Reading Comprehension and Question Answering with Bidirectional Attention Flow

Andrew Huang
Department of Computer Science
Stanford University
Stanford, CA 94305
ahuang98@stanford.edu

Michael Ko
Department of Psychology
Stanford University
Stanford, CA 94305
mlko53@stanford.edu

Abstract

Reading comprehension and question answering are difficult tasks in machine learning, due to the complex interactions inherent in natural language. The SQuAD dataset, a publicly available reading comprehension dataset, provides a benchmark for these critical tasks. We present a reimplementation of the BiDAF (Bidirectional Attention Flow) that was able to achieve 63.747 EM and 74.773 F1 on the test set of SQuAD on a single model.

1 Introduction

Machine comprehension is a specific task in machine learning where the model is given a question and context and then must answer it. The task is critical for digital assistants like Amazon Echo, accessibility software, and telehealth products. Because of these widespread applications and the potential gains to both linguistics and neural network research, the task has become increasingly popular as a research task.

The SQuAD dataset, published by Rajpurkar et al. [1] in 2016, was released in order to provide means for researchers to pursue human performance in machine comprehension. The dataset consists of over 100,000 question-answer pairs on over 500 Wikipedia articles, along with a public leaderboard. Many of the models use various attention mechanisms to capture the key sections of the context pertaining to the question. Multiple entries on the leaderboard [2] are already approaching human-level performance, reflecting the progress that has been made in the task.

2 Background and Related Work

Much work has been done of the SQuAD datasets, with researchers from prominent organizations all contributing high-performing models. As mentioned before, many of these models implement variations of attention mechanisms to increase their performance, including self-attention and coattention. While researching models that performed well, a few in particular caught our attention.

2.1 SQuAD Problem Definition

Each example of the Stanford Question Answering Dataset provides a context and a query. For any context with a word sequence of length T, $\mathbf{p} = \{p_1, p_2, ..., p_T\}$ and query with a word sequence with length J, $\mathbf{q} = \{q_1, q_2, ..., q_J\}$, the model must learn a function that maps context and query to two scalars that define the start a_1 and stop index, a_2, of the answer.

For the purposes of exploratory data analysis, it is helpful to plot the histogram of word sequence length for context, query, and answer span in order to understand the problem and exploit potential problem simplification. We also found the distribution of where in the context, relative to the length of the context paragraph, each answer was. Most answers tended to be towards the beginning and center of paragraphs. We noticed that most context lengths were greater than 300, and since generally answers were not found too close to the end, that we could clip our maximum context paragraph to 300 words. Similarly, we clipped out question and answer lengths to 30 words.

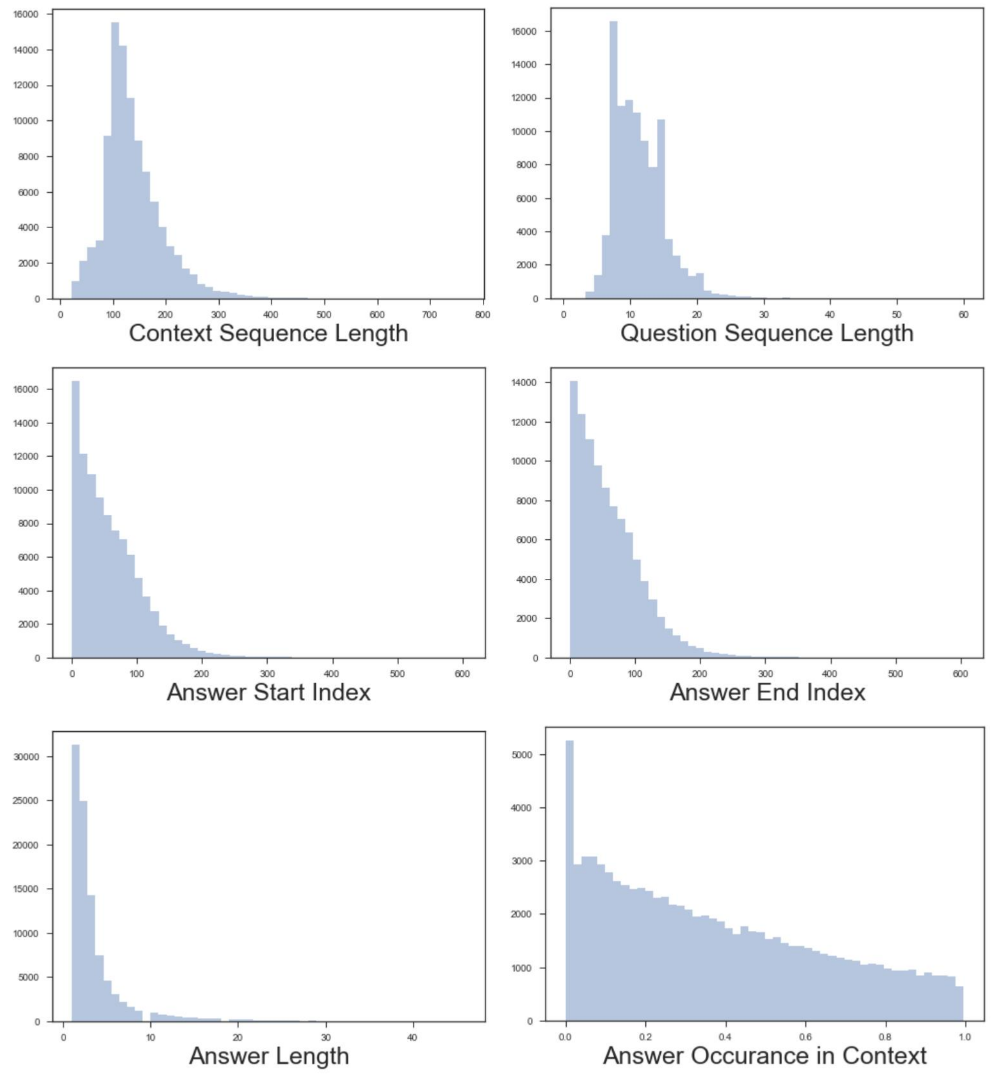


Figure 1: Histogram of key word sequences in SQuAD.

We observed that context length rarely exceeded 300 words and question length rarely exceeded 25 words. Furthermore, answers very rarely happened at the end of the context, suggesting that capping the sequence length to a maximum context length could reduce the complexity of the data without sacrificing performance. Looking at the where the answers are occurring relative to the context, we see that there is a substantial percentage of answers happening at the beginning of the context, but generally decline towards the middle and end of the context.

2.2 Bidirectional Attention Flow (BiDAF)

Developed by Minjoon Seo et al. [3], BiDAF is a hierarchical multi-stage architecture for modeling representations of the context paragraph at different levels. In their implementation, they utilize character embeddings and word embedding to create contextual embeddings, which are then fed through attention and modeling layers. The attended vectors are computed at each time step, and fed through to the modeling layer. There is attention from query to context and vice versa. This model performed extremely well on the SQuAD dataset, and we reimplemented it in our model.

2.3 R-Net

R-Net is a relatively simple model that performed very well on the SQuAD dataset [4]. Developed by researchers at Microsoft Research Asia, it utilizes a self-attention mechanism, to aggregate information from the whole passage. They also add gates to their attention-based recurrent networks. The model also performed extremely well on the SQuAD dataset. The self-attention mechanism caught our eye as an excellent idea that could be appended on top of existing attention layers.

3 Approach

3.1 Model Architecture

Our implementation of Bidirectional Attention Flow Model has 5 layers as follows:

1. **Word Embedded Layer** that maps each one hot encoded word to a vector representation. We used a set of pre-trained word vectors, GloVe [5] which has yielded good results in many natural language processing tasks.
2. **Contextual Embedding Layer** uses a Bidirectional LSTM to change each word vector to a word embedding that captures semantic meaning of nearby contextual words.
3. **Attention Flow Layer** uses both query embeddings and context embeddings to generate a query attending on context features (Query2Context) and a context attending on query features (Context2Query).
4. **Modeling Layer** uses the features generated from the Attention Flow Layer with a stacked Bidirectional LSTM to transform highlighted query and context attended features to task specific features.
5. **Output Layer** creates a probability distribution of the index of the start of the answer and the index of the end of the answer.

1. Word Embedded Layer: We transform the one hot encoded word vectors into a shared representation of the words using pretrained 300 dimensional GloVe embeddings. Thus we transform $\mathbf{p} = \{p_1, p_2, ..., p_T\}$ and $\mathbf{q} = \{q_1, q_2, ..., q_J\}$ into $\mathbf{X} \in \mathbb{R}^{d \times T}$ and $\mathbf{Q} \in \mathbb{R}^{d \times J}$.

2. Contextual Embedding Layer: In order to capture representations of each word dependent of contextual usage, we used Long Short-Term Memory cells that are placed bidirectionally. The embedding for both context and query is constructed by the concatenation of the forward LSTM and the backward LSTM. We opt to create a different LSTM cell for the forward direction and the backward direction because we believe that processing textual information should be fundamentally different in the reverse direction. However, we chose to share LSTM weights for the forward LSTM and backward LSTM used to process query and context. Low level semantic processing of the query and the context should be no different from each other. We obtained $\mathbf{H} \in \mathbb{R}^{2d \times T}$ and $\mathbf{U} \in \mathbb{R}^{2d \times J}$ from $\mathbf{X}$ and $\mathbf{Q}$.

3. Attention Flow Layer: From the contextual embeddings, we computed a similarity matrix S. The similarity score for the t-th context embedding and j-th query embedding is computed via $S_{tj} = w_{(S)}^T[\mathbf{h}; \mathbf{u}; \mathbf{h} \circ \mathbf{u}]$, where $w_{(S)}^T$ is a trainable vector. Attention in both directions (Context2Query) and (Query2Context) a is computed by softmax function. Attended vector representation is computed by a weighted summation of a and either $\tilde{\mathbf{H}}$ or $\tilde{\mathbf{U}}$. Finally, we used the newly computed attended vectors $\mathbf{U}$ and $\mathbf{H}$ is combined with the previous embedded feature by defining $\mathbf{G} = [\mathbf{h}; \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{h}}]$. Thus $\mathbf{G} \in \mathbb{R}^{8d \times T}$.

4. Modeling Layer: We used two stacked BiDirectional LSTM to transform our attended vectors. The output of the last Bidirectional LSTM is concatenated for both forward and backward direction, yielding $M \in \mathbb{R}^{2d \times T}$.

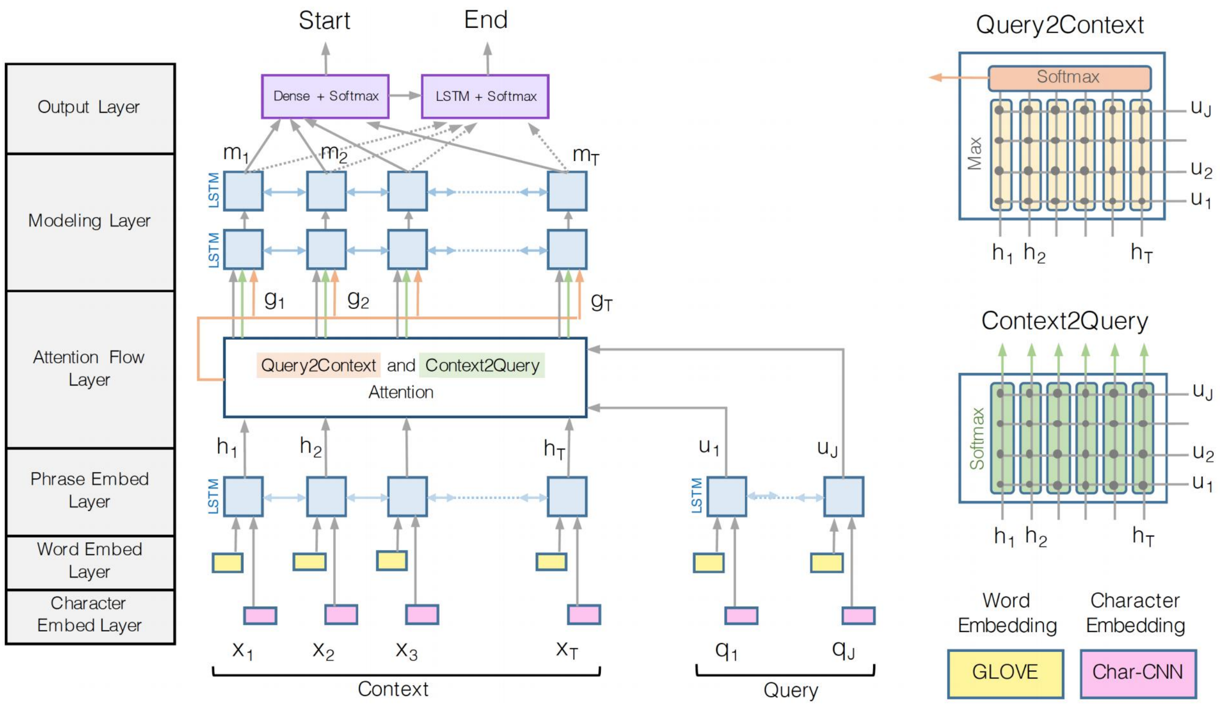


Figure 2: Architecture of Bidirectional Attention Flow Model; Adapted from Seo et. al., 2017. [3]

5. Output Layer: The output for SQuAD requires us to return a probability distribution over T of both the start index of the answer and the stop index of the answer. The start index p_1 is computed by $\text{softmax}(w_{p1}^T[\mathbf{G};\mathbf{M1}])$. Similarly, the end index $\text{softmax}(w_{p1}^T[\mathbf{G};\mathbf{M2}])$, where **M2** is the output of a last layer bidirectional LSTM. In this layer, we also implemented a dynamic programming algorithm; with this algorithm, we added constraints that the end index had to be after the start index, and that they couldn't be more than 30 words apart. In the original implementation, they simply took the argmax of both distributions, whereas here, we can efficiently narrow ourselves to more likely answers. We chose 30 words as our threshold after the initial exploratory visualizations in Figure 1.

3.2 Training

We used an Adam optimizer with an initial learning rate of 0.01. We also implemented learning rate annealing, where we multiplied the learning rate by 0.1 every 6,000 iterations so that it could learn more. Additionally, we implemented exponential moving average, with a decay value of 0.999. We experimented with a few values for dropout as a form of regularization, ultimately arriving at 0.2. We used batch size 100, and the 300-dimensional GLoVe vectors while training. After looking at our initial visualizations from the Data subsection, we decided to cap question length at 25 and context length and 300, in order to take memory usage into account while still maintaining high performance. We strongly considered ensembling, but decided against it for time and compute resource reasons.

4 Experiments

4.1 Data and Evaluation

The SQuAD dataset, as described before, consists of 100,000+ question-answer pairs and 500+ context paragraphs. Each answer pair came with three answers generated by humans as ground-truth. We evaluate the performance of our model using two metrics: Exact Matching score (EM), defined as the percentage of our predicted answers which matched one of the ground-truth answers, and F1, defined as:

$$F1 = \frac{2}{\frac{1}{\mathrm{P}} + \frac{1}{\mathrm{R}}}$$

$$\text{where, } P = \frac{\text{true positives}}{\text{predicted positives}}, R = \frac{\text{predicted positives}}{\text{all positives}}$$

Before modeling, we wanted to get a better sense of the data, and thus created the plots seen below, These plots became useful when deciding on hyperparameters for our model.

4.2 Results

We first evaluated our model on a small subset of the training set, to see if it could overfit while we were still tinkering with our model. We then trained on the whole training set. Our single model performed well on SQuAD, although we were not fully able to replicate the results of the original BiDAF paper. We achieved an EM of 63.747 and 74.773 on the test set, as seen in Table 1. We note that the dynamic programming component added to the output layer increased our performance quite a bit. The performance of our model relative to the others gave us ideas of improvements on our model for the future, mentioned later.

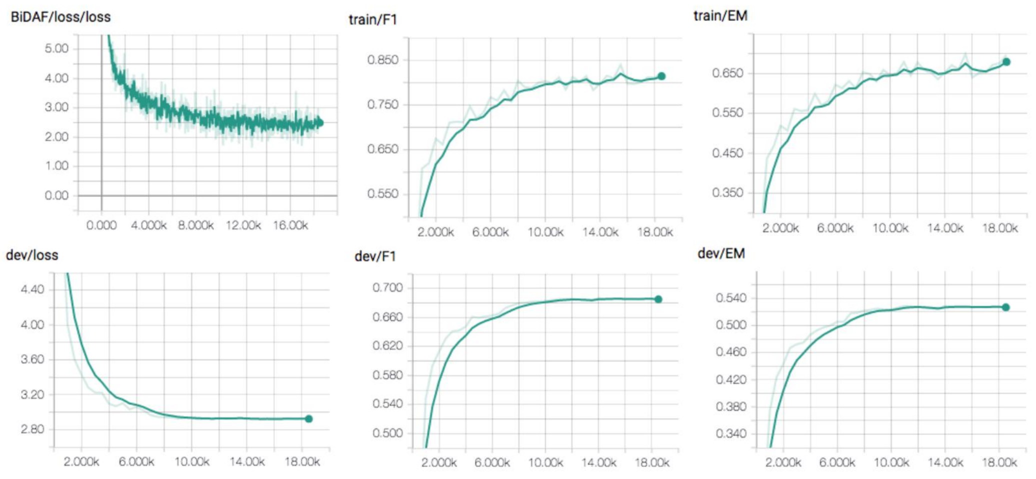


Figure 3: Loss, F1, and EM graphs for train and dev sets.

As seen in Figure 2, the model appeared to fit well. We decided to stop training around this time because we noticed that the dev set loss was starting to rise, and the training set evaluation metrics were starting to spike, which is indicative of the start of overfitting. We also note that this model performed better, and trained much faster, using the Adam optimizer [5] rather than the Adadelta optimizer, which was used in the original BiDAF paper. We believe that with better learning rate annealing, our performance could have been improved even more. We trained the same model using an Adadelta optimizer, and it was unable to reach the same performance as Adam on our model even after training for over 40,000 iterations, at which point it also began to overfit.

4.3 Analysis

4.3.1 Question Analysis

We wanted to break down our performance on different question types. We partitioned dev-1.1.json into six separate json files based on the presence of a question word, and then evaluated our model on each one.

As you can see in Table 2, we performed worst on ”why” questions, as we expected. We performed surprisingly well on ”how” questions, although it is likely many of those questions could have been

Table 1: Performance of Models on SQuAD

MODEL	EM	F1
Our Model	**63.747**	**74.773**
Our model w/o DP	60.946	71.768
Baseline (single model)	28	39
BiDAF (single model)	67.974	77.323
BiDAF (ensemble)	73.744	81.525
BiDAF + self-attention (single model)	72.139	81.048
r-net (single model)	76.461	84.265

Table 2: Performance of Our Model on Questions

QUESTION TYPE	EM	F1
Overall	62.829	74.076
Who	66.908	76.015
What	59.721	72.009
Where	60.521	73.064
When	75.556	82.596
Why	43.037	64.859
How	64.758	75.857

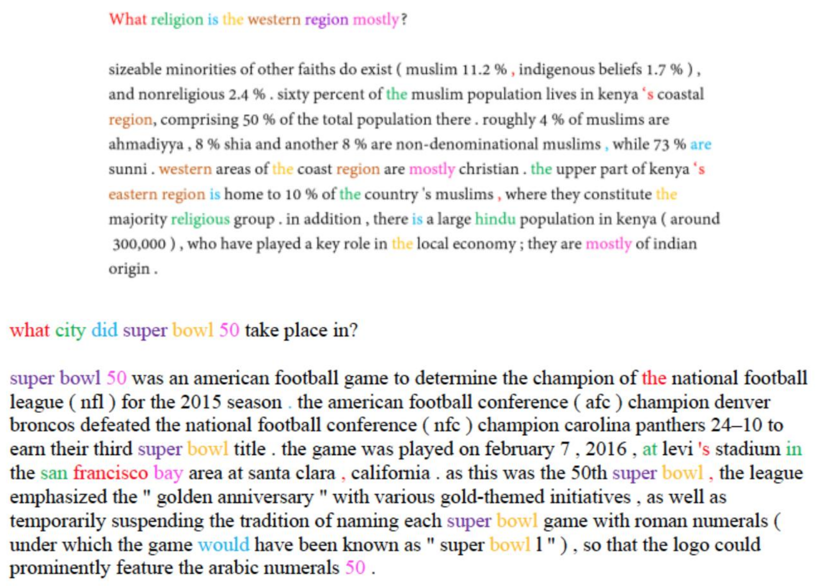
in the vein of ”how many,” which would not as difficult as a ”how” or ”why” question. Our model performed best on ”when” questions, which makes sense as those are often the most straightforward. ”What” questions comprised a vast majority of the questions, and our model about average on them. The length of the answers likely was a large factor in the difference in performance; one would expect ”why” questions to have significantly longer answers than ”when” questions.

In addition to the raw values, we were also interested in the differences between the EM and F1. Generally, the better the scores, the less the distance between the scores. ”Why” questions had not only the lowest scores, but also by far the largest difference between these two metrics. This makes sense given our assumptions about the length of the answers and the complexity of each question type.

We found this data bias interesting; we think it might be because a ”why” question-answer pair might be difficult to come up with given the constraint that it has to be an exact phrase from the context paragraph. Since these questions typically are most difficult to answer, and thus represent the largest hurdle for machine comprehension models, perhaps this reflects an inherent weakness in SQuAD.

4.3.2 Attention Analysis

For each question, we found the attention activations corresponding to each question word and color coded the four most attended words per question word. These visualizations are displayed in Figure 4. Through this visualization, we can see that words relating to the question word are highlighted, even when they aren’t direct stems/forms of each other. For example, religion also activated the word ”hindu” in addition to ”religious.” Although at first we were confused when ”religion” activated ”the,” we saw that the ones where it did activate were often in proximity to a religious word, making it a suitable choice for a starting index for an answer. We also noted that the question word activated a lot of punctuation and endings, like the possessive ”s.” We postulate that these are often indicators that useful information is close, and thus these would become activated. Words like ”the” and ”is” generally activated their same words. There was some overlap between words (i.e. words that were activated by multiple words in the query), and we did not color those words accordingly; these words tended to be articles (e.g. ”the”) or punctuation, supporting our earlier idea that these are often indicators of information within a passage.


What religion is the western region mostly?

sizeable minorities of other faiths do exist (muslim 11.2 % , indigenous beliefs 1.7 %) , and nonreligious 2.4 % . sixty percent of the muslim population lives in kenya 's coastal region, comprising 50 % of the total population there . roughly 4 % of muslims are ahmadiyya , 8 % shia and another 8 % are non-denominational muslims , while 73 % are sunni . western areas of the coast region are mostly christian . the upper part of kenya 's eastern region is home to 10 % of the country 's muslims , where they constitute the majority religious group . in addition , there is a large hindu population in kenya (around 300,000) , who have played a key role in the local economy ; they are mostly of indian origin .

what city did super bowl 50 take place in?

super bowl 50 was an american football game to determine the champion of the national football league (nfl) for the 2015 season . the american football conference (afc) champion denver broncos defeated the national football conference (nfc) champion carolina panthers 24–10 to earn their third super bowl title . the game was played on february 7 , 2016 , at levi 's stadium in the san francisco bay area at santa clara , california . as this was the 50th super bowl , the league emphasized the " golden anniversary " with various gold-themed initiatives , as well as temporarily suspending the tradition of naming each super bowl game with roman numerals (under which the game would have been known as " super bowl l ") , so that the logo could prominently feature the arabic numerals 50 .

Figure 4: Attention Paragraphs. Colored context words had the largest attention activations for question words of the corresponding color.

This first example was one on which the model performed well; we also performed the same visualization on a question-answer pair on which the model performed poorly. In this question, the correct answer was ”santa clara,” but the model guessed ”roman numerals.” Notice that none of the words ”santa,” ”clara,” or ”california” were attended to. We noticed that ”san,” ”francisco,” and ”bay” were all attended to. Perhaps it has to do with sparsity in GloVe; San Francisco is a much more prominent city than Santa Clara and thus is more likely to explain this context. The words relating to location were generally all attended to multiple times, so it was going in the right direction. Interestingly, ”roman numerals,” was the guess. Perhaps since it was late in the question and had lots of nearby attention activation from super bowl, the signal from the past was taken out, and since ”san francisco” and ”santa clara” didn’t activate the model enough, then it ended up with ”roman numerals.”

5 Conclusion

In this project, we reimplemented the BiDAF model, and were able to achieve similar results with our single model. We added dynamic programming in the output layer, which we showed had a fairly substantial performance bump, as well as changing the optimizer and training procedure (adding learning rate annealing). The model performed well on the SQuAD dataset, but had more difficulty with certain types of questions.

There are a lot of further directions to go in. One idea we like was to use a similar self-attention mechanism from R-net on top of BiDAF to further improve the context and query matching. We also think further adding in constraints to the output layer, using specialized features like POS-tagging other more traditional NLP techniques would also help boost the performance in order to integrate deep learning’s highly empirical models with more Bayesian ideas. Ensembling models would certainly improve the performance, although we do not believe that is where the larger and more significant/interesting performance gains are to be made. In the future, further hyperparameter tuning and testing of different optimization strategies could also have increased performance of the model.

We also think it would be interesting to test the ability of the model for different question types. For example, if a model wasn’t trained on ”when” questions, would it still be able to answer them? It could reveal deeper insights into how the model chooses its answers, and on the true relative difficulty of different kinds of questions. We would also be curious on testing our model on other datasets; to see if the model is good at both tasks, and if a model trained on SQuAD could still perform well on other datasets performing similar tasks.

6 Acknowledgements

We want to thank Richard Socher, Kevin Clark, Abi See, and the rest of the teaching team for being a great, responsive, supportive teaching staff! We both were enamored with the material and had a lot of fun working on the project. We would also like to thank Microsoft for computational resources!

7 References

[1] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250.*

[2] *https://rajpurkar.github.io/SQuAD-explorer/*

[3] Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603.*

[4] Wang, W., Yang, N., Wei, F., Chang, B., & Zhou, M. (2017). 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)* (Vol. 1, pp. 189-198).

[5] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[6] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*