IID SQuAD Track: Expanding BiDAF and QANet Architectures for SQuAD

Stanford CS224N Default Project

Eva Prakash
Department of Computer Science
Stanford University
eprakash@stanford.edu

Josh Singh
Department of Computer Science
Stanford University
jsingh5@stanford.edu

Abstract

In this work, we produced a question-answering (Q&A) system that builds off of the Bidirectional Attention Flow (BiDAF) [1] and QANet [2] architectures to perform well on the SQuAD 2.0 [3] dataset and improve upon the performance of the provided baseline BiDAF model. We first outperformed the accuracy of the baseline BiDAF model on the SQuAD 2.0 dataset by expanding the BiDAF model to incorporate character-level embeddings, both alone and in combination with word-level embeddings. We also experimented with adding self-attention and co-attention layers to the BiDAF model. We found that the BiDAF model with character-level embeddings slightly outperformed the BiDAF model with combined character-level and word-level embeddings. We also discovered that co-attention did not improve the baseline BiDAF model, but self-attention increased the performance of the BiDAF model with character-level and word-level embeddings. We then implemented the QANet architecture to improve upon the performances of the baseline and expanded BiDAF models and build an even more accurate Q&A system. We experimented with the character dimensions of the QANet architecture and found that smaller character dimensions slightly outperformed larger character dimensions. We also augmented the QANet model by adding a segment-level recurrence mechanism from Transformer-XL [4]. However, this QANet + Transformer-XL model proved considerably unwieldy to train. Our highest performing QANet model achieved **EM = 66.056** and **F1 = 69.415** on the dev set and **EM = 62.756** and **F1 = 66.325** on the test set, putting us at **16/114** on the non-PCE SQuAD track dev leaderboard and **23/100** on the non-PCE SQuAD track test leaderboard (submission name: Eva Prakash / Josh Singh).

1 Key Information

- Mentor: Vincent Li
- External Collaborators: None.
- Sharing project: No.

2 Introduction

Machine comprehension and Q&A models have have become critical domains in natural language processing (NLP) due to their applications in search engines and information retrieval for fields such as education, medicine, and law. The rise of these domains can also be attributed to the increase in public annotated datasets like SQuAD. Pre-trained contextual embedding (PCE) Q&A models such as Bidirectional Encoder Representations from Transformers (BERT) [5] and Embeddings from Language Models [6] have been finetuned on large corpora in order to optimize the necessary starting point for achieving high performance on smaller novel datasets. A popular approach to creating

non-PCE Q&A systems that train from scratch on specific datasets instead of generalizing to a wide variety of tasks is using recurrent neural networks (RNNs). However, RNNs cannot perform parallel computation and have frequently been seen to have limited long-term memory. A prominent NLP goal is thus replacing Q&A RNNs with convolution-and-attention-based architectures. In this project, we focused on improving the baseline BiDAF model with self-attention, co-attention, and various character-level and word-level embeddings. We also re-implemented QANet and experimented with augmenting the model with the segment-level recurrence mechanism from Transformer-XL in order to remember long-term contextual dependencies. We investigated the performance benefits and limitations of our various architectural expansions in extracting answer spans or determining non-answerability from the SQuAD 2.0 questions and context paragraphs. Most of our BiDAF-based models outperformed the provided baseline, and most of our QANet-based architectures outperformed our BiDAF-based models. Aside from EM, F1, AvNA, and NLL, we evaluated our architectures using a breakdown of question and answer type. We analyzed how the model performed on following question types: why, what, where, when, who, how, which, and other. We looked at correct and incorrect predictions and analyzed if non-answerable (NA) predictions or answerable (A) predictions were more correct.

3 Related Work

The BiDAF network [1] represents context paragraphs with character-level, word-level, and phrase-level specificity and pioneers a BiDAF attention mechanism to represent query-aware context without early summarization. The BiDAF context-query attention layer uses similarities between all pairs of context and query words to find both context-to-question (C2Q) attention and question-to-context (Q2C) attention. BiDAF is limited by its use of RNNs to process sequential inputs, which inhibits parallelization and slows the architecture down in real-world applications with large bodies of texts. While BiDAF generated state-of-the-art performance on the SQuAD 1.0 dataset in 2016, SQuAD 1.0 only contains answerable questions, whereas our SQuAD 2.0 dataset includes the extra complexity of non-answerable questions.

QANet [2] improves on the inefficiencies of the BiDAF architecture by forgoing recurrence models in favor of convolutions that learn localized structure and self-attentions that learn global interchange between words. QANet is the first efficient QA model to be constructed only with convolutions and self-attentions, building upon the previous work of Raiman and Miller [7], who sought to speed up QA with a model that replaced bi-directional attention with conditioning on search beams, and Weissenborn et al. [8], who also sought to speed up QA with a model that removed the context-query attention module. Both of these architectures remained inherently recurrent and intrinsically less efficient than QANet while also failing to demonstrate competitive accuracy on SQuAD 1.0. QANet introduces encoder blocks that stack convolution, self-attention, and feed-forward layers, successfully modeling both local and global interactions. It achieves F1 and EM scores on par with the state-of-the-art models for SQuAD 1.0. However, like BiDAF, the QANet paper does not describe any evaluation on non-answerable questions.

Transformer-XL [4] seeks to learn long-term contextual dependencies without the limitation of fixed-length contexts. It proposes a novel segment-level recurrence mechanism and positional encoding scheme that allows the model to learns dependencies beyond a fixed length without sacrificing temporal coherence.

The main contribution of our work is testing the robustness of these models on non-answerable SQuAD 2.0 questions and evaluating various architectural expansions to these models.

4 Approach

Our baseline model was the provided BiDAF model for the default project, which uses word-level embeddings.

4.1 Expanding BiDAF

We expanded the baseline BiDAF model by experimenting with character-level embeddings, replacing the BiDAF attention layer with a co-attention layer, and inserting a self-attention layer.

4.1.1 Character-Level Embeddings

Character-level embeddings allow Q&A models to condition on the morphology of words and focus on small, specific linguistic patterns. After obtaining the character-level embeddings of each context and query word, which were 64-dimensional pretrained GloVe character embeddings, we passed them as input into a two-dimensional convolutional layer and max-pooling layer in order to project these embeddings into the space of the word-level embeddings and create fixed-size vectors for each word. These embeddings were then passed through a two-layer highway network [1]. We also experimented with concatenating these character-level embeddings with the word-level embeddings of the baseline, which were 300-dimensional pretrained GloVe word embeddings. We used hyperparameter tuning to find that the best kernel size for the convolutional layer was 5 and the best kernel size for the max-pooling layer was 12. The maximum number of characters per word was 16.

4.1.2 Co-attention

Co-attention [9] involves two-way attention between context and query words and further attends a second time over C2Q and Q2C attention outputs. These second-level attention outputs are concatenated and fed through a bidirectional LSTM to produce the encoded hidden states. Given that co-attention can be beneficial in learning pairwise attentions, we experimented with swapping the BiDAF attention layer with a co-attention layer.

4.1.3 Self-Attention

Self-attention [10] attends over all hidden states in an embedded text sequence. The query (Q), key (K), and value (V) vectors all originate from the same query vectors, and the query-key dot products are used to compute a weighted sum of values as follows.

$$Attention(Q, K, V) = softmax(Q * K^T)V$$

Multi-headed self-attention [11] divides embeddings across multiple attention heads and averages the outputs. The multi-headed self-attention layer we implemented is inspired from one of the popular public QANet codebases linked with the paper [12]. We changed the hidden size from the default of 100 to 128 and used 8 attention heads in our self-attention layer in order to average attention outputs across a large number of heads.

4.2 QANet

The QANet architecture, as seen in Figure 1, is comprised of stacked encoder blocks that share weights between the context and question encoder as well as between the three output encoders. Our QANet implementation, though written from scratch, was guided by the QANet repository [12].

- **Input Embedding Layer**: In the same way as BiDAF, the input embedding layer concatenates word-level and character-level embeddings for each context and question word. QANet uses a hidden size of 128. We also experimented with 200-dimensional pretrained GloVe character embeddings.
- **Embedding Encoder Layer**: The embedding encoder layer uses positional encoding to keep track of positional placement in text. The positional encoding used is [5]

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

  where d_{model} is the embedding dimension, pos is the position, and i is the dimension. Our positional encoding function is referenced from the QANet repository [12]. The basic building block of the embedding encoder is a stack of 4 depthwise separable convolution layers [2] with a kernel size of 7, the previously described multi-headed self-attention layer, and two feed-forward layers that operate inside a residual block. Layer normalization is applied before each layer.
- **Context-Query Attention Layer**: The attention layer calculates S, a matrix of pairwise similarities between question word q and context word c, using the following trilinear function [1]

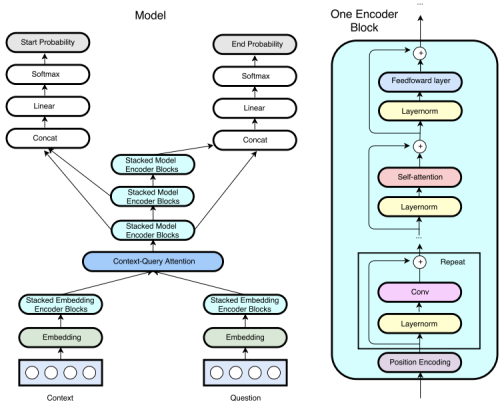
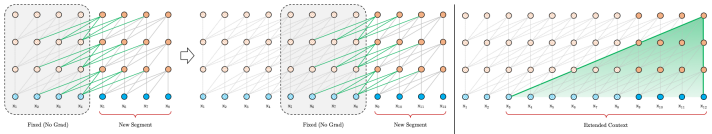


Figure 1: QANet architecture and encoder block inner workings[2].



(a) Training phase. (b) Evaluation phase.

Figure 2: Transformer-XL segment-level recurrence mechanism during training and evaluation [4].

$$similarity(q, c) = W_0[q, c, q \odot c]$$

where $\odot$ is element-wise multiplication and W_0 is a trainable parameter. C2Q and Q2C attention are then calculated as follows for the context matrix C and question matrix Q

$$C2Q = S' * Q^T$$
$$Q2C = S' * S'' * C^T$$

where S' is the result of the softmax function applied to S and S'' is the column-normalized version of S'. Our context-query attention layer implementation is inspired by the QANet repository [12].

- **Model Encoder Layer**: The model encoder layer has 3 repetitions of 7 blocks. The difference from the embedding encoder layer is that there are 2 depthwise separable convolutional layers with kernel size 5.
- **Output Layer**: The output layer predicts the start (s) and end position (e) probabilities of the answer using

$$s = softmax(W_1[M_0; M_1]), e = softmax(W_2[M_0; M_2])$$

where W_1 and W_2 are trainable parameters and M_0, M_1, M_2 are the three model encoder outputs.

4.2.1 Transformer-XL

We augmented the QANet model with the segment-level recurrence mechanism from Transformer-XL [4]. While we did not refer to a specific QANet + Transformer XL repository and put together our model from scratch, our implementation required adapting Transformer-XL modules from the Transformer-XL repository [13] and applying them to the QANet architecture. Transformer-XL splits up the context into fixed-size tokens during training and caches the hidden state sequence of the previous token to use as extended context for the current context token while the gradient stays within the current token, as seen in Figure 2. This allows for the model to exploit long-term contextual dependencies and information in the history during evaluation. Transformer-XL loses the ability to distinguish between token positions using standard positional encoding, so it instead uses a positional

embedding that retains relative positional information in hidden states. We tuned the size of the context tokens on our local machine to arrive at the length of 70 [4].

5 Experiments

5.1 Data

We used the SQuAD 2.0 dataset, created by splitting the official SQuAD's data set in half, while the training set is the same as the of SQuAD. The contexts are excerpts from Wikipedia and the answer is a portion of the context that answers the question. Our model takes in a context paragraph and question and either outputs an answer that is a subset of the context block or a prediction that the question is unanswerable.

5.2 Evaluation Methods

Our model is evaluated on two main evaluation metrics: EM and F1. EM (Exact Match) is a rigid, binary metric that measures whether the model output exactly matches the expected truth. F1 is the harmonic mean of recall and precision and thus is less rigid than EM. Recall measures how many words from the ground truth the predicted answer contains. Precision measures how much of the predicted output is part of or a subset of the ground truth.

We also recorded the metrics of AvNA and NLL. AvNA stands for "Answer versus No Answer" and helps us understand the classification accuracy of answer versus no answer forecasts. NLL is negative log likelihood and gives us insight into how well the model is learning.

We broadened our horizons past the provided evaluation metrics. We calculated the EM values for specific questions for the baseline, best BiDAF, and best QANet models. The specific types of questions are "why", "what", "when", "where", "how", "which", "who", and "other". We calculated the accuracy of A and NA answers and looked at a breakdown of correct and incorrect answers to see what percent of each was A vs NA. We also directly examined predictions and ground truths to conduct a more qualitative analysis.

5.3 Experimental details

The final models presented in this project are the BiDAF Baseline, BiDAF With Character-Level Embeddings, BiDAF With Character-Level + Word-Level Embeddings, BiDAF With Character-Level + Word-Level Embeddings and Co-Attention, BiDAF With Character-Level + Word-Level Embeddings and Self-Attention, QANet (64 Character Dim), QANet (200 Character Dim), and QANet + Transformer-XL.

We trained our models with 50K evaluation steps and no L2 weight decay. All models trained for 30 epochs except for QANet + Transformer-XL; our computational resources could only handle 5 epochs. For our BiDAF-based models, hyperparameter tuning led us to keep the learning rate at 0.5 and dropout at 0.2. Dropout was 0.1 for QANet-based models. In addition, QANet-based models had a learning rate of 0.001 with an exponential warm-up period over the first 1000 steps before remaining constant. For QANet-based models, word-level embedding dropout was 0.1 and character-level embedding dropout was 0.05. During training, QANet-based models dropped every second layer with probability 0.1 and sublayer l within the encoder layers with probability $\frac{l}{L}(1 - p_L)$, where L is the last layer and p_L is 0.9. Our BiDAF-based models were able to train in a few hours with a batch size of 64 using the Adadelta optimizer. The BiDAF model with co-attention took slightly longer due to the large dot products being calculated. Due to computational bottlenecks for large numbers of encoder blocks, our QANet-based models took 10+ hours to run with a batch size of 16. QANet + Transformer-XL in particular took 12+ hours for just 5 epochs. QANet used the Adam optimizer with β_1 = 0.8, β_2 = 0.999, ϵ = 1e-07, and weight decay = 3e-07. QANet + Transformer-XL used the Adam optimizer with β_1 = 0.9, β_2 = 0.999, ϵ = 1e-07, and weight decay = 3e-07.

5.4 Results

As seen in Figure 5, our highest performing model (QANet with 64 Character Dimensions) achieved EM = 66.056 and F1 = 69.415 on the dev set. **Our highest performing model achieved EM =**

Model	EM	F1	AvNA	NLL
BiDAF Baseline	56.86	60.39	67.65	03.20
BiDAF With Character-Level Embeddings	60.83	64.18	70.85	2.74
BiDAF With Character-Level + Word-Level Embeddings	59.28	62.78	69.08	3.04
BiDAF With Character-Level + Word-Level Embeddings and Co-Attention	52.19	52.19	52.14	5.18
BiDAF With Character-Level + Word-Level Embeddings and Self-Attention	59.70	63.08	69.53	2.94
QANet (64 Character Dim) - Dev Leaderboard	66.06	69.41	75.84	2.66
QANet (200 Character Dim)	65.70	69.12	75.26	2.56
QANet + Transformer-XL (5 epochs)	51.67	51.72	52.19	5.17

Figure 3: Dev statistics (EM, F1, AvNA, NLL) for our various models.

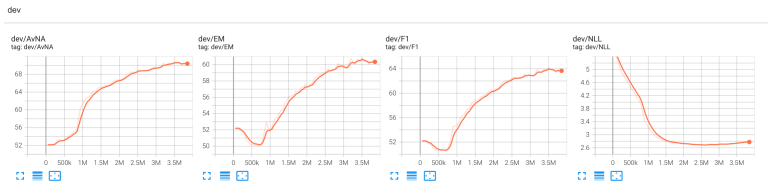


Figure 4: BiDAF+CharEmbeddings performance over 30 epochs as our highest-performing BiDAF-based model.

62.756 and F1 = 66.325 on the test set, putting us at 16/114 on the non-PCE SQuAD track dev leaderboard and 23/100 on the non-PCE SQuAD track test leaderboard (submission name: Eva Prakash / Josh Singh).

All of our model dev results are displayed in Figure 3. Among our BiDAF based models, BiDAF With Character-Level Embeddings, displayed in Figure 4, performed the highest. Character-level embeddings likely increased performance from the baseline by adding detailed information on the morphology of words to the model. Character-level embeddings alone may have outperformed concatenated character-level and word-level embeddings possibly due to the dilution of the specificity of low-level linguistic information with broader, higher-level relationships. Co-attention might not have increased performance from the baseline because the pairwise attentions might have modeled overly complex relationships. The addition of multi-headed self attention to the BiDAF model increased performance both relative to the baseline and the BiDAF model with concatenated character-level and word-level embeddings, showing the influence of multi-headed self-attention. This may be due to the fact that different embedding splits across heads can learn different interpretations.

Given the greater complexity modeled by the stacked convolution and self-attention layers of QANet, we fulfilled the expectation of QANet-based models outperforming BiDAF-based models. We found

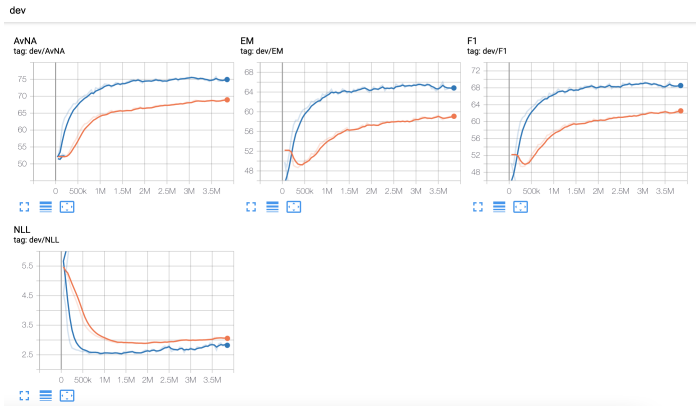


Figure 5: QANet (blue) and BiDAF+CharEmbeddings (orange) performance over 30 epochs.

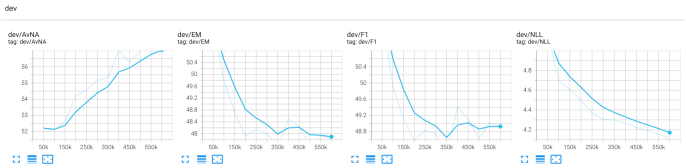


Figure 6: QANet + Transformer-XL performance over 5 epochs

that QANet with 64-dimensional character-level embeddings slightly outperformed QANet with 200-dimensional character-level embeddings. The difference is small, so this could potentially be due to uncontrollable differences in random initializations. However, it could also be possible that 200 dimensions are too many features and patterns for the model to control over as opposed to 64.

5.5 QANet + Transformer-XL

The 5 epochs that QANet + Transformer-XL ran for produced the results shown in Figure 6. Given that the 5 epochs took up large amounts of time and credits to run, we were not feasibly able to run more than 5 epochs given the computational bottlenecks of QANet and Transformer-XL memory segments combined. On top of this, we found that QANet + Transformer-XL was extremely sensitive to initializations, with F1 and EM starting points ranging from around 52 to around 48. Sensitivity to initializations and hyperparameters was a drawback acknowledged by the creator of the Transformer-XL codebase and caused us to frequently restart the training process[13]. In the displayed run, we observed that the F1 and EM dipped, yet the AvNA and NLL steadily increased and decreased without plateauing, which gives reason to believe that the model was still on track to genuinely learn and optimize, and the F1 and EM may have simply been suffering from the initial dip effects seen with BiDAF and QANet. Although 5 epochs is early to tell, our chosen context token length may have been too small and caused a memory overload with the number of tokens or too large and created slowness with the token size. The unwieldiness of QANet + Transformer-XL made it difficult for thorough hyperparameter tuning. Given that one self-proclaimed draw of QANet is training speed, our work with QANet + Transformer-XL indicates that while the model could technically be promising in terms of performance, its computational bottlenecks currently make it an unreliable choice for expanding QANet.

6 Analysis

When examining where certain models faced inaccuracies, we noticed that certain question types performed differently, more information is needed on answer vs no answer questions and predictions, and a predicted answer can be a subset of the ground truth or the ground truth can be a subset of the prediction. We designed our own evaluation metrics and scripts.

We first investigated how different question types performed. We looked at the EM scores for our predictions for eight questions types: "why", "what", "where", "when", "which", "how", "who", and "other".

	why	what	when	where	how	which	who	other
BiDAF-CharEmb	100.0	61.5	53.4	62.5	54.8	63.0	53.9	60.9
QANet	50.0	50.8	45.7	50.0	43.8	58.0	44.7	52.4
QANet200	75.0	66.1	61.2	70.0	61.6	70.0	67.1	65.7

EM Scores for BiDAF+CharEmbeddings, QANet64, and QANet200 for "why", "what", "when", "where", "how", "which", "who", and "other" questions.

The models were consistent in which questions they did best and worst in. The models predicted the best for "which", "what", and "where" questions and worst for "when" and "how" questions. "Why" questions' predictions were variable and other questions were moderately accurate compared to other data. We predicted that the answers for "which", "what", and "where" questions were straightforwardly stated in the context paragraphs. "How" and "when" questions' predictions performed the worst, as they have more open-ended answers and could be harder to answer.

We then investigated the answer vs no answer question accuracy breakdown.

	Correct: A	Correct: NA	Incorrect: A	Incorrect: NA
BiDAF-CharEmb	43.8	56.2	71.6	28.4
QANet	34.5	65.5	59.4	40.6
QANet200	44.7	55.3	74.1	25.9

The percentage of correct and incorrect predictions that were A vs NA

Of the predictions that the models got correct, the majority of the answer types were NA. Of the predictions that the model got incorrect, the majority of answer types where A.

	% of A Correct	% of A Incorrect	% of NA Correct	% of NA Incorrect
BiDAF-CharEmb	48.7	51.3	75.4	24.6
QANet	38.6	61.4	63.5	36.5
QANet200	53.6	46.4	80.4	19.6

The percentage of correct and incorrect predictions that were A vs NA

We also looked out how the model performed on NA and A answers individually. The overwhelming majority of NA were predicted correctly while about half of the A questions where predicting correctly. We predicted that these observations about NA and A were the case because it is much harder to predict that an answer exists and predict that answer than to return the default answer of NA.

Seeing these answers, we wanted to take our analysis further by looking at actual model outputs and comparing them with real answers. We found that, while there were some answers that completely predicted incorrect results, there were a lot of cases where the ground truth was a subset of the prediction or vice versa. For example, the triplet ("Why is Warsaw's flora very rich in species?", "The flora of the city...Science).", "location of Warsaw") received the prediction of "the location of Warsaw within the border region of several big floral regions" by BiDAF+CharEmbeddings, QANet, and the baseline model. This shows the limitation of our evaluation metrics (specifically, EM) as our prediction is as good, if not better, than the provided answer. This result means that our model might perform better than was previously thought from the evaluation metrics.

7 Conclusion

In this project, we augmented both the BiDAF and QANet architectures to examine their benefits and drawbacks on the SQuAD 2.0 dataset. After adding different combinations of embedding and attention layers to the BiDAF baseline model, we found that BiDAF with character-level embeddings performed the best. We implemented QANet with various character dimensions and outperformed all BiDAF+ models. QANet64 was our highest performing model by F1 and EM score. We expanded QANet with Transformer-XL mechanisms to increase long-term dependencies, but we found this to be an uncertain option, despite its promising initial direction, due to its computational bottlenecks. Our results are an indicator that experimenting with thoughtful additions to QA models is critical to understanding not only the extent of their pros but their cons as well. Training time and Azure credits were the main limitations we faced. For example, each epoch of QANet + Transformer-XL ran for many hours, meaning it would have taken many days and Azure credits to fully run. One of our teammates had COVID-19 and major digestive issues that temporarily hindered our progress and warranted our project extension. Our future work would focus on further examining the computational limitations of QANet + Transformer-XL by experimenting with a wide range of initializations and hyperparameters.

References

[1] Ali Farhadi Hannaneh Hajishirzi Minjoon Seo, Aniruddha Kembhavi. "bidirectional attention flow for machine comprehension".

[2] Minh-Thang Luong et al. Adams Wei Yu, David Dohan. "qanet: Combining local convolution with global self-attention for reading comprehension". 2018.

[3] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. 2018.

[4] Yiming Yang-et al. Zihang Dai, Zhilin Yang. "transformer-xl: Attentive language models beyond a fixed-length context". 2019.

[5] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. Bert: Pre-training of deep bidirectional transformers for language understanding. *cs.CL*, 2018.

[6] Mohit Iyyer Matt Gardner Christopher Clark Kenton Lee Luke Zettlemoyer Matthew E. Peters, Mark Neumann. Deep contextualized word representations. *NAACL*, 2017.

[7] Jonathan Raiman and John Miller. Globally normalized reader. page 1070–1080, 2017.

[8] Georg Wiese Dirk Weissenborn and Laura Seiffe. "making neural qa as simple as possible but not simpler.". page 271–280, 2017.

[9] Victor Zhong Caiming Xiong and Richard Socher. Dynamic coattention networks for question answering. *"arXiv preprint arXiv:1611.01604"*, 2016.

[10] Microsoft Research Asia Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. 2017.

[11] Vaswani et. al. Attention is all you need. 2017.

[12] Andy $andy840314.github.io$
https://github.com/andy840314/QANet pytorch.

[13] Zhilin Yang $kimiyoung.github.io$.
https://github.com/kimiyoung/transformer-xl.