IID SQuAD QANet Project

Stanford CS224N Default Project

Daniel Guillen
Department of Computer Science
Stanford University
eliasd@stanford.edu

Claire Mai
Department of Computer Science
Stanford University
cmai21@stanford.edu

Abstract

Our goal is to create a QANet [1] model for the reading comprehension task where given a context paragraph and question sentence, the model will find the start and ending word in the context paragraph that answers the question. We chose the QANet architecture for its novel use of transformer components on the question answering (Q&A) task. Previous and current models for Q&A heavily utilized RNNs with attention which lead to longer training and interference times in order to achieve good results. The QANet architecture avoids this by replacing the RNN with self-attention and residual blocks, thereby attaining a speedup in training time with comparable accuracy. Our final QANet model achieved an F1-score of 62.57 on the test non-PCE leader board.

1 Key Information to include

- Mentor: Chris Wolff
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

One of the most famous tasks of Natural Language Processing (NLP) is Question and Answering where given a context paragraph, a model must process the context paragraph and an input question and then output a span prediction i.e. a continuous sub-section from the original context paragraph that answers the given input question (if there is an answer). In the case where there is no answer available in the context paragraph, the model should abstain from answering. Consider the example in Figure 1. We see that the answer to the question was "Southern California Megaregion" which is a sub section from the context paragraph.

Solving this task has important applications in multiple professional fields to be used as a tool to provide information about a specific subject, e-invoicing, and digital billing efficiently [2]. This $Q\&A$ task is is challenging because it requires the model to piece together information across the entire context paragraph and understand various subtleties of the English language.

Our approach was to implement QANet. QANet was revolutionary because it replaced the RNN and attention components that were essential in previous state of the models with transformer encoder blocks with self-attention [1]. By excluding the RNN component, training times and inference times sped up significantly and allowed the authors to train on even more data produced through the method of back translation to gain better results compared to previous state of the art models.

3 Related Work

Previous work developed for the SQuAD challenge played with ideas of character-level embeddings, attention, and transformers. One popular model that capitalizes on attention is BiDAF [3]. The model

- **Context**: "The 8- and 10-county definitions are not used for the greater Southern California Megaregion, one of the 11 megaregions of the United States. The megaregion's area is more expansive, extending east into Las Vegas, Nevada, and south across the Mexican border into Tijuana."
- **Question**: "What is the name of the region that is not defined by the eight or 10 county definitions?"
- **Answer**: "Southern California Megaregion"

Figure 1: Example input context paragraph and input question, and ground truth answer the the input question.

utilizes a Bi-Directional Attention Flow network which allows attention to flow from the question to the context and from context to question. BiDAF also utilized not only word-level embeddings, but also character-level embeddings and contextual embeddings [3]. This allows the model to create a complex representation of the context paragraph and then utilize a recurrent neural network (RNN) such as long short-term memory network (LSTM) to take the representation and output the subsection answer. The RNN is used to process sequential inputs and the attention layer is used to remember long term interactions. BiDAF received an F1 of 77.3 and a EM (exact match) of 68.0.

Despite the remarkable results, the use of an RNN lead to slower training and inference times because of their recurrent nature. To speedup the reading comprehension, Raiman Miller proposed to replace bi-directional attention with search beams. However, their model attained similar results to BiDAF and the RNN was still a vital component, thereby having long training times [4]. Weissenborn et al then proposed to eliminate the context-query attention module, but this led to a poorer performance with minimal speedup [5].

Thus, some researchers attempted to remove RNNs from Q&A models and instead utilize transformers. The first use transformers instead of RNNs was QANet [1]. QANet leverages self-attention and transformer encoder blocks so that the model no longer needs to create an output in sequential order and can therefore parallelize a lot of the work. This resulted in training and inference speedups up to 13x [1]. Because of the shorter training times, QANet is able to train on larger datasets and thus obtain better F1 and EM scores compared to other state of the art models. More specifically, QANet obtained an F1 of 84.6.

4 Approach

4.1 BiDAF

Baseline : We used the BiDAF model provided by the CS224N staff as our baseline. The BiDAF model uses 300-dimensional pre-trained GloVe embeddings to represent words. These word embeddings are held constant during training time. Note that the BiDAF model introduced in Seo et al's paper [3] also has character embeddings, but this baseline version does not.

4.2 QANet

Our main approach is to implement a QANet architecture as described by Wei Yu et al. [1] and evaluate how well it performs on the SQuAD question-and-answering dataset. The QANet architecture consists of five main components: an embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer, and an output layer as seen in Figure 1 [1].

Below is a description of each of the 5 main components of QANet:

1. Input Embedding Layer: We first lookup the embeddings for each word and for each character in each word. Both the word and character vectors use a pre-trained GloVe word vectors to obtain a fix word and character embedding. The word embeddings are frozen in place during training. Let p_1 and p_2 denote the word and character embedding size respectively such that $p_1 = 300$ and $p_2 = 200$. We then apply dropout the char embeddings and then put the char embeddings through a Conv2d with a kernel size of (1, 5) and stride of 1. ReLU is then applied on the Conv2d output. We then take the maximum of each row

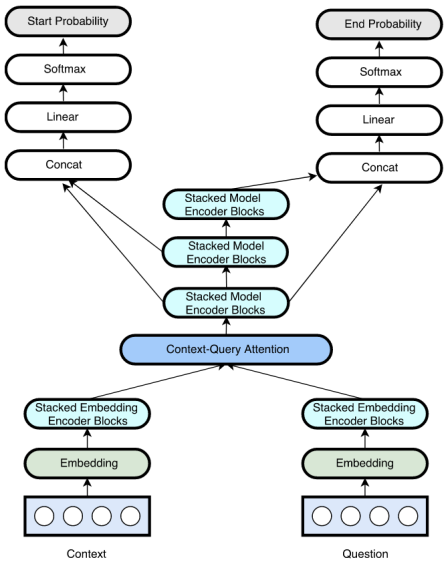

Figure 2: Diagram of the QANet architecture with the input context and question at the bottom of the diagram and outputs of the model located at the top of the diagram.

along the hidden size dimension in order to get a fixed-size vector representation of each word. Dropout is then applied to the word embeddings. This fixed-size vector representation is then concatenated to the word embeddings. This serves as the input into the Highway Encoder defined by Srivastava et al. [6]

2. Embedding Encoder Layer: This encoder block is a stack of positional encodings, several convolutional layers, a self attention layer, and feed forward layer, with a layernorm in between each of those layers as seen in Figure 3. Thus, we first add the positional encoding in order to incorporate this information into the model. The next several layers is a convolutional layer that utilizes depthwise seaparable convolutions. Wei Yu et al. decided on depthwise separable convolutions for their memory efficiency and for it being able to generalize better than regular convolutions [1]. The kernel is size 7 with 128 filters. After the conv layers is a multi-head self-attention layer with 8 heads as described by Vaswani et al. [7]. The last layer is the feed forward layer. Each of these layers has a residual connection the its input, thereby allowing the gradient to flow through the model efficiently.

3. Context-Query Attention Layer: We utilized the same implementation as the BiDAF model by Seo et al [3] by computing a similarity matrix consisting of the similarities between every pair of context and question words.

4. Model Encoder Layer: This layer is similar to the Embedding Encoder Layer and has the same parameters. The only difference is that there is an extra convolutional layer in the encoder block with a total number of 3 model encoder blocks. We repeat this step three different times with weights shared between each time in order to get three output matrices M_0, M_1, M_2.

5. Output Layer: The output layer predicts the probability of each position in the context paragraph as to whether it is the start or ending position of the answer. The probability of the start p^1 and ending position p^2 are modeled as

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

with M_0, M_1, M_2 are the outputs of the 3 model encoder blocks and W_1, W_2 are trainable variables. p^1 and p^2 are computed independently from each other.

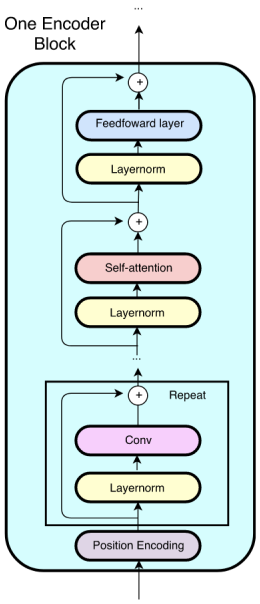

Figure 3: Diagram of a single encoder block used in the embedding encoder layer and model encoder layer.

4.3 Training

We use dropout two kinds of regularization: L2 weight decay and dropout. The L2 weight decay is set to $\lambda = 3x10^{-7}$. A dropout rate of 0.05 and 0.1 is applied to the character embeddings and word embeddings respectively. As stated in the QANet paper, we also apply layer dropout to each encoder block. For every encoder block each sublayer l has a survival probability of $p_l = 1 - \frac{l}{L}(1 - p_L)$ where L is the last layer and $p_L = 0.9$ [1].

We use the negative log likelihood as our loss function and model it as

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

where y^1_i is the ground truth starting position of the ith example and y^2_i is the ground truth ending position of the ith example. θ contains all the trainable variables. Our optimizer is Adam with $\beta_1 = 0.8$ and $\beta_2 = 0.999$.

During inference, we choose the predict span (start, end) that maximizes $p^2_{start}p^2_{end}$ such that $start \leq end$, i.e. the start token must come before the end token in terms of position.

5 Experiments

5.1 Data

We used the the Stanford Question Answering Dataset (SQuAD) [8]. More specifically, we utilized the SQuAD2.0 version, which is sourced from Wikipedia articles. The dataset contains around 150,000 questions. Of the 150,000 questions, there are 100,000 answerable questions and 50,000 adversarial unanswerable questions. The dataset was split into 3 separate datasets: train, dev, and test. The train set has 129,941 examples, while the dev set has 6078 examples and the test set has 5915 examples.

The input is a context paragraph with n words i.e. $C = \{c_1, \ldots, c_n\}$, an input question $Q = \{q_1, \ldots, q_m\}$ with m words. The output is then a span prediction i.e. a continuous sub-section from the original context paragraph C that answers the given input question Q (if there is an answer). In the case where there is no answer available in the context paragraph, the model should abstain from answering.

5.2 Evaluation method

The main evaluation metric we used is the F1-score, the harmonic mean of the precision and recall metrics. F1 measures the portion of overlap tokens between the predicted answer and ground truth [1]. We also use EM-score which stands for exact match. EM is a binary metric that is either 0 or 1. This means that if the predicted span is exactly the ground truth then it would receive an EM of 1, otherwise 0.

5.3 Experimental details

Report how you ran your experiments (e.g. model configurations, learning rate, training time, etc.)

Similar to Wei Yu et al., we used a warm up scheme learning rate with an inverse exponential increase from 0.0 to 0.001 for the first 1000 steps, and then keep a constant learning rate of 0.001 for the remainder of training for all experiments. We also trained all experiments for 2.5 million iterations.

QANet-v1 QANet-v1 only had the input embedding layer and context-query attention layer and output layer with a fixed learning rate of 0.001

QANet-v2 QANet-v2 with all layers (input embedding layer, embedding encoder layer, context-query attention layer, model encoder layer, output layer) with no regularization, stochastic depth regularization (also known as layer dropout), or warm-up scheme learning rate.

QANet-final QANet-final has all layers (input embedding layer, embedding encoder layer, context-query attention layer, model encoder layer, output layer) with a warm-up scheme learning rate as described above.

5.4 Results

Model	Optimizer	F1-score(dev)	F1-score(test)	EM-score(dev)	EM-score(test)
Baseline	Adadelta	58	N/A	55	N/A
QANet-v1	Adam	50.294	N/A	50.294	N/A
QANet-v2	Adam	52.193	N/A	52.193	N/A
QANet-final	Adam	64.5	62.570	61.334	58.969

Table 1: Table denoting the optimizer, F1-score, and EM-score for each model on the dev and test non-PCE leaderboard.

As seen in Table 1, both QANet-v1 and QANet-v2 did worse than the baseline with dev F1-scores of 50.294 and 52.193 respectively, while the baseline received a dev F1-score of 60.686. QANet-v2 had no regularization. Thus, by fixing the previous bug and implementing L2 decay, dropout, and layer dropout, we were able to achieve our best results in QANet-final with a dev F1-score of 64.5, improving the baseline dev F1-score by about 6.5 points. QANet-final also received a test F1-score of 62.57 on the non-PCE leaderboard.

We expected our QANet-final to perform better than the baseline because QANet included learned character embeddings and convolutional encoder blocks with self-attention. The character embeddings provide a finer grain representation of each word and can thus allow the model to train better. The convolutional encoder blocks with self-attention are better at capturing the information from the entire context paragraph compared to your typical RNN with attention. A RNN goes through word embeddings sequentially and can have a hard time retaining information from parts earlier in the context paragraph due to its sequential nature. That is why we expected the convolutional encoder blocks to capture the context paragraph information better and thereby have a larger F1-score and EM-score.

6 Analysis

We suspect QANet-v1 did worse than the baseline because we had not yet implemented the encoder block layers. However, after implementing them in QANet-v2, we did not receive a sizable F1-score improvement. This was due to a bug in our encoder blocks. In the encoder block we mis-implemented the separable depthwise convolutions and forgot a ReLU activation. We also had a bug in our

self-attention layer where we masked the attention maps incorrectly. This shows that self-attention is a key component the QANet model.

Self-attention allows every word in the input attend to every other word. Therefore, by masking the attention maps incorrectly, we essentially placed greater emphasis on non word elements which lead to worse performance. Thus, after fixing these bugs and implementing regularization techniques such as l2 decay, dropout, and layer dropout, we were able to achieve results that out performed the baseline BiDAF version.

		Ground Truth		
		Answer	No Answer	Total
Prediction	Answer	2410	1203	3613
	No Answer	438	1900	2338
	Total	2848	3103	5951

Model	TPR	TNR	FPR	FNR
QANet-final	84.62%	61.23%	38.77%	15.38%

Table 2: Confusion matrix of the QANet-final model's predictions on the SQuAD 2.0 dev set. Below the matrix are the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR) associated with the confusion matrix.

Table 2 shows the confusion matrix for out QANet-final model. We see that our model was more likely to predict an answer than a no-answer, predicting an answer a total of 3613 times, while predicting a no-answer 2338 times. In addition, we see that the true positive rate (TPR) is greater than the true negative rate (TNR) with percentages of 84.62% versus 61.23%, indicating that our model is slightly worse at predicting a no-answer when it should compared to predicting an answer when it should. This notion of predicting answers more often than no answer is further seen in false positive rate (FPR) and false negative rate (FNR). The FPR is more than double the FNR showing that the model is more biased towards predicting an answer versus predicting a no-answer.

7 Conclusion

In this paper, we implemented the QANet model as described Wei et al's paper [1]. As a result, we achieved great results on the SQuAD 2.0 dataset. Our final QANet model received an F1-score of 64.5 on the dev set and 62.57 on the test set. Our dev F1-score outperformed the Baseline dev score by about 6 points. We also found that despite the dev set having more unanswerable questions than answerable ones, our model predicted an answer more often than no-answer as indicated by the TPR being higher than the TNR.

Some limitations to our work is the fact that we only test our model on the SQuAD 2.0 dataset. We do not know if the QANet model will produce promising results on other Q&A datasets or perform well on other tasks. Furthermore, we did not extend the QANet model at all. Some future work that we can do is to incorporate ensemble methods. Ensemble methods has shown to increase performance effectively. This is done by combining multiple 'weaker' learners to build a stronger learn for the task. For instance, we could potentially average start/end probabilities that each model outputs in order to determine the final start and end tokens. We could also implore majority voting whether the most popular start and end tokens amongst the models will be the answer for the predicted span.

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. *CoRR*, abs/1804.09541, 2018.

[2] Francesca Alloatti, Luigi Di Caro, and Gianpiero Sportell. eal life application of a question answering system using bert language model. *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 250–253, 2019.

[3] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2018.

[4] Jonathan Raiman and John Miller. Globally normalized reader. *CoRR*, abs/1709.02828, 2017.

[5] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Fastqa: A simple and efficient neural architecture for question answering. *CoRR*, abs/1703.04816, 2017.

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

[7] 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.

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