Combining Bidirectional Attention Flow and Attention Pooling Pointer Networks for High Performance on the SQuAD Challenge

Anoop Manjunath
amanjuna@stanford.edu

Kiko Ilagan
ilaganf@stanford.edu

Abstract

Machine reading comprehension is an important task to solve en route to a generalized question answering system. The Stanford Question Answering Dataset (SQuAD) challenge is a popular challenge for machine reading comprehension. In this paper, we present a deep recurrent neural network architecture that utilizes bidirectional attention flow and an answer pointer network to achieve 74.605 F1 and 64.219 EM scores on the SQuAD challenge.

1 Introduction

1.1 Task Definition

The goal of SQuAD is to create a model that, given a context passage and a question about the passage, can correctly provide the start and end word of a continuous string of words constituting the answer within the context passage.

1.2 Background/Previous Work

The original SQuAD paper was published in 2016 and has since sparked much research progress on the task of reading comprehension. As of March of 2018, models, particularly the Hybrid AoA ensemble reader, are quickly approaching human performance [1].

Our model in particular draws upon and extends three major avenues of active research in the SQuAD task: attention, out of vocabulary (OOV) management through character level embeddings, and probabilistic condition of end word choice based upon start word prediction.

Attention is a powerful technique employed prolifically in the SQuAD task in almost all top models. In particular, we build our attention framework off of the Bi-Directional Attention Flow (BiDAF) framework for attention with context word embeddings being augmented with attention flowing both between the question and answer and answer to question [3].

The presence of words outside of the model vocabulary is a challenging obstacle in the question answering task. One popular technique for dealing with these OOV words, pioneered by Yoon Kim for the task of sentence classification, is augmenting the model's word embeddings with trainable character level embeddings generated from a convolution over each words' characters [2].

Another novel technique that has been shown to improve performance on SQuAD is conditioning end word prediction on the model's start word prediction. This improves performance by allowing the end word prediction to access the information used by the start word. In this paper we build upon the conditioning methods explored out by the Match-LSTM and Answer Pointer Model [4] as well as the R-net model [5].

In addition to implementing each of the above techniques among others, our model extends the existing research by developing each of the above techniques further and integrating aspects of different models in novel fashions.

2 Model Architecture

Our model's architecture can be broken down into several distinct layers, each of which was developed separately and cumulatively added to the baseline. We discuss each layer further:

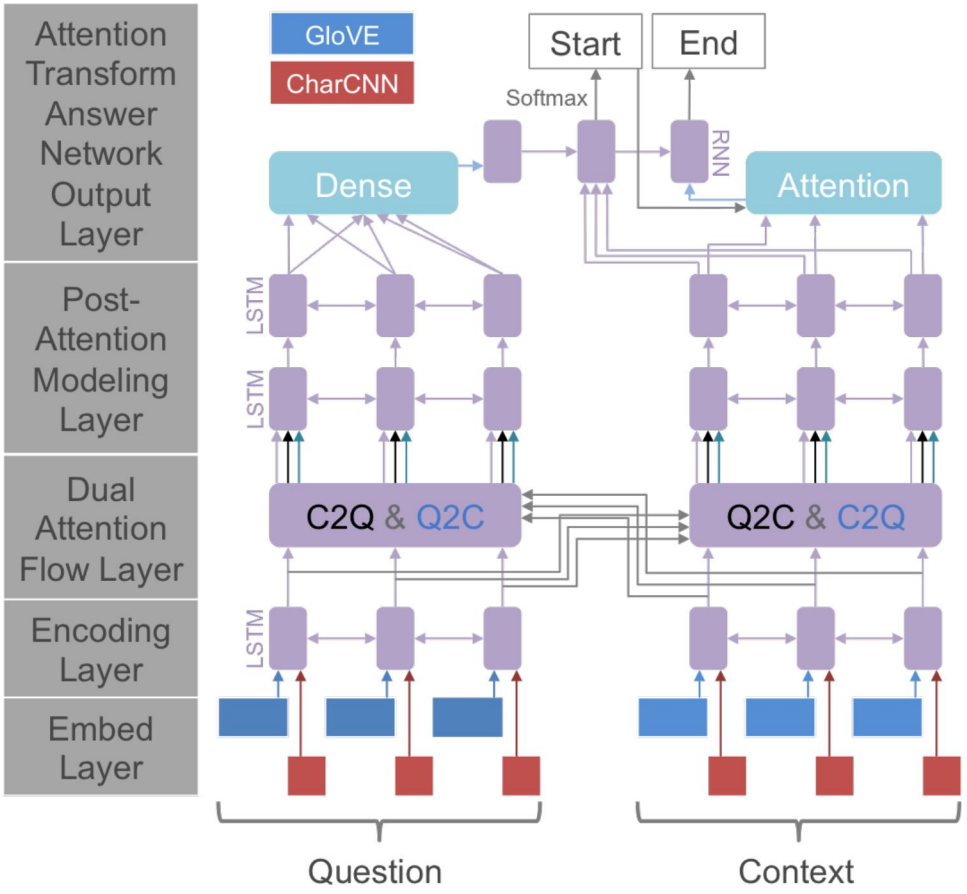

Figure 1: Diagram of our model's architecture

2.1 Word/Character Embedding Layer

For a single SQuAD example (context, question, answer), the context and question are both represented as sequences of pre-trained d-dimensional GloVE embeddings (we use $d = 300$) $x_1, ..., x_N \in \mathbb{R}^d$ and $y_1, ..., y_M \in \mathbb{R}^d$, respectively. Furthermore, for each word w comprised of characters $c_1, ..., c_L$, we create a character-level encoding $\text{emb}_{char}(w) \in \mathbb{R}^{100}$ by performing 1-dimensional convolution with 100 filters of size 5 and same padding on each word. This character level embedding is then concatenated with the word embedding to create the hybrid embedding inputs x_i^A and y_i^A.

2.2 Encoding Layer

The hybrid embeddings are then used as input for a single layer bidirectional LSTM encoding layer:

$$\{\overrightarrow{c_1}, \overleftarrow{c_2}, ..., \overrightarrow{c_N}, \overleftarrow{c_N}\} = BiLSTM(\{x_1^A, ...x_N^A\})$$
$$\{\overrightarrow{q_1}, \overleftarrow{q_2}, ..., \overrightarrow{q_M}, \overleftarrow{q_M}\} = BiLSTM(\{y_1^A, ...y_N^A\})$$

We concatenate these forward and backward hidden states to obtain the context hidden states c_i and question hidden states q_j:

$$c_i = [\overrightarrow{c_i} \| \overleftarrow{c_i}] \; \forall i \in \{1, ..., N\}$$
$$q_j = [\overrightarrow{q_j} \| \overleftarrow{q_j}] \; \forall j \in \{1, ..., M\}$$

2.3 Dual Bidirectional Attention Layer

We then apply bidirectional attention to the context and question hidden states. Following Seo et al., we calculate a similarity matrix S that contains similarity scores for every pair of context and question states (c_i, q_j):

$$S_{ij} = w_s^\top [c_i \| q_j \| c_i \circ q_j] \in \mathbb{R}$$

Where w_s is a learnable weight vector. We then perform Context to Question (C2Q) and Question to Context (Q2C) attention using S. For all $i \in \{1, ..., N\}$:

$$\alpha^i = softmax(S_{i,:}) \; \in \mathbb{R}^M$$
$$a^i = \sum_{j=1}^{M} \alpha_j^i q_j$$
$$m_i = max_j S_{ij}$$
$$\beta = softmax(m)$$
$$c'_C = \sum_{i=1}^{N} \beta_i c_i$$

We then use these attention outputs to create the blended context representation $b_i^C = [c_i \| a_i \| c_i \circ a_i \| c_i \circ c'_C]$. We also perform a parallel computation of bidirectional attention where we flip the roles of context and question to obtain blended question representations $b_j^Q = [q_j \| a_j \| q_j \circ a_j \| q_j \circ c'_Q]$.

2.4 Post-Attentional Modeling Layer

We then use the blended representations as input to a two layer bidirectional LSTM network to capture the relationships between context words after conditioning on the query.

$$\{\overrightarrow{r_1}, \overleftarrow{r_1}, ..., \overrightarrow{r_N}, \overleftarrow{r_N}\} \; = BiLSTM(BiLSTM(\{b_1^C, ..., b_N^C\}))$$
$$\{\overrightarrow{s_1}, \overleftarrow{s_1}, ..., \overrightarrow{s_M}, \overleftarrow{s_M}\} \; = BiLSTM(BiLSTM(\{b_1^Q, ..., b_M^Q\}))$$

Like in the embedding layer, we concatenate the forward and backward hidden states for each context word and each question word to obtain the post-attention context hidden states r_i and post-attention question hidden states s_j:

$$r_i = [\overrightarrow{r_i} \| \overleftarrow{r_i}] \; \forall i \in \{1, ..., N\}$$
$$s_j = [\overrightarrow{s_j} \| \overleftarrow{s_j}] \; \forall j \in \{1, ..., M\}$$

2.5 Attention Transforming Answer Network Output Layer

Now that we have the post-attention hidden states, we can use them as inputs to a unidirectional answer RNN, which is run for exactly two time steps. The initial hidden state h_0 of this answer

network is the output of a dense fully-connected layer that takes all the post-attention question hidden states s as input, with a tanh activation:

$$h_0 = tanh(W_s s + b_s)$$

This effectively "pools" then transforms the post-attention output. Next, at time step 1, the answer network uses the initial hidden state and a linear combination of all the post-attention context hidden states to calculate a score vector v_1 (which for this simple RNN is equal to the new hidden state h_1) that can be subsequently put through a softmax function to obtain the start location probability distribution:

$$v_1 = tanh(W_r r + W_h h_0 + b_v) \in \mathbb{R}^N$$

We can then get the start probability distribution: $\mathbb{P}(start) = softmax(v_1)$. This distribution is then used to perform attention - let p represent $\mathbb{P}(start)$:

$$u = \sum_{i=1}^{N} p_i r_i \in \mathbb{R}^N$$

The resulting context conditioned on the start distribution is then fed as the input for time step 2 of the answer network, which calculates the end score vector v_2:

$$v_2 = tanh(W_r u + W_h h_1 + b_v)$$

We can then obtain the end distribution - $\mathbb{P}(end) = softmax(v_2)$

2.6 Predictions

At test time, we take the argmax over the start distribution to obtain the start location, then take the argmax of the slice of the end distribution that corresponds to locations within 38 words of the start location. 38 was chosen as the maximum distance between the start and end prediction as this is the length of the longest answer in both the training and development sets.

3 Experiments

3.1 Data

First, we analyzed the given data to familiarize ourselves with the task. This analysis also has the added benefit of helping us choose intelligent default numbers for certain hyperparameters, namely the maximum allowable context length and question length. First, we examined the lengths of the contexts in the training and development sets:

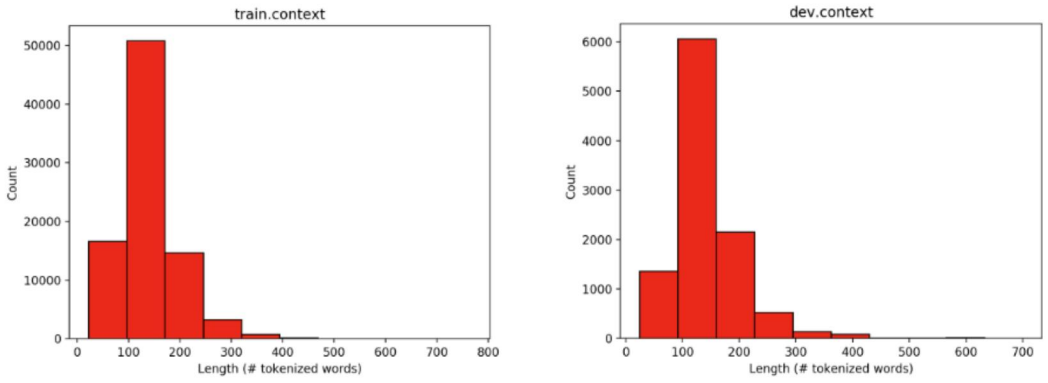

Figure 2: Length of contexts in the training and development sets

The contexts have mean lengths of around 138 words and 142 words in the training and dev sets, respectively. Both sets have contexts that reach around 700 words, but the vast majority of contexts do not exceed 500 words long. Performing a similar analysis on the questions:

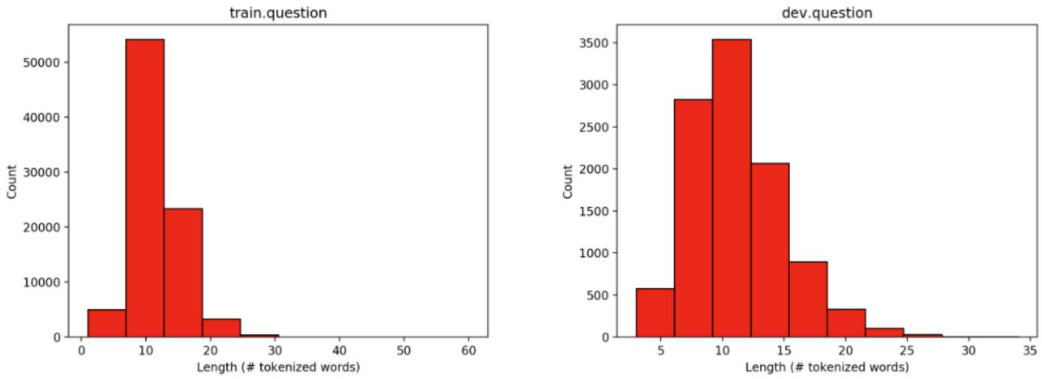

Figure 3: Length of questions in the training and development sets

Question lengths averaged around 11.3 and 11.4 for the train and dev sets, respectively, with maximum lengths of 60 and 34 words. Note that the sets actually both have fairly similar distributions - the outliers in the training set throw off the relative scales of the histograms.

Lastly, we performed an analysis of the answer span lengths and span locations in the context. First span length:

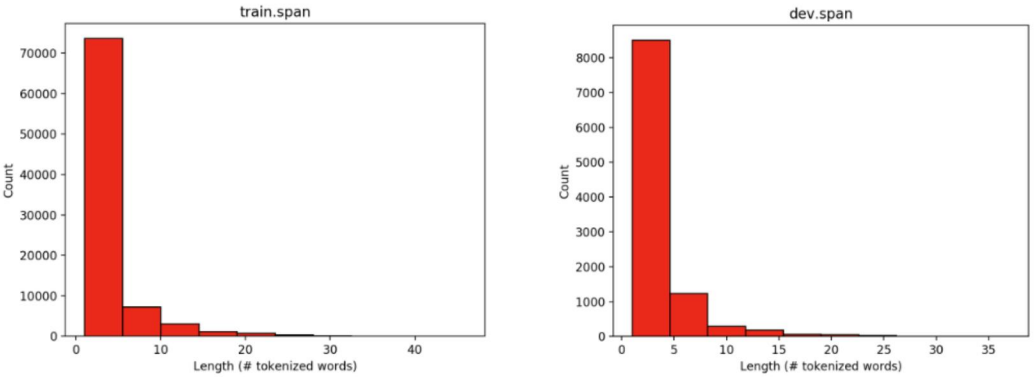

Figure 4: Length of answer spans in the training and development sets

Mean answer lengths were 3.4 and 3.2 words for the training and development sets, respectively, with maximum answer lengths of 46 and 37. Both sets are fairly dramatically skewed right. Furthermore, since most of the answers are quite short, it is not entirely unreasonable to approximate the location of the span in the context by just measuring what proportion of the way through the context contains the end index of the answer.

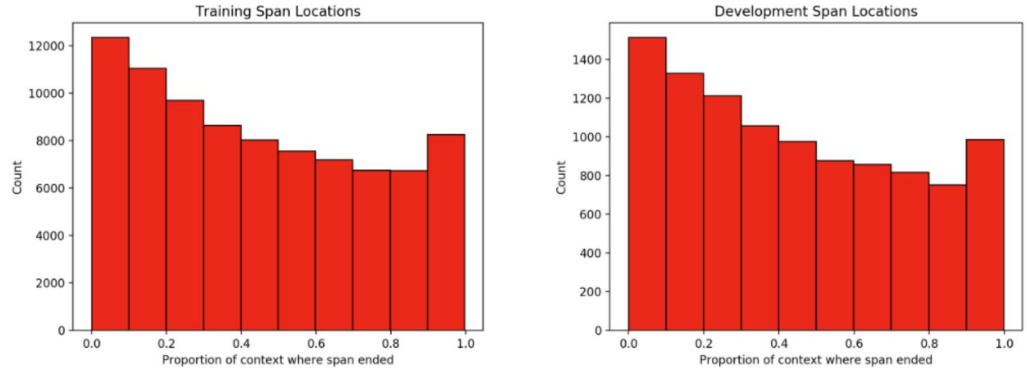

Figure 5: Span locations in the training and development sets

The average end position is about 45% and 44% of the way through the context for the training and development sets, respectively. The means coupled with the histograms tell us that most of the contexts tend to appear in the first half of the context, with a noticeable spike of answers appearing at the very end of the context.

With the insights from these data, we selected a cut-off context length of 500 and question length of 35.

4 Results

4.1 Constructive Analysis

Given the modular nature of model, we build it up in sequential stages, evaluating model performance on the dev set at each step. As a result, we have some interesting analysis of the features that most contributed to model success shown below:

Model Improvement	**F1 (%)**	**EM (%)**
Baseline	43.8	34.79
LSTM	44.7	36.3
Attention	52.5	41.7
CNN	60.2	47.4
Post-Attention BiLSTMs	69.431	60.423
Output conditioning	73.953	63.586
L2 regularization	73.553	62.791

Table 1: Progress of our model's performance as specified improvements were added

The majority of additions monotonically improved performance. The attention, character-level CNN, and post-attention RNNs made particularly large differences in model performance. The switch to LSTMs and more interestingly, the addition of output conditioning and L2 regularization offered only meager improvement. We also evaluated the final and baseline model performance on the dev set, stratified by question type:

Baseline Model Performance

	Who	**What**	**When**	**Where**	**Why**	**Which**	**How**
F1	38.5%	39.5%	48.0%	35.5%	28.0%	43.3%	50.0%
EM	34%	28%	42%	26%	8%	32%	35%

Final Model Performance

	Who	**What**	**When**	**Where**	**Why**	**Which**	**How**
F1	74.5%	71.7%	74.6%	65.8%	53.7%	69.2%	69.9%
EM	66%	57%	63%	52%	21%	58%	46%

Table 2: Model scores on 100 randomly chosen examples of each question type

Performance on each of these question types is varied due to their differing difficulties and frequencies in the training data. For example, "why" questions are the most difficult questions linguistically since they tend to require greater comprehension of the passage and reasoned decision making, and they are also the least frequent type of question in the data, comprising less than 2% of all training examples.

We now consider each improvement more closely:

4.1.1 LSTM

The slight uptick in performance from the addition of LSTMs versus the previous GRUs came from better results on questions with longer answers and passages where the answer to the question was located further away from the location of question words in the context. We hypothesize that this difference in performance is due to the LSTMs superior ability to gate information as compared to GRUs, however training time did increase substantially.

4.1.2 Attention

Introducing bidirectional attention into the model gave us our first significant boost in performance. We saw the greatest performance improvement on "which" questions upon implementing this module, which likely can be attributed to the model's new, enhanced ability to more carefully focus on the relevant parts of the question and context to pick correct answers out of a list.

4.1.3 CNN

For this improvement, we increased our default GloVe embedding size from 100 to 300 and implemented a character-level CNN to further augment the expressiveness of the input to our model. We consequently saw an overall increase in performance, particularly for "who" and "when" questions. We hypothesize that the character level representation of names and dates are more helpful for distinguishing the answer among other names/dates, since they tend to appear in similar contexts and thus are not easily distinguished with normal word vectors. The learned, augmented representations from the CharCNN combat this issue.

4.1.4 Post-Attention BiLSTMs

We realized that we were wasting some information from the attention layer - intuitively, the modified context and questions conditioned on each other could provide valuable information. We thus implemented a post-attention 2 layered bilateral LSTM and saw fairly significant improvement across the board.

4.1.5 Output conditioning

We then decided to implement an answer network and some dense layers that ultimately allow us to condition the model's end prediction on the start prediction. Previously, such predictions were made independently with two separate softmax functions. However, the fact that an end prediction should never go before a start prediction tells us that there is a conditional relationship between the two in the real world - implementing this final layer to capture this intuition gave us our last noticeable boost in performance.

4.1.6 L2 regularization

L2 regularization did not significantly improve performance, even after training for 15 thousand iterations. This indicates that dropout level of 0.2, which we applied to almost all the model's learnable parameters, is sufficient for preventing overfitting.

4.2 Error Analysis

4.2.1 Baseline

In order to better understand our model's performance and how to improve it, we regularly examined its qualitative outputs. In particular, for the baseline we observed the following. Note that the word chosen as the start is underlined while the end is *italized*.

Issue	Example	Solution
The model predicts an end locations before beginning locations	Q: Who was abc's third major rival in 1949? A: *dumont* television company	Constrainted span selection
The model performs poorly when answering phrased in the form of negations	Q: What can the non-elected members from the scottish government not do	Context to question and question to context attention to negations
Poor performance when identifying answers that are rare words	Q: What was the name of the 3d system effect in dimension in time? A: pulfrich effect	Character level CNN

4.2.2 Final Model

After developing the model in full, we examined the output to determine areas of improvement:

Issue	Example	Solution
Model answer is reasonable but incomplete	Q: Who acts as the coordinator? A: the architect Truth: the architect or engineer	More sophisticated search of high likelihood spans
The model overruns the ground truth answer prediction	A: constant velocity was associated with a lack of net force, Truth: a lack of net force	small penalty for answers above a certain length
Model cannot answer question that requires knowledge from outside of context	Q: Which architect, famous for building his work on St. Paul's Cathedral, is featured? A: Carlo Lodoli, Truth: Inigo Jones	Shared context between certain questions

There were other errors that involved poor labeling of question-answer pairs, but such examples are unimportant.

5 Conclusions

We present a model that combines aspects from various high-performing SQuAD models and have shown that such a model produced from this combination also performs well on the challenge. Each modular improvement was brought about to match human intuition about the challenge, but there are further improvements yet to make. Many of the errors yet to be overcome can be attributed to insufficiently nuanced start/end location prediction - the model can identify relevant content, but cannot narrow its focus enough. In the future, we would like to experiment with more sophisticated start and end conditioning (likely using joint probability distributions). Furthermore, it is likely that allowing the model to use self-reference and self-attention will allow it to better focus on the relevant words for prediction and stop overrunning the correct prediction length.

References

[1] CUI, Y., CHEN, Z., WEI, S., WANG, S., LIU, T., AND HU, G. Attention-over-attention neural networks for reading comprehension. *CoRR abs/1607.04423* (2016).

[2] KIM, Y. Convolutional neural networks for sentence classification. *CoRR abs/1408.5882* (2014).

[3] SEO, M. J., KEMBHAVI, A., FARHADI, A., AND HAJISHIRZI, H. Bidirectional attention flow for machine comprehension. *CoRR abs/1611.01603* (2016).

[4] WANG, S., AND JIANG, J. Machine comprehension using match-lstm and answer pointer. *CoRR abs/1608.07905* (2016).

[5] WANG, W., YANG, N., WEI, F., CHANG, B., AND ZHOU, M. Gated self-matching networks for reading comprehension and question answering. In *ACL* (2017).