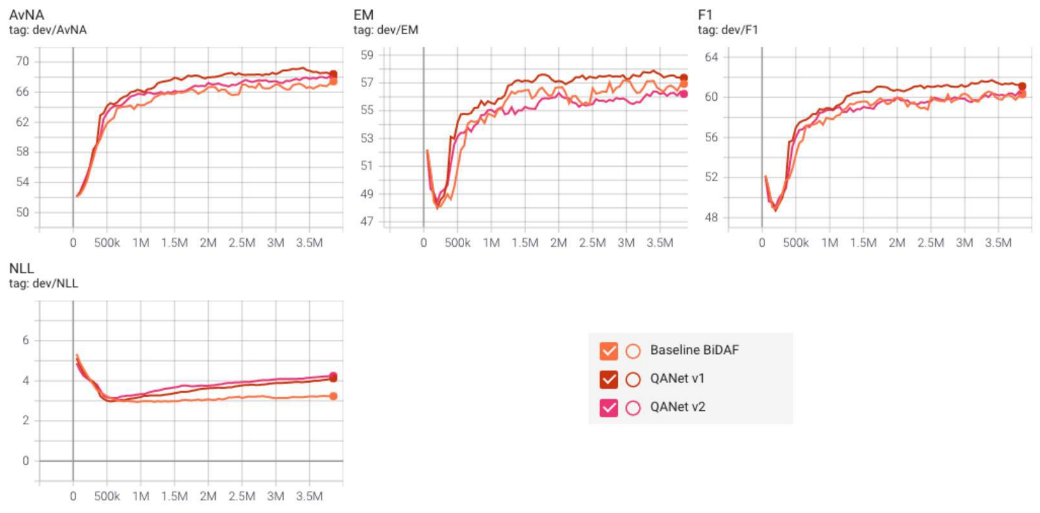


Figure 2: QANet v1 and QANet v2 results compared to baseline BiDAF.

significant boost, and in fact underperformed when compared to the baseline, or resulted in a model with very slow training progress, and therefore we will not discuss them.

However we picked other top-3 QANet implementations with slight variations in the architecture, where their results were comparable and above the baseline. In the following section we show the results of QANet v1, QANet v2, and QANet Final compared to the baseline and their differences, where QANet Final model's architecture is described in detail in Section 3.

4.4 Results

The following are quantitative results of different QANet implementations compared to baseline:

	Dev Set	Test Set
Model	EM/F1	EM/F1
Baseline BiDAF	57.130 / 60.580	-
QANet v1	57.890 / 61.720	-
QANet v2	56.360 / 60.550	-
QANet Final	**59.536 / 63.318**	**56.331 / 60.651**

QANet v1 compared to the QANet Final had non-trainable character embeddings, and non-trainable OOV token word embeddings, and completely separate (weights not shared) 3 stacks of 7 Encoder Blocks in Model Encoder Layer, where each of those stacks output is $M0, M1, M2$ respectively. The 30 epoch training of QANet v1 took 19 hours 53 minutes.

QANet v2 was similar to QANet v1, but the Model Encoder Layer is the same as in QANet Final. The 30 epoch training of QANet v2 took 16 hours 41 minutes.

The results of both QANet v1 and QANet v2 were comparable with baseline as shown in Figure 2.

Our best implementation - QANet Final, outperformed all other QANet implementations and achieved results that are above baseline as show in Figure 3. The 30 epoch training of QANet Final took 17 hours and 16 minutes.

Despite the higher scores of QANet Final over baseline BiDAF, we found that the training time of QANet is much slower, which is the opposite of what original paper claimed. Baseline's 30 epoch training time was 11 hours 20 minutes.

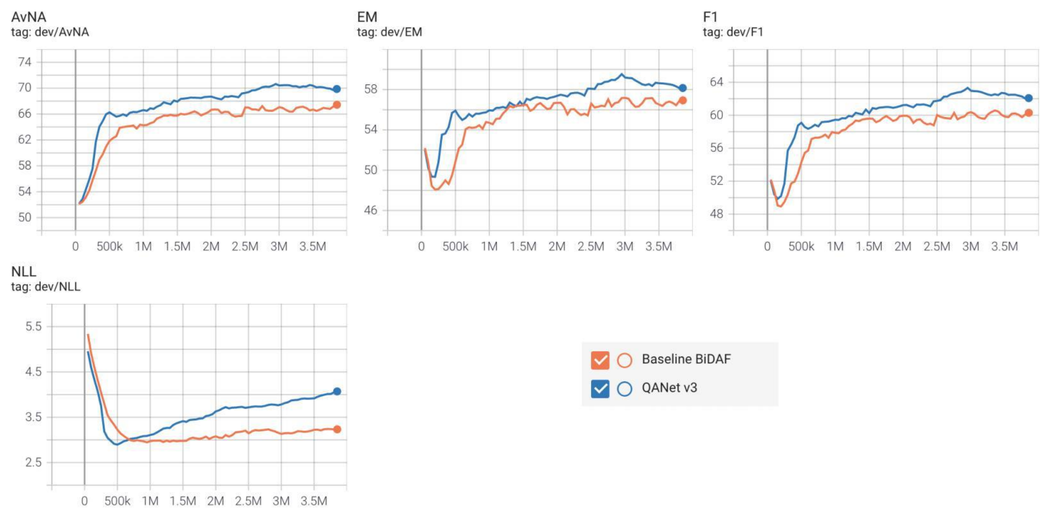
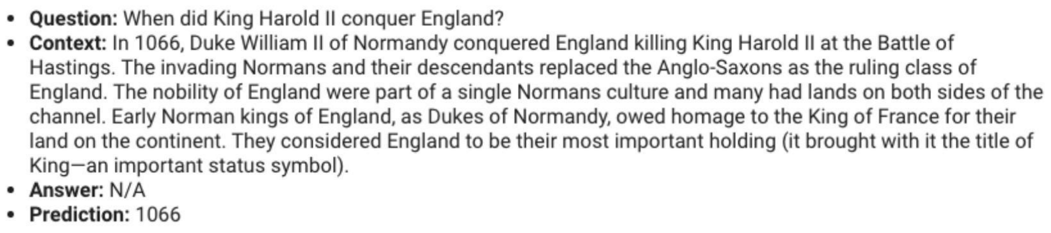


Figure 3: QANet Final results compared to baseline BiDAF.



- **Question:** When did King Harold II conquer England?
- **Context:** In 1066, Duke William II of Normandy conquered England killing King Harold II at the Battle of Hastings. The invading Normans and their descendants replaced the Anglo-Saxons as the ruling class of England. The nobility of England were part of a single Normans culture and many had lands on both sides of the channel. Early Norman kings of England, as Dukes of Normandy, owed homage to the King of France for their land on the continent. They considered England to be their most important holding (it brought with it the title of King—an important status symbol).
- **Answer:** N/A
- **Prediction:** 1066

Figure 4: An example of a question with a context paragraph.

5 Analysis

Upon analyzing answer span output of QANet Final, we see that both baseline BiDAF and our QANet Final give incorrect answers to questions where there is a few intersections of words in both context paragraph and query sentence. For example, as shown in Figure 4, both context and query contain: *conquer*, *England*, *King Harold II*, where a model learned to associate questions like *When* with integer strings in the sentence where there are "enough" intersections between context and query.

Other examples of incorrect answers given by QANet Final are in questions where the answer is logically deduced from the context. For example the question: *How many non-Muslims are in Greater London?* for the context paragraph:

Greater London has over 900,000 Muslims, (most of South Asian origins and concentrated in the East London boroughs of Newham, Tower Hamlets and Waltham Forest), and among them are some with a strong Islamist outlook. Their presence, combined with a perceived British policy of allowing them free rein, heightened by exposés such as the 2007 Channel 4 documentary programme Undercover Mosque, has given rise to the term Londonistan. Following the 9/11 attacks, however, Abu Hamza al-Masri, the imam of the Finsbury Park Mosque, was arrested and charged with incitement to terrorism which has caused many Islamists to leave the UK to avoid internment.

we see that model learned to attend to the correct parts of the context, but fails to deduce the correct logical answer.

6 Conclusion

In this project we implemented QANet [1] neural network architecture from scratch, where QANet borrowed ideas from Transformer [4] in its Encoder Blocks. A direct hands-on experience learned during implementation details of this architecture with PyTorch greatly increased our understanding and intuition behind self-attention mechanism of Transformer. Precisely diving deeper to understand the powerful self-attention mechanism was our motivation in choosing the original paper, and we feel that we have achieved our set goals.

7 Acknowledgements

The authors of this paper would like to thank the teaching staff of Stanford's CS224N course for providing very helpful guidance and instructions along with the implementation of the starter code and baseline of this project.

References

[1] 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, 2018.

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional Attention Flow for Machine Comprehension, 2018.

[3] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

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

[5] Chris Manning and CS224N course staff. CS224N Default Final Project: Building a QA system (IID SQuAD) track, 2021.

[6] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway Networks, 2015.

[7] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, 2017.