Question Answering on SQuAD

Conor McAvity
Department of Computer Science
Stanford University
cmcavity@stanford.edu

Abstract

The Stanford Question Answering Dataset (SQuAD) is a task for machine reading comprehension. An updated version of the task was recently released, SQuAD 2.0, which adds unanswerable questions to the original dataset. In this paper, I present an implementation of the QANet model [6] for SQuAD 2.0. On the hidden test set, the model obtained an F1 score of 66.9 and an EM score of 63.3.

1 Introduction

Reading comprehension is a popular NLP task which works as follows: a model is presented with a passage of text and a question about that text, and needs to predict the answer, which is in the form of a span of words from the text. The Stanford Question Answering Dataset (SQuAD) is a dataset for reading comprehension released by the Stanford NLP group. The original version, SQuAD 1.1, contained around 100,000 crowd-sourced questions written about Wikipedia passages [3]. In October 2018, the top model surpassed the human benchmark on the public leaderboard. As a result, an updated version of the challenge, SQuAD 2.0, was released, which adds 50,000 new questions to the original dataset [2]. These new questions pose the additional challenge that they are unanswerable, and are written to resemble answerable questions. In order for a model to perform well on the new challenge, it must be able to reliably determine from the context when a question has no answer.

In this project, I implement a version of the QANet model, an end-to-end reading comprehension model originally written for the SQuAD 1.1 challenge. In contrast to many other similar reading comprehension systems, QANet does not contain any recurrent layers in its architecture. Rather, it makes extensive use of self-attention as well as convolutional neural networks.

2 Related Work

There have been a variety of different end-to-end neural network models created for the SQuAD challenge. The Bidirectional Attention Flow model (BiDAF) was an early successful approach on the original SQuAD 1.1 dataset. The central contribution of this model is the attention flow layer, which not only computes attention from the context sequence to the query sequence, but from query to context as well. More recently, models that use pre-trained contextual embeddings have proven to perform especially well at the SQuAD challenge. In particular, the Bidirectional Encoder Representations from Transformers (BERT) model [1], which pre-trains language representations with a bidirectional Transformer, is a component in all of the top-performing models on the SQuAD 2.0 leaderboard, including the state of the art.

3 Approach

3.1 Baseline model

The baseline model is a version of the BiDAF model, but which omits the character-level embedding layer used in the original model and only includes a word-level embedding layer. For further details on the BiDAF model, refer to the paper [4].

3.2 QANet

QANet is an end-to-end model for question answering. Its high level structure is similar to that of other reading comprehension models of the time, such as BiDAF: it consists of an embedding layer and an encoding layer applied separately to the context and question sequences, a bidirectional attention layer between the context and question sequences, a further encoding layer, and a softmax output layer. The novel feature of this model is that unlike BiDAF, it does not make any use of recurrent neural networks in its encoding layers. The motivation for this design choice is to allow the model to better leverage parallelization. Instead of RNNs, its encoding architecture uses CNN sublayers, as well as a multi-head self-attention sublayer, drawing inspiration from the Transformer architecture [5]. The paper points out that the combination of CNNs and self-attention should allow the model to examine both local and global interactions within a sequence of words.

Like BiDAF, the original QANet paper was written for the SQuAD 1.1 dataset, which does not include examples labelled with "NoAnswer" introduced in SQuAD 2.0. To adapt the model to the new dataset, an "OOV" token is added to the beginning of each context sequence. A test time, the model's output is taken to be "NoAnswer" if it assigns a higher probability for start and end positions to this "OOV" token than to any other span.

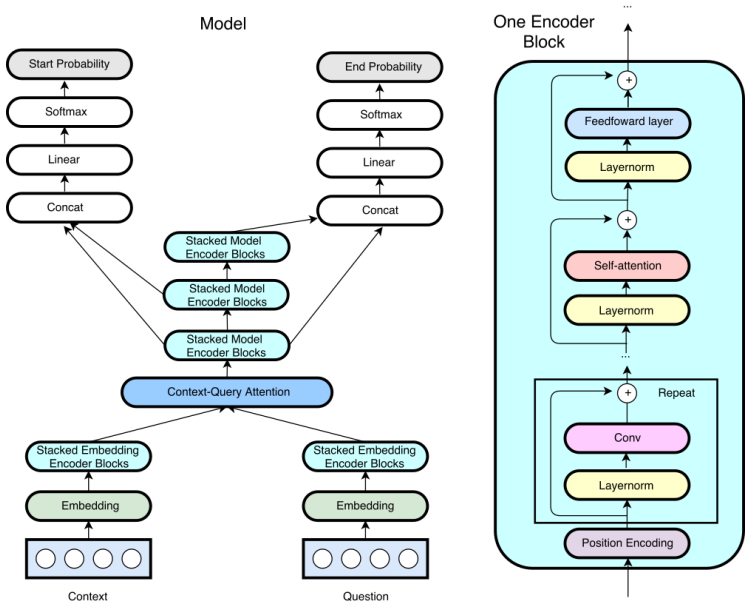

Figure 1: An overview of the QANet architecture

3.2.1 Embedding layer

The embedding layer is similar to its counterpart in the baseline BiDAF model. For each word w in the context or question sequence, it obtains the embedding by concatenating a 300-dimensional word-level GloVe embedding and a character-level embedding. In the character-level embedding layer, a randomly initialized character vector is looked up for each character c in w. A 1-dimensional convolutional neural network is applied to the resulting matrix, and max-pooling is applied to yield the embedding. After concatenating the two embeddings for a word, the resulting vector x is passed

through a two-layer highway network. To implement this layer, I adapted the baseline embedding layer for QANet by adding in the character-level embedding layer.

3.2.2 Embedding encoder layer

The encoder layer architecture is one of the main contributions of the QANet paper. It consists of a sinusoidal positional encoding followed by a stack of Encoder blocks. Like the Transformer encoder, the Encoder block uses a sublayer of multi-head self-attention followed by a feedforward sublayer [5], but it also precedes the self-attention sublayer with a variable number of depthwise-separable CNNs. Within each block, the sublayers are connected by layer norm and residual connections: specifically, the output of each sublayer is $x + \text{sublayer}(\text{layernorm}(x))$. This encoder is applied to the output of the embedding layer for both the question and context sequences.

I implemented most of the encoder architecture myself, but for the positional encoding[1] and self-attention sublayer[2], I adapted code from open-source PyTorch implementations of the Transformer architecture.

3.2.3 Bidirectional attention layer

The bidirectional attention layer is the same as its counterpart in the baseline BiDAF model. Briefly, this layer computes a similarity matrix between pair of elements in the context and question sequences according to the equation $S_{ij} = W[c_i; q_j; c_i \circ q_j]$ for parameter matrix W, where $c \circ q$ denotes elementwise product. Then, context-to-query attention is computed using the row-wise softmax of S, and query-to-context attention is computed using the column-wise softmax of S. The output is a sequence of representations $[c; a; c \circ a; c \circ b]$, where c is the input representation of a context word, a is a context-to-query output, and b is a query-to-context output.

3.2.4 Model encoder layer

The model encoder layer uses the same encoder architecture as the embedding encoder layer. First, a projection is taken to reduce the size of the output of the bidirectional attention layer from 4 times the hidden size down to the hidden size. Then 3 stacked encoder layers are applied to the sequence, yielding outputs M_0, M_1 and M_2.

3.2.5 Output layer

The output layer is similar to the output layer in the baseline BiDAF model. The 3 outputs of the model encoder layer are used to compute probability distributions of the start and end positions of the answer in the context, according to the following equations: $p_{\text{start}} = \text{softmax}(W_{\text{start}}[M_0; M_1])$ and $p_{\text{end}} = \text{softmax}(W_{\text{end}}[M_0; M_2])$, for parameter matrices W_{start} and W_{end}.

3.2.6 Loss function

The loss function is the sum of the negative log likelihood loss for the start and end probability distributions, averaged over all the examples in a batch. Specifically, the loss for each example is $-(\log p_{start}(i) + \log p_{end}(j))$, where i and j respectively denote the ground truth start and end indices.

3.2.7 Predicton

At test time, the predicted span (i, j) is chosen to maximize the probability of the product $p_{start}(i) \cdot p_{end}(j)$, subject to the condition $i \leq j$.

[1] http://nlp.seas.harvard.edu/2018/04/03/attention.html

[2] https://github.com/jadore801120/attention-is-all-you-need-pytorch/tree/master/transformer

4 Experiments

4.1 Dataset

I used the SQuAD 2.0 dataset for training and developing my models. The training set consists of 130,000 examples, and the dev set consists of 6000 examples. The maximum context length was set to 400 words, the maximum question length set to 50 words, and the maxmimum answer length set to 30 words. The maximum number of characters kept in a word was set to 16. Histogram of the original context and question lengths are shown in Figure 2.

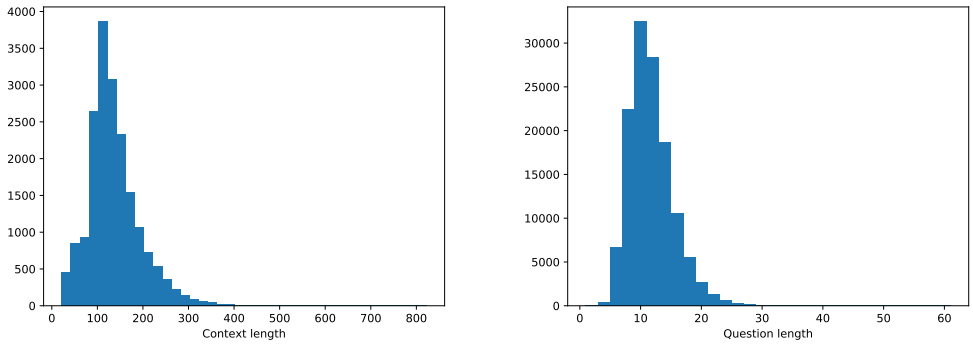

Figure 2: Histograms for context and question lengths

4.2 Evaluation method

The main evaluation metrics are the F1 and EM scores, based on 3 provided reference answers for each example. The EM (exact match) score is the percentage of test examples for which the model matches one of the reference translations. The F1 score for a single example is the harmonic mean of precison and recall when considering the model output and the reference translations as bags of words, and the final F1 score is the average of the individual scores.

4.3 Implementation details

For the most part, I followed the paper in selecting hyperparameters. I used a hidden size of 128 throughout the model. For the character-embedding layer, I used a kernel size of 5 and 200 output channels. For the embedding encoder layer I used 1 Encoder block with 4 CNN layers each with kernel size of 7, and for the model encoder layer, 7 Encoder blocks with 2 CNN layers each. I shared weights between applications of the embedding encoder layer to the question and context sequences, and also shared weights between each of the three applications of the model encoding layer. In each of the self-attention sublayers, I used 8 heads.

Following the paper, I used the Adam optimizer, with learning rate 0.001, $\beta_1 = 0.8$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. I used an exponential moving average (EMA) of the parameters with value 0.9999. For regularization, I used L2 weight decay with $\lambda = 3 \cdot 10^{-7}$. I also used dropout after every sublayer with value 0.1, and stochastic depth dropout within each encoding layer, with the probability of a sublayer l surviving equal to $1 - (l/L)(0.1)$, L being the total depth. Due to memory constraints, I used a batch size of 10 instead of 32 as used in the paper. Training time was approximately 90 minutes per epoch.

4.4 Other experiments

I also tried several other ideas that didn't end up working. One experiment was to increase the number of Encoder blocks in the model encoder layer from 7 to 9, but this did not improve performance, and was more computationally expensive. I also tried halving the learning rate when the dev NLL began to increase, but this didn't lead to noticeably different performance either.

4.5 Results

On the hidden test set, my QANet implementation obtained an F1 score of 66.9 and an EM score of 63.3, which was competitive on the non-PCE class leaderboard. The performance of this model on the dev set compared with two other models that I trained, the baseline BiDAF and improved baseline model, can be seen in Table 1.

Model	Dev Set EM	Dev Set F1
Baseline BiDAF without char-embedding	57.3	60.7
BiDAF with char-embedding	61.0	64.4
QANet	67.3	70.7

Table 1: F1/EM scores on the dev set

As can be seen, the QANet implementation outperformed the improved baseline BiDAF implementation. This was an expected outcome, because QANet was one of the top performing models on the original SQuAD 1.1 challenge. These results indicate that QANet can also obtain reasonable performance when adapted for SQuAD 2.0.

5 Analysis

5.1 Subsets of the data

In Table 2, we see the performance of the model on different subsets of the questions. The model achieves EM and F1 scores above those on the full dataset for questions beginning with "when" and "who". In particular, performance on "when" questions is dramatically higher, with increases of 7.0 EM and 4.6 F1. This likely indicates that the model is very good at picking out years from the context, as well as being quite good at identifying named entities. On the other hand, the model achieves a low EM score on questions staring with "why", despite achieving a reasonable F1 score. An explanation for this could be that when answering higher-level questions concerning cause and effect, the model is able to identify the general part of the context in which the answer lies, but is incapable of cutting out irrelevant information and honing in correctly on the precise location of the answer.

Question type	Count	Dev Set EM	Dev Set F1
When	440	74.3	75.3
Who	601	69.2	71.9
What	2759	67.3	70.6
Where	231	67.1	71.3
How	525	66.7	70.1
Which	146	63.7	68.4
Why	84	57.1	70.7

Table 2: Dev set F1/EM scores by question type

5.2 Selected outputs

An examination of some sample outputs for the model indicates that while it is largely capable of providing answers that are reasonable in a syntactic sense, it sometimes fails to understand basic semantic relationships between words. In some cases this leads to the model answering questions that have no answer. For instance:

Context: Tymnet was an international data communications network headquartered in San Jose, CA.
Question: Tymnet worked with what?
Answer: N/A
Prediction: data communications network

While the model seems to understand that there is a relationship between "Tymnet" and

"data communications netework", and that the question is looking for something with a relationship to "Tymnet", it fails to distinguish between the semantically distinct relationships expressed by the verbs "was" and "worked with".

However, in some cases, it is debatable whether questions are correctly labelled as unanswerable. For instance:

Context: Thus, on 1 August 1944, as the Red Army was nearing the city, the Warsaw Uprising began. The armed struggle, planned to last 48 hours, was partially successful, however it went on for 63 days.
Question: How many days did the Red Army Warsaw Uprising last?
Answer: N/A
Prediction: 63

Despite the example being labelled as unanswerable, it seems clear from the context that the Warsaw Uprising lasted for 63 days, which is what the model output.

5.3 Ablation analysis

As can be seen in Table 1, the addition of the character-level embedding layer to the baseline BiDAF model resulted in an increase of 3.7 EM and 3.7 F1. This big jump in performance is likely due to the character-level embedding layer giving the model the ability to more easily associate words with similar spellings. This might allow the model to more easily pick out words of the context that share sub-components with words in the question, which could be useful in locating the answer span.

I also tried replacing Encoder block in the embedding encoder layer with LSTM-based encoder from the baseline BiDAF model. This resulted in a relative decrease of 1.9 EM and 2.2 F1 after 1M steps, and it was also slightly more computationally expensive.

6 Conclusion

In this project, I implemented a version of the QANet model for the SQuAD 2.0 challenge. The model improved on the baseline EM and F1 scores, and achieved a competitive performance among the top models on non-PCE class leaderboard. The project demonstrates that the QANet model, which was originally written for SQuAD 1.1, is also capable of reasonable performance when adapted for SQuAD 2.0.

Since the model was quite computationally expensive to train, it was difficult to do extensive hyperparameter search in the available time. In future work, it could be interesting to experiment with changing the number and arrangement of sublayers in the Encoder blocks, and with using varying amounts of dropout in different parts of the model.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[6] 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. *arXiv preprint arXiv:1804.09541*, 2018.