Gated Self-Attention for SQuAD Question Answering

Stanford CS224N Default Project

Danny Tse
Department of Computer Science
Stanford University
dannytse@stanford.edu
Mentor: Yuyan Wang

Abstract

Machine comprehension and question answering are central questions in natural language processing, as they require modeling interactions between the passage and the question. In this paper, we build on the multi-stage hierarchical process BiDAF described in Seo et al. (2017)'s *Bi-Directional Attention Flow for Machine Comprehension*. We utilize tools from the R-Net model described in *R-Net: Machine Reading Comprehension with Self-Matching Networks*, testing different combinations of model components. We experiment with different types of encoding, such as using a Gated Recurrent Unit (GRU) or a Convolutional Neural Network (CNN), and attention mechanisms, such as comparing context-query attention layers and contemplating the usage of gates. We ultimately introduce a modified form of BiDAF which utilizes both an LSTM and a CNN in its encoding layer, as well as BiDAF's context-query attention layer followed by R-Net's self-attention layer. We conduct various experiments on the SQuAD datasets, yielding competitive results on the CS224N SQuAD Leaderboard.

1 Introduction

A common misconception regarding language is that it has to do with people and words, not the concepts they represent. In fact, one can realize that language is in fact merely a social construct; the sequential representation of glyphs mean nothing to those who don't know the language! Thus, an interesting problem that arises is: how do we get machines to understand our language? More specifically, given a short passage of text, can a machine "read" this text and answer questions on it? This research question is particularly difficult as (1) there are many nuances to language in which a computer has to learn and (2) such training data is scarce, as though texts are abundant, it is difficult to come across texts along with question-answer pairs.

In 2017, Seo et al. published *Bi-Directional Attention Flow for Machine Comprehension* [1], introducing a multi-stage hierarchical network (BiDAF) that represents the context at different levels, utilizing bidirectional attention to obtain a question-aware passage representation without early summarization. At time of publishing, this model achieved state-of-the-art results. Also in 2017, Microsoft Asia published *R-Net: Machine Reading Comprehension with Self-Matching Networks* [2], introducing an end-to-end neural network model utilizing gated attention-based recurrent networks, self-matching attention, and a pointer-network output layer. The R-Net paper primarily focuses on two datasets: the Stanford Question Answering Dataset (SQuAD) and the Microsoft Machine Reading Comprehension (MS-MARCO) datasets. At time of publishing, the model outperformed strong baselines, beating the BiDAF model and taking over first place on the Stanford Question and Answering Dataset (SQuAD) leaderboard and claiming the best published results on the MS-MARCO dataset.

In this paper, we wish to develop a similar neural network with reading comprehension skills - in other words, given a passage or document, the model could answer questions related to the text that require logical reasoning. To do so, we use methods and model architecture from the two papers

discussed above. As both models (BiDAF and R-Net) have similar architecture in that it consists of some embedding layer and some context-question embedding layer, we combine the techniques discussed in the R-Net paper to improve upon the BiDAF model. Precisely, we use Seo et al.'s BiDAF model as our baseline, as well as expanding on it, applying a gated self-attention layer on top of the context-query matching in BiDAF. Ultimately, our model (Figure 1) consists of six layers: 1) an embedding layer (obtaining word and character embeddings) 2) an encoding layer for the embeddings, with a recurrent network and a convolutional neural network (for character embeddings only), applied separately to the passage and query, 3) a contextual embedding layer, which uses context clues from surrounding words to refine the embedding of the words, 4) a gated self-matching attention mechanism, aggregating evidence from the passage to infer the answer, 5) a modeling layer applied on the output of the gated self-matching layer, which utilizes a Recurrent Neural Network to scan the context, and 6) the output layer, which provides an answer to the query.

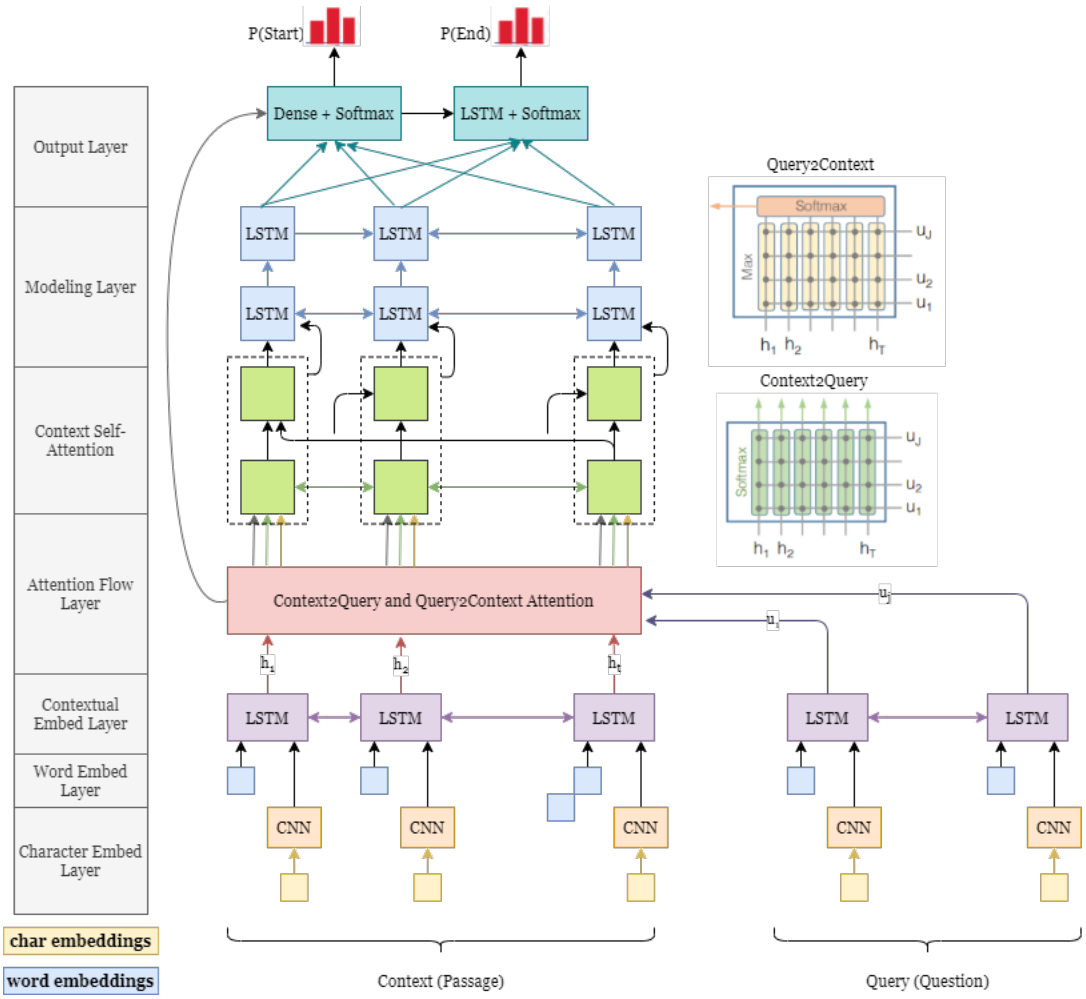

Figure 1: Model Overview. Adapted from *Bi-Directional Attention Flow for Machine Comprehension* and *R-Net: Machine Reading Comprehension with Self-Matching Networks*.

Initially, we had decided to re-implement the entire R-Net model, but the full implementation required too much training time and device memory. As a result, we had to simplify both attention layers and the output layer, resulting in much weaker model performance. (Will be discussed further in Section 4.) Unfortunately, with extensive changes to the model, the pointer network appeared to no longer be a compatible output layer for the model – we observed the loss stagnating, with performance below the BiDAF baseline. Thus, we began experimenting with different model combinations using our baseline, observing that the model architecture was similar - the BiDAF model had an extra modeling layer, while the R-Net model had an extra self-attention layer. We made sure to utilize what

made R-Net strong - the usage of a recurrent network in the embedding layer, along with the gated self-attention mechanism, and incorporated these elements into the given BiDAF model.

2 Related Work

Since the release of the SQuAD dataset, much progress has been made and we have seen the rise of several interesting NLP models. For example, character-level embeddings with Convolutional Neural Networks (CNN) were introduced in Kim (2014) [3], where we embed characters into vectors, which are then fed in as 1D inputs to the CNN, and then max-pooled over the entire width to obtain a fixed-size vector for each word. In fact, we use this method in our model. Some other related work that utilizes CNNs is in visual question answering. Some early works, such as Antol et al. [4], encode images using a CNN, then combine them to answer questions. Lu et al. [5] builds on this field, showing that attending from the image back to the question words improves the VQA task results. This is similar to work done in language and to a lesser extent, this paper, as we implement bi-directional attention between the query and context, improving our performance on question answering. Some work that R-Net builds on and improves on include Wang and Jiang (2016) [6], where they propose matching the context with the query from multiple perspectives, and Wang and Jiang (2016) [7], where match-LSTMs and answer pointers are utilized in unison to effectively find the answer in the context. Furthermore, when compared to previous attention-based recurrent network models like Bahdanau et al. (2014) [8], Rocktaschel et al. (2015) [9], the R-Net model utilizes a self-matching attention mechanism, where the model aggregates information from the context to infer the answer. This self-matching attention mechanism is implemented by using gated attention-based recurrent networks on the context against itself to emphasize question-relevant evidence.

3 Approach

To begin, we formally define the problem: given a context or passage and a query or question from SQuAD, we want our model to return an answer within the span of the passage/context, or indicate that no such answer exists.

The baseline we use is a Bi-Directional Attention Flow (BiDAF) model on the word embeddings. This model is described in the Default Final Project Handout (SQuAD Track), and consists of a Word Embedding Layer, an Encoder Layer, a Context-Query Attention Layer, a Modeling Layer, and an Output Layer. Our main adjustment to this model is a change in the encoding layer (adding a), as well as the addition of a self-attention layer (see Figure 1). We suspected that, since the self-attention mechanism allows hidden states to consider previous hidden states, this model can record long-distance dependencies, and as a result have more complete answers to questions. Variants of self-attention have proven to be vastly successful in modern natural language processing models (ex. usage of multi-headed self-attention in a Transformer Encoder-Decoder block introduced in Vaswani et al. (2017)'s *Attention Is All You Need* [10]), thus in theory, this model should outperform our Bidirectional Attention Flow baseline.

We describe the model architecture below. First, we obtain the query (question) and context (passage)'s word embeddings and then process their character embeddings through a convolutional neural network, followed by an encoding layer containing a bi-directional LSTM. Then, we apply a contextual embedding layer (as described in BiDAF), followed by R-Net's self-matching attention. This provides us with a question-aware representation for the passage. Then, we refine the passage representation with self-matching attention and feed it into the output layer to return the boundary of the answer span. The model architecture is:

1. **Word and Character Embedding** Our character embedding layer maps each word to a high-dimensional vector space. Let $C = \{w_t^C\}_{t=1}^N$ and $Q = \{w_t^Q\}_{t=1}^M$ represent the context paragraph and query respectively. As described in Kim (2014), we obtain the character-level embeddings of the words, which are then passed into a Convolutional Neural Network (CNN) as 1D inputs. Then, we max-pool over the entire width to obtain a fixed-sized vector c_t for each word. Furthermore, this layer obtains the word embeddings of C and Q e_t respectively, and returns the concatenation of these word embeddings and the fixed-size character embedding vectors obtained from the max-pool, or $[e_t^C, c_t^C]$ and $[e_t^Q, c_t^Q]$, for context and query respectively.

2. **Question and Passage Encoder** Passing in the concatenated word-character embeddings from the first level, we first pass in $[e_t^C, c_t^C]$ and $[e_t^Q, c_t^Q]$ into a projection layer, projecting each embedding to have dimensionality H. In other words,

$$h_i^C = W_{\text{proj}}[e_t^C, c_t^C] \in \mathbb{R}^H$$
$$h_i^Q = W_{\text{proj}}[e_t^Q, c_t^Q] \in \mathbb{R}^H$$

Then, we apply a Highway Network to both h_i^C and h_i^Q, as described in Srivastava (2015) [11], which given an input vector h_i, it computes

$$g = \sigma(W_g h_i + b_g)$$
$$t = \text{ReLU}(W_t h_i + b_i)$$
$$h_i' = g \odot t + (1 - g) \odot h_i,$$

where W_g, W_t, b_g, b_t are all parameters. We apply the above transformation twice to each of h_i^C and h_i^Q, each time with different learnable parameters. Finally, we apply a bidirectional one-layer LSTM to encode the representations for words in the question and the passage. That is,

$$c_i = \text{LSTM}_C(c_{i-1}^P, h_i'^C)$$
$$q_i = \text{LSTM}_Q(q_{i-1}^Q, h_i'^Q)$$

Note that our baseline embedding and encoding layer is the exact same as described above, except the baseline only considers word-embeddings, and does not use a GRU layer.

3. **BiDAF Attention Layer (Context2Question, Question2Context)** The BiDAF Attention Layer lies in the heart of the model - the main idea is that attention should flow both ways, from context to question and vice-versa. This "attention flow" is responsible for connecting information from context words with information from query words. Furthermore, this attention flow does not simply summarize the query and context into a single feature vector, but rather allows information to "flow" through the subsequent layers, mitigating information loss. Suppose H is the hidden size of the GRU from the previous layer. Then, we have context hidden states $c_1, \ldots, c_N \in \mathbb{R}^{2H}$ and question hidden states $q_1, \ldots, q_M \in \mathbb{R}^{2H}$. First, we compute a similarity matrix $S \in \mathbb{R}^{N \times M}$, which is defined as follows:

$$S_{ij} = w_{\text{sim}}^T[c_i,\ q_j,\ c_i \circ q_j]$$

Note that a similarity score S_{ij} is defined for each pair (c_i, q_j) of context and query hidden states. A special feature about S is that it stores information about both the query and the context; as a result, we can use S in both our Context-to-Question and Question-to-Context Attention.

Then, we take Context-to-Question Attention (C2Q). First, we take the softmax of S row-wise to obtain the attention distributions S' over question hidden states. We can then use S' to take weighted sums of the question hidden states q_j, obtaining our C2Q attention outputs a_i. In other words,

$$S'_{i,:} = \text{softmax}(S_{i,:})$$
$$a_i = \textstyle\sum_{j=1}^M S'_{i,j} q_j.$$

Then, we take Question-to-Context Attention (Q2C). This time, we take the softmax of S column-wise to obtain the attention distribution $\bar{S}$ over context hidden states. Then, we obtain b_i, the Q2C attention output, by multiplying S' with $\bar{S}$, and then using the resulting matrix to obtain a weighted sum over the context hidden states:

$$\bar{S}_{:,j} = \text{softmax}(S_{:,j})$$
$$\mathbf{S} = S'\bar{S}$$
$$b_i = \textstyle\sum_{j=1}^M \mathbf{S}_{i,j} c_j.$$

Then, for each $i \in \{1, \ldots, N\}$, we can obtain the output g_i of this layer by combining all the information computed above:

$$v_i = [c_i,\ a_i,\ c_i \circ a_i,\ c_i \circ b_i] \in \mathbb{R}^{8H} \ \ \forall i \in \{1, \ldots, N\}.$$

4. **Self-Matching Attention** Now, we are given some query-aware context representation from previous layers, or $\{v_i\}_{i=1}^N$. To provide extra information of the passage/context, we apply another layer of attention, matching $\{v_i\}_{i=1}^N$ against itself, obtaining h_i which encodes evidence from words in the passage and their matching question information:

$$h_i = \text{GRU}(h_{i-1}, [v_i, c_i]^*)$$

where c_i is given by the attention-pooling vector of the whole passage representation v:

$$s_j^t = u^T \tanh(W_v^P v_j + W_u^{\bar{P}} v_i)$$
$$a_k^t = \exp(s_k^t) / \sum_{j=1}^n \exp(s_j^t)$$
$$c_t = \sum_{k=1}^N a_k^t v_k$$

Note that the GRU is bidirectional in this layer. Also, a gate is applied to $[v_i, c_i]$ to obtain $[v_i, c_i]^*$:

$$g_i = \text{sigmoid}(W_g[v_i, c_i])$$
$$[v_i, c_i]^* = g_i \odot [v_i, c_i]$$

This additional gate allows the neural network to adaptively control the input to the RNN. Furthermore, in our specific model, we adjust the output size of this GRU so that it is equal to that of the input size. That is, the shape of h_i should be equal to the shape of v_i from the BiDAF Attention Layer.

5. **Modeling Layer** Using the modeling layer, we wish to refine the vectors after the self-attention layer. At this point, our vectors are conditioned on both the query and the passage. Given vectors h_i, we employ a bidirectional LSTM:

$$m_{i,\text{fwd}} = \text{LSTM}(m_{i-1}, h_i)$$
$$m_{i,\text{rev}} = \text{LSTM}(m_{i+1}, h_i)$$
$$m_i = [m_{i,\text{fwd}}, m_{i,\text{rev}}]$$

6. **Output Layer** Finally, with m_i, the output of the modeling layer, and v_i, the output of the BiDAF-attention layer, we predict a vector of probabilities with shape [batch size, number of words in context]. First, the output layer takes m_i and applies a bidirectional LSTM to it, generating a vector m_i' for each m_i:

$$m'_{i,\text{fwd}} = \text{LSTM}(m'_{i-1}, m_i)$$
$$m'_{i,\text{rev}} = \text{LSTM}(m'_{i+1}, m_i)$$
$$m'_i = [m'_{i,\text{fwd}}, m'_{i,\text{rev}}]$$

Then, let V be the matrix with v_i as the i-th column, M be the matrix with m_i as the i-th column, and M' be the matrix with m_i' as the i-th column. Then, we calculate p_{start} and p_{end} as

$$p_{\text{start}} = \log \text{softmax}(W_{\text{start}}[G, M])$$
$$p_{\text{end}} = \log \text{softmax}(W_{\text{end}}[G, M']),$$

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

We apply dropout at the end of each layer. Furthermore, we train this model using the Adadelta optimizer to minimize the negative log-likelihood loss for the start and end positions. For example, if the start and end locations are i, j respectively, then the loss is simply

$$\text{loss} = -\log p_{\text{start}}[i] - \log p_{\text{end}}[j].$$

4 Experiments

4.1 Data

As data, we use the Stanford Question Answering Dataset 2.0 (SQuAD 2.0), which has the splits train (129,941 examples), dev (6078 examples), and split (5915 examples). In this dataset, passages are selected from the English Wikipedia (usually 100-150 words), questions are crowd-sourced, and each

answer is either contained within the passage, or does not exist at all (not answerable). Also, each answerable SQuAD question has three answers, each from a different crowd worker. An example question-answer pair from SQuAD is as follows:

Passage: Tesla later approached Morgan to ask for more funds to build a more powerful transmitter. **When asked where all the money had gone, Tesla responded by saying that he was affected by the Panic of 1901, which he (Morgan) had caused**. Morgan was shocked by the reminder of his part in the stock market crash and by Tesla's breach of contract by asking for more funds. Tesla wrote another plea to Morgan, but it was also fruitless. Morgan still owed Tesla money on the original agreement, and Tesla had been facing foreclosure even before construction of the tower began.

Question: On what did Tesla blame for the loss of the initial money?

Answer: Panic of 1901.

4.2 Evaluation method

To evaluate model performance, we use two well-defined, numerical, automatic evaluation metrics: Exact Match (EM) and F1 score. The Exact Match (EM) captures the percentage of the prediction that completely matches with one of the ground truth answers. For example, if the ground truth was "Panic of 1901" and the model responded with "1901", then the ground truth score would be 0. On the other hand, the F1 score is less strict - it is the harmonic mean of precision and recall. An example of an F1 score is with "1901" (model answer) and "Panic of 1901" (ground truth). Since the model answer is completely contained in the ground truth, it would have 100% precision, though only 33.3% recall, as it only included one out of the three words in the ground truth answer. Thus we have an F1 score of $2 \times 100 \times 33.3/(100 + 33.3) \approx 50.0\%$. We first evaluate the baseline's performance and observe its EM and F1 scores, and then compare all subsequent models against it.

4.3 Experimental details

We first downloaded the BiDAF baseline from `https://github.com/minggg/squad` and ran it on standard hyper-parameters to have a preliminary score to compare to. Then, as the baseline did not include character embeddings, we saw that it was a good opportunity to implement an extra layer to increase our scores. Thus, we attempted to test out different methods for character level embeddings. Reading through both the BiDAF and R-Net papers, we were introduced to two different methods: (1) processing the character embeddings through a CNN, and then concatenating them with the word embeddings (BiDAF paper's suggestion) and (2) running word + character level embeddings through a GRU. We observed that the CNN method took around the same time to train, and yielded slightly higher EM and F1 scores. On the other hand, the GRU character embedding layer took around 2.5 times the amount of time to train (with the same default hyper-parameters), and performed moderately worse than the baseline.

As described in the introduction, we initially attempted to re-implement the entire R-Net model, but realized that the CNN method was much stronger with the BiDAF model than the GRU method. Thus, we implemented the Gated Attention-Based Recurrent Networks and Self-Matching Attention Layers (described in Appendix A) described in the R-Net paper. Unfortunately, we soon realized that training the R-Net model took far too long. When we implemented the final pointer network layer, the code would move far too slow, despite being at a batch size of 8. Furthermore, we realized that the machines were running out of memory midway despite a low batch and hidden size. This is because, just to vectorize the calculation for the attention mechanisms (calculating products of the weight matrices and the passage embeddings, the attention scores, and the attention pooling matrix c_t), it would require a 5-dimensional matrix computation. Thus we realized had to trade either runtime or memory, which we did not have in the given virtual machines. As a result, we removed the pointer network output layer; and as the R-Net architecture is similar to the BiDAF architecture in the sense that there is an embedding, encoding, query-context matching, and output layer, we began connecting components of the BiDAF model to the R-Net model. Here, we tried two different derived models. In one model, we used all the pieces of the R-Net model, minus the output layer, and passed it through the modeling and output layer described in Section 3. This model is named "R-Net + BiDAF Output" in the next section. The second model is the model described in Section 3, where we combine the encoding layers described in the BiDAF and R-Net papers, and use both

BiDAF query-context matching and R-Net self-attention, followed by a modeling and output layer. This model is called "Modified BiDAF" in Section 4.4.

We noticed that our "Modified BiDAF" model performed significantly better than the other models we were testing, and that it was oscillating and overfitting a little early. We noticed this "oscillation" when the model's EM and F1 scores would bounce back and forth, increasing and decreasing. We noticed overfitting when, around epoch 10, the EM and F1 scores began to drop for a few evaluation steps, and then go back up slightly, only to drop again. A quick fix to these issues were to gradually decrease the learning rate, as well as making sure to dropout after every layer. After hyper-parameter tuning, we found that a learning rate of 0.2, dropout of 0.2, kernel size 5 for the CNN, 3 layers for the GRU in the self-attention layer, and a hidden size of 100 worked best for training this model. The training process took around 18 hours on the Standard NC6sv3 machine. Ultimately, we obtained competitive results on the CS 224N IID SQuAD leaderboard.

4.4 Results

The EM and F1 scores of each model we tested are displayed in the following table:

F1 and EM Scores		
Model	**EM Score**	**F1 Score**
Baseline (BiDAF)	57.889	60.965
Baseline + CNN char lv. emb.	57.973	61.381
Baseline + GRU char lv. emb.	52.731	55.390
RNet + BiDAF Output	54.327	57.145
Modified BiDAF (Dev.)	**62.342**	**65.540**
Modified BiDAF (Test)	**61.133**	**64.695**

Analyzing the results, it seems that everything went as expected, besides the Baseline + GRU character level embedding model that performed under baseline. Particularly, I was puzzled because it took so much longer to train compared to the CNN character level embedding layer, the loss was going down, and that it was the exact same model as the baseline, but with more information. Initially, this signaled to me that a GRU processing character-level embeddings may be detrimental to the question-answering task, but with some extra thought, this is likely because the GRU character level embeddings are simply more compatible with the other parts of the R-Net model that we did not get to test fully due to time and memory limitations. This suggested that more information is not necessarily beneficial, and that synergy between layers is incredibly important. I made sure to remember this when assembling the Modified BiDAF model, which eventually became a success.

5 Analysis

As described above, we believe that the GRU character level embedding model failed because it was not compatible enough with the rest of the model. Similarly, we believe that the R-Net + BiDAF Output model fails for similar reasons. The pointer network described in the R-Net paper uses an additional attention mechanism connecting the output of the previous attention layer to the actual query. However, due to hardware limitations, we had to cut down on batch size, hidden size, and simplify functions, weakening the model overall. Furthermore, by replacing the pointer network altogether, we lose the final attention layer and thus, lose the fresh information regarding the query.

In the Modified BiDAF model, we compensate for this by passing in the self-attention output into the modeling layer, but passing in the Attention Flow Layer output, along with the output of the modeling layer, into the output layer. This way, when we output probabilities for the start and end, the output layer directly works with context-to-query and query-to-context attentions. And surely, when we kept the model and hyper-parameters the same, but tried inputting the self-attention layer's result directly into the output layer, with the modeling layer output (also generated from self-attention layer's result), we observed slightly weaker results (EM 59, F1 62). Also, we noticed that with the same learning rate and dropout, our Modified BiDAF model began to overfit faster than the RNet + BiDAF Output model. This can be attributed to the fact that the output layer interacts with the context-to-query and query-to-context attentions directly, so the model is able to draw easier connections between the expected answer and the words in the passage. Thus, we needed to ensure that we kept the learning rate lower than the other models, and that we were calling dropout at the end

of each function. Furthermore, the CNN + LSTM start in the Modified BiDAF model simply provides more information (compared to the baseline) – given character embeddings, the LSTM can process subword relationships that normal embeddings may not encode. This hypothesis is verified when you consider the outputs in the csv files: in the first 75 entries, the modified BiDAF model had almost all of the baseline's answers, and more. Finally, we can think of the Attention Flow Layer as an indirect upgrade to the R-Net Gated Attention-Based Recurrent Network Layer. This is because the former does not aim to summarize the query and context into single feature vectors, but rather, at each time step, the attention vector is factored into the subsequent modeling layers. This reduces information loss that may otherwise be a result of the R-Net's Gated Attention-Based Recurrent Network Layer, and thus our model is able to answer more abstract queries whose answers may otherwise have been eliminated or lost as the layer did not deem it relevant right away. These factors combined: adjusting for overfitting and dropout, passing in query-related data into the output layer, extra information from the CNN and its synergy with the LSTM, and the addition of powerful gated self-attention all allow the Modified BiDAF layer to outperform all other models we have tried, including the baseline.

6 Conclusion

In this paper, we present a modified version of BiDAF for reading comprehension and question answering. This modified version utilizes self-attention from Microsoft Asia's R-Net model, applying gated self-attention after the Attention Flow Layer, and then inputting it to the BiDAF modeling layer. We are able to retain BiDAF's strengths in mitigating information loss in context-query pooling or an output layer directly interacting with query data, while building upon it with powerful techniques introduced in the later R-Net paper, such as GRUs and gated self-attention. With this new model, we achieve competitive results on the Stanford CS 224N IID SQuAD leaderboard. For future work, we will try to fully implement R-Net at its full power, as well as test substituting R-Net's Gated Attention-Based Recurrent Network layer for BiDAF's Attention Flow Layer to test for possible improvements.

References

[1] Ali Farhadi Hananneh Hajishirzi Minjoon Seo, Aniruddha Kembhavi. Bi-directional attention flow for machine comprehension. In *arXiv preprint arXiv:1611.01603*, 2016.

[2] Microsoft Research Asia Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. 2017.

[3] Yoon Kim. Convolutional neural networks for sentence classification. 2014.

[4] Stanislaw Antol Margaret Mitchell C. Lawrence Zitnick Dhruv Batra Devi Parikh Aishwarya Agrawal, Jiasen Lu. Vqa: Visual question answering. 2015.

[5] Dhruv Batra Jiasen Lu, Jianwei Yang and Devi Parikh. Hierarchical question-image co-attention for visual question answering. 2016.

[6] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. In *arXiv preprint arXiv:1608.07905*, 2016.

[7] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA,*, 2016.

[8] Kyunghyun Cho Dzmitry Bahdanau and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *CoRR*, 2014.

[9] Karl Moritz Hermann Tomas Kocisky Tim Rocktaschel, Edward Grefenstette and Phil Blunsom. Reasoning about entailment with neural attention. In *CoRR*, 2015.

[10] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N Gomez Lukasz Kaiser Ashish Vaswani, Noam Shazeer and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[11] Jurgen Schmidhuber Rupesh Kumar Srivastava, Klaus Greff. Highway networks. 2015.

A Appendix

The R-Net + BiDAF Output model shown in 4.3/4 corresponds to the model consisting of the following layers:

1. **Question and Passage Encoder** Passing in a question $Q = \{w_t^Q\}_{t=1}^m$ and a passage $P = \{w_t^P\}_{t=1}^n$, we begin by converting the question and passage into their corresponding word embeddings ($\{e_t^Q\}_{t=1}^m$ and $\{e_t^P\}_{t=1}^n$) respectively). We obtain the character-level embeddings by taking the final hidden states of a bi-directional recurrent neural network (RNN) applied to the embeddings of characters in the token. Then, we apply a Gated Recurrent Unit (GRU) to obtain representations for words in the question and the passage. That is,

$$u_t^Q = \text{GRU}_Q(u_{t-1}^Q, [e_t^Q, c_t^Q])$$
$$u_t^P = \text{GRU}_P(u_{t-1}^P, [e_t^P, c_t^P])$$

2. **Gated Attention-Based Recurrent Networks** Now, given question and passage representations u_t^Q and u_t^P from the Question and Passage Encoder, we use a gated attention-based recurrent network to match question information with passage representation. Wang & Jiang (2016) [6] propose match-LSTM:

$$v_t^P = \text{GRU}(v_{t-1}^P, [u_t^P, c_t])$$

where c_t is given by the attention-pooling vector of the entire question u^Q:

$$s_j^t = v^T \tanh(W_u^Q u_j^Q + W_u^P u_t^P + W_v^P v_{t-1}^P)$$
$$a_i^t = \exp(s_i^t) / \sum_{j=1}^m \exp(s_j^t)$$
$$c_t = \sum_{i=1}^m a_i^t u_i^Q$$

Then, we add another gate to the input to the input $[u_t^P, c_t]$:

$$g_t = \text{sigmoid}(W_g[u_t^P, c_t])$$
$$[u_t^P, c_t]^* = g_t \odot [u_t^P, c_t],$$

which is establishes a relationship between the current passage word and its attention-pooling vector of the question. $[u_t^P, c_t]^*$ is utilized in subsequent calculations in place of $[u_t^P, c_t]$, and is called the "gated attention-based recurrent network".

3. **Gated Self-Matching Attention** Now, we are given the question-aware passage representation from previous layers, or $\{v_t^P\}_{t=1}^n$. An issue with such a representation is that it lacks extensive knowledge of context. To circumvent this issue, the paper proposes using attention to match $\{v_t^P\}_{t=1}^n$ against itself, obtaining h_t^P which encodes evidence from words in the passage and their matching question information:

$$h_t^P = \text{GRU}(h_{t-1}^P, [v_t^P, c_t])$$

where c_t is given by the attention-pooling vector of the whole passage v^P:

$$s_j^t = v^T \tanh(W_v^P u_j^P + W_u^{\bar{P}} u_t^P)$$
$$a_i^t = \exp(s_i^t) / \sum_{j=1}^n \exp(s_j^t)$$
$$c_t = \sum_{i=1}^n a_i^t v_i^P$$

Once again, an additional gate as stated above is applied to $[v_t^P, c_t]$.

4. **Output Layer** See layers 5 and 6 in Section 3.