A Multi-Attention Reading Comprehension Model for SQuAD Dataset

Tong Yang[*†], **Shuyang Shi** [‡]
[†] Dept. of Electrical Engineering, Stanford University, CA, 94305
[‡] Dept. of Computer Science, Stanford University, CA, 94305
{tongy,bsnsk}@stanford.edu

Abstract

In this report, we built a multi-attention reading comprehension model for the SQuAD dataset. The model encodes words by word vectors and character-level CNN, incorporates Bidirectional Attention Flow, Coattention, and Self attention in the attention layer, uses a bidirectional LSTM modeling layer, and at last performs a max-probability answer selection in the output layer. Within 5 epochs, we achieved F1 score 75.196 and EM score 64.565 on test set for a single model. In the experiment part, we used data statistics for hyper-parameter selection, performed ablative analysis to show contributions of different components, visualized attentions to illustrate their effects, and analyzed accuracies regarding other criteria like answer length.

1 Introduction

Reading Comprehension(RC) is a popular and challenging subtask in NLP where the machines are trained to read paragraphs, comprehend its meaning, integrate with its own knowledge and come up with an answer. SQuAD[1] is a large and high-quality reading comprehension dataset which contains 100K+ questions with answers directly taken from the given contexts.

In this project, we have built a reading comprehension model aiming at achieving good performance on the SQuAD dataset. Started with the neural baseline model, we have made several improvements regarding the Encoder Layer, Attention Layer and the Output Layer. Character-level CNN is added to the Encoder Layer so that the inner structure of words and the out-of-vocabulary words can be better handled with the character embedding. In the Attention Layer, the baseline attention model is replaced with the combination of 3 different attention models, including bidirectional attention, coattention and self-attention to better capture the attention flow in both directions. Modeling Layer is introduced before Output Layer to capture the interaction of the context words conditioned on the information provided by the question. In the Output Layer, instead of predicting starting and ending position independently, we predict the ending position conditioned on that of the starting position, and select the span that maximize the product of joint probability. Last but not least, hyper-parameters such as embedding size are also tested during the implementation. With these improvements to the model, we are able to achieve F1 score 75.196 and EM score 64.565 on test set for a single model.

The paper is organized as follows: Section 2 introduces the related work in the area that achieves high performance on reading comprehension tasks followed by Section 3 which defines the problem. Section 4 introduces our model in detail, and Section 5 discusses the results and error analyses based on the experiments. Finally, we conclude our project and discuss about possible future improvements.

2 Related Work

Among the extensive the works on building deep learning systems for the SQuAD dataset[1] , most of the top models adopted some form of attention mechanism, which has improved the accuracies for this task by a large margin.

BiDAF model[2] uses a bi-directional attention flow upon its encoder RNNs, which includes both the attention from contexts to questions and from questions to contexts. Dynamic Coattention Networks model[3] has a Coattention Layer, which also involves a two-way attention between the context and the question. Unlike BiDAF, coattention involves a second-level attention computation, which means it attends over representations that are themselves attention outputs.

Microsoft proposed R-Net model[4], which in addition to normal attention, has a layer of self-attention. By directly matching the question-aware passage representation against itself, it extracts evidence from the whole passage according to the current passage word. These attention mechanism help models to focus on specific part of the passage context, and therefore improve the answer accuracy, though they may rely on other parts of the neural model to have best effect.

Typically, these models use some form of RNN on top of the attention layer for further modeling and output layer. However, there are works that developed more complicated output layer to better model this task. Instead of predicting the start position and end position (because answer is always a segment in the passage context) independently, Wang *et al*. [5] uses a answer pointer layer to first predict the start position and then uses an RNN to predict end position conditionally. Yu *et al*. [6] further elevated the interdependence of these two positions by predicting among all the candidate answer spans, the representations of which are obtained by some layers of RNN. This method can be expensive because all possible spans of appropriate length need to have a hidden representation.

3 Problem Definition

The problem for the SQuAD dataset can be described as below.

Given two word sequences $c = (c_1, c_2, \ldots, c_N), q = (q_1, q_2, \ldots, q_M)$, which are respectively context and question. The model needs to learn a function $f(c, q)$ that takes these two sequences and maps to a pair of integers $(l^{\text{start}}, l^{\text{end}}) \in \mathbb{N}^2$ that are the start position and end position of the answer to the question in the context sequence. $1 \leq l^{\text{start}} \leq l^{\text{end}} \leq N$.

4 Model Architecture

As is shown in Figure 1, the model involves 4 neural layers: encoding layer, attention layer, modeling layer, and output layer.

Encoding layer: We use character-level CNN embedding and pretrained word embedding from GLoVe, and put them through a layer of bidirectional LSTM for context and question embeddings. The weights are shared between contexts and questions. The embedding dimension of character-level CNN is fixed to be 20. GLoVe embedding dimension 100 and 300 are tried out during experiment. Since higher embedding dimension does not help with increasing the F1 score, embedding dimension of 100 is used in the final model.

To better capture the inner structure of words and better handle out-of-vocabulary word, we introduced Character-level CNN into our model. The character vocabulary contains 70 commonly used characters in ASCII together with 2 additional ones (for unknown character and padding character). We represent each word w with characters $c_1, ..., c_L$ by $[e_1, ..., e_L]$, where $e_i \in \mathbb{R}^{d_c}$ is the trainable character embedding for character c_i. A single layer CNN contains a convolution layer with window width (filter size) k, a ReLU and a max-pooling layer. The output represents the character-level encoding $\text{embed}_{\text{char}}(w) \in \mathbb{R}^f$. In our model, we applied 3 CNN layers with different filter size $k = [3, 4, 5]$, and set $d_c = 20$, $f = 100$.

Attention layer: We include 3 different attention flow in our model: bidirectional attention flow, coattention, and self-attention. Both bidirectional attention and coattention use the idea that attention

[1] https://rajpurkar.github.io/SQuAD-explorer/

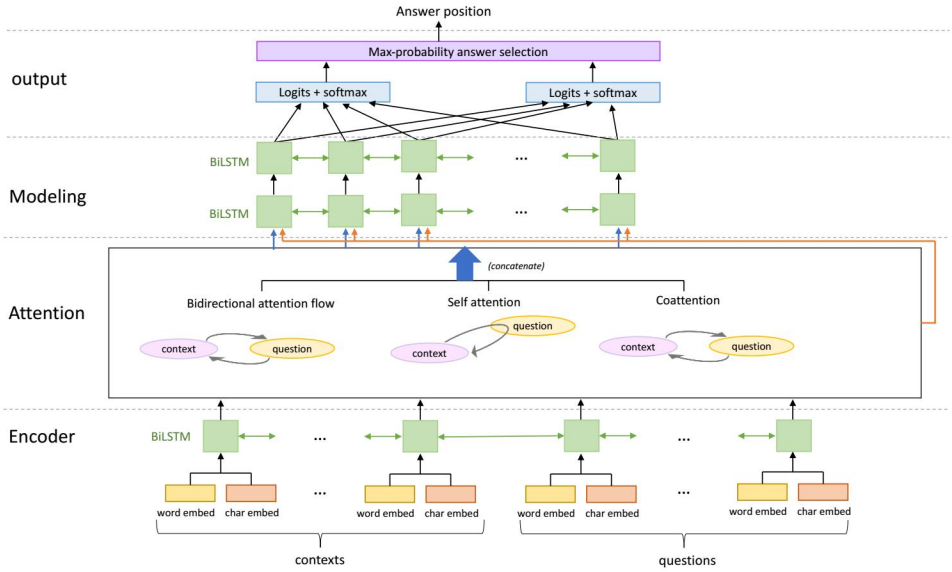


Figure 1: Overall model architecture

flows from question to context and from context to question. Self-attention addresses the context self matching, pinpointing the important parts of the context. The outputs of 3 Attention Layers are concatenated after the Encoder output before feeding into the Modeling Layer.

(a) Bidirectional Attention
For context hidden states $c_1, ..., c_N \in \mathbb{R}^{2h}$ and question hidden states $q_1, ..., q_N \in \mathbb{R}^{2h}$, the similarity matrix $S \in \mathbb{R}^{N \times M}$ represents the similarity between context and question, where $S_{ij} = \omega_{\text{sim}}^T [c_i; q_j, c_i \circ q_j]$.
Context-to-Question (C2Q) Attention outputs $a_i = \sum_{j=1}^{M} \alpha_j^i q_j \in \mathbb{R}^{2h}$, where $\alpha^i = \text{softmax}(S_{i,:}) \in \mathbb{R}^M \; \forall i \in \{1, ..., N\}$
Question-to-Context (Q2C) outputs $c' = \sum_{i=1}^{N} \beta_i c_i \in \mathbb{R}^{2h}$, where $\beta = \text{softmax}(m) \in \mathbb{R}^N$, $m_i = \max_j S_{ij} \in \mathbb{R} \; \forall i \in \{1, ..., N\}$

(b) Coattention
Coattention computes two-way attention through 2 levels of attention computation. The first layer starts with adding sentinel vectors c_ϕ, q_ϕ, which introduce the flexibility of not attend to any particular word in the input, to the end of the hidden states. A projected question hidden states $q'_1, ..., q'_M, q'_\phi$ is computed based on the question hidden states $q'_j = \tanh(Wq_j + b) \in \mathbb{R}^{2h+1} \; \forall j \in \{1, ..., M\}$. Affinity matrix $L \in \mathbb{R}^{(N+1)\times(M+1)}$ stores the affinity score for each pair of (c_i, q'_j) where $L_{ij} = c_i^T q'_j$. Context-to-Question (C2Q) Attention outputs $a_i = \sum_{j=1}^{M+1} \alpha_j^i q'_j \in \mathbb{R}^{2h}$, where $\alpha^i = \text{softmax}(L_{i,:}) \in \mathbb{R}^{M+1} \; \forall i \in \{1, ..., N\}$. Question-to-Context (Q2C) outputs $b_j = \sum_{i=1}^{N+1} \beta_i^j c_i \in \mathbb{R}^{2h}$, where $\beta^j = \text{softmax}(L_{i,:}) \in \mathbb{R}^{N+1}$
Second level attention outputs $s_i = \sum_{j=1}^{M+1} \alpha_j^i b_j \in \mathbb{R}^{2h} \; \forall i \in \{1, ..., N\}$ are concatenated with a_i before fed into a bidirectional LSTM. The final outputs of the Coattention Layer has form $\{u_1, ..., u_N\} = \text{biLSTM}(\{[s_1; a_1], ..., [s_N; a_N]\})$

(c) Self-attention
Self-attention directly matches the question-aware context representation against itself, which apparently needs a C2Q attention (we use the C2Q part in BiDAF for the model). Assume that representation is $v_1, \ldots, v_N \in \mathbb{R}^l$. We need each v_i to attend to all $\{v_1, \ldots, v_N\}$, thus $e_j^i = \nu^\top \tanh(W_1 v_j + W_2 v_i) \in \mathbb{R}$, $\alpha^i = \text{softmax}(e^i) \in \mathbb{R}^N$, $a_i = \sum_{j=1}^{N} \alpha_j^i v_j \in \mathbb{R}^l$, where W_1, W_2 are weight matrices and ν is a weight vector. After that, we feed $[a_i; v_i]$ to a bidirectional LSTM to obtain the final representation.

Modeling layer: For the modeling layer, we use 2-layer BiLSTM, which is borrowed from model BiDAF[2].

Output layer: The output layer contains two steps. In the first step, we use a fully connected layer, a ReLU activation, and a softmax to generate probability distributions $p^{(\text{start})}, p^{(\text{end})}$ for start position and end position respectively. In the second step, we perform a max-probability answer selection, and choose

$$(l^{(\text{start})}, l^{(\text{end})}) = \underset{i,j,i \leq j < i+L}{\arg\max} \{p_i^{(\text{start})} p_j^{(\text{end})}\}$$

as the starting and ending position of the answer, where L is some parameter to limit the length of the answer segment. This step is similar to the one in Chen *et al.*'s work[7], and can be achieved in linear time by a monotone queue.

5 Experiments

In this section we analyze the dataset, describe the model configuration and accuracy, and demonstrate some further analyses, which involve ablative analysis for model components, attention visualization in sentence examples, and model accuracies regarding answer lengths.

5.1 Data analysis

The 100K+ question-answer pairs in the dataset is split into $\%80$ training set, $\%10$ dev set, and $\%10$ hidden test set. Since we use both word and character embedding in the model, we need to pad the short words/characters and truncate those that are too long. In order to help better choose the threshold, we started with the analysis of the training data statistics.

Figure 2 shows the histogram of training context lengths, question lengths, and word length in terms of characters in both context and question. Figure 3 shows the histogram of answer length and the starting and ending position of answer in the context. Based on the statistics we obtained, we determined the parameters context_len $= 400$, question_len $= 30$, word_len $= 16$, max_answer_len $= 15$,.

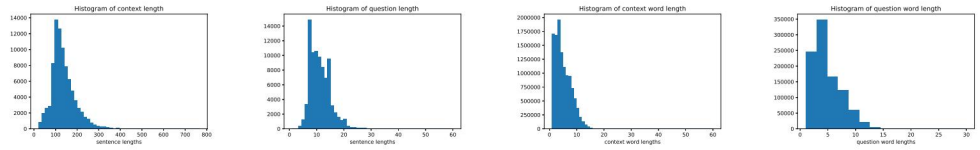


Figure 2: Context and Question Statistics

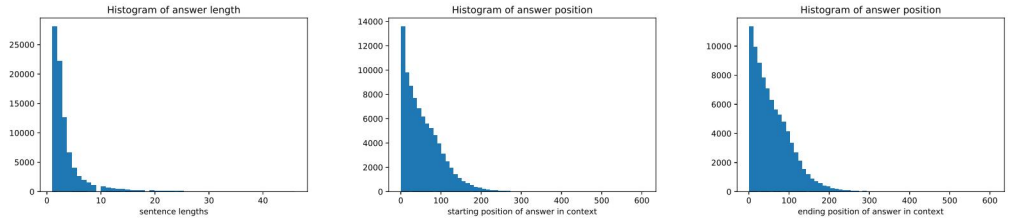


Figure 3: Answer Statistics

5.2 Configurations

In experiments, we set learning_rate $\alpha = 0.001$, dropout_rate $= 0.15$, max_gradient_norm $= 5.0$, hidden_size $= 200$, context_len $= 400$, question_len $= 30$, word_len $= 16$, max_ans_len $= 15$, embedding_size $= 100$. Due to the memory limit of Tesla M60 GPU, we use batch size 20.

5.3 Model Accuracy

F1 score and Exact Match (EM) score are used to evaluate model accuracy. For each question, precision is computed as the number of correct words divided by the length of predicted answer, while recall is computed as the number of correct words divided by the length of the ground truth answer. The F1 score is computed accordingly per question and then averaged across all questions. EM score is the number of questions that are answered exactly the same as the ground truth divided by total number of questions. Better models are expected to have higher F1 and EM scores.

In Table 1, we report the accuracies of our model on test set, and compare it to some state-of-art works. Apparently, our model has very competitive accuracy on this dataset when compared to single model approaches, and some more hyper-parameter tuning can possibly further elevate the accuracy. (More epochs can help our model achieve higher scores but we only report 5-epoch result here due to the delay of CodaLab test leaderboard)

Model	**Test F1**	**Test EM**
our model (5 epoch)	75.196	64.565
R-Net (single model)	80.7	72.3
BiDAF (single model)	77.323	67.947
Dynamic Coattention Networks (single model)	75.896	66.233

Table 1: Model accuracies

5.4 Ablative Analysis

For better understanding of different components of our model, we did an ablative analysis on the dev dataset, where we measure the accuracies while we remove one component by another in the model until it degrades back to the public baseline. From Table 2 we know that attentions and LSTM modeling layer can improve prediction accuracy by a considerable margin, while techniques like max-probability selection and character-level CNN only have small improvements. Also, the order of this analysis can have an impact on this analysis, thus the actual impact of self-attention may be more than what is showed here.

Model	**Dev F1**	**Dev EM**
our model	74.961	64.503
without char-level CNN	74.557	63.983
without max-probability selection	74.44	64.021
without self-attention	73.915	63.605
without answer length limit	72.321	62.119
without coattention	70.31	59.782
without modeling layer	49.137	39.328
without BiDAF attention (i.e. baseline)	43.522	34.276
baseline + coattention	64.478	54.011

Table 2: Ablative analysis for our model

5.5 Attention Visualization

Figure 4 plots the coefficients for context words regarding different types of attentions, for a sample question.

Question what kind of arches does norman architecture have ?

Context norman architecture typically stands out as a new stage in the architectural history of the regions they subdued . they spread a unique romanesque idiom to england and italy , and the encastellation of these regions with keeps in their north french style fundamentally altered the military landscape . their style was characterised by rounded arches , particularly over windows and doorways , and massive proportions .

Answer rounded

Figure 4a shows BiDAF similarity matrix of the example above. Darker color represents higher similarity score. From the plot we can see that same words in context and question have highest similarity score, context word **rounded** and question word **arches** have relative high similarity scores. The attention distributions of BiDAF are based on this similarity. Figure 4b shows the attention distribution over context location when performing Question-to-Context Attention in BiDAF. This distribution shows that Q2C attention extract words in the context that are highly related to the question. Similarly, C2Q attention extract highly related question words for each context word.

Figure 4c shows the heatmap of the C2Q and Q2C attention distribution of Coattention model. Darker color maps to higher scores. Each horizontal line depicts the importance of context words for a certain question word; Each vertical line shows the importance of question words for a certain context word. We can see that word **kind**, **of** and **arches** in the question are relatively important to many of the context words, and this is consistent with human sense when we analyze the question: "what *kind of arches* does norman architecture have ?"

Figure 4d visualizes distribution of self-attention for this example (darker color maps to higher scores). From the plot we can see that question-aware context can effectively find critical words in itself ("rounded arches"), and for them the attention scores are much higher. Within the attention scores for these words, more relevant parts have higher attention scores. This is consistent with the expectation that self-attention extracts important parts in context itself regarding to the question.

These illustrations demonstrate the effectiveness and different focuses of these attention mechanisms. Although they all make mistakes sometimes, combining them together will hopefully give a satisfying prediction result.

5.6 Accuracies by Answer Length

Answer length	Percentage	Average F1
1-5	94.40%	0.7514
6-10	4.74%	0.6663
11-15	0.71%	0.5530
16-20	0.12%	0.5128
21-25	0.02%	0.4334

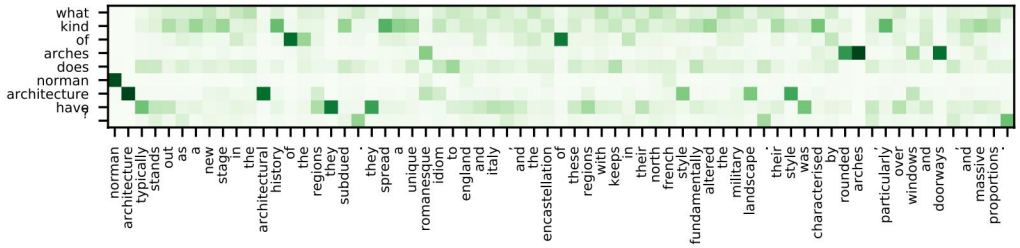
Table 3: Model accuracy on different length of ground-truth answer (dev set)

Table 3 demonstrates the model accuracies for different questions in dev set, which are classified by the length of ground truth answer. We can see that the accuracy for shorter answers are much higher than longer answers, and answers of length more than 15 generally have F1 no more than 0.5, because of the length limit in max-probability answer selection process. According to the length distribution, it is reasonable to emphasize on shorter answers.

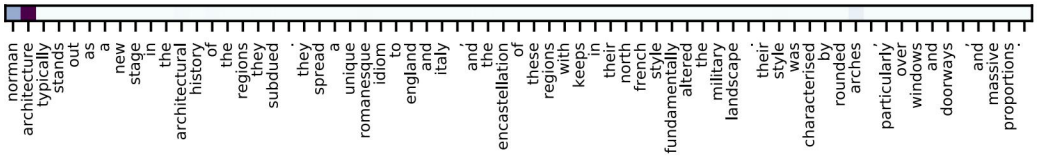
Figure 5 shows the distribution of different answer length both in prediction and ground truth answers for dev set. Generally the prediction answer lengths are very close ground truth answer lengths, and for some question the predicted answers are longer. The prediction line does not have answer longer than 15 because of the length limitation in output layer.

6 Possible Improvements

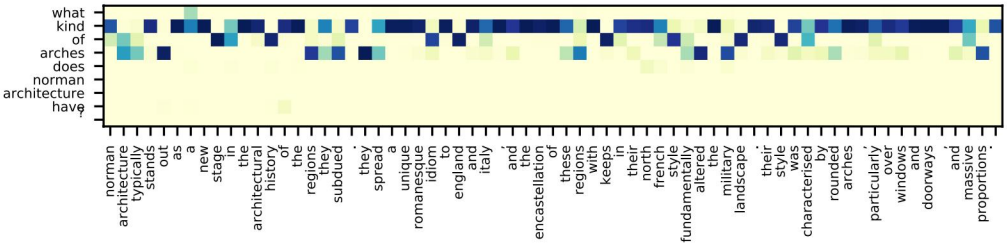
Further improvement can be made on different aspects. For one thing, instead of computing start and end position probabilities separately, a more intuitive way is to represent different candidate spans by vectors, and choose answer span accordingly. The challenge comes from the number of candidate spans, which requires a large neural network that is expensive regarding both time and space. For another, as an important part of the model, different attention mechanisms may have a better way to interact with each other, rather than simply being concatenated into long vectors.



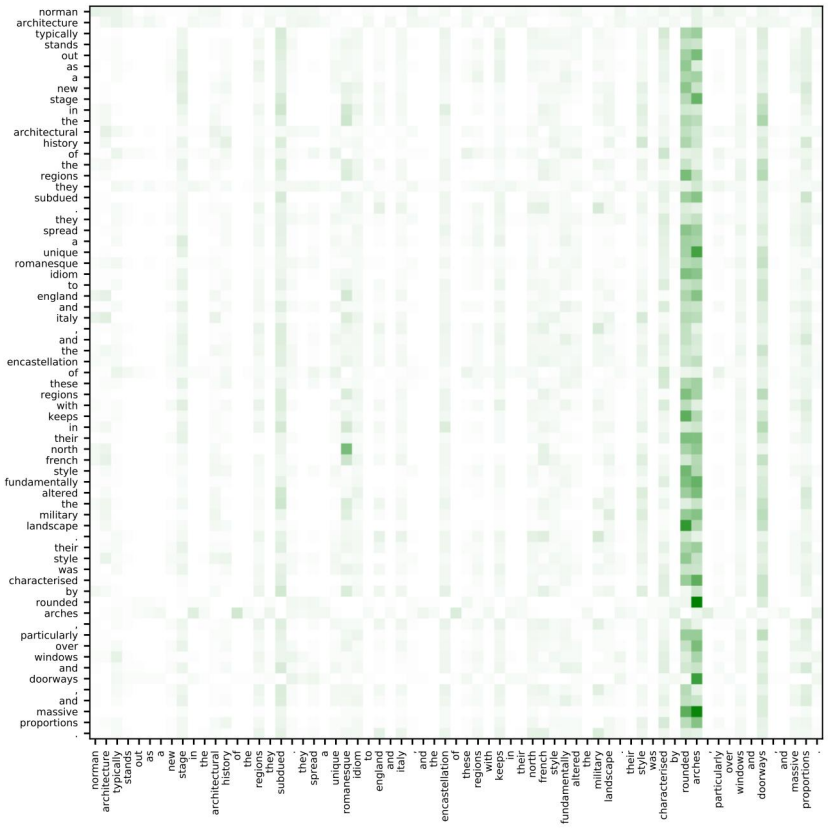
(a) Example of BiDAF Similarity Matrix



(b) Example for BiDAF (Q2C)



(c) Example of Coattention



(d) Example of Self-attention

Figure 4: Illustration of attentions

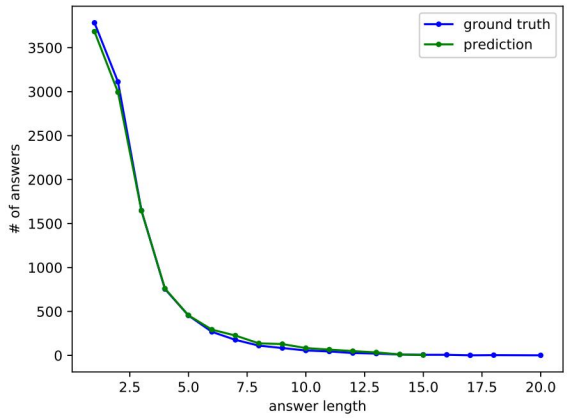


Figure 5: Answer length distribution

7 Conclusions

In this project, we built a reading comprehension model by incorporating different attention mechanisms, constructing suitable modeling layer, and performing max-probability answer selection. The implemented model achieves competitive accuracies (single model) on the SQuAD dataset. Ablative analysis demonstrate the contribution of different components; visualizations show that the attention mechanisms of the model is able to select correct context / question locations, and that they can possibly have different emphases. Analysis on answer length further shows the effectiveness of the model.

References

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

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

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

[4] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198, 2017.

[5] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016.

[6] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996*, 2016.

[7] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.