SQuAD GOALS
Guided Objective Advanced Learning System

Derek J. Phillips
Department of Computer Science
Stanford University
Stanford, CA 94305
djp42@stanford.edu

Abstract

In this work we present a neural-network based question answering model that is inspired by recent advances in the field, notably the bidirectional attention flow model. We guide the objective by conditioning the end of the answer span on the prediction for the start of the answer, resulting in an advanced system capable of learning to answer questions well. Our simple addition of differentiable conditioning results in significant improvement in the model's performance. Our final model attains an F1 score of 75.764% and an EM score of 65.96% on the test set, which is sufficient to demonstrate the improvement provided by our simple conditioning technique, which is just under 2% for each metric. We hope to present a thorough comparison with other conditioning methods in future work.

1 Introduction

The quest for artificial intelligence is usually centered around interaction with humans, which itself relies on the ability to communicate. This necessity coupled with recent advances in deep-learning have motivated further studies in natural language processing in recent years, with focus areas including chat-bots [3], speech-recognition [1], and question-answering [4], which is the focus of this work. The task of machine-based question-answering is similar to critical reading tests for humans, although in a simplified way. The goal is to identify the span of a contextual paragraph that corresponds to the answer of a given question.

To facilitate research in this area, Stanford's NLP laboratory created and released a dataset of context, questions, and crowd-sourced answers, called the Stanford Question Answering Dataset (SQuAD [4]) in 2016. The data-set promotes a centralized environment for researchers to compare and evaluate their models, and has seen much activity in recent years. We present a subset of the interesting models that exhibit exceptional performance on the dataset in Section 2.

Following the related work, we present a detailed outline of our approach including a description of our final model in Section 3. Then, we thoroughly describe the experiments we run in Section 4, as we encountered many different factors that demonstrate significant influence over the model performance. We by no means provide a comprehensive hyper-parameter tuning due to the combinatorial nature of the problem (which is often the case in deep-learning problems [2]), but we do perform numerous local searches. This section is also where we provide in-depth descriptions of the improvements included in our model, and Section 4.4 describes our simple approach to differentiable conditioning. We follow these results and descriptions of our experiments with error analysis in Section 5 and overall conclusions in Section 6.

2 Background/Related Work

Many researchers have developed models for question answering and subsequently tested them on SQuAD. The top performing models often involve interesting and novel applications of the concept of attention to the problem, which helps models identify the important part of paragraphs to use for finding an answer. Specific examples of this include the bidirectional attention flow technique [5], which we implement in this work, and co-attention to "attend" the question and context simultaneously [9]. There are other methods of attention as well, such as self-attention [6]. Other work includes the use of character-level embeddings, which have been shown to better handle out-of-vocabulary words [8], but only improve the model performance by a few points [5]. Thus, we opt to leave the addition of character-level embeddings to future work.

3 Approach

3.1 Data Analysis

One of the first steps to approaching any machine learning problem is to understand the data. In the problem of question answering, there are two key characteristics that we want to understand:

1. The lengths of the components of the data we are given.
2. Where does the answer generally appear?

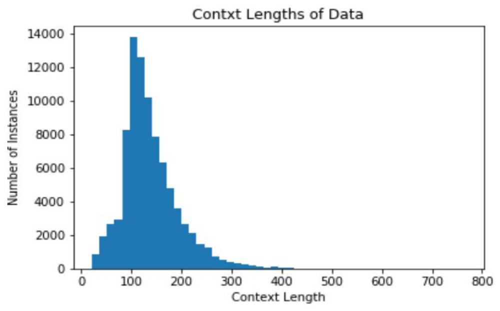

Figure 1: Lengths of the context paragraphs.

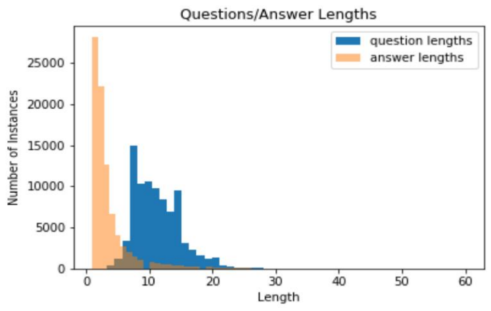

Figure 2: Lengths of the questions and answers.

We examined our data and found that the context is no more than 400 tokens long in the vast majority of examples, as seen in Fig. 1, which will be helpful when we design our model. In Fig. 2 we see that the question and answer are almost always less than 30 tokens long. Regarding our second characteristic, we first examined the absolute position of the start and end, which wasn't extremely useful and is displayed in Fig. 3. However, when we characterized the relative position of the answer as in Fig. 4 we observed that the answer tends to be closer to the start of the context, but this tendency is not significant; the answer seems to be well distributed throughout the context.

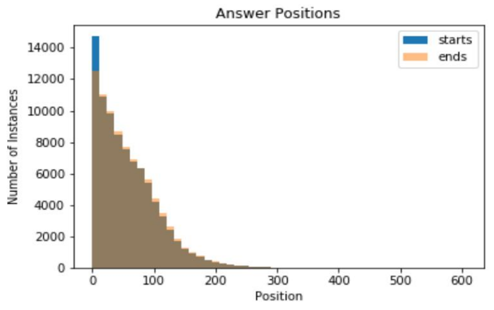

Figure 3: Start and end locations for answer spans.

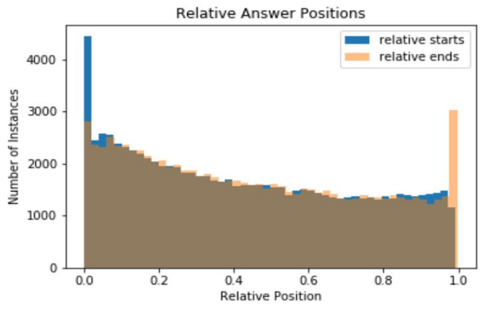

Figure 4: Relative start and end locations.

3.2 Model

The final model we use is similar to the BiDAF [5] model, but it is not exactly the same as we leave out a number of improvements, including the character-level embedding. We built up the model by starting with the baseline and incrementally testing improvements, keeping those the best instantiation. This process is detailed in Section 4. The final model we use is similar to most question answering models [5], and consists of four main layers: (1) LSTM Encoder, (2) Bidirectional Attention, (3) 2xLSTM Modeling, (4) Conditioned Softmax Output, as shown in Fig. 5. Our main contribution is the conditioning of the end location prediction on the predicted distribution of the start location, which we describe in Section 3.3.

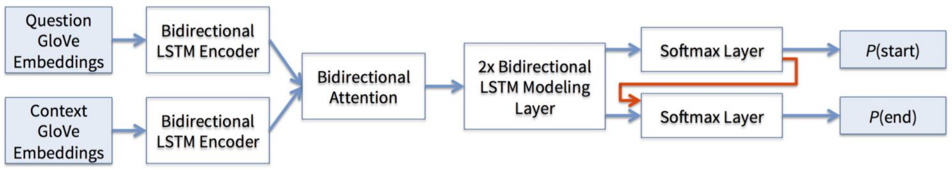
Question GloVe Embeddings
Context GloVe Embeddings
Bidirectional LSTM Encoder
Bidirectional LSTM Encoder
Bidirectional Attention
2x Bidirectional LSTM Modeling Layer
Softmax Layer
Softmax Layer
P(start)
P(end)

Figure 5: Model Layout

3.2.1 Encoder Layer

We leave the encoder layer largely unchanged from the baseline model. It consists of a bidirectional LSTM that shares weights between the context and the question encodings. The context and question are represented as sequences of word embeddings, $\boldsymbol{x_1}, \ldots, \boldsymbol{x_N} \in \mathbb{R}^d$ and $\boldsymbol{y_1}, \ldots, \boldsymbol{y_M} \in \mathbb{R}^d$ respectively, where d is the dimension of the embeddings. These are from pre-trained GloVe embeddings.We keep d at 100 for all of our tests, as we saw that previous attempts to increase the dimension often do not produce significant improvements.We omit many of the equations because they appear throughout the literature [5], but the key is that $\boldsymbol{x}$ and $\boldsymbol{y}$ are fed through a bidirectional LSTM with shared weights to output:

$$\boldsymbol{c}_i = [\overrightarrow{\boldsymbol{c}_i}; \overleftarrow{\boldsymbol{c}_i}], \boldsymbol{q}_j = [\overrightarrow{\boldsymbol{q}_j}; \overleftarrow{\boldsymbol{q}_j}] \in \mathbb{R}^{2h}, \forall i = 1, \ldots, N; j = 1, \ldots, M$$

3.2.2 Attention Layer

For the attention layer we use a bidirectional flow attention technique [5], and do not make any significant modifications. Below is a summary of the main computations. First, we compute a similarity matrix $\boldsymbol{S}$:

$$\boldsymbol{S}_{ij} = \boldsymbol{w}_S^\top [\boldsymbol{c}_i; \boldsymbol{q}_j; \boldsymbol{c}_i \circ \boldsymbol{q}_j] = \boldsymbol{w}_c^\top \boldsymbol{c}_i + \boldsymbol{w}_q^\top \boldsymbol{q}_j + \boldsymbol{w}_{cq}^\top (\boldsymbol{c}_i \circ \boldsymbol{q}_j) \in \mathbb{R}.$$

The second representation above represents the way we calculate the value, as it is a more efficient computation in terms of memory. $\boldsymbol{S}$ contains similarity scores for each pair of context and question hidden states. We then compute Context-To-Question attention, which computes attention values a by computing $\boldsymbol{a}_i = \sum_{j=1}^{M} \boldsymbol{\alpha}_j^i \boldsymbol{q}_j \in \mathbb{R}^{2h}$, where $\boldsymbol{\alpha}^i = \text{softmax}(\boldsymbol{S}_{i,:}) \in \mathbb{R}^M$.

Next we compute Question-To-Context attention: $\boldsymbol{c}' = \sum_{i=1}^{N} \text{softmax}(\boldsymbol{m})_i \boldsymbol{c}_i \in \mathbb{R}^{2h}$, where $\boldsymbol{m}_i = \max_j(\boldsymbol{S}_{i,j}) \in \mathbb{R}$. The output of this layer is the concatenation $\boldsymbol{b}_i = [\boldsymbol{c}_i; \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{c}'] \in \mathbb{R}^{8h}$. Due to the subtle nature of some of these computations, we created a unit testing framework to ensure the operations are completed as intended. This is included the the code for the project, entitled *test.py*.

3.2.3 Modeling Layer

Our modeling layer consists of two bidirectional LSTM layers. These take in $\boldsymbol{b}$ and outputs a matrix $\boldsymbol{M} \in \mathbb{R}^{N \times h}$. We do not do anything special here, as this is a standard approach [5]. However, we see that the addition of this layer is the single most significant improvement to our model, and the second layer improves performance even more.

3.2.4 Output Layer

The output layer is where we insert our significant contribution. See Section 3.3 for a detailed description. The main idea is that we compute the probability distributions for the start and end location of the answer span using the matrix $\boldsymbol{M}$ that is output from the modeling layer. Thus, if our output layer (which includes a fully-connected layer) is called O:

$$P(\text{start}) = \text{softmax}(O_s(\boldsymbol{M})) \in \mathbb{R}^N \tag{1}$$

$$P(\text{end}) = \text{softmax}(O_e([\boldsymbol{M}; p_{\text{start}}])) \in \mathbb{R}^N, \text{ where } [\boldsymbol{M}; p_s tart] \in \mathbb{R}^{N\times(h+1)} \tag{2}$$

3.2.5 Hyper-parameters

Some main hyper-parameters for our final model are a hidden layer size $h = 200$, a batch size of 100, embedding size $d = 100$, and a dropout probability of 0.25, although we do some fine tuning at the end of training to decrease it to 0.2. We limit the context length to 400 tokens, and the question length to 30. Our best learning rate begins at 0.001, and perform some fine tuning at the end of training our best model. We clip the gradients to a maximum norm of 5.0.

3.3 Contribution

Our most novel contribution is a simple method of accounting for the start position when predicting the end position of an answer, which we explain here. The goal of question-answering is to compute the span of the text that corresponds to the answer of a question:

$$P(\ell^{\text{start}}, \ell^{\text{end}} \mid \text{context}, \text{question}) \tag{3}$$

However, many approaches assume that the start and end positions for the answer are independent:

$$P(\ell^{\text{start}}, \ell^{\text{end}} \mid \text{context}, \text{question}) = (P(\ell^{\text{start}} \mid \text{context}, \text{question}), P(\ell^{\text{end}} \mid \text{context}, \text{question})) \tag{4}$$

This assumption is not strictly true given that the end position is at least after the start position based on the syntax of the English language. Furthermore, in problems with duplicate answers (see Section 5.5 for an example where the answer word appears multiple times in the text), it is intuitively helpful to know which of the answers was weighted higher by the model when predicting where to end the span. Research on conditioning the end position on the start position has produced some complicated models that require running an RNN for exactly two-steps and can be interpreted as non-differentiable if implemented with the use of an $\arg\max$ operation [7]. This results in:

$$P(\ell^{\text{start}}, \ell^{\text{end}} \mid \text{context}, \text{question}) = (P(\ell^{\text{start}} \mid \text{context}, \text{question}), P(\ell^{\text{end}} \mid \text{context}, \text{question}, \ell^{\text{start}})) \tag{5}$$

Our approach is much simpler and entirely differentiable. We do not rely on any use of RNNs, and simply use the probability distribution $P(\ell^{\text{start}})$ that we compute through a softmax layer as an additional set of features to our computation of $P(\ell^{\text{end}})$. This results in:

$$P(\ell^{\text{start}}, \ell^{\text{end}} \mid \text{context}, \text{question}) = (P(\ell^{\text{start}} \mid \text{context}, \text{question}), P(\ell^{\text{end}} \mid \text{context}, \text{question}, P(\ell^{\text{start}}))) \tag{6}$$

While much simpler than previous approaches, this formulation does significantly improve the performance of a model that does not include any conditioning, but we leave a direct comparison against other conditioning methods for future work.

4 Experiments

In this section we explain the tested improvements in roughly the order we explored them, describing the improvement and the effect on the performance, if any. The testing was not always performed until convergence, as we could occasionally distinctly seem trends emerge early in the process. Fig. 6 shows the observed performance benefit for the improvements.

4.1 LSTM vs GRU

The first improvement we explored was replacing the GRU RNN layers with LSTM layers. We never tested our final model with GRU layers, but the early testing provided a noticeable improvement of 1.56% on the Dev EM score and 1.76% on the Dev F1 score. We expect that testing this in the final model would produce more significant improvements, as we tripled the number of RNN layers.

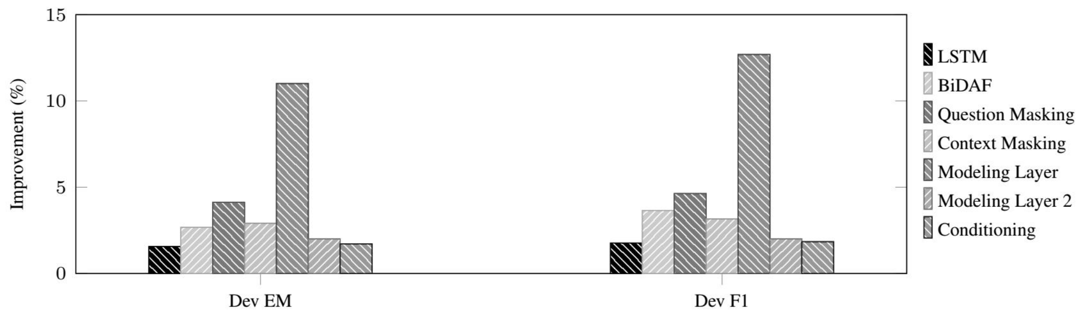

Figure 6: Performance increase for each improvement.

4.2 Basic Attention vs BiDAF

The main implementational challenge of our project was implementing Bidirectional Attention Flow [5], which necessitated subtle memory management techniques. The layer, described in Section 3.2.2, is much more expressive than the baseline's dot-product attention. The attention mechanisms resulted in respectable improvements of 2.67% and 3.65% for the Dev EM and F1 scores.

4.2.1 Question Masking

In the above tests we omitted the inclusion of question and context masking to isolate the improvement of just the bidirectional attention. We tested the addition of question masking, which effectively tells the model which tokens of the question should be ignored because they are just padding tokens. The improvement here was larger than from just the attention layer, which is interesting. The Dev EM and F1 increased by 4.12% and 4.63%, respectively.

4.2.2 Context Masking

Naturally, we also tested the inclusion of context masking, which also provided a significant increase of 2.91% for Dev EM and 3.16% for th Dev F1 scores. However, we did not test the addition of masking to convergence; these values indicate the general significance of the improvements.

4.3 Bidirectional RNN Modeling Layer

Our next improvement also followed from the original implementation of bidirectional attention [5], and consists of adding a bidirectional RNN modeling layer between the attention and output layers. The baseline's modeling layer was just a fully connected layer, so we saw a large performance jump (11.01% Dev EM, 12.69% Dev F1), but training also took about twice as long. We added a second RNN layer and saw performance increase another 1.84% for Dev EM, and 2.27% for Dev F1.

4.4 Conditioning End Prediction on Start Prediction

Our implementation of the conditioning we describe in Section 3.3 provided a small yet noticeable improvement in performance of 1.4% EM and 1.69% F1. We performed this test both early in our process and on our final model, and we saw slightly less significant results on our final model, possibly due to the additional expressibility of the modeling layer. This would justify further exploration that we were unfortunately unable to perform in this work.

4.5 Optimizers

Based on previous work we believed that using the AdaDelta optimizer may provide a useful boost. However, we tested many different learning rates for AdaDelta and found none of the subsequent models to perform on the same level as Adam with a learning rate of 0.001. Our hypothesis is that this is because many other hyper-parameters could affect the performance of the optimizer, including dropout. However, it is more likely that our hyper-parameter search was simply not fine-grained enough to find the optimal learning rate. We tested AdaDelta both early in our model development as well as with our final model, and in both cases saw significantly inferior performance.

4.6 ℓ_2 Regularization

We saw a lot of overfitting, so we looked into regularization and dropout (Section 4.7) to address the problem. ℓ_2 regularization (adding $\lambda||x||_2$ to the loss, x are the model parameters) did not have any improving effect. We tested with $\lambda \in 1 \times 10^{-3}, 1 \times 10^{-5}, 1 \times 10^{-8}$ and saw performance changes range from large drops to essentially inconsequential, as shown in Fig. 7.

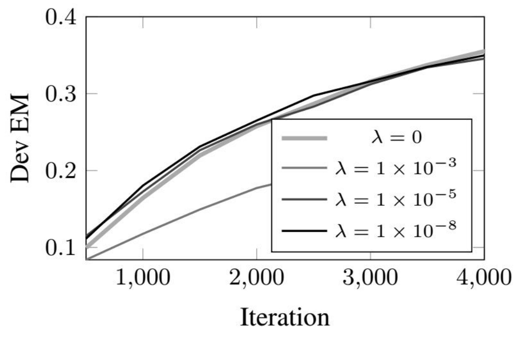

Figure 7: The effect of regularization on the model performance (F1 exhibits same trend).

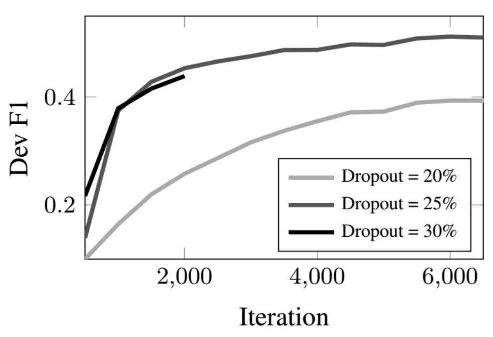

Figure 8: The effect of dropout on the model performance (F1 exhibits same trend).

4.7 Dropout

It seemed to depend greatly on the associated learning rate. We tested throughout the model improvement process, but the most interesting results are for our final model, where we performed a set of more fine-grained dropout search. Early on we were using a dropout of 0.15, but soon found that 0.2 performed slightly better. However, with the addition of the bidirectional modeling layer, we encountered more significant over-fitting, so we increased dropout to 0.25, where we saw a significant increase in performance, and proceeded to test other dropout levels, as depicted in Fig. 8. With our final model we performed more fine-grained testing with dropouts of 0.225 and 0.275, both of which did not provide significant changes to the model performance.

4.8 Learning Rate

We continuously checked the learning rate of the model as we observed it had significant impact on model performance. However, we performed a set of final tests on our final model, which are most important to present. We saw that 0.001 performed the best. Both increasing and decreasing the learning rate by an order of magnitude (0.01 and 0.0001) significantly decreased model performance, so we also tested with smaller deviations.

The result of this test was that even small deviations in the learning rate resulted in worse performance. A learning rate of 0.0012 decreased the F1 score very slightly (0.04), but the EM score decreased by slightly more (0.15). This could have been just a result of stochasticity of the model, but more extensive testing would be required to make any more significant conclusion. A learning rate of 0.0008 more noticeably decreased the performance, by 1.5 and 0.4 for F1 and EM, respectively.

5 Analysis

Analysis of the model performance is generally the most important aspect of working with neural networks, due to their notoriously poor interpretability. In this section we dive into examples of the model performance in an attempt to understand its behavior better. We examine the truth data for a random sample of answers in addition to the answers predicted by our model, focusing on the interesting cases. We find errors related to boundary positions, question misclassification, ambiguity, and others, in addition to noting some preprocessing errors we encountered.

5.1 Fine-grained Question Differentiation

We first highlight an example of the model performing well. Specifically, the model was able to correctly learn a rather specific question type, ones that start with "in what year." As an example, the questions "In what year was Nikola Tesla born?" and "When was Nikola Tesla born?" elicited different responses from the model, where the latter response included the month and date. This shows that the model is able to infer the type of answer desired from subtle changes to the question.

5.2 Data Errors

Much to our surprise, the first type of error we encountered was actually in the data. We examined the file *dev-v1.1.json*, and saw that answers were occasionally wrong. As one example, the question reads "Where was France's Huguenot population largely centered?" Two of the supplied answers are correct, but the third is entirely inaccurate, almost as if it was referring to the next question:

1. "the southern and central parts of France"
2. "southern and central parts of France,"
3. "about one-eighth"

However, when we examine the answer to the next question, all of the answers are correct, leading us to believe that this might actually be an error in the dataset. For reference, the question ID is 57105da9a58dae1900cd699e. Of course, it is extremely difficult to collect perfect data for complicated tasks like question answering, and SQuAD remains one of the best available.

5.3 Boundaries

The model's errors often result from incorrect labeling of the boundaries, which can be quantified by the large gap between EM and F1 scores. One example of this is the question "What does AC stand for?" In this case the model answers with "modern alternating current" even though the correct answer is simply "alternating current."

This error often presents itself in questions that need longer answers as well, which is intuitive due to the fact that it needs to be more accurate relative to the length of the answer. One example of this is the question "What are the two bodies that make up the European Union's legislature?", which elicits an answer so long we will omit it from here, but it is 52 words long even though the correct answer is simply "European Parliament and the Council of the European Union."

5.4 Question Misclassification

There is a subset of answers that are incorrect due to model misunderstanding between syntactically similar questions. The best example of this that we found is the question "What was the proportion of Huguenots to Catholics at their peak?" The correct answer is "about one-eighth," but the model predicted "two-million." Clearly, the model understood that the question wanted a quantitative answer, but it failed to distinguish between proportional and absolute quantities.

5.5 Ambiguity

Some questions present a great deal of ambiguity. For example, there is the case of question with ID 56e0b94b7aa994140058e6b9, "What is Tesla's home country?" First, the answer is "Serbian" even though Serbia is not a country, but beyond that we see that "Serbian" appears in multiple locations in the context (token 14 and 83). In fact, the true answers are split between these. It is unclear how the model would handle such a case of duplicate answers, but we believe it would not affect model accuracy because all correct answers would have high probability, but it would affect the score given because the true answer can only be one of these.

Another example of ambiguity is the question “What was the price of oil in March of 1974?” and the correct answer was “12$”, even though another acceptable answer in our opinion would have been “US $3 per barrel.” It is difficult to address such problems, but a sufficiently robust model would be able to learn that either answer was correct, and the problem is more about the data.

5.6 Miscellaneous

There are occasional questions which the model gets completely wrong and it is hard to understand why. One example of this is the question “Are atmosphere oxygen levels going up, down, or staying the same?” The model answers with “fossil-fuel burning” instead of “down,” which is surprising given that we often consider multiple-choice questions to be easier than open-ended questions. However, it is possible that the model interpreted the question more along the lines of a “why” question, which would make sense because the predicted answer is a cause.

6 Conclusion

We present a method of conditioning that uses the predicted probability distribution as a feature for the prediction of the end position of the answer, and we show that it provides an improvement of about 1.5% to a basic model. The first extension of this work is to compare our conditioning to other approaches. More future work includes creating a model ensemble, which many researchers find works well and adding the character-level embeddings using CNNs for another couple of points. We wanted to implement self-attention in this paper and compare different methods of combining it with BiDAF, but we leave it as future work. We hypothesize that a combination of the two modes of attention should lead to significant improvements.

Acknowledgments

We would like to thank Microsoft Corporation for providing credits to the Microsoft Azure cloud computing service to run our models. We also extend our thanks to the course staff for creating an informative, useful, and organized course, and for being extremely responsive, kind and helpful.

References

[1] Li Deng. “Deep learning: from speech recognition to language and multimodal processing”. In: *APSIPA Transactions on Signal and Information Processing* 5 (2016), e1.

[2] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. “Initializing Bayesian Hyperparameter Optimization via Meta-learning”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI’15. Austin, Texas: AAAI Press, 2015, pp. 1128–1135.

[3] H. N. Io and C. B. Lee. “Chatbots and conversational agents: A bibliometric analysis”. In: *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. 2017, pp. 215–219.

[4] Pranav Rajpurkar et al. “SQuAD: 100, 000+ Questions for Machine Comprehension of Text”. In: *CoRR* abs/1606.05250 (2016). arXiv: `1606.05250`.

[5] Min Joon Seo et al. “Bidirectional Attention Flow for Machine Comprehension”. In: *CoRR* abs/1611.01603 (2016). arXiv: `1611.01603`.

[6] Tao Shen et al. “Bi-Directional Block Self-Attention for Fast and Memory-Efficient Sequence Modeling”. In: *International Conference on Learning Representations*. 2018.

[7] Shuohang Wang and Jing Jiang. “Machine Comprehension Using Match-LSTM and Answer Pointer”. In: *CoRR* abs/1608.07905 (2016). arXiv: `1608.07905`.

[8] Zhiguo Wang, Wael Hamza, and Radu Florian. “Bilateral Multi-Perspective Matching for Natural Language Sentences”. In: *CoRR* abs/1702.03814 (2017). arXiv: `1702.03814`.

[9] Caiming Xiong, Victor Zhong, and Richard Socher. “Dynamic Coattention Networks For Question Answering”. In: *CoRR* abs/1611.01604 (2016). arXiv: `1611.01604`.