Exploring Attention in Question Answering Models

Ethan Shen*
Department of Computer Science
Stanford University
ezshen@stanford.edu

Anav Sood*
Department of Mathematics
Stanford University
anavsood@stanford.edu

Abstract

We examine the performance of bi-attention, self-attention, and co-attention layers in the framework of the Bi-Directional attention flow model (BiDAF) model in an effort to determine the flexibility of general attention layers in question answering. Noting the original bi-attention layer's faster training and better performance, we construct an ensemble of fine-tuned and modified BiDAF models to get an idea of attention focused models' effectiveness on the SQuAD dataset. Our final ensemble model achieved a 77.85 F1 score and 68.39 EM score on the dev set.

1 Introduction

Machine Comprehension (MC), the task of getting a machine to thoroughly understand and digest language, and Question Answering (QA), the task of getting a machine to answer a question through MC, are complex problems that have become approachable through the lens of deep learning. Due to their high level of difficulty, MC and QA tasks are often used a benchmark for measuring the progress of NLP. Thanks to the creation of the SQuAD dataset, a reading comprehension dataset with more than 100,000 context, question, answer tuples, serious progress has been made with respect to these tasks [1].

We present our results on SQuAD, which are organized as follows. In section 2 we formalize our problem statement. In Section 3 we present a baseline model and three separate attention-based models, all which operate in the BiDAF framework [2]. In Section 4 we present slight modifications to our original bi-attention based model, and introduce our ensembling method. In section 5 we give analysis for experiments run in Section 3 and Section 4. In section 6 we present our conclusions and potential next steps for future research. All of the work presented below follows the guidelines of the CS224N default final project [3].

2 Problem Statement

Given a finite sequence of question words $Q = \{q_i\}_{i=1}^{M}$ and a finite context $C = \{c_i\}_{i=1}^{N}$, we seek to determine a variable length finite sequence of answer words $A = \{a_i\}_{i=1}^{L(C,Q)}$, where there exists $K \in \mathbb{N}$, such that $a_k = c_k$ for all $K \leq k < K + L(C, Q)$. In particular this reduces to finding a starting and ending point, i_s, i_e respectively, and letting $A = c_{i_s}, c_{i_s+1}, \ldots, c_{i_e}$.

3 Attention-Based Models

As a reference, we implement the baseline model described in the CS224N default final project [3]. Below we first describe the framework for the BiDAF model, which we use as a means of comparing different variations of complex attention [2]. Then we provide the mathematics behind each different attention layer in vectorized form. The co-attention and co-attention layers are modified from their

original papers, so to get the best intuition regarding how the layers work it may help to view the math given in the original papers, which is in non-vectorized form.

3.1 BiDAF Framework

The BiDAF model is a very successful QA model and is highly cited among models currently leading the SQuAD leader-board. Aside from introducing bi-attention, the original paper presents a backbone for many attention based QA models. We provide this structure below, abstracting out the bi-attention layer to a general attention layer. We adopt the convention that the set of vectors $\{a_1, \dots, a_n\}$, where $a_i \in \mathbb{R}^m$, naturally induce a tensor $\mathbf{A} \in \mathbb{R}^{n \times m}$, where the i^{th} row of $\mathbf{A}$ is a_i. Similarly we will refer to the i^{th} row of some tensor $\mathbf{A}$ as a_i when appropriate. RNNs do not share weights unless otherwise specified.

1. **Initial Embedding Layer**. We represent both our question $\{q_i\}_{i=1}^M$ and context $\{c_i\}_{i=1}^N$ with dense word embeddings of dimension e. This gives a question tensor $\mathbf{Q} \in \mathbb{R}^{M \times e}$ and context tensor $\mathbf{C} \in \mathbb{R}^{N \times e}$. Hence forth q_i, c_i refer to either the word or its embedding depending on context.
2. **Contextual Embedding Layer**. To find our contextual embedding we find
$$u_t = \text{BiRNN}(u_{t-1}, q_t)$$
$$h_t = \text{BiRNN}(h_{t-1}, c_t)$$
where u_t, h_t are the resulting concatenation of forward and backwards RNNs with d-dimensional output and the two BiRNNs share weights. From this we have a contextual question tensor $\mathbf{U} \in \mathbb{R}^{M \times 2d}$ and a contextual context tensor $\mathbf{H} \in \mathbb{R}^{N \times 2d}$.
3. **Attention Layer**. Using the contextualized embeddings $\mathbf{U}, \mathbf{H}$, the attention layer comes up with finite attention embeddings $\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k$. We then define
$$\mathbf{G} = [\mathbf{H}; \mathbf{A}_{i_1}; \mathbf{A}_{i_2}; \dots; \mathbf{A}_{i_\ell}; \mathbf{H} \circ \mathbf{A}_{j_1}; \mathbf{H} \circ \mathbf{A}_{j_2}; \dots \mathbf{H} \circ \mathbf{A}_{j_{\ell'}}]$$
where $\{i_1, \dots, i_\ell\}$, $\{j_1, \dots, j_{\ell'}\} \subseteq \{1, \dots, k\}$. Note in general $\mathbf{G}$ is some tensor representing a combination of our attention embeddings and contextual context $\mathbf{H}$, but does not to need to be exactly in the form described above.
4. **Modeling Layer**. Now, supposing the t^{th} row of $\mathbf{G}$ is g_t, we find
$$m_t^{(1)} = \text{BiRNN}(m_{t-1}^{(1)}, g_t)$$
where $m_t^{(1)}$ is the resulting concatenation of forward and backwards RNNs with d-dimensional output. From this we have $\mathbf{M}^{(1)} \in \mathbb{R}^{N \times 2d}$.
5. **Output Layer**. We find
$$\mathbf{p}^{(1)} = \text{softmax}(\mathbf{w}_{\mathbf{p}_1}^\top [\mathbf{G}; \mathbf{M}^{(1)}]), \ \mathbf{p}^{(2)} = \text{softmax}(\mathbf{w}_{\mathbf{p}_2}^\top [\mathbf{G}; \mathbf{M}^{(2)}])$$
where $\mathbf{w}_{\mathbf{p}_1}^\top, \mathbf{w}_{\mathbf{p}_2}^\top$ are trainable weight matrices of the proper size, and $\mathbf{M}^{(2)}$ is such that
$$m_t^{(2)} = \text{BiRNN}(m_{t-1}^{(2)}, m_t^{(1)})$$

3.2 Bi-Attention

Our implementation for bi-attention follows the BiDAF paper. We first compute a similarity matrix between our contextualized context and question embeddings $\mathbf{S} \in \mathbb{R}^{N \times M}$. We let
$$\mathbf{S}_{ij} = \mathbf{w}_{(\mathbf{S})}^\top [h_i; u_j; h_i \circ u_j]$$
where $\mathbf{w}_{(\mathbf{S})} \in \mathbb{R}^{6d}$ is a trainable weight matrix. Note in general we can in-code similarity in many different ways, so long as it $\mathbf{S}_{ij}$ effectively combines information from h_i and u_j and is trainable.

To find context to question attention (C2Q), $\tilde{\mathbf{U}} \in \mathbb{R}^{N \times 2d}$, we compute $\tilde{\mathbf{U}} = \text{softmax}_{\text{row}}(\mathbf{S})\mathbf{U}$ where $\text{softmax}_{\text{row}}$ maps each row s_i of $\mathbf{S}$ to $\text{softmax}(s_i)$.

To find question to context attention (Q2C), $\tilde{\mathbf{H}} \in \mathbb{R}^{N \times 2d}$ we find $r \in \text{softmax}(\max_{\text{col}}(\mathbf{S})) \in \mathbb{R}^N$, where $\max_{\text{col}}$ finds the max over the columns. We then broadcast r into $\mathbf{R} \in \mathbb{R}^{N \times N}$ and find $\tilde{\mathbf{H}} = \mathbf{R}\mathbf{H}$ then we then define
$$\mathbf{G} = [\mathbf{H}; \tilde{\mathbf{U}}; \mathbf{H} \circ \tilde{\mathbf{U}}; \mathbf{H} \circ \tilde{\mathbf{H}}]$$

3.3 Co-Attention

We adapt a co-attention layer from the recently published paper "Dynamic coattention networks for question answering" [4]. So that we can compare its performance to that of the bi-attention layer, we modify the layer to mimic the bi-attention layer's structure, but still capture essence of co-attention rather than bi-attention.

We first apply a $\tanh$ non-linearity to our contextual embeddings matrix row-wise to give us $\mathbf{U}'$. In particular we have that $u'_t = \tanh(\mathbf{W}u_t + \mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{2d \times 2d}$ and $\mathbf{b} \in \mathbb{R}^{2d}$ are trainable weights. We then compute $\mathbf{S}$ and our C2Q attention, $\tilde{\mathbf{U}}$, exactly as in bi-attention, but replace $\mathbf{U}$ with $\mathbf{U}'$ throughout the process. To find our Q2C attention, $\tilde{\mathbf{H}} \in \mathbb{R}^{N \times 2h}$, we simply compute find $\tilde{\mathbf{H}} = \text{softmax}_{\text{row}}(\mathbf{S})\text{softmax}_{\text{col}}(\mathbf{S})^\top \mathbf{H}$.

Finally, we compute our final embeddings matrix $\mathbf{F}$ by finding

$$f_t = \text{BiRNN}(f_{t-1}, [\tilde{h}_t; \tilde{u}_t])$$

and then let

$$\mathbf{G} = [\mathbf{H}; \mathbf{F}; \mathbf{H} \circ \mathbf{F}]$$

3.4 Self-Attention

We adapt a self-attention layer from the recently published paper "r-net: Machine Reading Comprehension with Self-Matching Networks" [5]. Similarly, we modify it to mimic our bi-attention layer.

We first compute our C2Q attention $\tilde{\mathbf{U}}$ exactly as outlined in bi-attention. Then we find $\mathbf{V} \in \mathbb{R}^{N \times 2d}$ such that

$$v_t = \text{RNN}(v_{t-1}, \tilde{u}_t)$$

where v_t is the output of a forward RNN with $2d$-dimensional output. To compute our self-attention, $\mathbf{A} \in \mathbb{R}^{N \times 2d}$, we compute the similarity matrix $\mathbf{T}$ among the v, so

$$\mathbf{T}_{ij} = \mathbf{w}_{(\mathbf{T})}^\top [v_i; v_j; v_i \circ v_j]$$

where $\mathbf{w}_{(\mathbf{T})} \in \mathbb{R}^{6d}$ is trainable. We then find $\tilde{\mathbf{V}} = \text{softmax}_{\text{row}}(\mathbf{T})\mathbf{V}$, and from that find $\mathbf{A}$ by letting

$$a_t = \text{BiRNN}(a_{t-1}, [\tilde{v_t}; v_t])$$

We then let

$$\mathbf{G} = [\mathbf{H}; \mathbf{A}; \mathbf{H} \circ \mathbf{A}]$$

4 Hyper-paramater Tuning, Initial Results, and Model Modifications

4.1 Hyper-parameter Tuning

All our models use the GLoVe 100D word embeddings. While fine tuning our models, we evaluated on the dev set by setting the start position of our answer to be $\arg\max_i \mathbf{p}_i^{(1)}$ and the end position to be $\arg\max_i \mathbf{p}_i^{(2)}$ We provide plots of the length of contexts, questions, and answers of our dev and train sets in Figures 1, 2, 3. Motivated by these plots, we reduced our maximum context length from 600 to 400, as the vast majority of context paragraphs are under a length of 400. This results in faster training, and essentially identical results.

As an initial comparison, we ran our bi-attention model with the parameters given with the baseline, hidden-state size (d): 200, optimizer: Adam optimizer, initial learning rate: 0.001, regularization: dropout with parameter 0.15, BiRNNs: GRUs. In general we use a batch-size of 32, the one exception being when we train our self-attention model where we use batch-size 16 due to memory issues. Our initial test was to compare the previously stated parameters with those given in the BiDAF paper, hidden-state size (d): 100, optimizer: Ada-Delta, initial learning rate: 0.5, regularization: dropout with parameter 0.2, BiRNNs: LSTMs. The hyper-parameters in the paper appeared to be less efficient in minimizing our loss, however the loss did seem to decrease with a more consistent slope. To isolate what factors contribute to changing the training of the model, we compared each

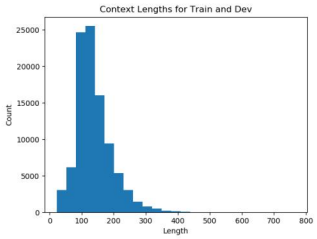

Figure 1: Length of context statements in train and dev sets.

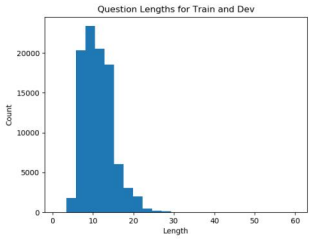

Figure 2: Length of question statements in train and dev sets.

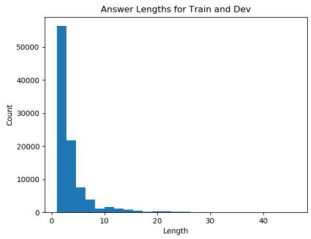

Figure 3: Length of answer statements in train and dev sets.

aspect of the model individually. Figure 6 displays that the Ada-Delta optimizer seems to perform similarly in nature to the Adam-Optimizer, but seems generally worse for the model throughout the training process. Also, Figure 6 displays the LSTMs effectiveness in leading to a smoother training process, where our loss decreases at a relatively regular rate for a comparably extended period, while the GRU loss tends to flatten earlier. This difference is subtle and best seen in the dev F1 (Figure 5) scores. Based on the above results, we choose our final model to use an Adam Optimizer with initial learning rate 0.001 and LSTMs as our BiRNNs or RNNs. The dropout change did not have a noticeable impact so we left dropout at 0.15, and since we could computationally afford to keep our hidden-state size at 200, we chose to do so.

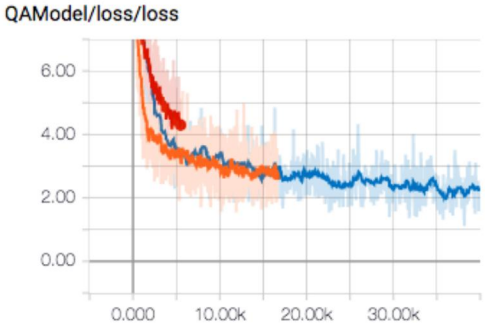

Figure 4: Training loss for bi-attention model.

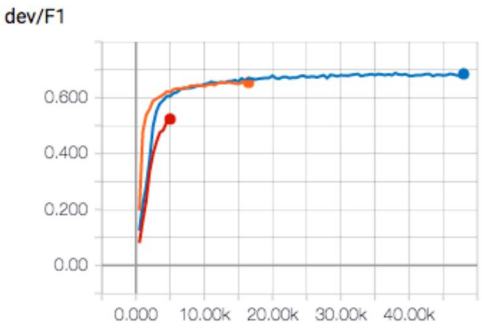

Figure 5: Dev F1 score for bi-attention model.

Figure 6: Plots of train loss and dev F1 for hyper-parameter tuning. Orange represents bi-attention with baseline parameters, while blue employs a LSTM, and red uses the Ada-delta optimizer.

4.2 Results on Different Attention Layers

Ideally we would do hyper-parameter tuning for each different attention layer we use with our model, but due to time constraints this was not feasible. Rather we use the hyper-parameters from above and run our model with a bi-attention, co-attention, and self-attention layer. Plots of train loss are shown in Figure 7.

Naturally, the model being designed for a bi-attention, performed very well with a bi-attention layer. We note that although the co-attention and self-attention models still are in the training process, they do not appear to be on trajectory to overtake BiDAF, and if so, certainly not by a notable margin. We discussed with other groups who tried training models based on similar intuition for longer, only to find the consensus was that no such model truly out-performs the original BiDAF. Our self-attention shows some promise, but the self-attention layer is far more complex and computationally expensive than the bi-attention layer, and takes significantly longer to train. Since the project is time-constrained, we deemed the trade-off between potential performance and time was net negative and we decided to focus our attention on modifying and tweaking the working bi-attention layer (BiDAF model) in effort to develop our own metric of attention's capability to solve MC and QA tasks.

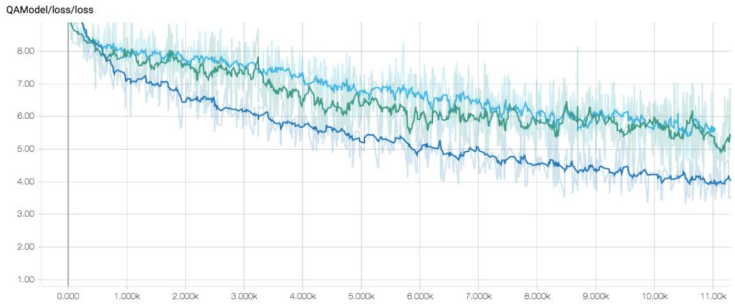

Figure 7: Train loss of bi (dark blue), co (light blue), and self-attention (green) over eleven thousand iterations.

4.3 Character Level Embeddings

We add character-level embeddings to our model, which help in dealing with out of vocabulary words (OOVs) and tend to boost performance in general. We use the model presented in "Learning Character-level Representations for Part-of-Speech Tagging"[6]. Suppose we have individual character embeddings where each character is represented by a vector $r \in \mathbb{R}^{d_{char}}$. Given a word $r_1, \ldots, r_\ell$, with ℓ characters, define $\mathbf{R} \in \mathbb{R}^{\ell \times d_{char}}$ where i^{th} row of $\mathbf{R}$ is the character embedding of the i^{th} character, $r_i \in \mathbb{R}^{d_{char}}$. We then apply a convolution with window size k and filter dimension f so that $\text{conv}(\mathbf{R}) \in \mathbb{R}^{\ell \times f}$. We then find our character-level word embedding by adding a max pooling layer $w_{ch} = \max_{\text{row}}(\text{conv}(\mathbf{R})) \in \mathbb{R}^f$. Now labelling our initial question, context word embedding tensors as $\mathbf{Q}_{word} \in \mathbb{R}^{M \times e}$, $\mathbf{C}_{word} \in \mathbb{R}^{N \times e}$, we can make new question, context character level word embedding tensors $\mathbf{Q}_{char} \in \mathbb{R}^{M \times f}$, $\mathbf{C}_{char} \in \mathbb{R}^{N \times e}$. We then define our final word embeddings as

$$\mathbf{Q} = [\mathbf{Q}_{char}; \mathbf{Q}_{word}] \in \mathbb{R}^{M \times (f+e)}, \mathbf{C} = [\mathbf{C}_{char}; \mathbf{C}_{word}] \in \mathbb{R}^{N \times (f+e)}$$

As an experiment, we also implement models using what we refer to as two layer CNNs, where after executing the above, we then redefine

$$\mathbf{C}_{char} := \max_{\text{row}}(\text{conv}(\text{ReLU}(\mathbf{C}_{char}))), \mathbf{Q}_{char} := \max_{\text{row}}(\text{conv}(\text{ReLU}(\mathbf{Q}_{char})))$$

Additionally, we also construct such a two layer CNN with a $\tanh$ non-linearity rather than a ReLU.

In practice we set $f = 100$ and $k = 5$ and use pre-trained character embeddings with from an open source repository and initialized them as a trainable variable [7]. We then additionally ran a model where we trained our own character embeddings with $d_{char} = 300$ (to mimic the pre-trained embeddings) using a ReLU double layer CNN. All perform relatively similarly, although the traditional one-layer CNN reaches peak performance with less training time.

4.4 Starting, Stopping, and Ensembling

From Figure 3, we see that over 90% of the questions have answers with less than or equal to 10 words. Thus in effort to have our output further mimic the structure of the data, instead outputting a start point i and end point j where $i = \arg\max_k \mathbf{p}_k^{(1)}$ and $j = \arg\max_k \mathbf{p}_k^{(2)}$, we collectively find the pair i, j such that $i, j = \underset{i,j:i \leq j \leq i+10}{\arg\max} \mathbf{p}_i^{(1)} \mathbf{p}_j^{(2)}$. This both prevents our stopping point from being before our ending point, and allows us to ensure that the start and end are within 10 words of one another. As an experiment we changed the range limit of our answer from 10 to 100, and although we did not see as large of an increase in score as with the range limitation set at 10, we still saw a non-trivial rise. This is an indication that the score rises not just due to length restriction, but also due to the decision to take the product of the two softmax probabilities and thus examine the start and end position collectively rather than individually. After training a variety of models in the fashion indicated above, we ensemble 10 BiDAF models with CNNs together using majority voting, 6 of which are double layer CNNs with ReLU and pre-trained character embeddings, 3 of which are single layer CNNs with pre-trained character embeddings, and one of which is double layer CNN with ReLU which trains its own character embedding. This gives us a final dev F1 score of 77.85 and dev EM score of 68.39.

5 Error and Attention Analysis

5.1 General Statistics

Table 1: Results of bi-attention models on dev set. Each subsequent model includes the modifications of the one before it.

Model	F1(%)	EM(%)
BiDAF (ensemble)	77.84	68.39
BiDAF (new span)	76.10	66.23
BiDAF (with CNN)	73.53	63.76
BiDAF (no CNN)	69.37	58.11
Baseline	44.22	35.07

Table 2: Performance of a single BiDAF (with CNN layer) on varying question types in the dev set.

Question Type	F1(%)	EM(%)
When	85.64	79.63
Who	78.03	71.02
Which	76.90	66.13
How	75.93	66.01
Where	73.75	63.43
What	74.52	61.81
Why	65.64	41.77

Table 1 from above displays the added benefit of F1 and EM score from each tweak and modification made to our original model. We clearly see that, in general, the BiDAF model is an effective change as it clearly outperforms our baseline. Table 2 displays a single BiDAF's performance on different types of questions, categorizing questions by chosen question word. The performance between questions is fairly similar, excluding "why" and "when". We find this to be the case as "why" questions tend to have longer answers (Figure 8). "When" questions, which are answered correctly with the highest accuracy, on the other hand, have many short answers (Figure 9). Naturally, the model tends to struggle with long answers, partially because questions with long answers are more complex, but also because of reasons further discussed in the following subsection.

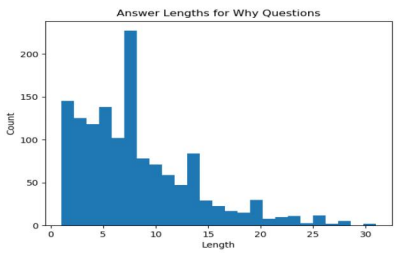

Figure 8: Length of answers to why questions in the train and dev set. Why is the only question word that doesn't have a slim tail for its higher lengths.

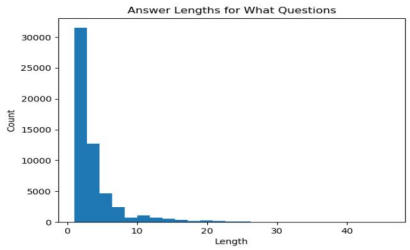

Figure 9: Length of answers to where questions in train and dev sets. All other answer length distributions for questions words are similar to that of "where", although "where" has the slimmest tail.

5.2 Error Analysis: Ambiguity and Improper Answer Spans

Many times, our answer will contain a understandable and correct subset of the true answer, but may contain additional other words or exclude some words from the true answer. We provide an example below, where the EM score and F1 score are penalized due to the prediction's slight differences with true answers, although the predicted answer definitely encapsulates the important necessary information. One can argue, however, that the true answers are more clear. Seeing as the model is already focusing on the proper key words, it is tough to fine-tune the model further and doing so may not result in a huge rise in accuracy. Potential solutions we propose for future work are having the model learn a minimum answer length from the question, thus forcing predictions which are small strict subsets of true answers to be extended, and having the model add in question words to the prediction should they be adjacent to the prediction. The latter will potentially help the model restate the question, a trait that is commonly found in clear true answers.

Context. hormones can act as immunomodulators , altering the sensitivity of the immune system . for example , female sex hormones are known immunostimulators of both adaptive and innate immune responses . some autoimmune diseases such as lupus erythematosus strike women preferentially...

Question. female sex hormones are immunostimulators of which immune responses ?

True Answers. ["adaptive and innate immune responses", "both adaptive and innate", "adaptive and innate immune responses"]

Predicted Answer. adaptive and innate

In certain cases, the improper span actually causes the model to output an answer that misses valuable information.

Context. eu competition law has its origins in the european coal and steel community (ecsc) agreement between france , italy , belgium , the netherlands , luxembourg and germany in 1951 following the second world war . the agreement aimed ...

Question. which countries were the european coal and steel community agreement between ?

True Answers. ["france , italy , belgium , the netherlands , luxembourg and germany", "france , italy , belgium , the netherlands , luxembourg and germany", "france , italy , belgium , the netherlands , luxembourg and germany"]

Predicted Answer. france , italy , belgium

5.3 Error Analysis: Miss-Attention

At times, present examples where our model incorrectly answers the question as it pays attention to improper parts of the context due to the words used in the question. We give an example below.

Context. in 1854 at ballarat there was an armed rebellion against the government of victoria by miners protesting against mining taxes (the "eureka stockade") . this was crushed by british troops , but the discontents prompted ...

Question. what armed group stopped the uprising at ballarat ?

True Answers. ["british troops", "british troops", "british"]

Predicted Answer. miners

Due to the nature of similarity encoding, the model heavily weighted "ballarat" and "armed" and thus picked an actor that was close to said words. To show this we present a heat-map of the similarity matrix (Figure 10). The clear correct answer "british troops", displays a key flaw in attention's ability to truly comprehend a passage. In this case, the attention model is unable to parse meaning and rather focuses on finding an appropriate word around high-attention buzz-words.

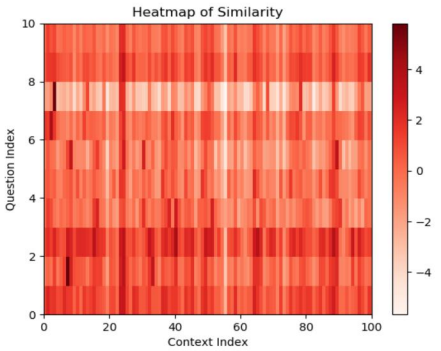

Figure 10: A heat-map of the similarity matrix for the above example. Note the two particular hot spots at (2, 8) and (4, 2), which correspond exactly to repeated words in the question and context. In general, we see hotter similarity close to where "miners" appears in the context than where "british troops" appears.

5.4 Error Analysis: Syntactic Ambiguities

One uncommon kind of error is when the model finds an answer that is technically true, but not likely what a human would use to answer the question. We give an example below.

Context. highly concentrated sources of oxygen promote rapid combustion . fire and explosion hazards exist when concentrated oxidants and fuels are brought into close proximity ; an ignition event , such as heat or a spark , is needed to trigger combustion . oxygen is the oxidant , not the fuel , but nevertheless the source of most of the chemical energy released in combustion . combustion hazards also apply to compounds of oxygen with a high oxidative potential , such as peroxides , chlorates , nitrates , perchlorates , and dichromates because they can donate oxygen to a fire .

Question. peroxides , nitrates and dichromates are examples of what type of compounds ?

True Answers. ["compounds of oxygen with a high oxidative", "compounds of oxygen with a high oxidative potential", "compounds of oxygen with a high oxidative potential"]

Predicted Answer. combustion hazards

6 Conclusion and Future Work

We find that complex attention has lead to vast improvements relating to QA and MC tasks, but the ideas behind attention, although intuitive, perhaps have not yet been expressed in their optimal mathematical format. In general, we find that different varieties of complex attention tend to be built in differing frameworks, presenting some ambiguity regarding how well the attention layer is performing its intended job vs. benefiting the model in other undocumented capacities. This is evidenced to some degree by our results on self and co-attention in the BiDAF framework, where the self and co-attention layers fail to fit the train data as well as bi-attention when placed in a framework catered to bi-attention. Yet, such layers perform similarly to BiDAF when implemented with a different model structure. Given more time we would like to continue running our co-attention and self-attention models to see how they compare to BiDAF in the long run. Interesting future work could involve developing a model structure in which the layers perform with a similarly high accuracy, or combining such attention layers either internally in the model or externally by ensembling.

In maximizing score outputs of our BiDAF ensemble, we find that character level word embeddings, methods for determining optimizing our answer span, and ensembling are all very effective. The ensemble model performs at an impressive level, and displays attention's ability to effectively recognize important parts of the context when given a question. We do see, however, that the model operates by focusing on key-words and discerning acceptable answers around them, and thus at times fails to address the MC task at a high level. Moving forward we believe it is important to construct models with specific layers that focus more directly on capturing the meaning of phrases in the context and question, rather than largely basing predictions only off of similarity measures between important words.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. CoRR, abs/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] Abi See, et al. CS 224N Default Final Project: Question Answering." Feb. 2018.

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

[5] Natural Language Computing Group, Microsoft Research Asia. R-Net: Machine Reading Comprehension with Self-Matching Networks. 2017.

[6] Santos, C., & Zadrozny, B. (2014). Learning Character-level Representations for Part-of-Speech Tagging. Proceedings of the 31st International Conference on Machine Learning, ICML-14(2011), 18181826.

[7] Max Woolf. Pretrained Character Embeddings for Deep Learning and Automatic Text Generation. April 2017