Question Answering using BiDAF and DrQA

Rui Fu
Stanford University
Stanford, CA 94305
ruif@stanford.edu

Xuan Yang
Stanford University
Stanford, CA 94305
xuany@stanford.edu

Abstract

We utilized character-level embedding, bi-directional attention flow, DrQA mode and smarter span selection. We achieved 77.566 and 68.058 for the dev F1 and EM scores, and 76.83 and 67.114 for the test F1 and EM scores.

1 Introduction

SQuAD is a reading comprehension dataset that provides a measure for the ability of a machine to understand text. To develop a deep learning system for SQuAD, we utilized various techniques described in previous research papers, and experimented with the values of hyperparameters to achieve a better performance.

2 Model

To improve the performance of our question-answer system against the baseline model, the following approaches are utilized.

2.1 Character-level CNN

In addition to word embedding layer, we add a character embedding layer to map each word into a high-dimensional vector space. This embedding layer is helpful as it can infer information about the words that not presented in GloVe vectors. We adopted the same Convolutional Neural Network (CNN) as Seo et al. proposed [1], to compute the character-level embedding of each word.

The word and character embedding vectors are concatenated through a two-layer Highway Network (Srivastava et al., [2]).

2.2 Bidirectional attention flow

We adopted the same structure of a bidirectional attention flow layer as described in [1]. Context hidden states and question hidden states are used to calculate a similarity matrix S. Then taking the row-wise softmax of S yields attention distribution over the question locations α, and the softmax over the max of each row of S yields attention distribution over the context locations β. We then compute Context-to-Question attention and Question-to-Context attention, and concatenating them with the context hidden state to obtain blended representations of context b_i.

2.3 Additional input features

To boost the performance of our model on SQuAD, we added some additional features alongside word and character vectors to represent context tokens [3]. Denote the context tokens as c_i for $i \in \{1, ..., N\}$, and the corresponding question as q, the additional feature vectors we incorporated are described below.

Table 1: Results of our model with various aq_mapping_size's

mapping size	Train F1	Train EM	Dev F1	Dev EM
30	0.8252	0.701	0.7071	0.5578
100	0.8712	0.767	0.6941	0.5457

- *Exact Match*: $f_{exact\_match}(c_i) = \mathbb{I}(c_i \in q)$. This is a binary feature to indicate whether c_i can be exactly matched to a word in q.
- *Token features*: $f_{token}(c_i) = TF(c_i)$. We calculated the normalized term frequency (TF), which to some extent reflects the relative importance of each token.
- *Aligned question embedding*: $f_{align}(c_i) = \sum_j a_{i,j}\mathbf{E}(q_j)$. $\mathbf{E}(\cdot)$ is the word vector obtained in section 2.1. For each pair of c_i and q_j an attention score $a_{i,j}$ is calculated as:

$$a_{i,j} = \frac{\exp(F(\mathbf{E}(c_i))F(\mathbf{E}(q_j)))}{\sum_k \exp(F(\mathbf{E}(c_i))F(\mathbf{E}(q_k)))},$$

  where $F(\cdot)$ is a single fully connected layer with RELU nonlinearity, the dimension of its output is a tunable hyperparameter, we call it aq_mapping_size. This feature identifies similar question words for each context word, and supplements the exact match feature as a soft alignment.

We concatenate the above features with blended representation b_i to obtain the final representation of context tokens. They are passed as the input to a two-layer bidirectional LSTM network to obtain m_i. Finally, we apply a downprojecting linear layer and softmax function to obtain a distribution of the start index p^{start} over the context locations, and a single LSTM then downprojecting linear layer to obtain a probability distribution of the end index p^{end}.

2.4 Smarter span selection at test time

We implemented a better rule as in [3] to predict start index and end index simultaneously, by maximizing $p^{start}(i)p^{end}(j)$ where (i, j) satisfies $i \leq j \leq i + M$. The maximum length of answer span M is a trainable parameter.

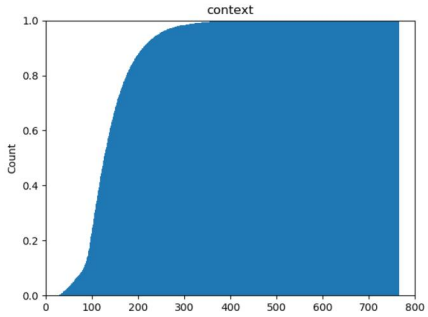
3 Experiments

We generated the cumulative distribution of context and answer length in the training data, as shown in Figure 1(a) and Figure 1(b). We found that length of context paragraph in more than 98% of the training dataset is below 300, thus we set the hyperparameter context_len to 300. Also this is the maximum memory footprint that can be used in our GPU. Another observation we made is answer length is smaller than 16 in 90% of the training cases, thus we set the hyperparameter M to be 15.

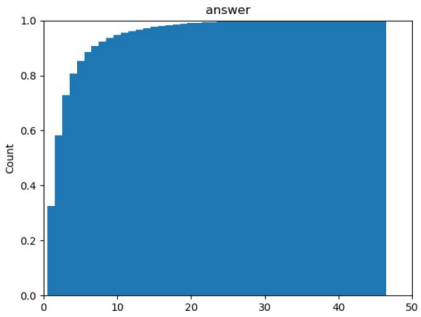
We conducted a set of experiments to choose the optimal hyper-parameters for our model, including the aq mapping size and dropout rate. Table 1 illustrates the impact of the aq_mapping_size on the overall performance. With aq_mapping_size of 100, the training performance is improved, but the performance of the dev set is degraded, implying the system is over-fitting with a larger aq_mapping_size. Therefore we choose aq_mapping_size to be 30, since it generates better performance and requires less memory.

From Table 2, we can see with increasing the dropout rate, both F1 and EM scores on the test and dev sets are improved at first, due to less overfitting. However, when exceeding certain dropout rate - in our case 0.5, the performance starts to decay, as the system becomes more difficult to train. Based on the above observations, we choose the dropout rate 0.3 for our model.

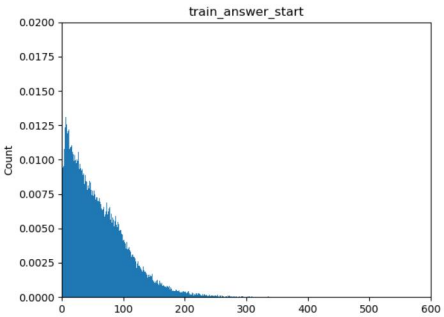
With using the above techniques and hyper-parameters, our final F1 and EM score on the test and dev leader board are shown in Table 3.



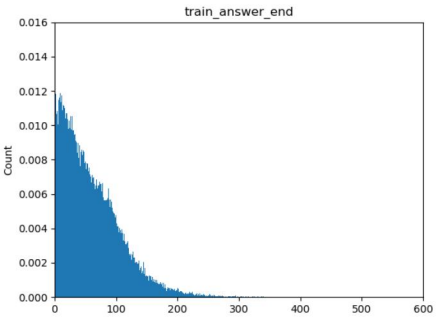
(a) Cumulative distribution of context lengths in training set



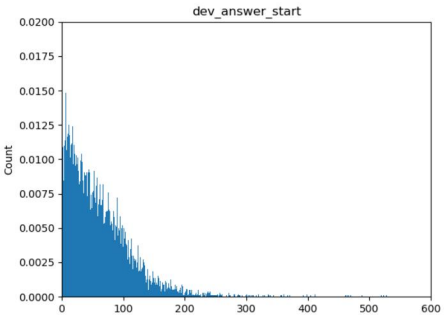
(b) Cumulative distribution of answer lengths in training set



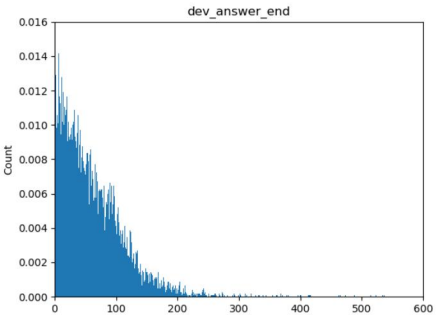
(c) Histogram of answer start position in training set



(d) Histogram of answer end position in training set



(e) Histogram of answer start position in dev set



(f) Histogram of answer end position in dev set

Figure 1: Distributions of context and answer lengths, answer start and end positions

Table 2: Result of our model with various dropout rates

dropout rate	Train F1	Train EM	Dev F1	Dev EM
0.15	0.8252	0.701	0.7071	0.5578
0.2	0.8405	0.704	0.7085	0.5605
0.3	0.8866	0.753	0.7097	0.5588
0.5	0.7751	0.626	0.6828	0.5308

Table 3: Results of our model on leader board

	F1	EM
Dev	77.566	68.058
Test	76.83	67.114

4 Conclusion

We utilized character-level embedding, bi-directional attention flow, DrQA mode and smarter span selection. We achieved 77.566 and 68.058 for the dev F1 and EM scores, and 76.83 and 67.114 for the test F1 and EM scores.

References

[1] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[2] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017.