Bi-Directional Attention Flow and Co-Attention Models for Question Answering on the SQuAD Dataset

Rafael Musa
Department of Computer Science
Stanford University
rmusa@stanford.edu

March 20, 2018

Abstract

In this paper, we outline the architecture of a neural question answering model for the SQuAD dataset using a bi-directional GRU, bi-directional attention flow, self-attention, a fully connected output layer, and smart span selection. The final model was submitted to the Codalab test leaderboard and achieved 71.653 F1 and 60.38 EM.

1 Introduction

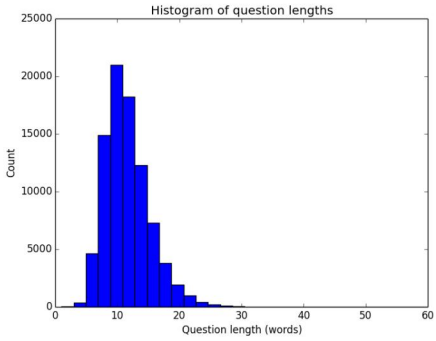
This paper addresses the problem of question answering, specifically when the answers to the questions are taken straight from the paragraphs instead of being generated by the system. More formally, given a question and context paragraph $c_1, \ldots, c_M$, the goal is to output a pair of indexes $i, j, j > i$ such that $c_i, \ldots, c_j$ is the answer to the question given.

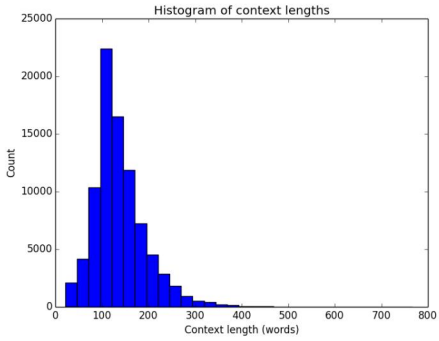
1.1 Dataset

The dataset used is the Stanford Question Answering Dataset (SQuAD). We trained on 86326 question-context pairs, and saved 10391 pairs as a dev dataset. The context paragraphs come from Wikipedia, and the ground truth answers were obtained through Amazon Mechanical Turk. We adopt a neural approach with an embedding layer using pre-trained GloVe word vectors, an encoder layer using a bi-directional GRU, a bi-directional attention layer, a self-attention layer, and a fully connected output layer. The complete architecture is described more thoroughly in section 3.

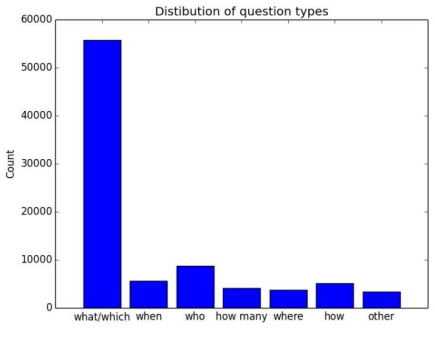
In order to inform our approach, we ran some basic data analysis on the training data. Our system pads questions and context paragraphs to the same length, so, in order

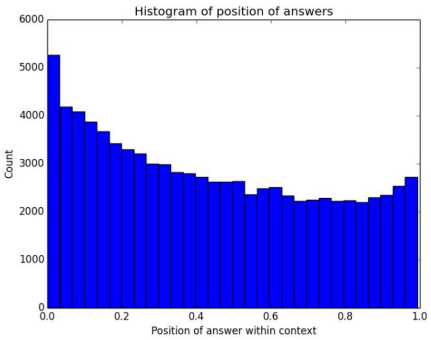
to determine an appropriate cutoff, we plotted histograms of the lengths of each.





With this information in mind, we decided to set the question length to 30 and context length to 400, since the number of examples exceeding this length was negligible. Next, we hoped to understand the types of questions asked and whether answers were more likely to appear close to the beginning or end of the context.





We note that the great majority of questions is of the 'what/which type'. We kept this data so we could evaluate potential improvement ideas that were specific to a question type, such as looking for numbers for a 'how many' question or names for a 'who' question.

For the position of answers, we plotted the position of the middle of the answer span with respect to the context paragraph with 0 being the very beginning of the context and 1 being the end. We found that answers were slightly more likely to appear at the beginning of the context. This suggests that a possible improvement of the model outlined later in this paper is to slightly bias in favor of spans toward the beginning of the context.

2 Background and Related Work

We use F1 and Exact Match (EM) metrics to calculate performance on the SQuAD dataset. The stricter EM metric is binary and deems an answer correct only if it exactly matches the ground truth answer. F1 is the harmonic mean of precision and recall, and

allows for 'partial credit' for answers that contain only a subset of the ground truth, or that include some additional tokens.

Rajpurkar et al. [2] assessed the performance of human answerers to be 82.304 EM and 91.221 F1. Many state-of-the-art models submitted to the public SQuAD leaderboard have performance similar to or even exceeding that of humans. For example, the Reinforced Mnemonic Reader model by Hu et al. [5] achieves 82.849 EM and 88.764 F1.

Our efforts are based primarily of the Bi-Directional Attention Flow (BiDAF) model by Seo et al. [3] and the Self-Attention mechanism outlined in the R-Net paper by Microsoft Research Asia [6]. Note that each paper implements some techniques that we do not attempt, as well as some that we either adapt or implement exactly. Full details can be found in section 3.

3 Approach

Embedding Layer

This layer just performs an embedding lookup to convert the question and context into word-vector representations. This layer outputs $x_1, \ldots, x_{30} \in \mathbb{R}^{100}$ as the question representation and $y_1, \ldots, y_{400} \in \mathbb{R}^{100}$ as the context representation. Note that questions and contexts are not all length 30 and 400. In order to get around this we pad the inputs up to those lengths and then apply a mask when necessary to ensure the padding does not interfere with the model.

RNN Encoding Layer

This layer converts the word vector representations into hidden states for the question and context using a 1-layer bidirectional GRU. The GRU uses a dropout probability of 0.15. The GRU is shared between context and question. The GRU outputs a sequence of forward and backward states for the question and context: $\{\overrightarrow{q_1}, \overleftarrow{q_1}, \ldots, \overrightarrow{q_{30}}, \overleftarrow{q_{30}}\}$ and $\{\overrightarrow{c_1}, \overleftarrow{c_1}, \ldots, \overrightarrow{c_{30}}, \overleftarrow{c_{30}}\}$. Each output has dimension size 200. We simply concatenate the forwards and backwards representations to obtain a single representation of length 400 for each word in each.

Bi-Directional Attention Layer

This layer takes as input the question and context hidden representations output by the bidirectional GRU, $q_1, \ldots, q_{30}$ and $c_1, \ldots, c_{400}$. It follows the implementation by Seo et al. and the outline in the project handout. The main improvement of this type of attention over simpler models is that BiDAF ensures that both the question attends to the context and the context attends to the question.

We define a trainable weight vector $w_{sim} \in \mathbb{R}^{600}$ and compute a similarity matrix S such that $S_{ij} = w_{sim}^T[c_i; q_j; c_i \circ q_j]$. For Context-to-Question (C2Q) attention, we take a row-wise softmax of S which yields attention distributions α_i for $i \in \{1, \ldots, 400\}$. We then compute $a_i = \Sigma_{j=1}^{30} \alpha_j^i q_j$. These a_i are the C2Q outputs.

For Question-to-Context (Q2C) attention, first we compute vectors $m_i = \max_j S_{ij}$ and attention distributions $\beta_i = \text{softmax}(m_i)$. Then we compute $c' = \Sigma_{i=1}^{400} \beta_i c_i$. These c' are the Q2C outputs.

We combine both C2Q and Q2C into a single output vector b_i by outputting $b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c']$. Here each b vector has dimension 1600.

Self-Attention Layer

This layer takes as input the output on the BiDAF layer. We considered an alternative architecture where the BiDAF and Self-Attention layers ran in parallel and had their outputs concatenated before the output layer, but this sequential layer architecture performed better. The implementation follows that outlined in the R-Net paper by Microsoft Research Asia and the outline in the project handout.

We define a trainable weight vector w and weight matrices W_1 and W_2. We set the free dimension of these to 50. We call the input to this layer (which are the 1600-dimensional outputs from BiDAF) $v_1, \ldots, v_{400}$. First, we compute e vectors such that $e_j^i = w^T \text{tanh}(W_1 v_j + W_2 v_i)$. We then obtain attention distributions $\alpha_i = \text{softmax}(e_i)$. Then, we obtain $a_i = \Sigma_{j=1}^{400} \alpha_j^i v_j$. Finally, we concatenate the input with these a_i and feed them into a bidirectional GRU using droupout probability 0.15. We get $\{h_1, \ldots, h_{400}\} = \text{biGRU}(\{[v_1; a_1], \ldots, [v_{400}; a_{400}]\})$. The h_i are the output of the self-attention layer.

Output Layer

The input to this layer is the output of the self-attention layer. This layer is a simple fully-connected layer with a ReLU non-linearity which outputs vectors b'. We use these b' to obtain separate distributions for the start and end of the span. Each b' is passed through two downprojecting linear layers, each with different weights for the span start and end. We apply a softmax to the outputs of the linear layers to obtain distributions for the start and end of the span. Then, we return the answer of length at most 30 that maximizes the product of the span start and span end. The length was decided by examining a histogram of span lengths from the training set, which showed a negligible number of questions with answers of length above 30.

4 Experiments

All experiments were run on the SQuAD dataset described in section 1.1, and evaluation was done using the F1 and EM metrics explained in section 2. We used the Adam optimizer with learning rate 0.001. All models were trained for 12k to 20k iterations on an Azure GPU machine. Each run full training run took 8-12 hours.

We compare our improvements with a baseline model with an RRN layer, a simple Context-to-Question attention layer, a fully connected output layer, and naive span selection where separate argmaxes are taken over the start and end span distributions. The baseline had 42.856 F1 and 33.945 EM.

Here is a summary of all experiments run:

Experiment	Dev F1	Dev EM
Baseline	42.86	33.95
BiDAF	49.91	39.93
Self-Attention + BiDAF in parallel	55.02	44.85
Self-Attention + BiDAF sequentially	68.24	56.71
Self-Attention + BiDAF (seq) + exact match feature	64.90	53.24
Self-Attention + BiDAF (seq) with 200 GloVe dimension	68.16	56.58
Self-Attention + BiDAF (seq) + 2 extra fully connected layers	67.13	56.20
Self-Attention with 75 hidden size + BiDAF (seq) + 0.2 dropout	68.00	56.60
Self-Attention + BiDAF sequentially + smarter span selection	**71.06**	**59.39**

The model with best dev set performance was the one described in detail in section 3. That model was submitted to the test leaderboard and obtained 71.653 F1 and 60.38 EM.

Due to time and resource constraints, we were unable to run a grid search to tune hyperparameters, or to independently tune each of the four main improvements implemented over the baseline model (BiDAF, Self-Attention, exact match features, and smarter span selection). Note further that all parameter tuning was done on the model with naive span selection since smarter span selection was implemented only after the experiments were run. The first two and last improvements were described in depth in the approach section, so we detail only the exact match features attempt here.

For this improvement, we took the output of the embedding layer for the context words, and appended a 1 to the word vector if that word was present in the question and a 0 if it was not. The reasoning was that words that are present in the question are more likely to be in or close to the correct answer span, so this information would be useful to our model. We also had to append 0 to all the question word vectors since we reused the RNN encoding layer weights between question and context and therefore needed both representations to be the same length.

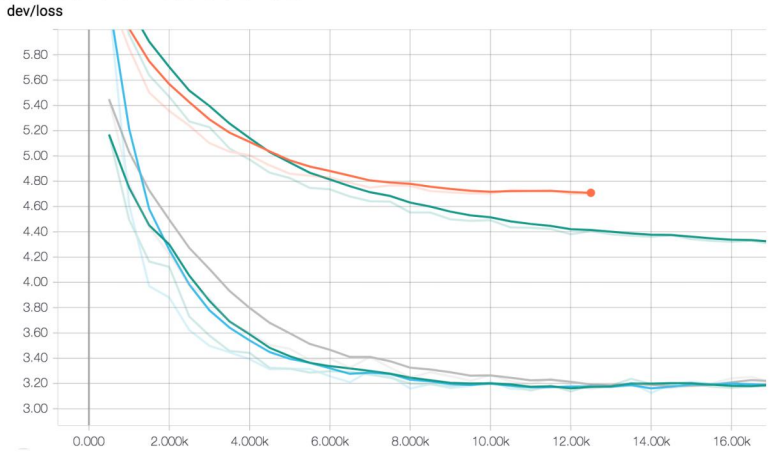
Unfortunately, it turned out that this additional feature decreased our model performance when combined with BiDAF and Self-Attention so we kept it out of our final model. We also tried adding two additional fully connected layers in the output layer, but that had a very small effect on performance. Therefore, we opted for the simpler model with fewer layers.

After determining that using BiDAF and Self-Attention sequentially and ignoring the exact match feature was the best overall architecture, we attempted to tweak some hyperparameters. We tried switching the GloVe dimension to 100 from 200, changing dropout to 0.2, and increasing the self-attention hidden size from 50 to 75. None of these changes led to improvements in the dev F1 score. More time and computing power would have allowed us to perform a grid search with these and other hyperparameters but given the limited experiments we ran we determined the optimal was to keep GloVe dimension at 100, dropout at 0.15, and self-attention hidden size at 50[1].

[1]Microsoft Research Asia uses 75 in their R-Net paper introducing the architecture.

5 Error Analysis

The following plot shows how the loss on the dev set goes down per iteration for a subset of the models trained.



Here, the orange line is the baseline, and the green line right below it is the simple BiDAF implementation. The other three lines that are clustered together are all for BiDAF and self-attention with a different combination of hyperparameters. We see that they converge to around the same loss which makes sense since their dev F1 performance is very similar as seen in the experiments table above.

Now we analyze some specific incorrect answers to gain insight into how the model fails and how we could improve it.

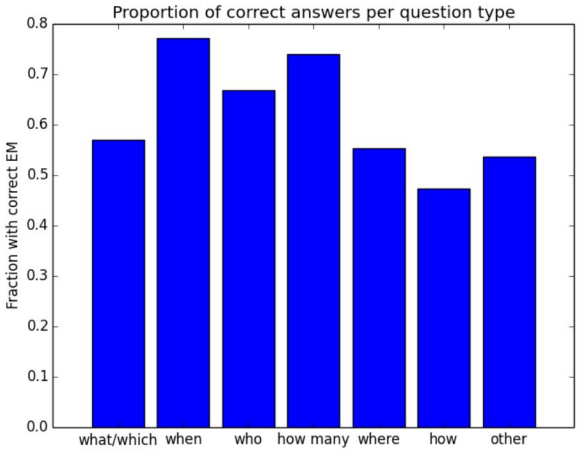
Context: a piece of paper was later found on which luther had written his last statement . the statement was in latin , apart from " we are beggars , " which was in german
Question: what was later discovered written by luther ?
Answer: his last statement
Our prediction: piece of paper
Analysis: This is an example of a very difficult question for a model to answer correctly. Our prediction of 'a piece of paper' is technically correct in that it was found by Luther, but the ground truth answer is the statement since a human can tell form that passage that that is more significant over the fact that it was written on a piece of paper. Since the prediction is correct syntactically, the model would need nuanced understanding of how the semantics of each word interact in order to get this one right.

Context: most _platyctenida_ have oval bodies that are flattened in the _oral-aboral_ direction , with a pair of _tentilla-bearing_ tentacles on the a boral surface . they cling to and creep on surfaces by _everting_ the pharynx and using it as a muscular " foot " . all but one of the known _platyctenid_ species lack _comb-rows_ . _platyctenids_ are usually cryptically colored , live on rocks [...]
Question: what do platyctenida use their pharynx for ?

Answer: cling to and creep on surfaces
Our Prediction: a muscular ” foot ”
Analysis: For this context paragraph, all words surrounded by underscores were unknowns. Since this passage uses some technical jargon from a specific subfield, several terms here were unknown. Our model would be more likely to answer a question on this passage correctly if it had a better representation for unknown words. For example, implementing a character-level CNN is likely to improve performance on this passage.

Context: With the opening of the Dorothy and Michael Hintze sculpture galleries in 2006 it was decided to extend the chronology of the works on display up to 1950; this has involved loans by other museums, including Tate Britain, so works by Henry Moore and Jacob Epstein along with other of their contemporaries are now on view. [54 words omitted] Then there is a section that covers late 19th-century and early 20th-century sculpture, this includes work by Rodin and other French sculptors such as Dalou who spent several years in Britain where he taught sculpture.
Question: Which museum was among those that loaned more modern works for the new sculpture galleries?
Answer: Tate Britain
Our Prediction: Rodin
Analysis: Here we are clearly attending to the wrong portion of the context since the answer we select is several sentences away from the correct answer. Note that the question contains ‘museum’ and the word ‘museums’ occurs in the passage very close to the correct answer. Some smarter way to incorporate this feature, perhaps with stemming, could give the model a clue to attend to the correct section in this case which would give it a chance at answering correctly.

We also take a look at the proportion of answers the model got correct (using the EM metric) per question type



We note that the ‘when’ and ‘how many’ questions have the best correct answer

rates. This is likely because these answers are usually answered by a day/month/year or number, and these patterns are easy for the model to pick up and act on. 'How' questions have the worst performance, at under 50% EM. These 'how' questions can be answered in a great variety of ways so it is harder for the model to pick up on patterns.

To improve performance on some of these question types, using part-of-speech or named-entity recognition tags would be very helpful. Our analysis in section 1.1 revealed that the vast majority of questions are of the 'what/which' type, and those are almost always answered by nouns, so having the POS annotation could help the model. Other types would benefit too: 'where' performance would likely be increased by location tags, and 'who' by person tags.

6 Conclusion and Future Work

Using BiDAF, self-attention, and smarter span selections, we were able to achieve 71.653 F1 on the test set on the SQuAD Codalab leaderboard. Many state-of-the-art models achieve near-human performance of around 90 F1. In order to get to that level of performance, there are several improvements our model could use. Many were discussed in previous sections: using character-level CNNs to improve unknown word handling, POS and NES tags as additional features, running a more thorough grid search to tune hyperparameters, and trying to incorporate the fact that answers are slightly more likely to appear in the first half of the passages. Improvements like these appear in many published papers that we used as starting points for this project, but we did not have time to implement them all.

References

[1] CS224N Winter 2018 Staff. CS224N Default Final Project: Question Answering. [Online, accessed 19-March-2018]

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

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

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

[5] Minghao Hu, Yuxing Peng, and Xipeng Qiu. Reinforced Mnemonic Readoer for Machine Comprehension. *arXiv preprint arXiv:1705.02798*, 2017

[6] Natural Language Computing Group, Microsoft Research Asia. R-Net: Machine Reading Comprehension With Self-Matching Networks. [Online, accessed 17-March-2017]