Improved BiDAF with Self-Attention

Mingchen Li
Department of Electrical Engineering
Stanford University
limc@stanford.edu

Gendong Zhang
Department of Electrical Engineering
Stanford University
zgdsh29@stanford.edu

Zixuan Zhou
Department of Electrical Engineering
Stanford University
zixuan95@stanford.edu

Abstract

We performed machine reading and question answering on SQuAD 2.0 using an improved version of BiDAF model. We added character embedding layer, self-attention layer and a novel designed learnable weighted average-attention layer. The final F1 and EM score of the model were 66% and 62%, respectively. We also tried a two-stage system approach, which in theory can classify whether a question has an answer and convert SQuAD 2.0 to SQuAD 1.1 for the second stage. However, we found no benefit on dividing the system into two stages since neither traditional machine learning nor deep learning can give a high enough accuracy on the binary classification task of the first stage.

1 Introduction

Machine reading and question answering (Q&A) is one of the most important testing methods for evaluating how well computer systems understand human languages. It is also a critical ability for language related artificial intelligent system, such as chatting robots and search engines. Therefore, the research community has created many Q&A-oriented dataset based on abundant resource of text on the Internet, such as SQuAD[1], TriviaQA[2], etc. This project focuses on the SQuAD 2.0 dataset.

There has been extensive researches conducted on conquering SQuAD dataset and Q&A tasks in general, among which QANet[3] and Bi-Directional Attention Flow (BiDAF)[4] are the most popular ones. In this project, BiDAF model is provided to us as the baseline mode. We investigate both networks and proposed improvements on the latter one. The QANet is implemented as a comparison model for our improved BiDAF. For BiDAF, we perform feature engineering on the embedding layers and add self-attention[5] layer to the model structure. In addition, we design learnable weighted average-attention layer which itself improved the accuracy of the baseline model by nearly 5%. With the combination of these three features, the resulted F1/EM score of the improved model is increased by an average of 7% on a backbone of Gated Recurrent Unit (GRU) network. The overall model architecture is shown in Figure 1.

We also investigate a novel two-stage system architecture, which intends to perform a binary classification at the first stage to determine whether a question is answerable or unanswerable, and a normal Q&A model at the second stage. This system should smoothly transfer a SQuAD 2.0 task to a SQuAD 1.1 task with the help of the binary classifier. However, the results of this system do not satisfy our expectations. The analysis of the experimental results can be referred to section 5 of this report.

In summary, the contribution of this project are as follows:

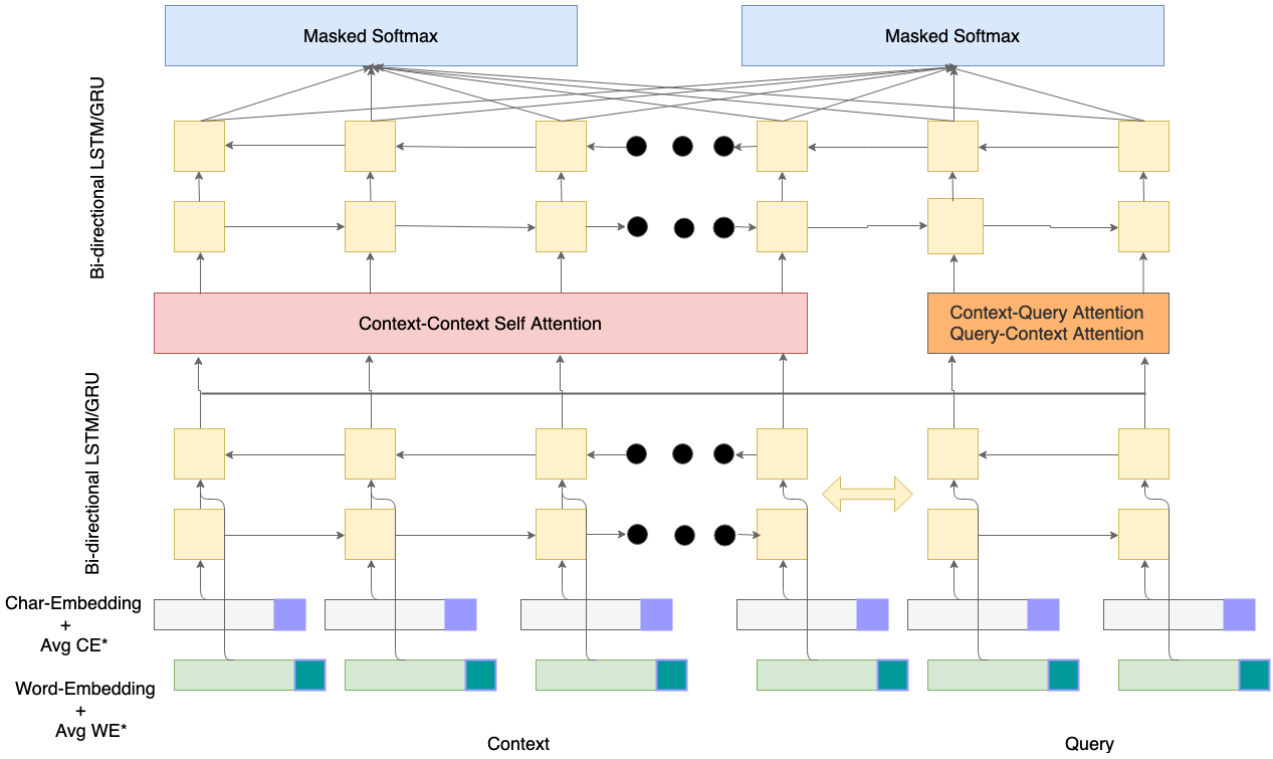


Figure 1: An overview of our model architecture. CE stands for char-embedding. WE stands for word-embedding.

- We combined character embedding, self-attention and average-attention layers to a BiDAF model using GRU network to improve the accuracy of the baseline model. We experimentally prove that adding our learnable weighted average-attention layer is beneficial based on the significant improvement of model performance and negligible extra computational cost.
- We test the idea of two-stage architecture on the SQuAD 2.0 Q&A task. Based on the experimental results, we conclude that a binary classifier does not introduce extra benefit on the task.
- We implemented QANet and enabled it to train on SQuAD 2.0 dataset. The results of it is compared with the other model described above.

2 Related Work

2.1 Self-Attention

Prior to self-attention, the sentence embedding was always represented by vectors. Self-attention, however, introduces a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence. This method of computing attention has significantly increased the accuracy of machine translation and machine comprehension tasks. Later, [5] introduces Transformer model, which relies entirely on self-attention compute representations of its input and output without using sequence-aligned recurrent or convolution layers.

2.2 SQuAD 1.1 Specific Models

BiDAF is a hierarchical multi-stage architecture for modeling the representations of the context paragraph at different levels of granularity. It successfully combines recurrent models to process sequential inputs, and attention components to cope with long term interactions. This architecture is based on a Markov-chain assumption that the attention at each time step is a function of only the question and the context at the current time step and does not directly depend on the attention at the

previous time step. Therefore, the model is able to focus more on the interaction between query and context at each time step, and a previous incorrect attention does not influence the current attention. This attention mechanism is implemented in both directions. The model out performs all its previous approaches on SQuAD 1.1 dataset.

QANet comes after BiDAF. It aims to increase the training speed of the model by discarding the recurrent nature of the models. It exclusively uses convolutions and self-attentions as the building blocks of encoders that separately encodes the query and context. Standard attentions are used to learn the interactions between context and questions. The final answers are then predicted based on recurrent-free encoded representations.

2.3 Binary Classifier on Answerable/Unanswerable Questions

The most significant different between SQuAD 1.1 and SQuAD 2.0 is that the latter one contains both answerable and unanswerable questions. Despite this difference, by padding prefixing zeros to the contexts, most of the systems aiming at SQuAD dataset still treat all the questions equally, and use end-to-end architecture to train the models. There are only a few models that focus on distinguishing between answerable and unanswerable questions. NAQ detector introduced by [6] is one of the first of such models. Instead of using deep learning techniques, the researcher implements traditional machine learning methods to build the binary classifier. By assembling decision trees, linear regression and TF-IDF scores, the model achieves around 80% of accuracy on the binary classification task. This accuracy indicates that the traditional machine learning is probably not a good choice for this project. The best Non-PCE model on public SQuAD 1.0 leaderboard is around 90% of F1. If there are 20% of the dataset contains unanswerable questions, and those questions are sent to a model that only predicts answerable questions, the resulting accuracy will be lowered drastically. In addition, due to the time constraint of this project, we don't have enough time to re-implement such models. Therefore we decide to use the baseline deep learning model to perform the task.

3 Approach

3.1 Character Embedding

The baseline of this project is provided by the teaching group, and originally, it only used pretrained Glove word embedding for the BiDAF model. From the hint of the project handout and the code, we conclude that character-level embedding is missing, and the original BiDAF does include both word-level embedding and character-level embedding. Character-level embedding can be an extra feature for both questions and contexts, which is useful for improving the model performance since the embedding can contains more information.

There are several benefits of using char-level embedding, the first and foremost important benefit is providing extra information of the word meanings and its most important representation in the character level. Another benefit is that for misspelling words, typos and new words can be dramatically alleviated, and especially some newly invented words might not have a pre-trained word embedding, while they can be encoded using character embedding. Therefore, having the character embedding can be formed for every single word's vector even it is out-of-vocabulary.

From the start code, the character vector file is already included and can be really handy, and the data loader of the start code can load the character indices and corresponding word. Therefore, we add an additional character embedding layer in the layers files, in the character embedding layer, it contains a convolutional layer to extract the character-level information, and after several further dimension manipulation, the character embedding is processed and can be concatenated with the original word embedding to get a 2 times embedding size representation. This word-character embedding then follows the original procedure for the downstream jobs.

3.2 Learnable Weigthed Average-attention

Since the addition of character embedding boosting up the performance of our model, we decide to dig more into the feature of the word vectors and character vectors in order to further improve

the F1/EM scores. Due to the insight and inspiration of [6], we inspect that adding the mean of the word and character vectors can be beneficial to the performance, and hence, we append the mean of each vector (character and word) to its original vector. However, we do not want to brutally take the mean of each vector since it is fixed and might not be good as a feature, we want our model to take a weighted average of the vector elements and the weights can be learned by the model itself. Therefore, we take the learnable weighted average-attention method (named by ourselves) to compute the average of each vector. This mechanism can summarize a collections of vectors, words or other unordered collections into a single vector, and be more powerful than simply taking averaging or max-pooling. It learns to compute the weights and pay different attentions to different vector elements.

3.3 Self-attention

We also get inspirations from [5,7], so that we insert an extra layer of self-attention to further increase the performance of our model. The BiDAF baseline already included the bi-directional attention flow, which captures the context-query and query-context attentions. Self-attention is another type of attentions that can be used to help the coreference resolution. It computes the attention scores between context and context. We first take the attention score tensor from BiDAF original baseline, and put it into a linear layer to change the dimensions. After going through some non-linear activation function and dropout operation, another RNN encoder to encode the attention scores, and then compute the similarity scores between context and context. Finally we use masked softmax function to normalised these scores, and after unrolling to the original dimensions as the input, the self-attention scores is mixed with the original input, keeping the same dimension for the downstream job like before. We implemented the self attention in a different way than the multi-headed way to keep things simple.

3.4 Fine-tuning

We have also explored different training approaches and fine-tuned our models, such as changing LSTM to GRU to cut down the training time, adding two layers of RNN, using Adam optimizer instead of AdaDelta. However, the improvement was very limited, and for the purpose of this project, we decide to focus on the model architecture instead of manually trying different parameters.

3.5 Two-stage Model

We also tried to divided the question answering into two different stages. The first is choosing the answerable questions and the second it predicting the answer. But the training and performance did not work well. We decided to move on two other approaches as mentioned above.

4 Experiments

4.1 Data

The dataset used in this project is SQuAD 2.0. This dataset contains more than 100000 question-answer pairs on more than 500 articles. In addition, it also contains more than 50000 unanswerable questions. To do well on SQuAD 2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering. The original SQuAD dataset has three splits, train, dev and test, with the first two publicly accessible and the last one held privately. In this project, we split the original dev set into two sets, one for dev and another for test. We also analyzed the amount of questions that start with different key words (“how”, “what”, “why”, “which”, “who”, ”where”, “when”), as shown in Figure 2. We can see that "what" dominates both training set and dev set.

4.2 Evaluation Methods

There are two standard evaluation metrics for this type of Q&A systems, EM and F1 scores. EM score measures the exact match rates of the ground truth answers and predict answers. The F1 score measures the harmonic mean of precision and recall. The equation of F1 is shown in Equation 1. The final score of each question is the maximum of each EM and F1 taken across all the answers.

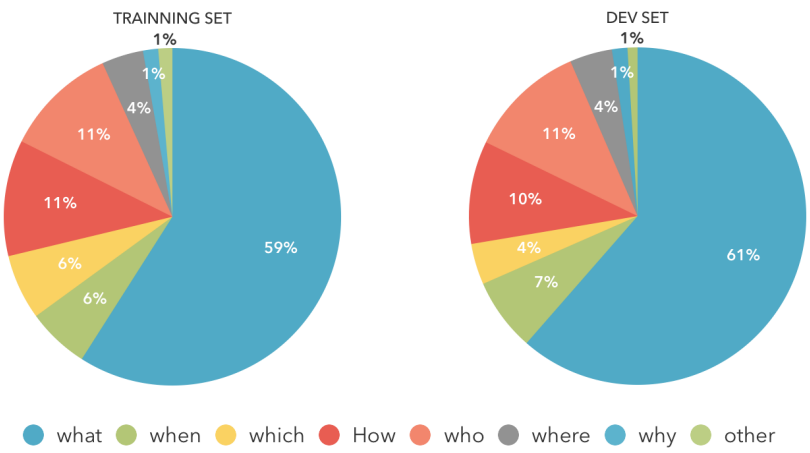


Figure 2: An overview of our model architecture. CE stands for char-embedding. WE stands for word-embedding.

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{1}$$

Besides, there is also a ratio between answerable and unanswerable questions during the training process, AvNA. This ratio can appeal the correlation between the EM and F1 scores with the distribution of the dataset.

4.3 Experimental Details

For this project, we have several parallel work going on at the same time. One direction is to explore the potential of the baseline model and based on that trying to improve the performance. Another one is to compare with specific models targeting question answering, such as QANet, and use them as a comparison with our improved models.

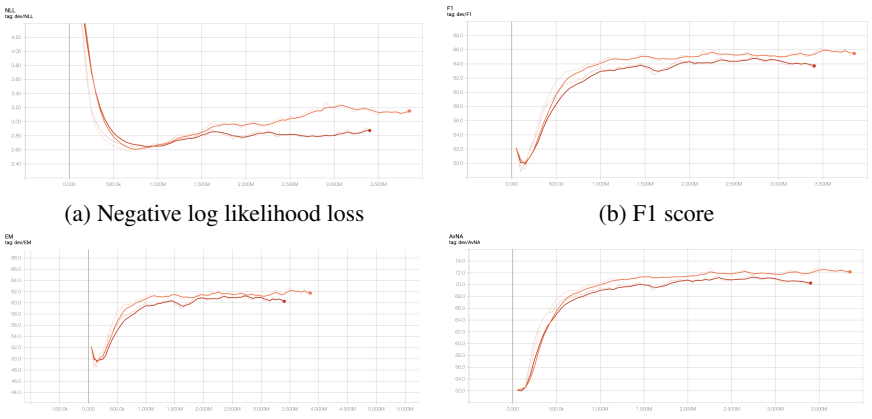
Our initial thought was to divide the task into two stages, one was to identify whether the question is answerable or not, and another was to try to answer the answerable questions, and as mentioned before, it was not as effective as we thought, and the loss was fluctuating while we trained it, so we decided to move on and try other approaches.

We first examined the baseline provided by the teaching staff, and understood the way of handling the new SQuAD 2.0 dataset and general question answering task. We trained on Azure NV6 GPU for approximately 13 hours, and the result is shown in Table 1. One simple way to improve the performance is to add character-level embedding to our baseline models, and an increase in both F1 and EM score has been observed. For adding character embedding, we followed the spirits of Assignment 5, and successfully implemented. The result can be seen in Table 1. Learnable weighted average-attention was then added to the embedding layer to further boost the performance. In order to decrease the training time, we replaced LSTM with GRU to simplify and accelerate the training process. Learnable weighted average -attention also increased the performance further. Our final approach was to add self-attention layer to our model. Since adding more layers made our model more complicated, the training time increased to 18 hours, and we tried to use other optimizer including Adam to stabilize the training, and also experimented with different number of RNN layers, but found the performance was not as we expected, and the details is elaborated in Section 4.4 and 5.

For the purpose of comparison, we used open-source QANet code of SQuAD 1.1, and modified the data loader and evaluation method to make it functional for SQuAD 2.0. QANet is a more complicated network, and due to the limited resource and time frame, we only trained it for 24 hours on Azure. The results and comparison details are shown in in Section 4.4.

4.4 Results

We tested our best self-designed model on the dev set of SQuAD 2.0. The model contains char-embedding layers, weighted average-attention and self-attention layers. To best visualize the results of the model, we trained the final version of the model with two settings, one with one-layer of embedding RNN and another one with two-layer of embedding RNN. Intuitively, the two-layer should perform better than the one-layer model. However, the validation results show the opposite. As shown in Figure 2, the performance of two-layer model is better than the one-layer model during the first few iterations, and then becomes worse for the rest of the time. The reason that a more complicated model performs worse can be due to overfitting. Since our embedding vectors are not so long, the model is able to overfit the training dataset by memorizing the correlation of data points inside a deep structure.



(a) Negative log likelihood loss

(b) F1 score

(c) EM score

(d) Answerable vs unanswerable questions

Figure 3: The experimental results of two models with one-layer and two-layer of embeddings. The orange line is the one-layer model, and the red line is the two-layer model.

We have trained two models, QANet and improved BiDAF. The results of the BiDAF models at every improvement level is shown in Table 1. The last row of the table is the results of QANet.

5 Analysis

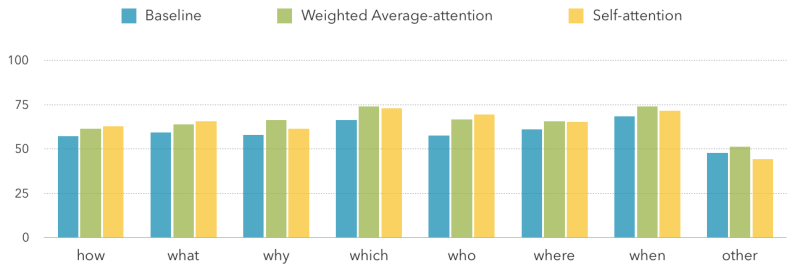


Figure 4: An break-down F1 scores of different question types.

We break down the scores and analyze the performances of models on questions that start with “how”, “what”, “why”, “which”, “who”,”where”, and “when” separately, as shown in Figure 4 and Figure 5.

Both F1 and EM scores exhibit similar improvements of our extended models. Compared with baseline, the weighted average-attention model and the self-attention model obtain better scores among most types of questions. But we also notice that performance of “other" drop down after adding self-attention mechanism.

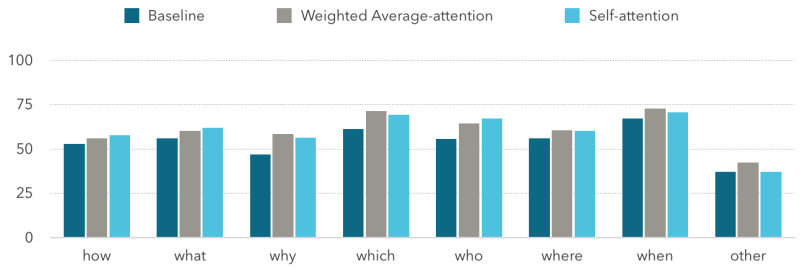


Figure 5: An break-down EM scores of different question types.

An interesting fact is that the weighted average-attention model performs better on more types of questions. To be more specific, the self-attention model outperforms others on questions that start with "how" (11%), "what"(59%) and "who"(11%), and the model with weighted average-attention reaches the highest scores on "why"(1%), "which"(6%), "where"(4%), "when"(6%), and "other"(1%).

According to the break-down scores, one "strategy" of the self-attention model is to adjust the balance and pay more attention on dominant types of question, by which it can better the overall scores. For example, the "other" only takes account tiny amount (1 %) of training set, the self-attention model does not put a lot effort on figuring out how to address it.

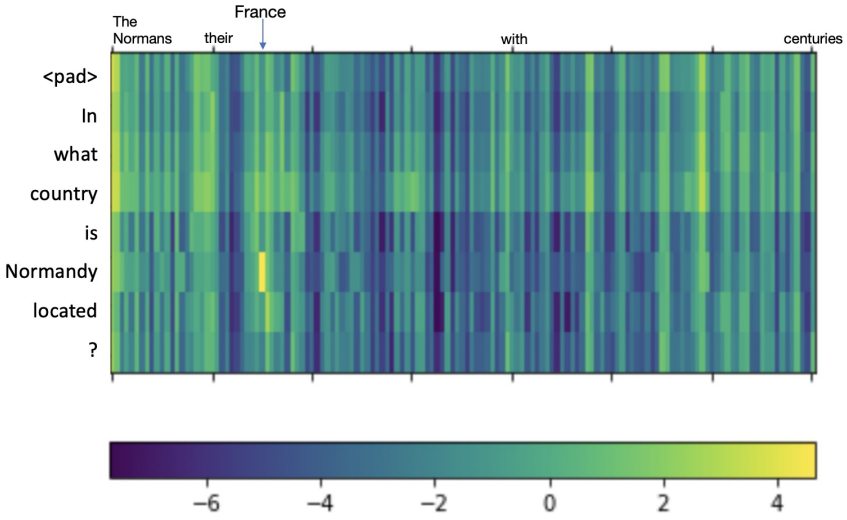


Figure 6: BiDAF attention visualization based on similarity matrix: the context is "The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France...it continues to evolve over the succeeding centuries." , the question is "In what country is Normandy located?" and the answer is "France".

We also visualize BiDAF attention using similarity matrix. Compared with SQuAD 1.1, this example from SQuAD 2.0 contains OOV token at the beginning for both context and question, which has been taken into consideration. As exhibited in Figure 6, the lighter area represents higher attention between context and question in word level, and the brightest area corresponds to "France", which is the ground truth (answer). Therefore, our model tends to obtain high attention for the most relevant content.

Table 1: Model results at each implementation level

Model	F1	EM
Baseline	60.758	57.469
Baseline + Char Embedding	62.517	59.183
Baseline + Char Embedding + Learnable Weighted Average-attention	64.734	61.049
Baseline + Char Embedding + Learnable Weighted Average-attention + Self-attention	66.241	62.679
Baseline + Char Embedding + Learnable Weighted Average-attention + Self-attention +2-Layer RNN	63.098	60.025
QANets [1]	68.013	64.251

6 Conclusion

In this project, we explored a variety of methods for question answering task on SQuAD 2.0 dataset. Character-embedding, weighted average-attention, and self-attention were implemented, and the ablation study shows that all of them contribute to improvements, compared with the baseline. The best F1 (66%) and EM (62%) scores were from our self-attention model. Besides, the experiment exhibited that adding extra layers might not help to boost the performance but in some ways it makes neural network more challenging to train. In addition, we also tried to establish a two stage model but it did not reach the performance as we expected, which shows that adding extra binary stage might not benefit the system.

7 Future Work

Looking forward, if we have more time and computational power, we will finetune our two-layer model further. Also, we can implement the recent discovery of AdaBound optimizer, which should give better performance than Adam and AdaDelta. In addition, if we are implementing the model in wild, we can also add ELMO pre-trained embeddings, which should further improve the performance of the model.

[1]

References

[1] Pranav Rajpurkar & Robin Jia & Percy Liang 2018 Know what you don't know: Unanswerable questions for squad. In Proceedings of the Association for Computational Linguistics.

[2] Mandar Joshi & Eunsol Choi & Daniel S Weld & and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension.

[3] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, andQuoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension.CoRR, abs/1804.09541, 2018. URLhttp://arxiv.org/abs/1804.095

[4] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," arXiv preprint arXiv:1611.01603, 2016

[5] A. Vaswani & N. Shazeer & N. Parmar & J. Uszkoreit & L. Jones & A. N. Gomez & L. Kaiser & I. Polosukhin. 2017 Attention is all you need. In Neural Information Processing Systems (NIPS)

[6] Nakanishi, M. & Kobayashi, T. & Hayashi, Y., 2018. Answerable or Not: Devising a Dataset for Extending Machine Reading Comprehension. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 973-983).

[1] QANet serves as a comparison guideline, we did not submit QANet result to the learderboard, the result is from our own evaluation

[7] Clark, C. & Gardner, M., 2017. Simple and effective multi-paragraph reading comprehension.