Improving BiDAF on SQuAD 2.0

Stanford CS224N Default Project

Zachary Chen
Department of Computer Science
Stanford University
zachchen@stanford.edu

Abstract

Question Answering (QA) is a task in which a model is given a paragraph as well as a question about that paragraph, and whose goal is to correctly answer the question using a chunk of text from the paragraph, or in some cases, identify that the question is unanswerable given the paragraph. In this project, I aim to improve upon a baseline question answering model based on Bi-Directional Attention Flow (BiDAF) [1] that does not include a character-level embedding layer, and that is evaluated on SQuAD 2.0 [2]. The first improvement I made was implementing character-level embeddings and concatenating them with the baseline word-level embeddings, similar to that in the original BiDAF paper. I then modified the existing BiDAF model's attention layer to include self attention, trying different hyperparameters along the way. As a final step, I evaluated these changes on SQuAD 2.0 compared to the baseline model, and found that while adding character embeddings improved my performance compared to the baseline, my method of self-attention was not as successful.

1 Key Information to include

- Mentor: Yian Zhang
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Question answering is an interesting task as it can provide a measure for how well systems can understand text. Furthermore, it is an incredibly applicable and practical task. For example, people often want to be able to get a direct answer to a question rather than just web pages or paragraphs regarding their question. In cases like smart speakers and virtual assistants, it is important to be able to answer a question from a user as it wouldn't make sense to read out paragraphs of information, and it is also essential to know when a question is unanswerable so the device/model doesn't give false information. The SQuAD 2.0 dataset improves on the original SQuAD dataset and challenges models that had performed well on the original dataset by including questions that are unanswerable.

As our baseline only utilizes 300 dimensional GloVe word-embeddings and not character-embeddings, this means that the model may miss out on information that can be extracted inside individual words. By adding character-level embeddings, the model will be able to condition on the internal structure of words. Furthremore, without character-level embeddings, the baseline model may also have trouble with words that are out of the vocabulary, which is another key idea behind why I added character-embeddings fed through a Convolutional Neural Network (CNN) to the word-embeddings.

In addition, the baseline attention flow layer only relates the question (referred to as query) words to the passage (referred to as context) words and the context words to the query words, but not the context words to themselves, i.e., self-attention. That is, it calculates which question words are most relevant to each context word and which context words are relevant to some question words, but not which other context words are relevant to each context word. As it may be helpful to know which context words relate to each other, I added context to context self-attention to the existing bidirectional attention flow layer in the baseline, and found improvements in performance on the SQuAD 2.0 dataset using the added character embeddings and self-attention.

3 Related Work

One of the most influential datasets for question answering is the original Stanford Question Answering Dataset (SQuAD) [3]. In this dataset, there are over 100,000+ question and paragraph pairs of which the paragraphs were taken from Wikipedia and the questions and answers were crowdsourced using Amazon Mechanical Turk. This dataset was improved upon to create SQuAD 2.0 [2], which was created at the time to address the weakness of some datasets focusing only on answerable questions or only having easily identifiable unanswerable questions. It was shown that at the time, a state-of-the-art model performing close to humans on SQuAD 1.1 performed drastically worse on SQuAD 2.0, showing the need for improvement on identifying unanswerable questions.

The baseline in this paper is heavily based off the BiDAF [1] model developed by Seo et al in 2018, and evaluated on SQuAD 2.0. In this paper, the authors created a multi-layered model that includes both word and character-level embeddings, and more importantly a bidirectional attention flow layer (hence the name) that links and fuses information between the context in query words, computing attention both from query to context and context to query. While attention was already a previously popular mechanism, the attention flow layer in this paper doesn't summarize query and context into single feature vectors but rather allows the attention vectors to flow through, reducing information loss. The character-level embedding layer I implemented is based on the one implemented in this paper, and the attention flow layer in the baseline I modified is the same attention flow layer described in this paper.

Another paper that strongly relates and inspires the self-attention implemented in this paper is Microsoft's paper introducing R-NET [4], in which the authors implemented gated attention-based recurrent networks to obtain question-aware passage representations, pointer networks to locate the positions of answers from the given context, and most importantly a self-matching attention mechanism that attends the context with itself in addition to with the query and inspired the self-attention I implemented. At the time of publication, R-NET achieved best results on both the Microsoft MAchine Reading COmprehension (MS-MARCO) dataset and the original SQuAD.

The Dynamic Coattention Network [5] is another high performing SQuAD model that similar to BiDAF, has a two-way attention. However, it includes a second-level attention computation that attends to the attention outputs of the first level. This is similar to my implementation of self-attention, as I am also concatenating addition attention to the existing context to question and question to context attention, except instead of concatenating second-level attention outputs based on the first-level attention outputs, I am concatenating context to context attention outputs.

Finally, another model worth mentioning that uses a different approach to that in this paper and also the other papers mentioned already is QANet [6], which adapts ideas from Transformers and replaces RNNs entirely with self-attention and convolution. Using an Encoder Block consisting of positional encoding, residual connections, layer normalization, self-attention sublayers, and feed-forward sublayers, QANet was able to achieve very high performance on SQuAD 1.1 prior to BERT.

4 Approach

4.1 Baseline

My baseline model was based on BiDAF [1], but with only the 300 dimensional GloVe word-level embeddings and without a character-level embedding layer. In the baseline, we start in the embeddings layer, where we are first given some word indices for both the context and query, then use an embedding lookup to convert the indices to word embeddings. We then project each embedding to have dimensionality H, the hidden size of the model, and apply a Highway Network to refine the representations. Then, in the encoder layer we use the embeddings as input and use a bidirectional LSTM to encode them, and pass the outputs into the attention layer.

In the attention layer, which will be relevant later in the paper, for each pair (c_i, q_j) of context and question hidden states, we compute similarity score $S_{ij} = w_{sim}^T[c_i; q_j; c_i \circ q_j]$, where $\circ$ represents an element-wise product and w_{sim}^T is a learnable parameter. Then, to get context to question attention outputs a_i and question to context attention outputs b_i, we can take the softmax of S and take the weighted sums with the corresponding hidden states, as follows:

$$\bar{\boldsymbol{S}}_{i,:} = \text{softmax}(\boldsymbol{S}_{i,:}) \in \mathbb{R}^M \quad \forall i \in \{1, \dots, N\} \qquad \bar{\bar{\boldsymbol{S}}}_{:,j} = \text{softmax}(\boldsymbol{S}_{:,j}) \in \mathbb{R}^N \quad \forall j \in \{1, \dots, M\}$$

$$\boldsymbol{S'} = \bar{\boldsymbol{S}}\bar{\bar{\boldsymbol{S}}}^T \in \mathbb{R}^{N \times N}$$

$$\boldsymbol{a}_i = \sum_{j=1}^{M} \bar{\boldsymbol{S}}_{i,j}\boldsymbol{q}_j \in \mathbb{R}^{2H} \quad \forall i \in \{1, \dots, N\}. \qquad \boldsymbol{b}_i = \sum_{j=1}^{N} \boldsymbol{S'}_{i,j}\boldsymbol{c}_j \in \mathbb{R}^{2H} \quad \forall i \in \{1, \dots, N\}.$$

Then, we can concatenate the results and combine the hidden state c_i to get the outputs $g_i = [c_i; a_i; c_i \circ a_i; c_i \circ b_i]$.

In the modeling layer, we use another bidirectional LSTM with two layers this time, taking in as input the outputs g_i from the previous layer and computing:

$$\boldsymbol{m}_{i,\text{fwd}} = \text{LSTM}(\boldsymbol{m}_{i-1,\text{fwd}}, \boldsymbol{g}_i) \in \mathbb{R}^H$$
$$\boldsymbol{m}_{i,\text{rev}} = \text{LSTM}(\boldsymbol{m}_{i+1,\text{rev}}, \boldsymbol{g}_i) \in \mathbb{R}^H$$
$$\boldsymbol{m}_i = [\boldsymbol{m}_{i,\text{fwd}}; \boldsymbol{m}_{i,\text{rev}}] \in \mathbb{R}^{2H}.$$

Finally, in the output layer we use the previously calculated m_i to produce a vector m_i' for each m_i given by:

$$\boldsymbol{m'}_{i,\text{fwd}} = \text{LSTM}(\boldsymbol{m'}_{i-1,\text{fwd}}, \boldsymbol{m}_i) \in \mathbb{R}^H$$
$$\boldsymbol{m'}_{i,\text{rev}} = \text{LSTM}(\boldsymbol{m'}_{i+1,\text{rev}}, \boldsymbol{m}_i) \in \mathbb{R}^H$$
$$\boldsymbol{m'}_i = [\boldsymbol{m'}_{i,\text{fwd}}; \boldsymbol{m'}_{i,\text{rev}}] \in \mathbb{R}^{2H}.$$

and letting G, M, and $M\prime$, be matrices with columns corresponding to $g_i, ..., g_N$, $m_i, ..., m_N$, and $m_i\prime, ..., m_N\prime$ respectively, we can take the softmax to get the start and end probabilities as follows:

$$\boldsymbol{p}_{\text{start}} = \text{softmax}(\boldsymbol{W}_{\text{start}}[\boldsymbol{G}; \boldsymbol{M}]) \qquad \boldsymbol{p}_{\text{end}} = \text{softmax}(\boldsymbol{W}_{\text{end}}[\boldsymbol{G}; \boldsymbol{M'}]),$$

where W_{start} and W_{end} are learnable parameters.

Our training loss is then just $-\log p_{start}(i) - \log p_{end}(j)$ where i and j are the gold start and end locations, and out prediction is the pair (i, j) that maximizes $p_{start}(i) \cdot p_{end}(j)$. We prepend an Out Of Vocabulary token to each context so that if $p_{start}(0) \cdot p_{end}(0)$ is the maximum we predict no-answer. For more detailed explanation about the layers, please refer to the original paper [1]. (Note that the embedding layer in the baseline doesn't have character-level embeddings.)

4.2 Character-Level Embeddings

My first main approach was to implement character-level embeddings so the model would be able to condition on the internal structure of words and better handle out-of-vocabulary words. This approach followed that from the original BiDAF paper [1]. Following the paper, I first took in as input character indices for the given context and queries, then similar to the word embeddings I used an embedding lookup to convert the indices to character embeddings. Then, I applied dropout, and passed the embeddings as input into a CNN implemented through PyTorch's Conv2d, tried both applying and not applying ReLU to the output of the CNN, then max pooled the outputs across the word length dimension to get a final character embedding for each word. Then, I concatenated the resulting character embeddings to the word embeddings from the baseline, and fed that as input into the encoder layer. Note that since I concatenated the character embeddings to the word embeddings instead of replacing them, the hidden size doubled.

4.3 Context To Context Self-Attention

My second approach was to modify the attention flow layer of the baseline model so that it also includes context to context attention. To do this, I followed a similar approach used in the baseline to calculate the context to query attention, except I essentially replaced the query with the context. That is, I computed the similarity score $S'_{ij} = w_{sim}^{T\prime}[c_i; c_i \circ c_j]$, where $\circ$ represents an element-wise product and $w_{sim}^{T\prime}$ is a learnable parameter, and added a learnable bias to the similarity score. Note that the context to query similarity score includes q_i, but I only have one c_i as it is redundant to have two identical c_i. To confirm this, I trained and evaluated my model using $c_i \circ c_j$ in my S'_{ij} calculation, using $[c_i; c_i \circ c_j]$, and using $[c_i; c_j; c_i \circ c_j]$ (where c_i and c_j have different dropouts and their own learnable weights). Then, I took the softmax along the j dimension similarly to the context to question approach (applying a mask to account for padding characters), and took the weighted sums with the corresponding context hidden states, giving me the following equations:

$$\bar{\mathbf{S}}'_{i,:} = \texttt{softmax}(\mathbf{S}'_{i,:}) \in \mathbb{R}^M, \forall i \in \{1, ..., N\}$$

$$\mathbf{d_i} = \sum_{j=1}^{M} \bar{\mathbf{S}}'_{i,j} \mathbf{c_j} \in \mathbb{R}^{2H}, \forall i \in \{1, ..., N\}$$

Then, I concatenate d_i and $c_i \circ d_i$ onto the original g_i to get a new $g_i = [c_i; a_i; c_i \circ a_i; c_i \circ b_i; d_i; c_i \circ d_i] \in \mathbb{R}^{12H}$ which will be the new input into the modeling layer and also for the output layer.

5 Experiments

5.1 Data

I evaluated my models on SQuAD 2.0 [2], which provides about 150,000 question-paragraph (query-context) pairs in total, of which roughly half cannot be answered using the provided paragraph. The paragraphs are from Wikipedia and the questions and answers were crowdsourced using Amazon Mechanical Turk. If a question is answerable, the answer will be taken directly from a chunk of the corresponding paragraph, meaning the models don't have to generate text but rather select the start and end position of the answer, or output that there is no answer.

5.2 Evaluation method

The main metrics that the models were evaluated on were the Exact Match (EM) metric, which measures whether a given output matches the ground truth answer exactly, and the F1 metric, which is the harmonic mean of the precision and recall of the model answer compared to the ground truth answer. There are three human provided answers for each question, and the maximum EM and F1 scores across the three answers for each question is used, making evaluation more forgiving. These scores are then averaged across the entire dataset to get the final scores. The splits for the evaluation are as follows:

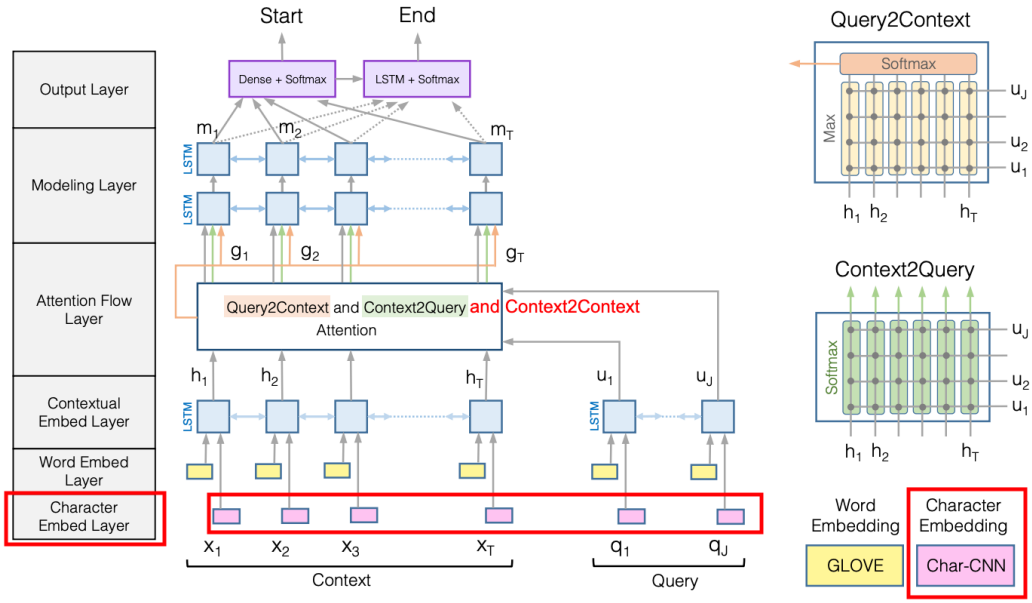


Figure 1: BiDAF Baseline Model [1] with my changes in red

- `train` (129,941 examples): All taken from the official SQuAD 2.0 training set.
- `dev` (6078 examples): Roughly half of the official dev set, randomly selected.
- `test` (5915 examples): The remaining examples from the official dev set, plus hand-labeled examples.

I used only the train and dev set to train, tune, and evaluate the models, and submitted my best models to be evaluated on the test set.

5.3 Experimental details

Models:

1. Baseline
2. Word/Char Embed no ReLU
3. Word/Char Embed w/ ReLU
4. Word/Char Embed w/ ReLU Dropout=0.15
5. Word/Char Embed w/ ReLU Self Attention $[c_i \circ c_j]$
6. Word/Char Embed w/ ReLU Self Attention $[c_i; c_i \circ c_j]$
7. Word/Char Embed w/ ReLU Self Attention $[c_i; c_j; c_i \circ c_j]$

Unless further specified in the model, for all models I used the following configurations:

- Word Embed Dim = 300
- Char Embed Dim = 64
- Learning Rate = 0.5
- Dropout Prob = 0.2
- Epochs = 30
- Eval Steps = 50000

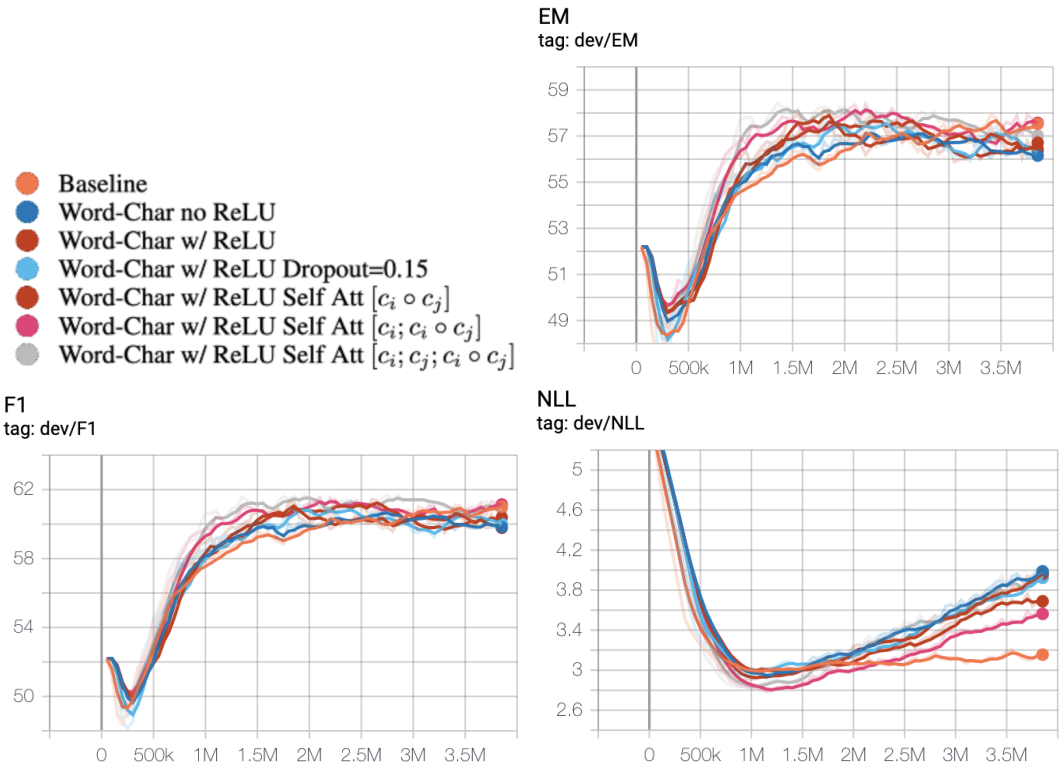
- Ema Decay = 0.999
- Hidden Size = 100
- Batch Size = 64

The training time for all models was between 4-5 hours, except for the baseline which took about 3 hours.

5.4 Results

Model	Dev EM	Dev F1	Test EM	Test F1
Baseline	57.755	61.186		
Word-Char no ReLU	57.822	61.428		
Word-Char w/ ReLU	**58.343**	**61.560**	**57.194**	**60.789**
Word-Char w/ ReLU Dropout=0.15	58.041	61.675		
Word-Char w/ ReLU Self Att $[c_i \circ c_j]$	57.453	61.018		
Word-Char w/ ReLU Self Att $[c_i; c_i \circ c_j]$	58.041	61.336	57.041	60.507
Word-Char w/ ReLU Self Att $[c_i; c_j; c_i \circ c_j]$	57.637	61.123		

Table 1: Model Performances



Overall, I found that the word-character embeddings with ReLU model narrowly performed the best of both the Dev and Test set. When compared to the word-character embeddings without ReLU model, it makes sense that it performs better, as in general adding an activation function after a CNN is beneficial. While I expected improvement over the baseline using the word-character embeddings, the results using self-attention are not as high as I had hoped. Although using both the word-character embeddings and the self-attention outperformed the baseline, it underperformed when compared to not using self-attention at all. As self-attention has been shown to be beneficial in question answering, this likely points to my implementation of self-attention being suboptimal. Since there are various ways to implement self-attention, the results tell me that I should perhaps consider a different approach to implementing self-attention, such as through a separate layer, or at least further look into my current implementation and try to identify areas that could be preventing improvement.

6 Analysis

Below are some examples where my best model (word-char embeddings w/ ReLU) is able to correctly identify the answer, and some where it fails.

• **Question:** What is the rarest cause of poor immune function in developing countries?
• **Context:** Immunodeficiencies occur when one or more of the components of the immune system are inactive. The ability of the immune system to respond to pathogens is diminished in both the young and the elderly, with immune responses beginning to decline at around 50 years of age due to immunosenescence. In developed countries, obesity, alcoholism, and drug use are common causes of poor immune function. However, malnutrition is the most common cause of immunodeficiency in developing countries. Diets lacking sufficient protein are associated with impaired cell-mediated immunity, complement activity, phagocyte function, IgA antibody concentrations, and cytokine production. Additionally, the loss of the thymus at an early age through genetic mutation or surgical removal results in severe immunodeficiency and a high susceptibility to infection.
• **Answer:** N/A
• **Prediction:** N/A
• **Analysis:** In this example, the model correctly predicts that there is no answer, whereas the baseline model incorrectly predicts obesity, alcoholism, and drug use. In the baseline model, it may see words like cause and countries and think that there is an answer, but one reason why the word-character model may be able to get this question correct could be because it is able to realize that the "ed" in the word "developed" from the context and the "ing" in the word "developing" from the query make a big difference to the answer of the question.

• **Question:** In what area of this British colony were Huguenot land grants?
• **Context:** In 1700 several hundred French Huguenots migrated from England to the colony of Virginia, where the English Crown had promised them land grants in Lower Norfolk County. When they arrived, colonial authorities offered them instead land 20 miles above the falls of the James River, at the abandoned Monacan village known as Manakin Town, now in Powhatan County. Some settlers landed in present-day Chesterfield County. On 12 May 1705, the Virginia General Assembly passed an act to naturalise the 148 Huguenots still resident at Manakintown. Of the original 390 settlers in the isolated settlement, many had died; others lived outside town on farms in the English style; and others moved to different areas. Gradually they intermarried with their English neighbors. Through the 18th and 19th centuries, descendants of the French migrated west into the Piedmont, and across the Appalachian Mountains into the West of what became Kentucky, Tennessee, Missouri, and other states. In the Manakintown area, the Huguenot Memorial Bridge across the James River and Huguenot Road were named in their honor, as were many local features, including several schools, including Huguenot High School.
• **Answer:** Lower Norfolk County
• **Prediction:** Lower Norfolk County
• **Analysis:** In this example, the baseline incorrectly answers Manakintown but our model is able to correctly answer Lower Norfolk County. In this case, our model is able to tell that the land grants were in Lower Norfolk County, despite Manakintown coming up as a much more frequent location, and is also able to perhaps realize that English Crown and British colony are similar, showing an ability to consider synonyms. Furthermore, "land grants" only comes up once in the context, and the model is able to realize that it is important to answering the question.

• **Question:** How did old oil affect motorists in many countries?
• **Context:** Price controls exacerbated the crisis in the US. The system limited the price of "old oil" (that which had already been discovered) while allowing newly discovered oil to be sold at a higher price to encourage investment. Predictably, old oil was withdrawn from the market, creating greater scarcity. The rule also discouraged development of alternative energies. The rule had been intended to promote oil exploration. Scarcity was addressed by rationing (as in many countries). Motorists faced long lines at gas stations beginning in summer 1972 and increasing by summer 1973.
• **Answer:** N/A
• **Prediction:** withdrawn from the market
• **Analysis:** In this example the model incorrectly identifies "withdrawn from the market" as the

correct answer when in fact there is no correct answer. It seems that the model is unable to identify that the question is asking more about the affect on motorists as opposed to what happened to old oil. Notice that the model didn't pick "motorists faced long lines" either, so it could be focusing more on "old oil".

• **Question:** What town in upstate New York was settled by Huguenots?
• **Context:** Huguenot immigrants did not disperse or settle in different parts of the country, but rather, formed three societies or congregations; one in the city of New York, another 21 miles north of New York in a town which they named New Rochelle, and a third further upstate in New Paltz. The "Huguenot Street Historic District" in New Paltz has been designated a National Historic Landmark site and contains the oldest street in the United States of America. A small group of Huguenots also settled on the south shore of Staten Island along the New York Harbor, for which the current neighborhood of Huguenot was named.
• **Answer:** New Paltz
• **Prediction:** New Rochelle
• **Analysis:** In this example the model isn't able to distinguish that the upstate area is New Paltz and not New Rochelle. This shows an inability of the model to recognize "upstate" as being a key part of the question, and one reason why the model may have failed to get the right answer in this case could be because "town" is closer to New Rochelle and the question is asking "What town".

7 Conclusion

In this project, I explored integrating character embeddings and context to context self attention on a baseline BiDAF model, then compared my results on SQuAD 2.0 to the baseline model. I found that while adding character embeddings gave better results than the baseline, my method of implementing self attention was not as successful as I had hoped. Given the time restraints of the project and time required to train the models, I was also not able to explore as much finetining as I had hoped, largely because the majority of the time went into trying to integrate self-attention. In the future, I hope to further explore self-attention and other methods of implementing self-attention, as well as potential other areas of improvement, such as changing the output layer to one similar to R-NET, or implementing coattention.

References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bi-directional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*, 2017.

[2] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[4] Natural Language Computing Group and Microsoft Research Asia. R-net: Machine reading comprehension with self-matching networks. `https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf`. 2017.

[5] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. In *International Conference on Learning Representations (ICLR)*, 2017.

[6] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations (ICLR)*, 2018.