# Question Answering on SQuAD2.0

**Mengyu Li**
Department of Statistics
Stanford University
lmy18@stanford.edu

**Boyao Sun**
Department of Civil and Environmental Engineering
Stanford University
boysun@stanford.edu

**Qiwen Wang**
Department of Computer Science
Stanford University
qwang26@stanford.edu

## Abstract

The purpose of this project is to investigate the performance of Recurrent Neural Network Natural Language Processing models on question answering tasks. The data used for this project is the Stanford Question Answering Dataset (SQuAD) v2.0 [1]. We choose to experiment on non-PCE (Pre-trained Contextual Embeddings) models, do ablation studies, and figure out which features could help the model understand the texts the best. The experiments start from applying Bi-Directional Attention Flow (BiDAF)[2] on the dataset, and the results of those models will be the baseline of the project. We make modifications and adding more features on BiDAF model to achieve better performance on SQuAD.

## 1 Introduction

Question answering is one of the most challenging tasks in Natural Languages Processing, where the machine tries to comprehend a given passage of text and correctly gives an answer to the questions. To measure the performance of the various efforts toward the task, Stanford NLP group released Stanford Question Answering Dataset (SQuAD), which is a reading comprehension dataset consisting of 100,000 questions and their corresponding contexts based on Wikipedia articles, with answers can be directly generated from the span of the context. The dataset we choose to work on for this project is its successor, SQuAD2.0, which adds 50,000 new unanswerable questions and requires us to generate no answer predictions. PCE methods, such as ELMo[3] or BERT [4], are widely used on question answering tasks, and the modifications of BERT reach the state-of-art performance on SQuAD2.0. Although non-PCE models didn't work so well as the PCE ones, we still choose a non-PCE model as our baseline because they are more efficient to implement given our limited computation resources. In our experiment, we made several modifications to the original BiDAF model[2] to improve its performance, including adding character-level embeddings and word features in the embedding layer, applying self-attention layer, changing the type of RNN layer, optimizer, and learning rate to improve the result. The approach and results will be introduced detailedly in Section 3 and 4.

## 2 Related Work

PCE methods provide better word embeddings including the information for surrounding contexts. ELMo [3] trains a two-layer bidirectional LSTM on a large corpus, and the pre-trained layer could be used as the embedding layer in multiply NLP tasks. BERT takes the advantage of Transformers[5],

.

outperforms ELMo, and achieves the state-of-art of many NLP tasks, including question answering. Since PCE methods requires fine-tuning a large neural network.

In our project, we use the non-PCE method, BiDAF as our base model and baseline. BiDAF is a bi-directional network for question answering that represents the interactions of query and context at different levels of granularity. BiDAF mainly consists of 6 layers: word embedding layers; a contextual embedding layer, a bi-directional LSTM on both query and context; an attention flow layer that compute the flow from context to query and vice versa; a modeling layer, a bi-directional LSTM that encodes context words; and an output layer. Besides BiDAF, R-Net[6] introduces a self-attention layer, and QANet[7] applies an Encoder Block inspired by Transformers. They both achieve good results on SQuAD, and can be easily applied on to BiDAF.

DrQA [8] uses Wikipedia as sources to do question answering tasks. DrQA proposes several ideas to boost the results, such as fine-tuning the most frequent question words or adding a exact matching feature to the word embedding. Several modifications on our model are inspired by DrQA. We will provide more details of those methods in the approach section.

# 3 Approach

## 3.1 Baseline Model

The given baseline BiDAF model has the following layers. A word embedding layer provides GloVe vector representations. The embedded word vectors are fed into a bidirectional LSTM encoder layer to get the hidden states. Note that the questions and texts share the same encoder layer. The question hidden states and context hidden states are fed into an attention layer, which combines the information from contexts and questions. The combined information is then pass into another bidirectional LSTM encoder. The output hidden state of the encoder is forwarded to the last output layer. The output layer firstly uses a fully-connected layer and a softmax layer to predict the start pointer position, and then use another LSTM encoder, a fully-connected layer, and a softmax layer to predict the end pointer position.

## 3.2 Modified BiDAF Model Overview

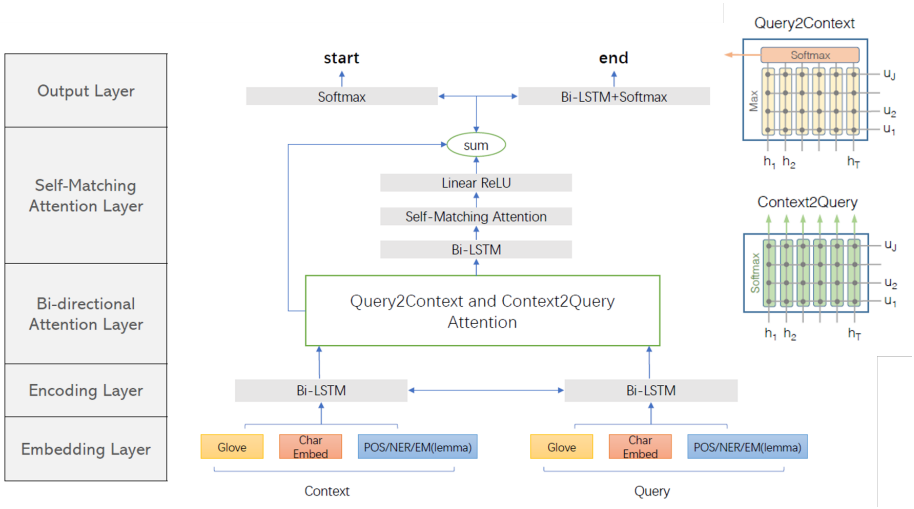



Figure 1: Model Architecture

We modified BiDAF model by adding a character level embedding, word features and a self-attention layer. The architecture of our model is shown in Figure. 1. The following section provides details about each layer.

### 3.3 Word Level and Character Level Embedding

Based on the BiDAF model implemented by CS224N staff team, we realize that the character-level word embedding is not included in this implementation. Character-level word embedding has been proved to be effective for NLP tasks: it represents the word meaning clearer, handles unknown words efficiently, and is presented by the original BiDAF paper. Therefore we included an Character Embedding Layer in our model.

To implement the character-level word embedding, after mapping the character indices corresponding to each word, we initialized an embedding layer for the characters, and applied an CNN layer followed by max pooling to get the desired char-level embedding. We then concatenated the character-level embedding and the word-level embedding from pretrained GloVe vectors to obtain a new representation for words[9]. Since we changed the length of the embedding, the dimensions of the parameters that need to learn in contextual embedding layer, the attention flow layer, the modeling layer and the output layer are changed to dimensions in terms of the new embedding dimension.

The word level embedding and character level embedding are concatenated together as the final representation of each word, and the result vector is fed into a Highway Encoder.

We have also added word features in additional to the GloVe vector to make the word level embedding more powerful, as inspired by DrQA. We first added two feature indicators to represent if the word or its lemma form could be exact matched to a question word. To be specific, if the word in the context can be exactly matched to a word in its corresponding question, we set its feature indicator to be 1, otherwise it remains 0. By this simple approach, we can build connections between context and question words. We also consider the part-of-speech (POS) and named entity recognition (NER) of each word. An embedding layer is used to encode POS and NER information, and the output is concatenated with the word embedding.

### 3.4 Encoding Layer

After the contexts (C) and questions (Q) are seperately embedded, we used a bidirectional LSTM encoders to encode the embedding vectors of context and question into the new representation $\mathbf{H} = [h_1, \cdots, h_N]$ and $\mathbf{U} = [u_1, \cdots, u_M]$, accordingly, where $N$ is the maximum word length of a paragraph, and $M$ is the maximum word length of a question.

$$h_t = biLSTM(h_{t-1}, c_t) \tag{1}$$
$$u_t = biLSTM(u_{t-1}, q_t) \tag{2}$$

### 3.5 Bidirectional Attention Layer

Attention layers are designed to fuse the context representation and question representation and link the context and questions together. The first step of attention layer is to get a similarity matrix $S$, where $S_{ij}$ represents the similarity between the $i$-th word in context and the $j$-th word in question. The similarity matrix $S$ is defined as

$$S_{ij} = \mathbf{w}_{sim}^T [h_i; u_j; h_i \circ u_j],$$

where $c_i \circ q_j$ is an elementwise product and $\mathbf{w}_{sim}^T$ is a weight vector.

After obtaining $S$, we compute context to question attention (C2Q) and question to context attention (Q2C). C2Q attention outputs $a_i$ are computed by taking row-wise softmax on columns of S to get attention distributions and then take weighted sums. In equation,

$$\bar{S}_{i,:} = softmax(S_{i:}) \tag{3}$$
$$a_i = \sum_{j=1}^{M} \bar{S}_{i,j} u_j \tag{4}$$

And Q2C attentions $b_i$ are computed by:

$$\bar{S}_{:,j} = softmax(\bar{S}_{:,j}) \tag{5}$$

$$S' = \bar{S}\bar{\bar{S}}^T \tag{6}$$

$$b_i = \sum_{j=1}^{N} S'_{i,j} c_j \tag{7}$$

At last, we combine our attention outputs to generate a question-aware context representation:

$$g_i = [h_i; a_i; h_i \circ a_i; h_i \circ b_i]$$

where $\circ$ denotes element multiplication.

### 3.6 Self-Matching attention Layer

The C2Q and Q2C layer uses a bi-directional attention flow to generate a question-aware context representation, but this is not sufficient, we still face problems in learning long-term dependencies. Therefore, we propose a self-maching attention layer inspired by R-Net[6] to learn a self-aware context representation by matching the context against itself. In this way, we can effectively encode the information of the whole paragraph and refine the representation.

In particular, we first use a bi-directional LSTM to refine the sequence of vectors. Then, we compute our similarity matrix similarly to the one we computed in the BiDAF attention layer, but subtracting a large number when the word is matched with itself, so that the similarity of the diagonal approaches to 0 after applying the softmax. After that, we compute the attention vectors and sum together the question and self-aware representations.

$$g'_i = biLSTM(g'_{i-1}, g_i) \tag{8}$$

$$S'_{ij} = w^T_{self_sim} \cdot [g'_i; g'_j; g'_i \circ g'_j] - 10^7 I_{i=j} \tag{9}$$

$$\bar{S'}_{i,:} = softmax(S'_{i:}) \tag{10}$$

$$d_i = \sum_{i=1}^{N} \bar{S'}_{i,:} g'_i \tag{11}$$

$$\tilde{g}_i = g_i + ReLU(w_d \cdot [d_i; g_i; d_i \circ g_i + b_d) \tag{12}$$

### 3.7 Output Layer

In the output layer, we first use a bi-directional LSTM to refine the representations, then we calculate the probability of the answer span $p_{start}$ and $p_{end}$:

$$\tilde{g}'_i = biLSTM(\tilde{g}'_{i-1}, \tilde{g}_i) \tag{13}$$

$$p_{start} = softmax(W_{start}[\tilde{G}]) \tag{14}$$

$$p_{end} = softmax(W_{end}[\tilde{G'}]) \tag{15}$$

$$loss = -logp_{start}(i) - logp_{end}(j) \tag{16}$$

where $W_{start}$ and $W_{end}$ are learnable parameters, and the columns of $\tilde{G}$ and $\tilde{G'}$ are $\tilde{g}_i$ and $\tilde{g'_i}$.

To finally produce our predictions, we minimize our loss using the Adadelta optimizer. It turns our that Adadelta performs much better than using Adabound and AdaMax in this problem. We also restrict the span of the answer to be less or equal to 12 ($i < j \leq i + 12$).

## 4 Experiments

### 4.1 Data

The training data for the SQuAD dataset is consisted of two parts: Paragraph and Question. Our training set, dev set, and test set contain 129,941, 6078, and 5915 paragraph and question pairs.

The statistics for the training data is collected in assist of understanding the models and data. We observe that the length of context is right tilt, with a mode of 120 characters. The number of words in answers is also right tilt, where most of the questions that are possible to answer have less than 3 words in the answers. Furthermore, the longest answer in the training set have 17 words. Using the distribution of the length of the answer helps us finetuning the model parameter for the maximum answer length. The distributions for the question length for those that are possible to answer and impossible to answer are roughly the same. We also calculate the frequency of iterrogative words. Surprisingly, the frequency of having "what" is 4 times more frequent than the second most frequent iterrogative words, "who". By transferring the iterrogative words as a feature in a model, we could improve the metrics of our model.

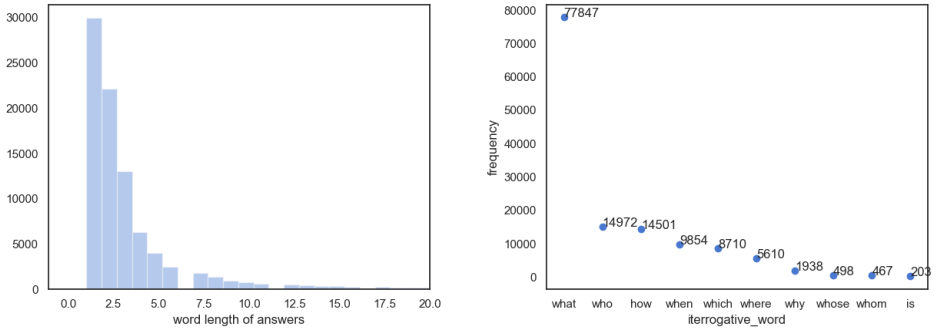



Figure 2: Features of SQuAD2.0

### 4.2 Evaluation method

The evaluation metrics we used are F1, EM, and AvNA scores. EM score is a rather strict metric, which reflects the 1/0 accuracy on whether our answer exactly matches one of the 3 true answers. While F1 score takes each gold answer as bags of words and doesn't require choosing the exact same span as human's, which is seen as more reliable. It is computed as follows:

$$F1 = \frac{2PR}{P+R}$$

where $P = \frac{tp}{tp+fp}$, $R = \frac{tp}{tp+tn}$[1]

Since SQuAD 2.0 is composed of questions that can or cannot be answered, we use AvNA score as a measurement for the classification accuracy of Answer and No Answer. Formally,

$$AvNA = 100 \times \frac{\sum_i \mathbb{I}[pred\_has\_ans_i = gold\_has\_ans_i]}{\text{the number of total data}},$$

where $pred\_has\_ans_i$ and $gold\_has\_ans_i$ are boolean variables indicating whether the prediction/gold answer is "has answer".

### 4.3 Experimental details

In our implementation described in Section 3, we limited the maximum number of words in a paragraph to 400, the number of words from a question to 50, and the maximum length of a predicted answer to 12 based on the length distribution of the training data. During training, we use Adam Delta with learning rate 0.2, and batch size 100 and the 0.999 decay rate for exponential moving average of parameters. We apply dropout with rate 0.2 throughout all bi-LSTM layers only at the training time. Training a single epoch on Azure NV12 machine takes about 50 minutes with the above parameters.

[1]P: Precision, R: Recall, tp: true positive, fp: false positive, tn: true negative

### 4.4 Results

In this session, we compare our models with the baseline BiDAF model, analysis the effect of layers and features, and visualize the training curves of the models.

Table 1: Model Performance on SQuAD v2.0 dev set (Non-PCE)[2]

| BiDAF Model | F1 | EM | AvNA |
|---|---|---|---|
| Baseline | 58.64 | 55.57 | 65.18 |
| With Adabound | 58.42 | 54.90 | 65.64 |
| With character embedding* | 62.19 | 58.66 | 68.74 |
| With character embedding, POS/NER* | 61.56 | 58.21 | 68.34 |
| With character embedding, exact match* | 63.31 | 60.52 | 69.76 |
| With character embedding, self attention* | 61.77 | 61.54 | 69.25 |
| With character embedding, exact match, POS/NER, self attention* | 64.49 | 61.55 | 71.38 |
| With character embedding | 64.00 | 60.71 | 70.06 |
| With character embedding, exact match | 64.69 | 61.54 | 71.20 |
| With character embedding, exact match, self attention | **66.74** | **63.57** | **73.20** |

Table 2: Model Performance on SQuAD v2.0 test set (Non-PCE)

| BiDAF Model | F1 | EM |
|---|---|---|
| With character embedding, exact match, POS/NER, self attention* | 62.105 | 58.884 |
| With character embedding, exact match, self attention | 63.931 | 60.626 |

#### 4.4.1 Model Performance

To understand the performance of our model, we use F1, EM, and AvNA to evaluate our models summarized in Table. 1. Our model with full features achieves 66.74 F1, 63.57 EM, and 73.20 AvNA, around 8% improvement from the baseline model on the dev set, 63.931 F1, 60.626 EM on the test set. With the modification we suggested, the model outperforms the BiDAF model, and the improved result is what we expect. Our approach that adopts the self attention and the additional word features are effective.

To further explore the detail performance of the model, we summarize the F1, EM scores along with the answer lengths and the question types. We observe that when the length of the answer grows, the F1 score drastically drops. The fact is not obvious from the global F1 score. But since most of the questions have shorter answers, the predictions on the longer answers don't effect the overall performance much.

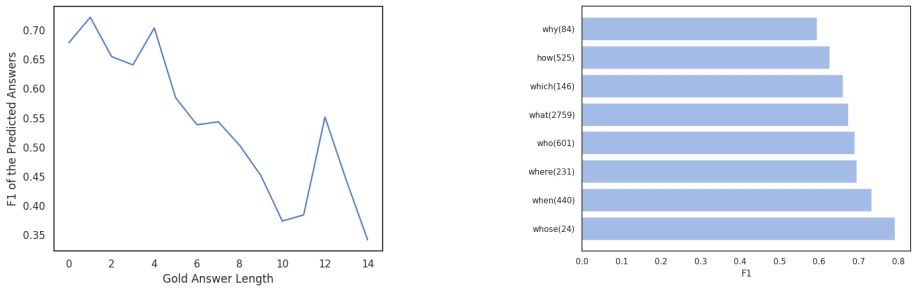
We have also group the questions by the most popular question words. In Figure. 3b, questions start with "whose" and "when" are answered correctly with higher probability. With POS/NER features, the type of the words can be recognized, and is helpful to answer the questions related to the time and the name of a person. On the other hand, the "why", and "how" type questions have relatively lower F1 scores, probably because these type of questions require deeper understand of the paragraph.

## 5 Analysis

### 5.1 Error example analysis

Arond 27% of the errors in our model are the wrong classification of Answer vs No Answer prediction. Here we examine two specific samples to analyze the reasons for it.

[2]*: Due to the computational limit on Azure NV6, we added a linear layer followed by ReLu activation after the attention layer to reduce the hidden layer dimension. These model performs relatively worse than the models without dimension reduction. We mark these models with *.




(a) Average F1 score for Answers with Different Length (b) Average F1 score for Questions of Different Type

Figure 3: Average F1 Score for Answers of Different Length and Different Types of Questions

#### 5.1.1 Predict Answer for No answer questions

**Context:**By far the most famous work of Norman art is the Bayeux Tapestry, which is not a tapestry but a work of embroidery. It was commissioned by Odo, the Bishop of Bayeux and first Earl of Kent, employing natives from Kent who were learned in the Nordic traditions imported in the previous half century by the Danish Vikings.
**Question:** What is the oldest work of Norman art?
**Answer:** N/A
**Prediction:** Bayeux Tapestry

In this example, our model predicts an answer while there is no answer. By comparing the question to the context, we can see that question "the oldest work of Norman art" resembles the context sentence "the most famous work of Norman art" a lot, with difference in only one word "oldest" vs "most famous". Our model uses question-aware representation and exact matches for words both in the context and the question, and thus cannot tell the slight difference between highly resemble sentences and predicts wrong answer.

#### 5.1.2 Predict No Answer for questions have answer

**Context:**The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.
**Question:** What was the Italian title of Polo's book?
**Answer:** Il milione
**Prediction:** N/A

In this example, the keywords "Italian title" and "book" are not directly mentioned in the context. To correctly generate the answer requires logical reasoning like human beings, which our model cannot achieve currently.

### 5.2 Abalation analysis

We further explore the effect of each modification to the network. One of the most powerful features is the character embedding. It significantly improves the baseline model by 6% F1 and 4.5%. The main reason for such a great improvement is because the character embedding can represent the OOV (Out-of-Vocabulary) words, whereas the word embedding treat all the OOV words the same. On the other hand, the POS/NER feature doesn't improve the model much. If we consider the POS/NER of the nearby words, the feature could probably be more effective to the model. We also observe that the combination of using the exact match indicator and lemma match indicator increases 1% of the

F1, EM, AvNA score. With the indicators, the model can better capture the alignment between the context and question.

## 6 Conclusion

In this paper, we implemented a non-PCE Question Answering model on SQuAD 2.0 and achieved several improvements compared with the original BiDAF model. We modified our baseline model by adding character embedding, word features (POS/NER/exact match in lemma form) and self-attention layer. We also tried different optimizers, drop-out rates, hidden layer dimensions and alternate types of RNN. Our model has significant improvement compare to BiDAF model. Through further investigation, the model is not performing well when the question resembles the context, but the answer is not in the context, and when more complicated logical reasoning is needed. Future work could involve penalizing the key words that are dissimilar to the context, or considering a deeper network with more self attention layers.

## 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] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237, 2018.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[6] 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.

[7] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

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

[9] Richard Socher Jeffrey Pennington and Christopher D. Manning. Glove: Global vectors for word representation. 2014.