BERTNet

Hongtao Sun
s3sunht@stanford.edu

Brett Szalapski
brettski@stanford.edu

Yang Wang
leonwy12@stanford.edu

Abstract

The QANet architecture achieved state-of-art performance on the SQuaD 1.1 Q&A challenge with only attention and convolution structures. The BERT model, which creates pre-trained word-piece embeddings and requires simply fine-tuning task-specific output layers, also achieved state-of-art performance on SQuAD 1.1 and 2.0. For this exploration, these two powerful models are combined to achieve higher performance than either model did alone, with query and context embeddings from BERT replacing the encoding layers of QANet. Finally, using just the Context-Query Attention layer from BiDAF on top of BERT-large, an F1 score of 80.23 was achieved on the test set.

1 Introduction

The Question-Answering task (Q&A) is one of the most important Natural Language Processing challenges of modern machine learning. In addition to being a proxy for how well a computer is capable of understanding text, it can massively improve the response-time and user-experience of information retrieval. The goal of a Q&A system is to take in a context paragraph or document, receive a question about the context, and answer the question by selecting a span from the context. For the development work in this paper, the SQuAD 2.0 data set created by Rajpurkar et al. is used [Rajpurkar et al., 2018], slightly modified for the purposes of evaluating in the CS224N class setting. Each row of the data consists of one context and one question, though it is easy to see how the usefulness of this task could be extended to general information retrieval across many documents.

A recent trend on the SQuAD 2.0 leader board is the use of models with pre-trained contextual embeddings (PCE) —rather than fixed embeddings such as GLoVe —combined with other successful Q&A models. Among these entries are "BERT + MMFT + ADA", "BERT + N-Gram Masking + Synthetic Self-Training", and "PAML + BERT" [Rajpurkar and Jia, 2019]. However, one of the most successful models from the SQuAD 1.1 challenge, QANet, does not appear on the SQuAD 2.0 leader board either independently or in conjunction with a PCE. One of the strongest advantages of QANet is its ability to train much faster than other Q&A models [Yu et al., 2018], so the primary goal of the model discussed in this paper is to demonstrate that the QANet model can be used in conjunction with the powerful embeddings of BERT to achieve competitive results on the SQuAD 2.0 data set.

The structure of the remainder of the paper is as follows: in Section 2 discuss related works; Section 3 details the relevant model architectures; Section 4 outlines the experiments carried out, including quantitative results; Section 5 discusses the quantitative and qualitative performance of the model; and Section 6 highlights potential future work and concludes this exploration.

2 Related Work

The architecture of the model described in this paper is a combination of two previous architectures, each known to perform very well on the SQuAD Q&A challenge. QANet [Yu et al., 2018], an architecture that achieved state-of-the-art results on SQuAD 1.1, is altered for use with the SQuAD 2.0 data set, which includes unanswerable questions that were not part of the 1.1 challenge. Underneath the QANet implementation, BERT [Devlin et al., 2018] is used to obtain contextual embeddings for

the passages and associated questions. It is believed that the combination of these two models can produce state-of-the-art results, above and beyond either models' capabilities in isolation. These models are described next, followed by the specific modifications that allow them to be combined for use with the SQuAD 2.0 challenge.

2.1 QANet

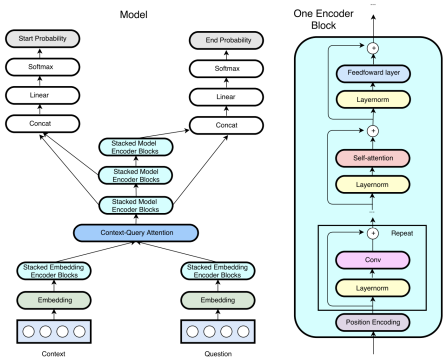


Figure 1: Original QANet structure

Similar to many other Q&A models, QANet uses five major components: an embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer, and an output layer. The major difference is that within each of these layers, only convolution and self-attention mechanisms are applied without recurrent structures. The architecture, as shown in figure 1, consists of 5 layers:

The **Input Embedding Layer**, in the original implementation of QANet, used the fixed, 300-dimensional GloVe word embeddings. This implementation will use the output of BERT for its embeddings —more on this shortly.

The **Embedding Encoder Layer** consists of a convolution layer, a self-attention layer, and a feed-forward-layer. The convolution layer consists of depth-wise separable convolutions for memory efficiency. The self-attention layer uses the multi-headed attention structure, governed by the following equations:

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \quad (1)$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2)$$

and Q, K, and V are the query, key, and value matrices [Vaswani et al., 2017]. With this attention, positional embeddings are encoded into QANet as each word in the input is compared to all of the other input words [Yu et al., 2018].

In the **Context-Query Attention Layer**, QANet also uses a standard technique for context-query attention, a context-query similarity matrix. The similarity function used is

$$f(q, c) = W_0[q, c, q \odot c] \quad (3)$$

where W_0 is a trainable parameter.

In the **Model Encoder Layer**, the inputs are concatenated from the rows of attention matrices A and B in the following fashion: $[c, a, c \odot a, c \odot b]$. The other layer parameters are similar to the Embedding Encoder Layer.

The **Output Layer** is task specific. The strategy adopted in this paper for the SQuAD 1.1 dataset was to predict the probability of each position in the context being the start or end of an answer span Seo et al. [2016].

For the implementation of this paper's architecture, QANet is recreated from scratch according to the architecture described in Yu et al. [2018].

2.2 BERT

BERT is a "multi-layer bidirectional Transformer encoder" [Devlin et al., 2018] that draws on work from Vaswani et al. [2017]. Using word-pieces (e.g. "playing" becomes "play" and "##ing") to form pairs of sentences, BERT is used to pre-train contextual word embeddings which can then be fed into task-specific architectures. The embedding scheme is a concatenation of three parts: token embeddings, segment embeddings (e.g. Question or Context segment), and position embeddings (see Figure 2). The embeddings are trained on two unsupervised prediction tasks: Masked Language Modeling (MLM) and Next Sentence Prediction. MLM involves masking out a word at random

and having the model predict the word that should be behind the mask. This helps prevent word fromsattending to themselves, a common pitfall of the bidirectional structure of the model. Next Sentence Prediction involves feeding the model two sentences from a large corpus. Half of the time, the sentences are sequential, and half of the time, the second sentence is a random sentence from the corpus. The model attempts to predict whether sentence two follows sentence one or not.

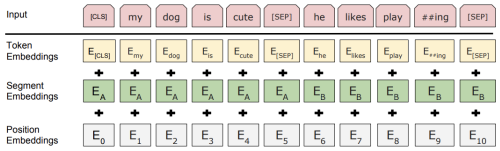


Figure 2: BERT Embedding Structure

Once the embedding weights have been trained, they can be used in a variety of language tasks, including SQuAD Q&A. Upon its release, BERT surpassed the then-state-of-the-art models by a wide margin on SQuAD 1.1. It has since been used both independently and in conjunction with a variety of architectures on SQuAD 2.0 as well, as seen on the SQuAD 2.0 leaderboard. To do so, the pre-trained embeddings are fed into, at a minimum, a task-specific output layer. The BERT weights can be trained on the task-specific data along with the output layer, or can be frozen to tune only the output layer itself. For the implementation of this paper's architecture, the Wolf et al. [2019] Huggingface code base is used.

3 Approach

In order to combine the two models outlined in Section 2, some modifications were needed. In addition, a subset of the layers from QANet, the Context-Query Attention layer, was used as the output layer for BERT. Next, the hybridization scheme is described, followe by a simpler hybrid model.

3.1 Hybrid Model

In order to combine these two architectures, some modifications must be made. First of all, as can be seen in figure 1), QANet expects to receive two separate blocks of batched inputs: the batch of queries, Q, and the batch of context paragraphs, C. However, the output of BERT is an embedding of the form

`[CLS]qqq...qqq[SEP]ccc...cccc<PAD>...<PAD>`

for a single example (see Figure 2). However, QANet was designed to take the query and context as separate blocks. Several different methods of resolving this difference were explored. The results of these methods can be seen in Table 1 found in Section 4.3:

- **QANet w/ BERT-small, padding between C/Q** (*Not included in the results table due to poor performance*): Individually split question from context after obtaining the BERT embeddings and pad both to appropriate matching lengths before feeding them into QANet. Zeros are used for padding, without applying BERT contextual embeddings to these tokens.
- **QANet w/ BERT-small, padding between C/Q**: Pad each question to a pre-determined maximum length before the BERT structure. This allows for easy separation of the question and context sequences after applying BERT to obtain the contextual embeddings. This resolves the concern of missing BERT embeddings on the pad tokens.
- **QANet w/ Sep. BERT for Query and Context**: Similar to the above approach, put the pre-padded, separate question and context blocks through two separate BERT models to obtain prior to feeding into QANet. The drawback of this approach is that the contextual word embeddings are not shared between question and context blocks.
- **QANet w/ BERT-{small, large}**: Use two copies of the original post-BERT sequence for both the question and context blocks of QANet, but with masking to view only the question or context as appropriate.

Ultimately, the final option was selected, so the question and context blocks in QANet are each given a copy of the full BERT embedding along with masks that isolate the question and the context appropriately. The resulting architecture is shown in Figure 3.

Additionally, because QANet was originally implemented on SQuAD 1.1, which always had an answer, the model must be modified to enable predicting No Answer. To supply a No Answer, the start and end positions are set to 0, predicting the <CLS> token. This has proven to allow the model make No Answer predictions.

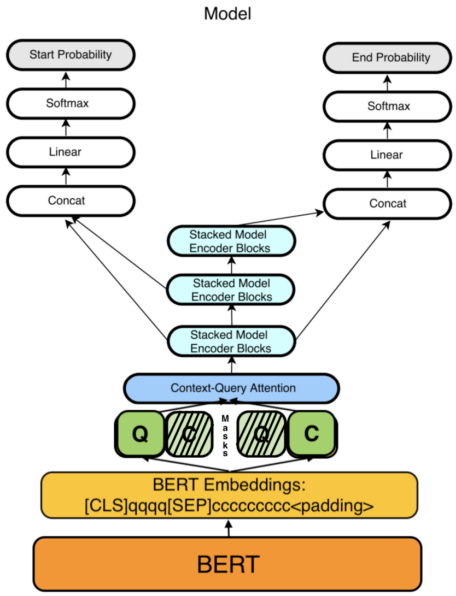


Figure 3: QANet + BERT Architecture

3.2 BERT with Context-Query Attention

Because BERT is a pre-trained contextual embedding, it has already performed some of the tasks included in the layers of QANet, such as the stacked model encoder blocks. As such, a simpler model was implemented using only the context-query attention layer from BiDAF followed by a linear output and softmax layer —CQ-BERT. This architecture significantly decreases model complexity and improves training speed. The result of this modification also proves superior to the entire QANet structure, most likely due to less over-fitting over the training data. The BiDAF attention module from Chute [2019] provides a memory-efficient implementation for a context-query attention layer.

4 Experiments

4.1 Data and Evaluation

The dataset is the SQuAD 2.0 labeled dataset,
modified slightly for the purposes of the Stanford course CS224n so that validation data can be used for testing. It consists of paragraphs and associated questions along with answers to these questions (or No Answer, as appropriate). The sample test-set, drawn from the production validation data, is augmented with additional hand-crafted examples. This data requires no pre-processing and consists of over 150,000 questions, both answerable and unanswerable.

The typical metrics applied to SQuAD performance are used for evaluation: F1 and EM Scores.

4.2 Experimental Details

Three different single-model experiments were run so as to establish appropriate baselines: QANet, BERT-small, and BERT-large. QANet used the framework code from Chute [2019], while the BERT models used the Wolf et al. [2019] base code. Both large and small used a single linear output layer to adapt them to the Q&A task.

Next, two variations on BERT-small were trained to gain insight into working with BERT and expand on the base model. Neither outperformed the baseline. One variation built on the BERT baseline model with the last two embedding layers concatenated to form each word's embedding. The other was the baseline model with the last three embedding layers concatenated, as well as two output hidden layers with ReLU activations and dropout applied. Dropout rate was set to 0.1.

Following these initial experiments, the combined architecture using QANet and BERT was run, using the QANet implementation from Hackiey [2019], replacing the Context-Query Attention layer with the memory-efficient version from Chute [2019]. QANet experiments were carried out using both BERT-small and BERT-large. In addition, the padding methods outlined in Section 3.1 were

each carried out in conjunction with BERT-small. Dropout was tuned at values of 0.1, 0.2, and 0.3, with optimal performance at 0.1.

Finally, the Context-Query Attention output layer described in Section 3.2 was trained along with BERT-small and BERT-large. The dropout hyperparameter was tuned with values of 0.1, 0.2, and 0.3, with optimal performance at a value of 0.2.

The results of all of these models are summarized in Table 1.

Each of the models using BERT-small, excluding QANet, was trained on a P100 for two to three epochs at a rate of 1.6 hours per epoch. The QANet + BERT-small model was trained on a P100 at four hours per epoch and achieved optimal performance at two epochs.

BERT-large models were trained on a V100. Non-QANet models trained at a rate of 1.5 hours per epoch and achieved optimal performance around two to three epochs. QANet + BERT-large took 4 hours per epoch and achieved optimal performance at two epochs.

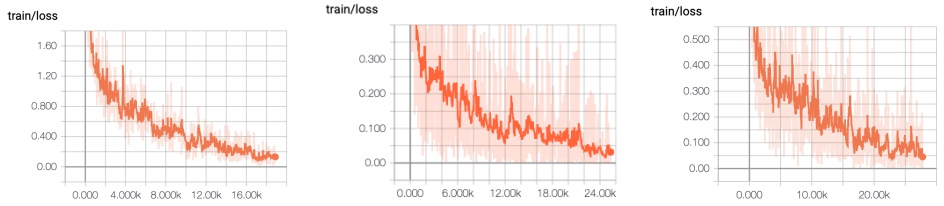
4.3 Results

Table 1: Full Results

Model	Dev			Test	
	EM	F1	AVNA	EM	F1
QANet	57.44	60.97	68.22	x	x
BERT-small	72.97	76.41	80.40	x	x
BERT-small, 2 embedding layers	72.06	75.54	79.62	x	x
BERT-small, 3 embedding layers, 2 output layers	68.28	72.07	76.18	x	x
BERT-small	72.97	76.41	80.40	x	x
QANet w/ sep. BERT for Query and Context	35.64	36.58	61.30	x	x
QANet w/ BERT-small, padding between C/Q	43.58	44.43	70.80	x	x
QANet w/ BERT-small	**74.50**	77.34	80.49	x	x
CQ-BERT-small	74.38	**77.98**	81.67	x	x
BERT-large	**78.89**	**82.18**	85.29	**77.38**	**81.10**
CQ-BERT-large, Dropout = 0.1	77.90	80.81	83.81	77.31	80.23
CQ-BERT-large, Dropout = 0.2	77.90	81.01	84.17	x	x
QANet w/ BERT-large	77.34	80.21	83.46	73.76	76.96

CQ-BERT and QANet + BERT narrowly outperformed BERT-small, though the results for QANet were not quite as strong as expected. This is attributable to two factors. First, the propensity for over-fitting. Between the embedding parameters in BERT and the number of parameters included in the QANet module, this architecture has a massive amount of representation power. Furthermore, this exploration did not recreate the language-translation data augmentation carried out by the original QANet paper Yu et al. [2018], in which Wei Yu et. al. used a Neural Machine Translation to translate the data from English to a foreign language, and then back to English, resulting in paraphrased versions of the original context and query. This paraphrase could then be used as an additional example, increasing the size of the training data. The strong results from CQ-BERT were a surprise, demonstrating the power of that individual layer.

Though the models surpassed the BERT-small benchmark, they were unable to outperform vanilla BERT-large. This is primarily due to lack of time and training resources with which to tune hyperparameters, as well as the issues mentioned above for QANet. The volatility of the loss curves in Figures 4b and 4c are a strong indicator that these models would have benefited from further hyperparameter tuning.



(a) BERT-large (b) QANet + BERT-large (c) C-Q Attention w/ BERT-large

Figure 4: Training Loss Plots for Different Models

5 Analysis

5.1 Different Methods to Split Context and Query

Three different methods were used to split the context from the query to make the output of BERT compatible with the inputs of QANet. As previously discussed, it was necessary to successfully separate the query from the context in order to have fixed-length inputs for the context and query batches fed into QANet. Adding additional padding and training two separate BERT models produced unsatisfactory performance. Ultimately, providing two copies from the output of BERT along with appropriate masks resulted in the best performance.

The first method, which operated on the data after it had passed through the BERT module, padding zeros between the context and the query, produced the worst results with an F1-score of 44.43. This is because the BERT model was trained on contiguous texts with padding after the context. Artificially adding zeros in between the context and the query does not allow for the contextual embedding scheme that gives BERT its success.

A second method explored for splitting context and query was to train two BERT models separately, one each for query embeddings and context embeddings. This method was the most similar to the original QANet structure, which encoded the query and the context separately. However, this approach resulted in a paltry F1-score of 36.58. The failure of this approach was due to lack of contextual representation shared between context and query, which is one of the strongest advantages of BERT. Because the embeddings for each were learned separately, the model was unable to reconcile the embeddings of the query and the context.

As discussed previously, the simplest method was the most effective. Passing the full BERT embedding to both the query and the context blocks of QANet, along with a mask to mask out the query and context where required, proved most successful. One additional change was required to allow for no-answer predictions —the [CLS] token was grouped with the context block, rather than the query block. In this way, the model was able to select the [CLS] span for a No Answer response.

5.2 Performance Analysis

Table 1 shows that the Context-Query Attention + BERT-small and QANet + BERT-small models both outperformed the baseline BERT-small. This vanilla BERT-small model turned out to be a difficult baseline to surpass according to both the leaderboard and discussion with peers.

The original BERT computes interactions between all words in the input. However, for Q&A, the interactions between context and query as separate groups may require more emphasis. Thus, the Context-Query attention layer from Seo et al. [2016] was adapted, with the flow of attention calculated in both directions. With this succinct modification, model performance increased over BERT-small from F1 of 76.41 to 77.98. With the additional layers from the QANet module added on top of the Context-Query attention layer, the F1 was still higher than that of BERT-small, at 77.34, though

not as high as BERT with the Context-Query attention layer. As discussed in section 4.3, this may have promoted overfitting, hurting development and test set performance. However, with further hyperparameter tuning, and especially with the data augmentation techniques used by the original QANet implementation, the team believes that these models could be further improved.

In contrast to the results over the BERT-small baseline, the BERT-large baseline was unbeatable. Again, with more resources to perform further hyperparameter tuning and the additional implementation of the data augmentation technique for QANet, it is likely that this result could be overturned.

5.3 Other Observations

One last observation drawn from analyzing missed examples is that C-Q Attention BERT is more likely to predict an answer when it should not, whereas BERT-large tends towards errant No Answer predictions. Further analysis of this trend is left to future work.

6 Conclusion

In summary, QANet and BERT can be combined to achieve near-state-of-the-art results on the SQuAD 2.0 Q&A challenge, although using Context-Query Attention as the output layer for BERT may be even more powerful. With an F1-score of 80.23, this model places 5th on the CS224N 2019 Winter class leader board. A couple of avenues exist for expanding on this work, including a more thorough exploration of possible hyperparameters used, especially for the models that include BERT-large, as well as augmenting the QANet data set according to the translation paraphrasing performed by the original QANet implementation used on the SQuAD 1.1 challenge.

References

Chris Chute. Starter code for stanford cs224n default final project on squad 2.0. https://github.com/chrischute/squad, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Hackiey. A pytorch implementation of qanet. https://github.com/hackiey/QAnet-pytorch, 2019.

Pranav Rajpurkar and Robin Jia. Squad2.0 the stanford question answering dataset. https://github.com/huggingface/pytorch-pretrained-BERT, March 2019.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018. URL http://arxiv.org/abs/1806.03822.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. URL http://arxiv.org/abs/1611.01603.

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. URL http://arxiv.org/abs/1706.03762.

Thomas Wolf, Victor Sanh, and Gregory Chatel et al. Pytorch pretrained bert: The big & extending repository of pretrained transformers. https://github.com/huggingface/pytorch-pretrained-BERT, 2019.

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. URL http://arxiv.org/abs/1804.09541.