Combining Attention Approaches for the SQuAD Challenge

Luke Asperger
CS 224N: Natural Language Processing with Deep Learning
Stanford University
lukea17@stanford.edu

Abstract

This paper describes an approach to machine comprehension that specifically addresses the Stanford Question Answering Dataset. This paper describes a model that combines multiple effective ways of using attention, including bidirectional attention flow and self-matching attention, to achieve strong results on the SQuAD challenge. Other important features of this model included using several layers of bidirectional recurring networks for modeling as well as convolutional networks to model character-level interactions. Although with more time, this model could have been further fine-tuned and additional features could have been added, the final model achieved results competitive with single-model results from many of the highest-performing research papers.

1 Introduction

Machine comprehension is a difficult challenge in natural language processing. Understand language requires recognizing complex interactions and relationships between words and sentences. However, huge advancements have been made in recent years with the rising popularity of neural networks, which have the ability to model highly complex phenomena when trained with enough data.

Question answering in particular is an important area of study because it has endless applications. The Stanford Question Dataset (SQuAD) was developed at Stanford in 2016 and has since become the de-facto dataset for researchers and tech companies to design and run models on. For this class, students were given a baseline model that function but performed rather poorly and were asked to build upon that model.

2 Background

The Stanford Question Dataset is a reading comprehension dataset consisting of context passages from Wikipedia and questions about these passages. Answers sourced from Amazon Mechanical Turk point to specific spans of text in the passage.

The two performance metrics we use are Exact Match (EM), the percentage of questions for which the models prediction exactly matches the ground truth, and F1, the harmonic mean of precision (the percentage of answer that is part of the ground truth) and recall (the percentage of the ground truth that is included in the predicted answer).

The dataset is broken up into three parts: a training set, a dev set and a test set. As different models are trained on the training set, their performances can be compared using the dev set. To avoid overfitting, a final test set is used that is kept secret and only used to evaluate a final model.

3 Approach

The overall design of my model pulls key aspects from two of the currently highest-performing models, *Bidirectional Attention for Machine Comprehension*[1] (hereafter referred to as BiDAF) and *R-NET: Machine Reading Comprehension with Self-Matching Networks*[2]. Both of these models introduced novel and effective ways of applying multiple layers of attention to the embeddings, which seems one of the most effective ways of improving performance.

After adding the attention layers described in the above papers, I experimented with various extra modeling layers, added character-level convolutional neural nets and tuned various hyperparameters to boost performance. Figure 1 shows the overall design of the full question-answering model. The specific implementations of each layer are described in the following section.

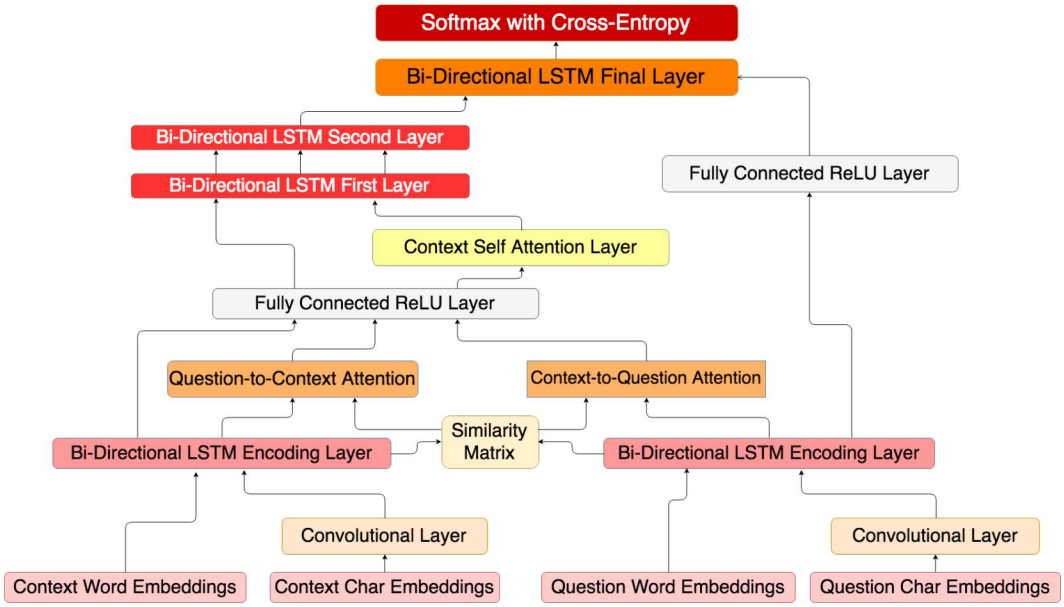

Figure 1: High-level structure of full model

3.1 RNN Encoding Layer with Character-Level CNNs

Once context paragraphs and questions are preprocessed, including tokenization and the mapping of each word to its corresponding index in the embedding matrix, each word is mapped to a 300-dimensional embedding vector. One important detail is that the final training run was performed using pretrained embeddings from the GLoVe 840B Common Crawl set, which included 2.2 million case-sensitive embeddings and was trained over a much larger corpus than the 400,000 case-insensitive embeddings from the 6B set used in the baseline model.

Each character of each word is mapped to a 20-dimensional trainable character embedding. The character embeddings are passed through convolutional neural networks of window sizes 2, 3, 4, and 5, each with 50 filters. The resulting CNN outputs are concatenated to the GLoVe embeddings, resulting in a final embedding size of 500 for each token. These embedding are fed through a bidirectional LSTM encoding layer, which outputs forwards and backwards hidden state vectors of dimension 200 each.

[1]Seo, Minjoon, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. "Bidirectional attention flow for machine comprehension." arXiv preprint arXiv:1611.01603 (2016).

[2]Wang, Wenhui, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. "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. 2017.

3.2 Bidirectional Attention Flow

There are two parts to the bidirectional attention flow as outlined in the BiDAF paper, Context-to-Question attention and Question-to-Context attention. Both rely on the output of a similarity matrix, calculated as follows, where c_i and q_i represent the context and question hidden states, respectively.

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

To calculate the Context-to-Question attention outputs, we apply a softmax to each row of the similatity matrix, which gives us an attention distribution, which is then used to take a weighted sum of the question hidden-states.

The Question-to-Context outputs are calculated similarly, but first we take the max of each row of the similarity matrix. This gives us a single vector of length context length, which we then apply a softmax over and take a weighted sum of the context hidden states. We then produce a final bidirectional attention output b_i by taking concatenation the outputs as follows, where a_i is the Context-to-Question attention output and c' is the Question-to-Context attention output. For further specifics how the weighted sums are calculated, see the BiDAF paper.

$$b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c']$$

3.3 Self-Attention

The bidirectional attention layer produced an output vector of length $hidden\ size \cdot 8$ for each context word, so we pass these representations through a simple fully-connected layer with a ReLU activation to reduce the dimensionality, this time to a hidden size of 100 because the memory requirements of the Self-Attention layer are quite demanding.

To apply Self-Matching Attention, we construct another similarity matrix, this time applying additive attention between context words as follows.

$$e_j^i = \mathbf{v}^T \tanh(\mathbf{W}_1\mathbf{v}_j + \mathbf{W}_2\mathbf{v}_i)$$

Just as in the bidirectional attention layer, we then apply a row-wise softmax max to obtain an attention distribution, which we use to take a weighted sum of the context vectors. These outputs are concatenated to the output of the fully-connected layer and are then passed on to the modeling layer.

3.4 Stacked RNN Modeling Layer

The modeling layer is critical for performance, and is inspired by the modeling layer in BiDAF. We use a bidirectional LSTM in this layer similarly to the encoding layer, but now the RNN inputs have been encoded with more information about both the question and the other context words. To increase the modeling ability of this layer, we stack two LSTMs on top of each other.

3.5 Pointer Network

The final layer of the model is another bidirectional LSTM just as in the modeling layer, but this time we feed in information about the questions one more time by setting the initial states of the LSTM to be the question representations from the encoding layer (fed once through a fully connected ReLU layer). The outputs of this last RNN are fed into a down projecting fully-connected layer so that a softmax can be applied to generate probability distribution for start and end points of the answer spans. The span's end location is conditioned upon the start location such that the end location must come after the start location.

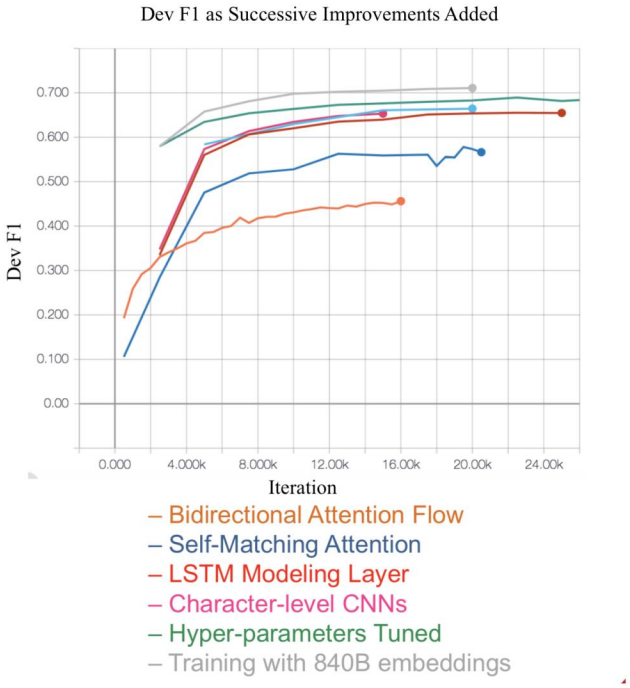

Figure 2: Plot showing improvements in F1 score as improvements were made to model

4 Experiments

4.1 Process

In order to evaluate the performance of each improvement individual, the majority of experiments were run with only one module changed or added at a time. Figure 2 shows the improvements in Dev F1 Score as changes were made. As the graph shows, the additions of the self-attention layer and the stacked LSTM modeling layer provided quite significant boosts of around 10% each. The character-level CNN was added after the modeling layer and provided a boost of around 2%.

4.2 Hyper-parameters

One major hyperparameter change made from the baseline was reducing the maximum question and context lengths from 30 and 600 to 20 and 350 respectively. This reduced memory constraints and allowed batch size to be increased from 10 to 25, resulting in much more efficient training. Another notable improvement is that using the 840B GLoVe embedding yielded a final F1 improvement of around 2%. This gain is likely because the number of out-of-vocab words decreased significantly.

Table 1 shows a full list of the main hyperparameters in the final model. I adopted a dropout rate of 0.2 since the majority of papers I reviewed used that, but with more time I would have attempted to further tune regularization. The sizes of the embeddings and hidden layers were chosen mainly to meet memory constraints for the GPU used to train the model. Potential gains could be realized by increasing these sizes and training on more powerful machines.

4.3 Results

Overall, the results for my model turned out quite well. The final Test F1 score was 77.0 and the final Test EM of 67.9. As Table 2 shows, my model's performance is essentially right on par with BiDAF and R-Net. Because I did not have time to train an ensemble model, I only compare to the single model performances of these models. Additionally, I only had time to train my final model

Table 1: Key hyper-parameters in final model

PARAMETER	VALUE
Max Context Length	350 Words
Max Question Lenth	20 Words
Pretrained Word Embedding Source .	GLoVe Common Crawl (Trained on 840B Tokens)
Word Embeddings Size	300
Character Embedding Size	20
CNN Window Sizes	2, 3, 4, and 5 Letters
Filters Per CNN	50
Batch Size	25
First Hidden Layer Size .	200
Second Hidden Layer Size	100
Dropout Rate	0.2
Max Gradient Norm	5.0
SGD Optimizer	Adam (Learning Rate = .001)

for 7.5 epochs, whereas BiDAF, for example, was trained for 12 epochs. I believe a longer training period would be a very simple way to obtain slightly higher results.

Table 2: Performance of Various SQuAD Models

Model	Test F1	Test F1
My Model .	77.0	67.9
BiDAF (Single Model)	77.3	68.0
R-Net (Single Model) .	77.5	68.4

4.4 Attention Visualizations

One of the most important aspects of this model is the combination of the bidirectional attention layers and the self-attention layers. The following figures help visualize what is actually going on in these layers for an example. For brevity, only the first 20 words of the context paragraph are shown.

Context: North American Aviation won the contract to build the CSM, and also the second stage of the Saturn V...

Question: Who was rewarded with building the CSM?

Answer: North American Aviation

Figure 3 shows attention distribution when Context-to-Question attention is applied. The intuitive way to look at this is that the context words are highlighting which parts of the question are most important. We see the darkest cells for ”Who”, ”building” and ”CSM”, which one could easily argue are the most important words of the question. It is also interesting to note that that, not surprisingly, the attention scores for ”CSM” matched with ”CSM” and ”build” match ”building” are quite high.

Conversely, Figure 4 shows the attention distribution for Question-to-Context attention. It is notable from this figure that ”CSM” and ”Saturn V” still appear to be emphasized more than the correct answer by the attention layer, but ”Aviation” is still highlighted to some extent.

Finally, Figure 5 shows the attention distribution for the Self-Matching layer. Here, ”Aviation” appears to be highlighted darker than before, so the interactions between ”Aviation” and the rest of the context appears to have done something to increase its perceived importance. Although ”Saturn V” is highlighted darkly as well, the model did go on to select the right answer, so presumably something in the modeling layer helped determine that ”North American Aviation” was a better answer than ”Saturn V”.

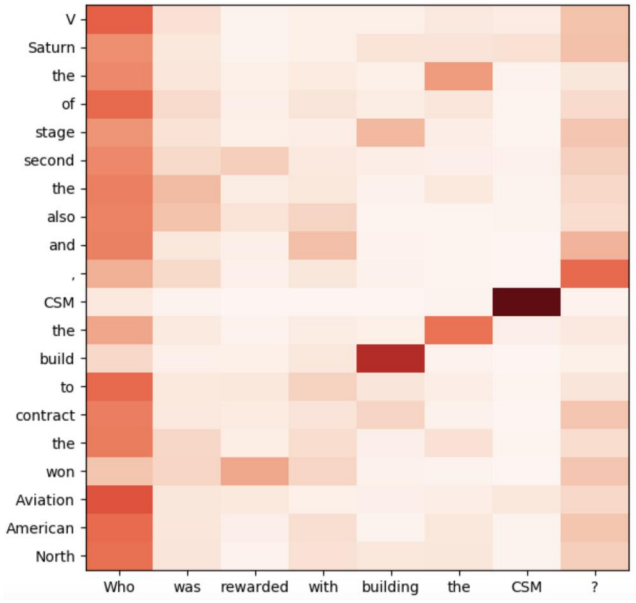

Figure 3: Visualization of Context-to-Question attention distribution

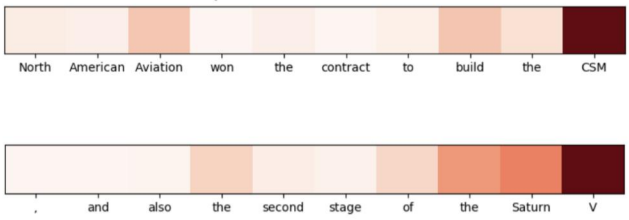

Figure 4: Visualization of Question-to-Context attention distribution

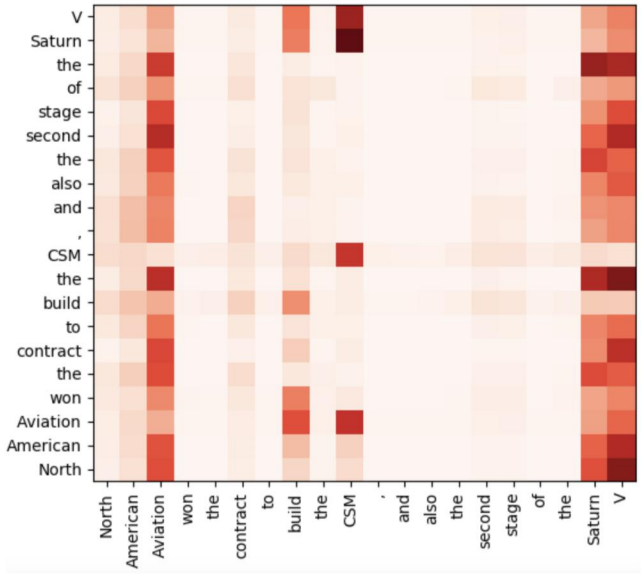

Figure 5: Visualization of Self-Matching attention distribution

5 Conclusion

This model successfully combined several key aspects of R-Net and BiDAF to obtain strong results on the SQuAD test set. Bidirectional attention and self-matching attention work well together. A bidirectional stacked LSTM modeling layer serves as an effective final layer before prediction. Additional gains were achieved by adding character embeddings with convolutional neural nets as well as tuning context and question length to allow for efficient training.

There are certainly several improvements that could be made to this model to improve results. One extension I would have liked to add was another Question-to-Context attention layer after the modeling layer. In my model, the self-attention layer and modeling layers have little interaction with the question, so it seems logical that a final interaction with the question at the end could be useful. I explored this concept and ran several experiments but was unable to tune the design successfully in time.

There are also simpler improvements that would be quite likely to improve performance. Features such as part-of-speech and name-entity tags could be added to the initial embedding in order to encode more information. The model could also be trained using two GPU's, as the memory capacity would allow the model to more easily tolerate longer question and context lengths. A final improvement worth testing is using a pretrained RNN encoding layer such as ELMo.

6 References

[1] Seo, Minjoon, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. "Bidirectional attention flow for machine comprehension." arXiv preprint arXiv:1611.01603 (2016).

[2] Wang, Wenhui, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. "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. 2017.