Improving SQUAD 2.0 Performance using BERT + X

Danny Takeuchi
Stanford University
Department of Computer Science
dtakeuch@stanford.edu

Kevin Tran
Stanford University
Department of Computer Science
ktran23@stanford.edu

Abstract

Bidirectional Encoder Representations from Transformers (BERT) is the current state-of-the-art language model which pretrains deep bidirectional representations for a wide-range of tasks. We aim to finetune this model for the task of SQUAD 2.0 question-answering by incorporating various deep learning techniques. First, we utilized dot-product attention and Convolutional Neural Networks to generate overall representation layers. Next, we modified the existing loss function to penalize the BERT model for predicting an answer when there isn't one. Finally, we increased the complexity of BERT model to be more specific to the question-answering task by incorporating highway networks, additional transformer layers and Bidirectional Attention Flow (BiDAF). Although well-motivated, some of these approaches increase performance because they either introduced noise or overfitted the model. However, running the model through a highway network and tuning hyper-parameters did nonetheless generate small gains in performance.

1 Introduction

Natural Language Comprehension tasks have significantly risen in popularity over the past years due to its plethora of real-world applications. We are attempting to tackle the task of question-answering on the Squad 2.0 dataset, which is composed of over 100,000 questions on crowdsourced Wikipedia articles. In the Squad machine comprehension task, the machine would dissect the semantic meaning of the question and the context paragraph, and generate an answer that spans a portion of the context. Compared to Squad 1.1, the Squad 2.0 dataset also asks questions about contexts that don't contain the true answer. This provides scores that are more indicative of the machine's language comprehension.

In recent years, Bidirectional Encoder Representations from Transformers (BERT) implementations have made significant gains compared to previous state-of-the-art QA models. However, many of these BERT models are still fooled into answering questions about contexts that don't contain the answer. We are leveraging the BERT huggingface pytorch implementation and attempting to improve upon this baseline model, especially with false positives where a question has no answer but we predict one anyway. We have adjusted BERT's attention structure and loss function, and fine-tuned the model by running the contextual BERT embeddings through various layers before computing logits.

2 Related Work

We relied upon the original BERT paper published in October 11, 2018 for our original baseline[Devlin et al., 2018]. There are a few related works that influenced some of the approaches we decided to take to improve upon the existing BERT model. First, Bi-Directional Attention Flow (BIDAF) network is a hierarchical multi-stage architecture well-suited for question-answering [Seo et al., 2016]. Its main feature, the bi-directional attention flow layer consists of context to question

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

attention and question to context attention, which helps generate query-aware context representations. Since the BERT model is just a general language model, we hope that adding BiDAF on top of the existing BERT model will make our model more specific to question-answering. In addition, highway networks have been shown to be very useful as a gating mechanism to filtering out potentially irrelevant information [Srivastava et al., 2015]. We hope that by passing our final hidden layers into a highway network before computing logits, we can achieve increases in performance. Finally, 1d Convolutional Neural Networks have been used in the context of pooling together information from character embeddings to generate word embeddings [Kim et al., 2016]. We hypothesize that perhaps, following the same line of logic, we can generate overall representation layers by passing multiple hidden layers through a CNN.

3 Approaches

3.1 Baseline

We are using BERT as our baseline model architecture for the SQUAD 2.0 question-answering task. The BERT model consists of 12 stacked bi-directional encoder transformers, each containing a multi-head self-attention layer and a feed-forward layer [Delvin et al., 2018]. This BERT model already has pre-trained embeddings generated from masked language modeling and next sentence prediction, so we simply leveraged these embeddings to encode our input question and context. To adapt BERT to our question-answering task, we utilized the following structure as our input representation for each sequence:

['CLS', 'qw1', ... 'qwN', 'SEP', 'cw1', …., 'cwM', 'SEP']

For clarification purposes, 'CLS' represents a classification token, 'SEP' represents a separation token, 'qw' represents a question word, and 'cw' represents a context word. N represents the number of question words and M represents the number of context words. The first encoder transformer will take this representation and retrieve embeddings for each of the words.

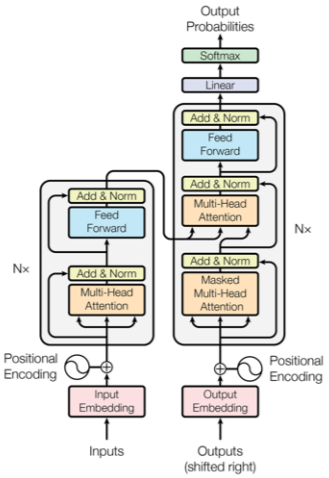

Figure 1: BERT Base

In each of the 12 transformers, multi-head attention is computed using key, query, and value vectors. Multi-head attention is defined as follows:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

The result of the multi-head attention is passed in through a single feed-forward layer and then into the next encoder. The inputs and outputs of each of the encoders are of size (batch-size, seq-len, emb-dim) except for the first encoder which takes in inputs of size (batch-size, seq-len) and outputs hidden states of size (batch-size, emb-dim).

After running the input representation through the entire BERT model, we perform fine-tuning by taking the set of final hidden states and running them through a linear layer to obtain start and end logits. Applying a soft-max over these start and end logits yields p-start and p-end which represent the probability distribution of our start/end index predictions.

3.2 BERT with Dot Product Attention over Hidden Layers (BERT + WHL)

In the current baseline BERT model, fine-tuning only consists of passing the final hidden states through a linear layer and a softmax function to obtain p-start and p-end. This is a reasonable approach because the final representation is likely to contain the most information about our words in relation to each other. However, by only using the final hidden state, we are potentially missing additional information encoded in prior hidden states. In the BERT + WHL model, we aim to combine hidden states from previous time steps by weighting them to generate an overall representation that

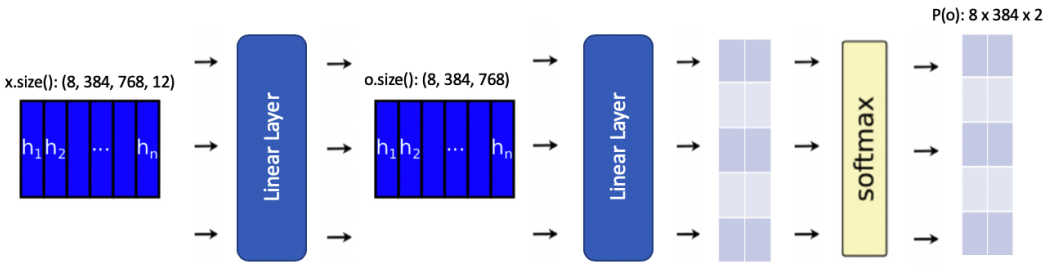

Figure 2: BERT + WHL

may be able to capture additional semantic and positional patterns. More concretely, we concatenate the last x hidden states into a matrix of size (batch-size, input-len, emb-dim, x). Then, we pass this matrix through a linear layer which will essentially weight the hidden states and output a tensor of size (batch-size, input-len, emb-dim) that contains the overall representation. The linear layer applies the transformation y = xAT + b to our tensor where A is matrix of weights for each hidden state and b is the bias term. We believe that using an overall representation that incorporates information from multiple hidden layers will boost performance.

3.3 BERT with CNN (BERT + CNN)

In this model, we attempted to achieve a similar goal as in the BERT + WHL model, which is to incorporate information from multiple hidden layers to generate an overall representation layer. However, instead of using attention to weight the various hidden layers, we utilized a 1-dimensional Convolutional Neural Network to pool information from multiple hidden layers. More concretely, we first concatenated the last four hidden layers of the BERT model to generate a tensor of (batch-size, question-size + context-size + 3, hidden-size, 4). Then we ran this through tensor through a 1-dimensional CNN with a kernel size of 2, input size of hidden-size and output size of hidden-size to obtain our overall representation layers, which is a tensor of size (batch-size, question-size + context-size + 3, hidden-size). Ultimately, we expect for this approach to be able to capture additional dependencies and information between consecutive layers that the BERT + WHL model couldn't. This is because the 1d CNN utilizes sliding windows to pool together information between consecutive layers whereas the BERT + WHL approach simply weights the various hidden layers. We expect that using the overall representation layers we get from our BERT + CNN model to compute logits will improve performance.

3.4 BERT with Weighted Cross Entropy Loss (BERT + WCEL)

We used Cross Entropy Loss in our model because it excels in training multi-classification problems. However, we believed we could tune it to train better on our Squad dataset. The BERT model contains C classes where C = 348, the maximum word length of the context paragraph. At each training iteration, it compiles two tensors of mini-batch size B for each of the true start and end positions of the answer span. It also compiles the start logits and end logits, each of size (B, C). It then compares the true start and end positions for each context in the mini-batch with the predicted start and end logits. When the associated context has no answer, the true start and end positions are 0. We further penalize the loss for cases in which there is no answer, but the model predicts an answer by creating a $weight$ tensor of size C: [2, 1, 1, 1, ...]. The loss for the start and end positions are separately calculated using the following equation:

$$\text{loss}(x, class) = weight[class](-x[class] + \log(\textstyle\sum_j exp)(x[j]))$$

The start and end losses are then averaged to generate the final loss.

3.5 BERT + Highway

Another approach that we tried was running the final hidden states through a highway network before computing logits. Highway networks can help improve performance because they contain gating mechanisms that can filter out potentially irrelevant information.

In this model, we ran the final hidden states through a highway network using the following equations:

$$x_{proj} = ReLU(W_{proj}x + b_{proj})$$
$$x_{gate} = \sigma(W_{gate}x + b_{gate})$$

Next, we utilized a gate to combine the highway projection with the skip connection.

$$x_{highway} = x_{gate} \odot x_{proj} + (1 - x_{gate}) \odot x$$

And then finally, we passed the output of the highway network through a dropout layer and linear layer to obtain the overall representations of our hidden states, which is a tensor of size (batch-size, input-len, emb-dim)

3.6 BERT with additional Transformer Layers (BERT + Transformers)

The baseline BERT model was originally developed as a language model and not for a specific task. Therefore, we suspect that our baseline model may actually be underfitting. Therefore, in this BERT + Transformers model we introduced two additional transformer layers on top of the pre-trained BERT model to fine-tune the number of encoder transformers from 12 to 14 and increase its complexity.

3.7 BERT with BiDAF (BERT + BiDAF)

As previously mentioned, the baseline BERT model was originally developed as a language model and not for a specific task. Therefore, our baseline model may not be optimized for the question-answering task and may be underfitting. We hypothesize that running the BERT model through an architecture designed for such task may increase performance. In particular, we segment the final hidden states of BERT into question hidden states and context hidden states and feed these hidden states into the BiDAF model. We took out the Embedding and Highway layers of the BiDAF model because our BERT final hidden states are essentially contextual word embeddings, but the rest of the BiDAF model remains the same.

4 Experiments

4.1 Data

We use the Squad 2.0 question-answering dataset which is composed of over 100,000 questions on crowdsourced Wikipedia articles. The Squad 2.0 also contains question, context pairs that have no correct answer as opposed to the Squad 1.1 dataset. Our two main evaluation metrics are F1 score and Exact Match(EM). We also developed our own scripts to analyze additional metrics of EM without non-answers, percentage of non-answer false positives (FP), and percentage of non-answer false negatives (FN). An example of a false positive would be if an answer is predicted for a question, context pair that contains no valid answer. An example of a false negative would be if no answer is predicted for a question, context pair that contains a valid answer.

4.2 Parameters

Each of our models ran on NV6 virtual machines. We ran the bert-base-uncased model from the huggingface repository on the VM as a baseline. This model has 12 transformer encoders, a hidden size of 768, 12 attention heads, and 110M parameters. We weren't able to use bert-large-uncased because the NV12 VM didn't have enough memory, even with a batch size of 4. Meanwhile, bert-base-uncased boosted our training speed to 5 hours per epoch. We trained each of our models for 2 epochs to more efficiently test our model. We used a batch size of 6 since we were limited by the NV6 VM's memory capacity. By using a gradient accumulation with step-size two, we were able

to boost our model's performance by half a point. The learning rate that we used was 3e-5. The max-seq-length was 384 and the doc-stride was 128.

4.3 Additional Experiments

We first implemented each of our initial outlined approaches. Additionally, we implemented two other variations of BERT + WHL. In BERT + Last3WHL, we take only the final 3 hidden states and concatenate them into a matrix of size (batch-size, input-len, emb-dim, 3). In BERT + Last4WHL + ReLU, we concatenate the last 4 hidden states, run them through a linear layer, and then run the final tensor through a ReLU layer.

We also tried combining our two approaches into BERT + WHL + Highway by first taking every hidden state generated by BERT and concatenating them into a matrix of size (batch-size, input-len, emb-dim, num-hidden-states). This would run through a linear layer to generate our new BERT representations and then run through a highway network.

5 Results and Quantitative Analysis

Table 1: BERT PCE Scores

Model	EM	F1	EM(No non-answers)	FP	FN
BERT Baseline + Hyperparam. Tuning	73.288	76.376	73.68	26.70	13.06
BERT + Highway	73.001	76.035	74.45	27.81	12.64
BERT Baseline	72.606	75.843	73.60	27.17	13.06
BERT + WCEL	72.359	75.379	74.25	27.56	13.92
BERT + Last3WHL	66.535	69.402	71.83	32.26	19.48
BERT + Last4WHL + ReLU	65.729	68.613	70.32	28.35	24.98
BERT + BiDAF	65.093	67.697	68.29	29.53	25.32
BERT + Transformers	63.123	65.813	67.91	31.53	23.32
BERT + WHL	61.056	64.322	67.80	37.31	22.20
BERT + WHL + Highway	57.947	61.508	63.45	39.80	22.23
BERT + CNN	51.468	56.352	58.93	44.58	26.89

5.1 Ablation Studies on BERT + WHL

The BERT + WHL approaches performed significantly worse than the BERT Baseline. In order to understand why, we tried removing layers from our BERT + WHL model. First, we then condensed hidden layers from 12 to 3 and we saw a 5 point increase in EM and F1 scores. Next, we tried using just the last hidden layer and saw that removing WHL increased our EM and F1 scores by another 6 points. This is likely because the linear layer forced the BERT model to learn noise, since few hidden dependencies existed across hidden states. We had hoped that using multiple hidden states would encode more relevant information, but our results show that the last hidden state contains the vast majority of valuable information. A common error we saw when analyzing results from the BERT + WHL models is the correct answer being present in the BERT + WHL prediction, but the prediction span boundary is incorrect. Consider the following example:

Context: "This relationship eventually produced closer ties of blood through the marriage of Emma, sister of Duke Richard II of Normandy, and King Ethelred II of England."

Question: "Who was Emma's brother?"

Correct Answer: "Duke Richard III"

Bert Baseline Answer: "Duke Richard III"

BERT + WHL: "sister of Duke Richard II of Normandy"

The imprecision in the prediction span boundary is most likely due to the noise inserted by the first few hidden layers of the BERT model, which are not as representative as the final hidden layer because at those stages the model is only starting to narrow its prediction boundary.

5.2 CNN Analysis

The BERT + CNN model was by far our worst performing model. It had the worse EM, F1, FN, and FP scores. Originally, we hypothesized that applying a 1-dimensional Convolutional Neural Network over the last four hidden layers would pool together knowledge learned between different consecutive layers to generate a better overall representation than the final hidden state by itself. However, for similar reasons why we thought the CNN approach would be effective, it was actually ineffective. Attempting to learn dependencies between hidden layers that didn't exist made the logits for each particular word in our overall hidden layer less representative. In examples where the BERT Baseline(which only uses the final hidden layer) predicted correctly, the predictions made by BERT + CNN for those same examples, contained only part of the correct answer. This is illustrated by the following example:

Context: "A function problem is a computational problem where a single output (of a total function) is expected for every input."

Question: "A function problem is an example of what?"

Correct Answer: "computational problem"

Bert Baseline Answer: "computational problem"

BERT + CNN Answer: "computational"

This example demonstrates how pooling together different hidden layers using a 1-dimensional CNN actually generates an overall layer that is actually less representative of a particular word than the original final hidden layer. The final hidden layer usually has the most information and as we can see in this example, some of that information is lost when we pool together knowledge of multiple hidden layers using a CNN.

5.3 Loss Function Analysis

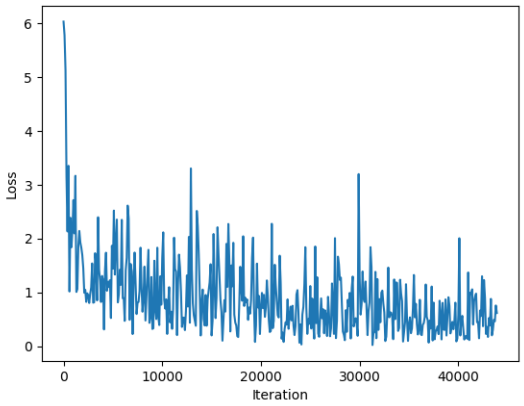

Figure 3: BERT + WCLE

BERT + WCEL did not perform as well as we expected, and actually performed slightly worse than the baseline model. In this model, we adjusted the loss function to further penalize false positives (predicting an answer when there is no answer) by a factor of 2. However, the EM and F1 score both decreased by roughly .5 points. We graphed the training loss over two epochs as seen in Figure 3 and noticed that there were intermittent fluctuations in the training loss that were increased in magnitude by the modified Cross Entropy Loss function. However, the model was not able to significantly improve upon baseline in avoiding false positives. The baseline had an FP rate of 27.17 and BERT + WCEL had an FP of 27.56. The most likely reason why the original baseline is performing better than our current modified loss function is because a penalizing factor of 2 is probably not optimal. We chose 2 as our factor because we originally wanted to weight against false positives 2 times as much.

We didn't have time to experiment with other penalizing factors, but perhaps with more fine-tuning of the penalizing factor, our modified loss function approach will help improve the EM and F1 scores.

5.4 Gated Networks Analysis

BERT + Highway made some minor improvements compared to our original baseline. We expected this increase because highway networks contain gating mechanisms that allow BERT to be more selective about what information it wants from the final hidden layer to calculate logits. And as expected, the BERT + Highway approach resulted in a predictable increase in EM w/o non-answers and F1 score. However, it also predicted more false positives (no true answer but an answer is predicted) than BERT Baseline. From examining various false positive examples of the BERT + Highway approach, we observed that the highway gating mechanism's selection process was still not sophisticated enough. Consider the following example:

Context: "The Normans were famed for their martial spirit and eventually for their Christian piety."

Question: "Who was famed for their Christian Spirit?"

Correct Answer: No Answer

Bert Baseline Answer: No Answer

BERT + Highway Answer: "The Normans"

This error BERT + Highway made is mainly attributed to both the "spirit" and "Christian" parts of the sentence being passed through the gating mechanisms and the model being unable to realize that these two words are not referring to each other.

5.5 Transformers Analysis

BERT + Transformers did not perform as well as the baseline. Initially, we hypothesized that the original baseline BERT model could be underfitting because BERT was a general language model and not a specific Question-Answering model. Therefore, we attempted to introduce two additional transformer layers on top of the pre-trained BERT model to fine-tune the number of encoder transformers from 12 to 14 and increase its complexity. However, based on our results, adding additional transformers on top of the current BERT model decreased performance. One trend that we observed was that in many cases when BERT Baseline predicts no answer and there actually is no answer to the question, BERT + Transformers still makes a prediction. We see this in the following example:

Context: "Frederick William, Elector of Brandenburg invited Huguenots to settle in his realms. Several prominent German military, cultural, and political figures were ethnic Huguenot, including poet Theodor Fontane and General Hermann von François."

Question: "What ethnicity was Frederick William, Elector of Brandenburg?"

Correct Answer: No Answer

Bert Baseline Answer: No Answer

BERT + Transformers Answer: "Huguenot"

Therefore, the BERT Baseline model may actually already be slightly overfitting as opposed to underfitting. We didn't have time to experiment thoroughly with various dropout probabilities, but if we had more time, we would increase our default dropout probability to see if our performance on BERT baseline improves. Another possibility is that perhaps 12 encoders is optimal when in conjunction with the current hyperparameters in the model. We originally expected performance to improve by adding more transformer layers because BERT Large contains 24 encoders (compared to BERT Base's 12 encoders), but BERT Large also has bigger hidden states (1024 for BERT Large and 768 for BERT Base) and more attention heads (16 for BERT Large and 12 for BERT Base). Therefore it is possible that with a hidden-size of 768 and 12 attention heads, 12 is the optimal number of encoders which is why we did not see an improvement in performance when adding additional transformer layers on top of BERT Base. We were unable to test this hypothesis however because we were using pre-trained weights and therefore could not adjust the hidden-size and number of attention-heads.

5.6 BiDAF Analysis

We expected BERT + BiDAF to perform significantly better than the baseline, but this model actually ended up performing much worse than BERT baseline. As previously mentioned, we passed our hidden states into BiDAF in hopes of making more our model more specific for question answering. However, introducing this additional complexity seems to have harmed performance by overfitting. Like BERT + Transformers, in examples in which the baseline model predicts no answer, and there is actually no answer to the question, BERT + BiDAF still makes a prediction. In addition, another interesting observation about the errors that this model makes is that it seems to overcomplicate simple questions. Consider the following example:

Context: "They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Francia."

Question: "Who was the Norse leader?"

Correct Answer: "Rollo"

Bert Baseline Answer: "Rollo"

BERT + BiDAF Answer: "King Charles III"

This example demonstrates how the BERT + BiDAF model seems to be overcomplicating a simple question by inferring some additional knowledge. Since the Norse swore fealty to King Charles III, he is now the new leader. So overall, based on the EM and F1 scores, adding a BiDAF on top of BERT doesn't actually help improve performance because doing so seemingly leads to overfitting and overcomplication.

6 Conclusion

Despite trying many different approaches, our best performing model was BERT Baseline + Hyperparameter Tuning. In this model, we incorporated a gradient accumulation with step size 2 and improved upon the baseline by 1 EM and 1 F1 point. Although several of our approaches, such as weighting the last three hidden layers to generate an overall representation layer, made intuitive sense our ablation studies showed us that introducing additional complexities may not always improve model performance. In our case, trying to combine knowledge from different layers or incorporate question-answering specific knowledge resulted in a loss of critical information and the addition of noise in the final layer of representation.

6.1 Future Improvements

Although our attempts to build on-top of the BERT Baseline model did not result in any large improvements in performance, there are a still few things that can be implemented and would very likely improve model performance. First, we can perform further hyperparameter tuning, specifically with the dropout probability because it is possible that the BERT Baseline model is slightly overfitting and with the penalizing factor for our modified loss function. Next, we could concatenate the last two layers of BERT model to generate an overall representation layer. This approach incorporates more information when calculating logits, and there won't be any information loss like the BERT + CNN and BERT + WHL models. Another approach would be to incorporate hand-crafted features and concatenate them to the final hidden layers of the BERT model before computing logits. This approach is discussed in [Chen et al., 2017] in which we will have a binary feature of whether the current context word is in the question, as well as its part of speech tag, name entity recognition tag, and term frequency tag. Lastly, ensembling multiple models (in particular vanilla BiDAF, QANet and R-Net) and combining their start,end predictions with our baseline BERT model, will most likely improve performance.

6.2 Acknowledgements

We would like to thank Professor Christopher Manning for a wonderful experience and all of the CS224N TAs for being so supportive and patient throughout the quarter. In addition, special thanks for Microsoft Azure for providing us for computational resources.

7 References

1. Chen, Danqi, et al. "Reading wikipedia to answer open-domain questions." arXiv preprint arXiv:1704.00051 (2017).
2. Cui, Yiming, et al. "Attention-over-attention neural networks for reading comprehension." arXiv preprint arXiv:1607.04423 (2016).
3. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
4. Kim, Yoon, et al. "Character-aware neural language models." Thirtieth AAAI Conference on Artificial Intelligence. 2016.
5. Seo, Minjoon, et al. "Bidirectional attention flow for machine comprehension." arXiv preprint arXiv:1611.01603 (2016).
6. Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. "Highway networks." arXiv preprint arXiv:1505.00387 (2015).
7. Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems. 2017.
8. Yu, Adams Wei, et al. "Qanet: Combining local convolution with global self-attention for reading comprehension." arXiv preprint arXiv:1804.09541 (2018).