CS224N Final Project SQUAD Challenge

Saahil Agrawal
Stanford University
saahil@stanford.edu

Nicholas Johnson
Stanford University
nickj@stanford.edu

Abstract

There has been significant progress on applying end-to end neural network based models for solving question answering tasks. While there have been several techniques that have been developed and tried, attention remains the common building block across most of these models. In this report we have applied BiDaff, Bi-directional attention flow, with a modified loss function that conditions end prediction on start prediction for the machine comprehension task of question-answering on SQuAD[1].

1 Introduction

Machine comprehension is a challenging task because it requires machines to do several complex tasks simultaneously like coreference resolution, commonsense reasoning, causal relations, and spatio-temporal relations[5]. The SQuAD dataset is an open source research dataset for machine comprehension that has invoked the interest of several academic and industrial researches alike. It is a highly respected test of measuring the ability of machines in comparison to humans on complicated intellectual tasks. It is an active area of research [1].

In this paper we introduce a modified loss function, customized specifically to the task of question-answering, where the answer is a contiguous subset of the context. The idea originates from a construction that the end prediction position has to be conditioned on the start position. While different researchers have implemented it differently, we decided to leverage the existing context mask and the fact that start position is predicted before the end position. In our implementation we have defined a loss function built over the existing cross entropy loss function by leveraging it's certain characteristics. There are different types of attention mechanisms, first where the attention vector has a temporal dependence on the previous attention vector, second spatial where either context is summarized to a reduced vector, capturing relevance basis the attention weights. Bi-directional attention remains to be one of the better performers on this dataset and hence we have tried to implement the same. Other attention mechanisms like LSTM-match [2] are different mathematical variations of the same idea.

2 Related Work

We looked at the approaches taken by 3 of the top performing single models [2, 3, 4]. End to end deep neural networks integrating LSTMs currently dominate the top of the charts for the SQUAD challenge. All look to use multiple attention matrices in order to build relationships and structures between question and context. Ultimately, we chose to follow the direction of Bi-Directional Attention flow because top performing groups in this course implemented it last year. Additionally, they included convolutional neural networks which was academically interesting for our group. More specifics on parallels between our approach and these groups work are detailed in the rest of the paper.

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

3 Approach

3.1 BiDaff Implementation

As our personal baseline, the first modification implemented on the course provided baseline was reimplementing the BiDaff. The intuitive idea behind is the way humans solve praragraph/comprehension reading questions. Most of us first read the paragraph, then the question, and then go back to paragraph to see which part of the context is relevant to the question we just read. But in doing so, while reading the question, we also use the subconscious memory to identify relationships between context, question and which part of the context you want search. The bidirectional attention flow in some ways simulates this process for the machines. This can be mathematically expressed as follows. Assume we have context hidden states $c_1, ..., c_N \in R^{2h}$ and question hidden states $q_1, ..., q_M \in R^{2h}$. We compute the similarity matrix S $\in R^{N \times M}$, which contains a similarity score S_{ij} for each pair (c_i, q_j) of context and question hidden states. The similarity matrix defined such that,

$$S_{ij} = \mathbf{w_{sim}^T}[c_i; q_j; c_i \circ q_j] \in \mathbb{R}$$

Here, $c_i \circ q_j$ is an element wise product and $\mathbf{w_{sim}} \in \mathbb{R}^6 h$ is a weight vector.

The dot product with weight vector allows us to have flexibility in the way we want to measure similarity between two word vectors. Since, both context and question are of different lengths, a bunch of matrix operations have to be applied in order to calculate the similarity matrix in a memory efficient way. Next, we perform Context-to-Question (C2Q) Attention. We take the row-wise softmax of S to obtain attention distributions α_i, which can be though of as the similarity between context token i and question token j. We then scale each question token with it's similarity score with i'th context token and take the sum of these two to form a context aware representation of the question.

In Question to Context (Q2C), here we find for every context word what is the corresponding question word with which it has maximum similarity. We take this probability/similarity and define a single vector that is a sum of the context vectors weighted by the above probabilities. The mathematical expression is:

$$\begin{aligned} m_i &= \max_j S_{ij} \in R \quad \forall \quad i \in \quad [1, 2, ..., N] \\ \beta &= softmax(m) \in R^N \\ c' &= \sum_i^N \beta_i c_i \end{aligned}$$

All the above vectors are combined along with interactions of attention vectors with context vectors into the blended layer that goes as input to the modeling layer. The modeling layer then passes the input vector to two bi-directional LSTM and concatenates the output vector with the input which is used for start position prediction. Similarly for end the output of second layer is passed through another layer before being concatenated and to be used for prediction.

3.2 Modifications

To make a full Bidaff model, we added the character embedding layer and a highway network to incorporate the character embedding with the word vectors[6, 7]. In order to evaluate unique word vectors for words out of our GLOVE vocabulary we followed the method outlined in [6]. A trainable character embedding was used to map words into $R^n d$ where n is length of word and d is the dimensionality of the character embeddings. In order to reduce this large space into a reasonable word embedding convolutions across the characters in a specific word were performed. These convolutions were then max pooled to generate a single word feature. Therefore the number of convolutional filters defined the dimensionality of the word embedding derived from characters. The highway network simply acts as a fully connected layer that determines how to combine the word and character embedding.

3.2.1 Self Attention

In continuation with the intuition for Bi-directional attention, when we search for answer in the text we also try to use our learned information about the relationships between the context words to select/eliminate certain possibilities. With this logic we intended to implement a self attention layer, which is a variant of the one published in [2]. Here, $\mathbf{W_1}\&\mathbf{W_2}$ are weight matrices with shapes $2h * l$ where l is a hyper-parameter. Similarly v is a vector of size $\mathbb{R}^l$ The mathematical expressions are given by -

$$\mathbf{e_j^i} = \mathbf{v^T} tanh(\mathbf{W_1} c_j + \mathbf{W_2} c_i)$$
$$\alpha^i = softmax(e^i) \in \mathbb{R}$$
$$a^i = \sum_i^N \alpha_j^i c_j$$

The output of this layer is then encoded using a bi-directional RNN-LSTM and the output vector is then concatenated to the blended input vector.

3.2.2 Loss Function Modification

Cross entropy loss is very popular because it is fast to compute and provides nice derivatives for back propagation. However as previously noted cross entropy loss on the true start and end position of the answer with the context isn't a perfect representative of our true evaluation metrics [3]. When we were evaluating our baseline BiDaff implementation we noted that 5% of our answers predicted a start position that was located after our predicted end location. This solution pair is by default invalid. The obvious solution was to impose a hard constraint that the end location be after the start location. From the BiDaff implementation the end position is already conditioned upon the start position prediction, so this would more explicitly define that relation. We implemented this by creating a sub-sampling of the context mask that would constrain the start and end positions from which to evaluate the end prediction. However; when we began to test this we would see giant (e^{30}) losses. After some time debugging, it was discovered that this giant loss occurred when the predicted start position was after the true end position so the cross entropy was capturing the value of the mask we applied. This led to a consideration of what information was being imputed to the model by taking the loss as the sum of the cross entropy loss of the start and end true positions. Implicitly this is stating that both losses are equally as valuable. However as we stated and demonstrated this isn't true. When the start position is after the true end position there is no change for positive F1 or EM, ignoring repeated phrases. Additionally, reiterating if the predicted end is before predicted start the loss also doesn't make sense. Therefore, we looked to modify this loss function to better represent our desire, similar in motivation to [3]. We consider two modifications in the same direction.

$$L_{start}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} log(p^{start}_{sample_i}) \tag{1}$$

$$L_{end}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \lambda_i log(p^{start}_{sample_i}) + (1-\lambda_i) log(p^{end}_{sample_i}) \tag{2}$$

$$\lambda_i = \begin{cases} 0 & argmax(p_{sample_i}) \leq end_i \\ 1 & o.w. \end{cases} \tag{3}$$

$$L_{start}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} log(p^{start}_{sample_i}) \tag{4}$$

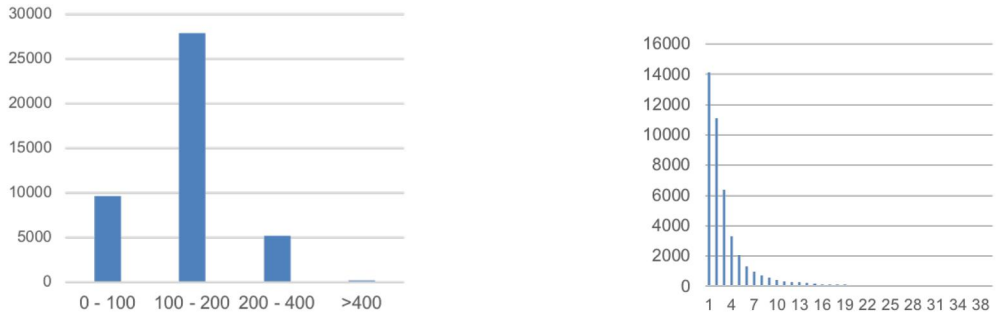
$$L_{end}(\theta) = -\frac{\lambda}{N}\sum_{i=1}^{N} log(p^{start}_{sample_i}) - \frac{(1-\lambda)}{N}\sum_{i=1}^{N} log(p^{end}_{sample_i}) \tag{5}$$

$$\lambda = \begin{cases} 0 & \exists i s.t. argmax(p_{sample_i}) \leq end_i \\ 1 & o.w. \end{cases} \tag{6}$$

One of the modified loss functions acts on the entire batch uniformly and the other acts on specific samples from the batch. From a practical standpoint the batch version should allow for more efficient back propagation and ensure the auto back propagation is only training the end predictions when the start predictions are predicting valid start points. We have defined our λ in a simple way that imputes knowledges that is application specific to the output. Presumably there are a variety of other functional definitions of lambda that could be explore to better relate the interplay between the loss due to the start and end position predictions. For $lambda \in [0, 1]$ this could be interpreted as the confidence in the end prediction conditioned upon the start prediction, outside of this range a more clever interpretation would have to be imagined.

4 Experiments

4.1 Delineate



(a) Distribution of context length

(b) Distribution of answer span

Figure 1: Exploratory statistics of inputs to initialize hyperparameters

The experiments were designed and structured in a staggered fashion such that we were inherently able to conduct an ablation study of our model. The first improvement we made was by replacing the existing attention layer in the baseline model with a bidirectional attention network. We also modified the modeling layer in our first experiment. While we already anticipated that character embedding using CNN would not add a lot of improvement to the model, we implemented the model with character embedding with and without the highway network. As anticipated the dev f1 score in all three cases was almost the same in the range of 66.5 - 68. However we observed we observed the Highway network did give some more explanatory power to our model and hence we decided to keep it in our future model. With our improved understanding of loss function and model prediction we then defined two new models, one with batch wise modified loss function and other with element wise modified loss function. As expected and observed in figure 4 this helped us eliminate all cases where the end and start positions were swapped. This also improved the overall model performance. As a parallel though, we also experimented with self attention combined with bidirectional attention flow. This increases the memory requirements of the model significantly and hence we had to considerably reduce the batch size to 20. The model was also extremely slow to train. Because of the time and resource constraints we decided to not combine this with other improvements.

4.1.1 Bi-Directional Attention

Table 1: Model Params

PARAM	VALUE
GLOVE Dimension	200
Hidden Layer Size	200
Context Length	400
Question Length	30
Batch Size	50

The model was run with architecture as defined by table 1. Context length of 400 was selected as there are negligible contexts with length more than 400 (refer figure 1a). The model was run for both 100 and 200 glove vector size and 200 gave better performance than 100 as expected, although not significant. Hence, in the subsequent experiments the glove size was kept constant because of the memory constraints of GPUs and the added complexity of other layers. The dropout rate, learning rate and gradient clipping limit were left at default of .15, .001 and 5 respectively.

4.1.2 Full BiDaff

Table 2: Model Params

PARAM	VALUE
GLOVE Dimension	100
Character Filter Number	100
Character Filter Width	5
Character Embedding Size	20
Max Word Length	15

The complete BiDaff with character embedding and highway network was implemented with architecture defined by the hyperparameters defined in table 2. The network was tested with both Ada Delta[4] as well as Adam optimizer. Ada Delta is slower than Adam and did not lead into any improvement and hence we selected Adam Optimizer for future tests. Character embedding size was kept small since a single alphabet has very little meaning and hence it is better to project in a much lower dimensional space then the word vector. The max word length was selected to be 15 as more than 90% of the words were observed to have less than 15 characters in them. The hyperparameters used for this model were slightly modified to learning rate of 0.5, dropout rate of 0.2, and max gradient norm to be 10.

4.1.3 Full BiDaff Sample Loss Modification

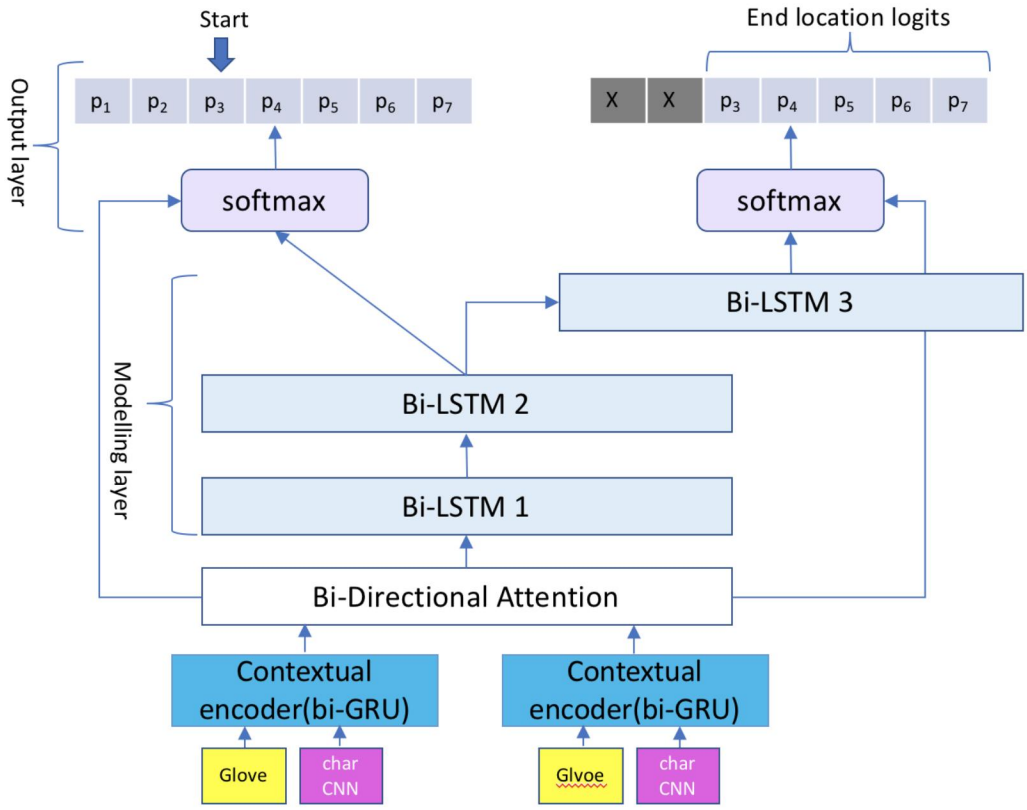


Figure 2: Model architecture schematic with masked end prediction

This corresponds to the first loss function defined in 3.2.2. All model parameters were retained. As has been described earlier, we modified the loss function in the Full BiDaff model. Our technique and works and our intuition is validated as our new model has **0%** invalid answer as shown in figure 5. Refer to the figure 2 for schematic of model architecture.

4.1.4 Full BiDaff Batch Loss Modification

We also tried our model with the batch wise loss function as defined in 3.2.2. This model was extremely slow to train. It is expected since it will first train only on start prediction until the start is not predicted before the true end. Only after this is achieved 100% in a batch it starts training for th end prediction for that batch. As you would observe in figure it shows an interesting step-wise improvement in f1 score.

We took a conscious call of experimenting with different types of architectures and analyze their performance rather than improving the performance of the architecture using hyperparameter tuning or ensemble of different models.

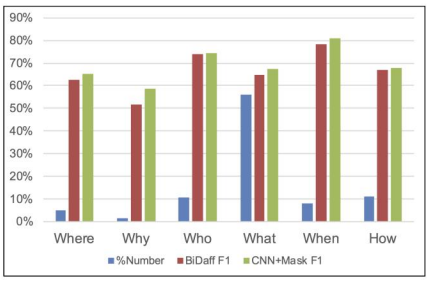
Results Table

Table 3: Model Performance on Dev Leaderboard

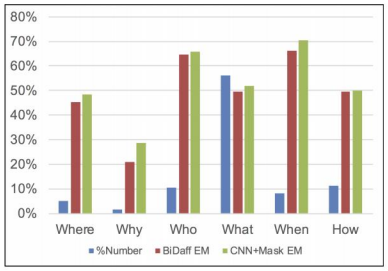
Model	F1	EM
Baseline	43.82	34.72
Bidirectional attention layer	72.21	61.50
BiDaff with self-attention	72.35	61.91
Full BiDaff with Highway	73.50	63.55
Full BiDaff with Element wise loss function	75.01	65.13

4.2 Statistics/Visualizations

We observe in figure 3 that the model under-performs on *Why* questions. We also observe in figure 4 that, irrespective of the modifications the model gets a 0 score on about 20% examples. The above two results indicate that the model is not complex enough to understand intricate relationships to answer why questions and also identify the answer locations.



(a) F1 performance



(b) EM performance

Figure 3: performance of BIDAF and Full Bidaff with modified loss for different questions types.

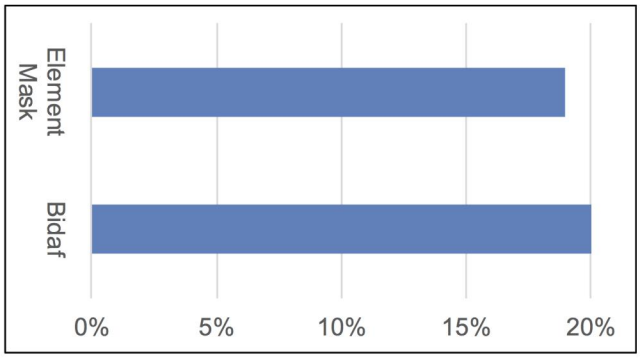


Figure 4: Percentage of answers where F1 is 0

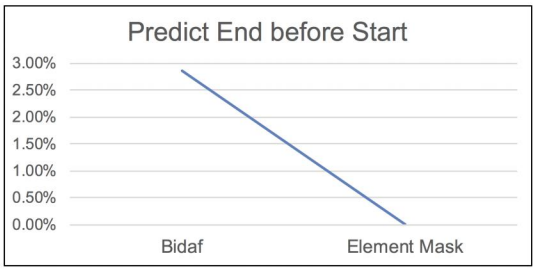


Figure 5: Percentage of end predictions before the start predictions

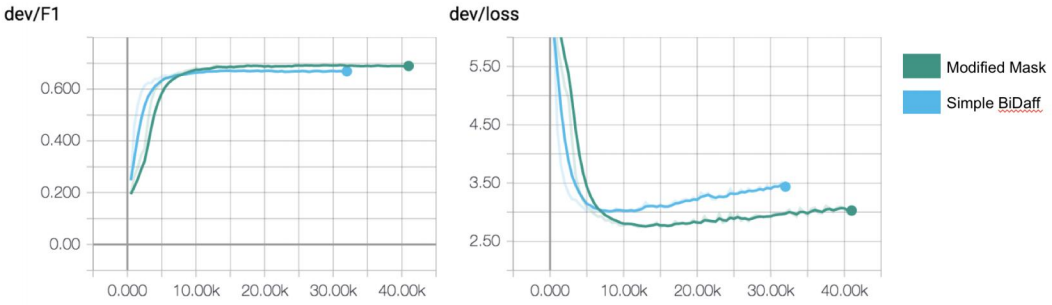


Figure 6: Dev F1 scores and loss respectively for

5 Conclusions

Our analysis of the different models that we have experimented with suggests that attention as a technique definitely helps improve performance at the task of machine comprehension. However, it would be interesting to see how much of an effect does complicated attention network have as compared to just having a basic attention layer with many more Bi-directional GRU/LSTM layers. It is extremely important to condition the prediction of end location prediction on start location. There are several ways to do it, and we have found a simple yet elegant solution which definitely applies very well to this particular task. It is evident that char embedding would be helpful only in the case where we have too many out of vocabulary words or hyphenated words in the text which is rarely the case. Even though the SQuAD challenge is a fairly new challenge, it is surprising to see how quickly we have been able to develop techniques to perform at an acceptable level. It is both astonishing and exciting that machines come close to the human performance in such complex tasks.

6 Future Work

The first thing we would want to do is to extend the modification of our loss function so that we cover the second extreme case where the end is predicted before the true start. This along with the previous modification should considerably improve the model f1 score. We believe that the existing model can definitely be improved upon by fine tuning the hyperparameters and adding more layers to the network. An interesting analysis would also be to combine the self attention with BiDaff and the new loss function to see how the whole system works in tandem. As we see that attention has a significant role to play in the performance of our model, it would be interesting to see how a Transformer model [8] performs on this dataset. Also, we observe that the model has variable performance for different types of questions and hence, some additions like the indication of question category, count of match of question in context etc. can be useful additions to the word vector before the output layer.

Acknowledgments

We would like to thank Microsoft for donating cloud credits that facilitated the training of all of our models. We would like to thank the CS224N course staff for their help at office hours, their prompt responses amongst all of the struggles with CodaLab and ultimately for the extension in the deadline which allowed for most of our exploratory work in the loss function. Finally, we would like to thank everyone in the course who contributed to the discussions on piazza as related to Bidaff.

References

[1] Pranav Rajpurkar et al. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*.

[2] Microsoft Research Asia *R-NET: Machine Reading Comprehension With Self-Matching Networks*

[3] Minghao Hu et al *Reinforced Mnemonic Reader for Machine Comprehension*

[4] Minjoon Seo et al. *BiDirectional Attention Flow For Machine Comprehension*

[5] S. Sugawara and A. Aizawa *An analysis of prerequisite skills for reading comprehension*, EMNLP 2016, p. 1, 2016

[6] Rupesh Kumar Srivastava et al *Highway Networks*

[7] Yoon Kim et al *Character-Aware Neural Language Models*

[8] Ashish Vaswani et al *Attention Is All You Need*