Using QANet to Perform Question Answering on the SQuAD 2.0 Dataset

Stanford CS224N Default Project

Eric Zeng
Department of Computer Science
Stanford University
erzeng@stanford.edu

Ricky Grannis-Vu
Department of Computer Science
Stanford University
rickygv@stanford.edu

Leon Bi
Department of Computer Science
Stanford University
leonbi@stanford.edu

Abstract

Machine comprehension and question answering (QA) systems are quickly becoming a staple of the modern world, with widespread commercial use from virtual assistants to search engines. The recently-proposed QANet model performs well on the earlier SQuAD 1.0 dataset, in large part due to its more parallelizable architecture which enables it to have significantly faster training and inference. In this project, we present our implementation of a custom QANet model and analyze our QANet model's performance on the newer SQuAD 2.0 dataset, which includes unanswerable questions. We found that the QANet model underperformed and hypothesize that this is in large part due to class imbalance within the SQuAD 2.0 dataset as well as memory and compute limitations with our model architecture and training. We recommend several avenues for future work to explore to overcome these limitations and achieve better performance on the SQuAD 2.0 dataset.

1 Key Information to include

- Mentor: Yian Zhang
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Machine comprehension and question answering (QA) systems are quickly becoming a staple of the modern world, with widespread commercial use from virtual assistants to search engines. Since recurrent neural networks (RNNs) are designed to process sequential data, they are typically used within QA systems in order to process textual data, which is sequential. However, because RNNs process data sequentially, they are unable to process language tokens in parallel, increasing both their training and their inference time. While ongoing research into using reinforcement learning techniques to skip unimportant language tokens may offer some potential speedup, such techniques are not always effective when processing long sequences of text, as is necessary for QA systems.

The long training time of RNNs makes it difficult for researchers and AI developers to iterate on these models, and so achieving good performance in QA systems is time and resource intensive. Furthermore, the long inference time of RNNs makes them less suitable for application in QA systems, as users may be turned off by the model's latency. A new model named QANet [1] was proposed,

which is capable of performing question answering with a more parallelizable model in place of a recurrent neural network component.

Rather than using recurrent neural nets, QANet proposes to exclusively use convolutions and self-attentions for encoders. The idea is that the convolution captures the local structure of the text while the self-attention learns the global interaction between each pair of words. Consequently the model is 3x to 13x faster in training and 4x to 9x faster in inference on the SQuAD 1.0 dataset. The speed-up in training allows for more iterations and therefore better results.

While the QANet model performs well on the SQuAD 1.0 dataset, we seek to investigate its performance on the SQuAD 2.0 dataset, which adds 50,000 unanswerable questions. We compare its performance with that of a baseline BiDAF model on the same dataset. We build upon the starter code provided in the project handout and spent significant time and effort implementing the complex QANet model architecture from scratch, including sub-components such as positional encodings, a multi-head attention mechanism, and depthwise-separable convolutions. We have made available all of our project code here: `https://github.com/rickygv99/squad`.

3 Related Work

A number of end-to-end neural networks have already been created for machine comprehension and question answering that we used to inform the way we designed our approach. For example, BiDAF [2] is a network that uses a bi-directional attention flow mechanism to get a context representation without early summarization. This model is our baseline and therefore we looked into its details to get insights into how we could improve their results. Another example of a related paper is r-net [3]. R-net is a gated self-matching network that creates a question aware passage representation, uses a self-matching attention mechanism to refine the representation, and finally uses a pointer network to locate the position of answers from passages. We looked into this paper because at the time of its submission it held the first place on the SQuAD leaderboard for both single and ensemble model.

The progress in end-to-end neural networks for machine comprehension and question answering can be attributed to the number of datasets available. Some datasets such as (Richardson et al., 2013; Berant et al., 2014; Yang et al., 2015) [4] are labeled by humans and high in quality but consequently are too small for training data intensive models. Conversely there are datasets that are automatically generated that are very large and allow for training of more expressive models. MS MARCO is an example of a large-scale dataset [5]. In this dataset specifically the questions are queries from Bing or Cortana and the passages are webpages and there are several related passages provided for each question in the dataset. Interestingly enough, the answers are human generated whereas in SQuAD the answers are a span of the passage.

4 Approach

4.1 Baseline Model

As our baseline, we are using a modified Bidirectional Attention Flow (BiDAF) [2] model. Unlike the BiDAF model from the original paper, our baseline BiDAF model does not include a character-level embedding layer.

At a high-level, the BiDAF model uses a bidirectional attention mechanism, computing both query-to-context and context-to-query attention. The two attention vectors are then combined together to produce a matrix $\mathbf{G}$, in which each column vector represents a query-aware representation of a context word. A two-layer bidirectional LSTM [6] network takes this matrix $\mathbf{G}$ as input and produces a matrix $\mathbf{M}$, in which each column vector contains contextual information about a context word with regard to both the whole context paragraph and the query. These matrices $\mathbf{G}$ and $\mathbf{M}$ can be used by the model's output layer (which is application-specific) in order to generate predictions.

4.2 Second Iteration

For the second iteration of our model, we sought to improve upon our BiDAF model by implementing a QANet model [1]. As described in (Yu et al., 2018), the QANet model consists of five main

components: an input embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer, and an output layer.

Input Embedding Layer: Our input embedding layer utilizes both the word embedding and the character embedding for a given word. Word embeddings are fixed during training from $p_1 = 300$ dimensional pre-trained GloVe word vectors [7]. All out of vocabulary words are replaced with an <UNK> token which has a random embedding initialization. Character embeddings are initialized similarly, with dimensionality $p_2 = 200$, and are an alternative way of representing a given word – as a concatenation of its individual character embeddings.

Input embeddings are generated for both the context paragraph C and the query sentence Q. When generating the input embedding for a given word, both its word and its character embeddings are passed through a dropout layer ($p = 0.1$). Its character embedding is then passed through a convolutional layer (kernel size = (1,5)) and then the maximum value of each row is taken. These maximum values are concatenated to the word embedding, producing $[x_w; x_c] \in \mathbf{R}^{p_1+p_2}$, where x_w is the word embedding and x_c is the maximum values computed from the character embeddings of x. Finally, this result $[x_w; x_c]$ is passed through a two-layer highway network [8], and its output is returned from this layer.

Embedding Encoder Layer: Our embedding encoder layer is applied separately to the input embeddings of the context paragraph C and the query sentence Q. The inputs to this layer have a dimension corresponding to word vector length of length $p_1 + p_2 = 500$ – we apply a depthwise-separable convolution [9] to map this to the hidden size of the network (d=100). In order to represent not just the meaning of each word in its embedding but also its position within the sentence, we concatenate the positional encoding of each word to its embedding, as described in the paper [1]. We used fixed positional encodings, which are calculated as follows:

$$PE_{pos,2i} = sin(pos/10000^{2i/d})$$
$$PE_{pos,2i+1} = cos(pos/10000^{2i/d})$$

where pos is the position of the word within the text, i is the position of the word within the input embedding, and d is the hidden size of the network.

We then perform 4 depthwise-separable convolutions on the resulting concatenation, each with d filters and a kernel size of 7. Next, we pass the result through a multi-head attention mechanism, as described in (Vaswani et al., 2017) [10]. This result is passed through one final feed-forward layer.

For each of these operations (conv/attention/ffn), the input is first layer-normalized [11]. Each operation is also contained within a residual block. Thus, each operation is computed as $f(layernorm(x)) + x$, where x is the input to the operation and f is the operation to be applied.

Context-Query Attention Layer: Our context-query attention layer takes in our query encoding and context encoding as input and outputs the attention scores between the context and query. In this layer, we first compute the similarities between each pair of context and query words, creating a similarity matrix $S \in \mathbf{R}^{nxm}$. Then each row of S is normalized with a softmax function resulting in matrix $\bar{S}$. We then compute the context-to-query attention as $A = \bar{S} \cdot Q^T \in \mathbf{R}^{nxd}$.

Model Encoder Layer: Our model encoder layer consists of three model encoder blocks, each of which generates a model encoding: M_0, M_1, M_2. The attention passed into this layer has length 400 – to map this to the hidden size $d = 100$ of the network we apply a depthwise-separable convolution with d filters. We then apply our first model encoder block to the attention scores to generate M_0. Each model encoder block consists of 7 embedding encoders, the latter 6 of which each take the output of the previous encoder as input. The embedding encoders are nearly identical to the earlier embedding encoder layer – the only difference is that 2 convolution operations are performed instead of 4. We calculate M_1 and M_2 similarly – the only difference is that they take M_0 and M_1 respectively as inputs rather than the attention scores. M_0, M_1, M_2 are returned from this layer as output.

Output Layer: Our output layer predicts the probability of each position in the context to be either the start or end of an answer span. The starting and ending positions are modeled as follows:

$$p^1 = softmax(W_1[M_0; M_1])$$
$$p^2 = softmax(W_2[M_0; M_2])$$

where W_1 and W_2 are trainable variables and M_0, M_1, M_2 are the outputs of our model encoder layer. p^1 and p^2 are returned from this layer as output.

Objective Function: Our objective function is the sum of the negative log-likelihoods of the distributions p^1 and p^2 (outputted from our output layer) over all the training examples.

$$L(\theta) = -\frac{1}{N}\sum_{i}^{N}[log(p^1_{y^1_i}) + log(p^2_{y^2_i})]$$

where y^1_i and y^2_i are the starting and ending positions of the i'th training example, and θ contains our trainable variables.

5 Experiments

5.1 Data

We trained and evaluated our model on examples from the SQuAD 2.0 dataset [12]. The dataset consists of context, question, answer triples with the answer being a span of text from the context that answers the question. SQuAD 2.0 differs from SQuAD 1.0 by including unanswerable questions that do not contain an answer to the question within the context. The train split contains 129,941 examples with the dev set containing 6078 examples and the test set containing 5915 examples. The dev and test set each contain about half of the official SquAD 2.0 dev set.

5.2 Evaluation method

We evaluated our model's performance quantitatively, using F1 score and exact match (EM) score averaged over all testing examples.

5.3 Experimental details

We trained the baseline model for 30 epochs and our QANet models for 20 epochs since early stopping showed signs of overfitting near that number of epochs. We made several modifications to the original QANet model from the paper. These include a larger QANet model with a hidden size of 192 instead of 128 for the convolution filters and 11 encoder blocks instead of 7 within the model encoder layer. We also trained a QANet model with the Adadelta optimizer used for the BiDAF model, and another QANet model with the AdamW optimizer, which modifies the Adam optimizer by decoupling weight decay from optimizations to our loss function, in order to improve our model's ability to generalize [13]. We also tried both a learning rate warmup and a constant learning rate setup when training our models. The learning rate warmup was identical to the QANet paper with an inverse exponential increase of the learning rate from 0 to 0.001 in the first 1000 steps of training whereas with the constant learning rate we kept it at 0.001. Additionally, we used $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 * 10^{-7}$, and a weight decay of $5 * 10^{-8}$ for Adam and AdamW. We did some manual hyperparameter tuning for several parameters and ended up making some small changes to the values from the QANet paper. Within the QANet model, we used a hidden size of 100 and a dropout rate of 0.1 along with an L2 weight decay of $\theta = 3 * 10^(-7)$ on trainable parameters. Unlike the original QANet paper, we used a batch size of 16 rather than 32 in order to remain within memory limits on the Azure virtual machine. Training our baseline BIDAF model took around 4 hours. Training for the QANet models took about 16 hours which was likely due to memory constraints since QANet is very memory intensive.

5.4 Results

Model Type	EM	F1
Baseline	57.621	60.886
QANet Larger Model	52.193	52.193
Adam with learning rate warmup	52.193	52.193
Adadelta with constant learning rate	52.126	52.126
Adam with constant learning rate	51.941	51.951
AdamW with constant learning rate	52.159	52.159

Table 1: Performance of various models on the dev set for SQuAD 2.0.

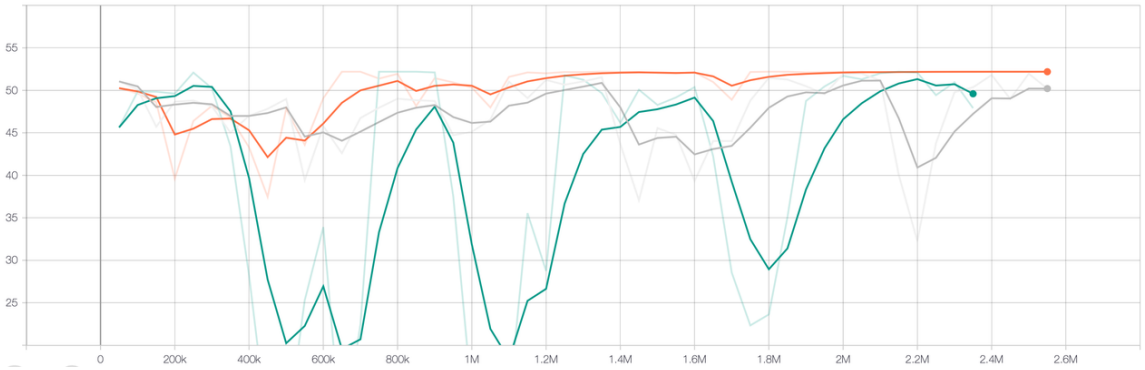

Figure 1: EM score vs steps for various models on the dev set
Green - Adam with learning rate warmup,
Grey - Adam with constant learning rate,
Orange- AdamW with constant learning rate

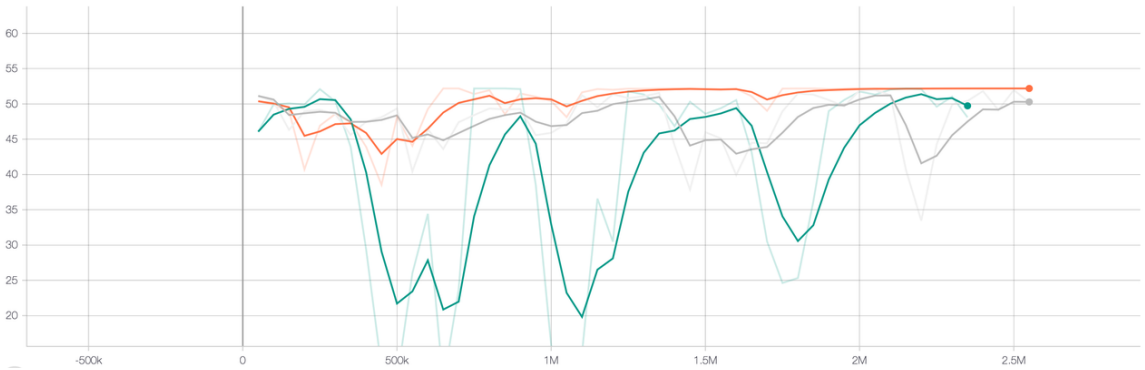

Figure 2: F1 score vs steps for various models on the dev set
Green - Adam with learning rate warmup,
Grey - Adam with constant learning rate,
Orange- AdamW with constant learning rate

The results on the test set for the QANet model using AdamW were EM: 47.878 and F1: 47.878. These are lower than the dev results which is expected due to some potential overfitting. However,

the results overall were lower than expected for QANet given that the model performed very well on SQuAD 1.0. There are several potential explanations for this such as differences from the original model with a lower batch size and the use of a constant learning rate. The above figures show extremely erratic training results over the 20 epochs for the experiment using Adam and the learning rate warmup scheme with the EM and F1 scores fluctuating heavily even after the initial training iterations. Because of this, we decided to go with a constant learning rate when trying AdamW which gave a much smoother curve, but it does reveal the possibility that there were implementation errors with the learning rate warmup that negatively impacted the model's performance. It is also odd that the EM and F1 scores ended up being identical for several of our experiments although this may not be relevant.

6 Analysis

By inspecting the outputs of our model, we found that QANet had difficulties with answering various types of questions which helps to explain the poor performance. All analysis done for QANet was with the AdamW and constant learning rate model. Because of the drastic drop-off in performance of QANet on the official SQuAD2.0 leaderboard compared to SQuAD1.0, we expected QANet to mostly have issues with unanswerable questions. However, this was not the case as the model was actually very consistent in predicting no-answer for unanswerable questions. On the other hand, the reason for this was that the model was predicting no-answer very commonly which gave it poor performance on answerable questions. Below are two examples where the first shows QANet outperforming BiDAF in predicting no-answer for an unaswerable question and the second showing BiDAF outperforming QANet in predicting the correct answer for an answerable question.

Question: Unsurprisingly, the mujahideen's victory with the Soviets in the 1980s succeeded to produce what?

Context: In Afghanistan, the mujahideen's victory against the Soviet Union in the 1980s did not lead to justice and prosperity, due to a vicious and destructive civil war between political and tribal warlords, making Afghanistan one of the poorest countries on earth. In 1992, the Democratic Republic of Afghanistan ruled by communist forces collapsed, and democratic Islamist elements of mujahdeen founded the Islamic State of Afghanistan. In 1996, a more conservative and anti-democratic Islamist movement known as the Taliban rose to power, defeated most of the warlords and took over roughly 80% of Afghanistan.

Answer: N/A

BiDAF prediction: justice and prosperity

QANet prediction: N/A

Question: What castle currently houses the Centre for Contemporary Art?

Context: The 17th century Royal Ujazdów Castle currently houses Centre for Contemporary Art, with some permanent and temporary exhibitions, concerts, shows and creative workshops. The Centre currently realizes about 500 projects a year. Zachęta National Gallery of Art, the oldest exhibition site in Warsaw, with a tradition stretching back to the mid-19th century organises exhibitions of modern art by Polish and international artists and promotes art in many other ways. Since 2011 Warsaw Gallery Weekend is held on last weekend of September.

Answer: Royal Ujazdów Castle

BiDAF prediction: Royal Ujazdów Castle

QANet prediction: N/A

Question Type	Who	What	Where	When	Why	How	Other
QANet F1	53.2	51.1	52.6	54.5	49.2	51.2	47.4

Table 2: QANet F1 score for different question types.

Additionally, we computed the F1 scores of the QANet model for questions that start with "Who", "What", "Where", "When", "Why", "How", and "Other" questions that started with something else. This shows that QANet struggled the most with "Other" questions which makes sense because they start with a wide range of different words so the model is less able to determine patterns for these types of questions.

QANet has several advantages over BiDAF due to transformers being able to follow longer dependencies in text and model stronger relationships between words with positional encoding when compared to recurrent models. In the below example, BiDAF incorrectly answers "Informal rule" whereas QANet correctly deduces that there is no answer because the question asks about imperialism and QANet recognizes that imperialism and rule are different terms so any context about informal rule is not relevant to the question.

Question: Which is less costly, formal, or informal imperialism?

Context: The definition of imperialism has not been finalized for centuries and was confusedly seen to represent the policies of major powers, or simply, general-purpose aggressiveness. Further on, some writers[who?] used the term imperialism, in slightly more discriminating fashion, to mean all kinds of domination or control by a group of people over another. To clear out this confusion about the definition of imperialism one could speak of "formal" and "informal" imperialism, the first meaning physical control or "full-fledged colonial rule" while the second implied less direct rule though still containing perceivable kinds of dominance. Informal rule is generally less costly than taking over territories formally. This is because, with informal rule, the control is spread more subtly through technological superiority, enforcing land officials into large debts that cannot be repaid, ownership of private industries thus expanding the controlled area, or having countries agree to uneven trade agreements forcefully.

Answer: N/A

BiDAF prediction: Informal rule

QANet prediction: N/A

7 Conclusion

In this paper, we implement the QANet model architecture and analyze its performance on the SQuAD 2.0 dataset. Interestingly, while QANet performs well on the SQuAD 1.0 dataset, it struggles to achieve good performance on the SQuAD 2.0 dataset and underperforms our baseline BiDAF model. We found that our QANet displayed bias towards predicting "no answer" on the SQuAD 2.0 dataset. This is likely because since SQuAD 2.0 contains 100,000 answerable questions and 50,000 unanswerable questions, it is imbalanced – a model is able to trivially predict 1/3 of the answers correctly by simply predicting "no answer" for everything. Possible avenues for future work to overcome this issue include exploring the effects of random under-sampling, under-sampling through Tomek links, and/or penalizing mistakes on answerable questions more strongly in our objective function.

Our model architecture and training was limited somewhat by memory limits and limited compute power on Microsoft Azure. We found that increasing the batch size from 8 to 16 improved our model's results; however, we were unable to use a batch size of 32 as in the original QANet paper due to memory limitations. It would be interesting to explore the effect of even larger batch sizes on the performance of our QANet model. We found the increasing the hidden size and number of attention heads for our model also increased our model performance. However, the larger the hidden size and number of attention heads, the more training time our model required, and consequently the more compute power we used. We had a limited amount of compute power available on Microsoft Azure, and so we had to limit the scale of our model. It would be interesting to explore the effect of a larger hidden size and more attention heads on our model performance.

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] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[4] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[5] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@NIPS*, 2016.

[6] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm, 2015.

[7] 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.

[8] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks, 2015.

[9] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2016.

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

[11] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

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

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017.