Question Answering System with Bi-Directional Attention Flow

Junjie Ke, Yuanfang Wang
Department of Computer Science,
Stanford University
{junjiek, yolanda.wang}@stanford.edu

Fei Xia
Department of Electrical Engineering,
Stanford University
feixia@stanford.edu

Abstract

For the Assignment 4 of CS 224n course, we did a reimplementation of the Bi-Directional attention flow model (BiDAF). We built the architecture from scratch, tuned the network and tried different regularization and out-of-vocabulary handling strategies. Eventually, we are able to get F1 score 76.5 and EM 66.3 on test set with our ensemble model of five single models.

1 Introduction

Machine Comprehension(MC) is a special task of Question Answering(QA), where the machine is given a query about a given context and is required to predict the answer. Such problems gain significant popularity over the years not only because of their vast applications, but also theoretical values for language and neural network research. To solve this problem, usually some attention mechanism is adopted to focus on only a small portion of the context. This task also requires modeling of the interaction between query and context. To motivate this line of research, Stanford NLP group released the SQuAD[1] dataset, which consists of 100K question-answer pairs, along with a context paragraph for each pair. There is also a public leader board available. The state of the art is already very competitive, as there are many methods that are approaching human level performance.

The rest of the paper is organized as following: Section 2 defines the problem; Section 3 introduces our method and changes to original BiDAF model. We also introduces our tricks and practices for improving performance; In section 4 we analyze both quantitative and qualitative results and different types of error. We also analyze the effectiveness of attention mechanism by visualizing the attention matrices. Finally in section 5, we discussed the experiences we learned from this project and future direction.

2 Problem Definition

Same as the evaluation criterion of the leader board, we define our problem as the following:

Given word sequence of context with length T, $\mathbf{p} = \{p_1, p_2, ..., p_T\}$ and question with length J, $\mathbf{q} = \{q_1, q_2, ..., q_J\}$, the model needs to learn a function $f : (\mathbf{p}, \mathbf{q}) \rightarrow \{a_s, a_e\}$, with the condition $1 \leq a_s \leq a_e \leq T$. $\{a_s, a_e\}$ is a pair of scalar indices pointing to the start position and end position respectively in the context $\mathbf{p}$, indicating the answer to the question $\mathbf{q}$.

3 Model

In this section, we first present the incrementally development of our deep neural network architecture, from baseline to the final model. We then introduce the additional training and fine-tuning strategies for performance boosting.

3.1 Architecture

Starting from the plain code framework provided in the Assignment 4 of *CS224n: Natural Language Processing with Deep Learning (2016-2017 Winter Quarter)* , we built the simple and straight forward baseline model with a Bi-Directional LSTM contextual layer, a context-to-query attention layer, a modeling layer and a output layer.

The contextual layer encodes the embedded sequences of context $\tilde{\mathbf{p}} = \{\tilde{p}_1, \tilde{p}_2, ..., \tilde{p}_T\}$ and question $\tilde{\mathbf{q}} = \{\tilde{q}_1, \tilde{q}_2, ..., \tilde{q}_J\}$ respectively using Bi-Directional LSTM with independent parameters. We get the representation matrix of context words $\mathbf{H} \in R^{T \times 2d}$ and question words $\mathbf{U} \in R^{J \times 2d}$, where d is equal to the size of embedding. The attention layer is the vanilla version: similarity matrix $\mathbf{S} = \mathbf{H}_{t:}\mathbf{U}^T_{j:}$, attention vectors $\mathbf{a}_t = softmax(\mathbf{S}_{t:}) \in R^J$ and attended question vectors $\tilde{\mathbf{U}}_{t:} = \sum_j \mathbf{a}_{tj}\mathbf{U}_{j:} \in R^{T \times 2d}$. Then the attended question matrix $\tilde{\mathbf{U}}_{t:}$ is sent to the modeling and output layer to predict the start and end indices. This process contains two Bi-Direction LSTM layers and the final logistic regression with softmax. Logits was post-processed with padding masks. Cross-entropy loss is used for training. The F1 score of this baseline model on dev set was around 50.

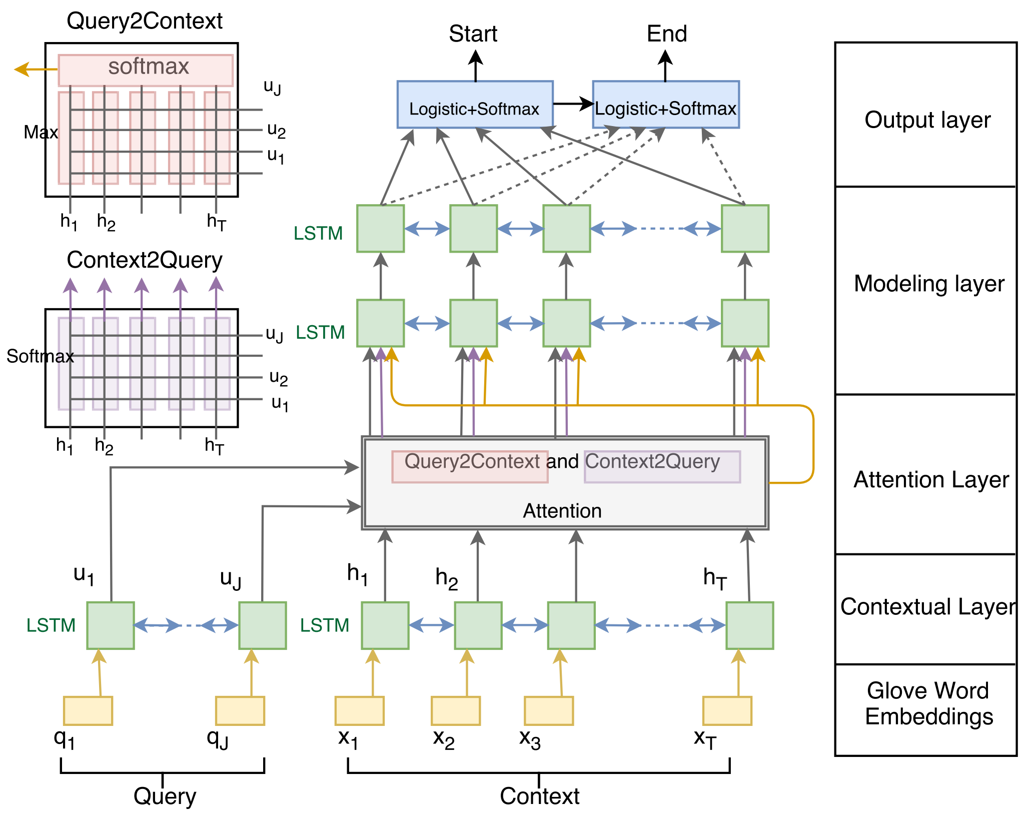

Figure 1: Architecture of our model

Our incremental development of baseline was guided by the BiDAF method [2]. We open source our implementation on Github after the course leader board competition[1].

Embedding layer: Increasing our word embedding size from 100 to 200 does not provide any contribution to the performance of our model. So we finalized the embedding size to 100. Besides, we skipped the hard-training word characters encoding process which is popular used in state-of-the-art models.

Attention layer: Firstly, padding masks were added to the similarity matrix in the attention layer. Query-to-context attention was also added with the result of attended context vectors: $\tilde{\mathbf{H}}_{t:} = \sum_t \mathbf{a}^q_{jt}\mathbf{H}_{t:} \in R^{J\times 2d}$, where $\mathbf{a}^q = softmax(\max_{dimension} \mathbf{S}_{t:}) \in R^T$. Moreover, we embraced more complex similarity function, $\mathbf{S} = \mathbf{w}[\mathbf{H}_{t:}; \mathbf{U}_{j:}; \mathbf{H}_{t:} \circ \mathbf{U}_{j:}]$. This single modification of similarity matrix added around 2 points for our F1 score.

Modeling layer: We still used two layers of Bi-Directional LSTM, which is the same as BiDAF[2]. $\mathbf{M_1}$ and $\mathbf{M_2}$ represent the outputs of these two layers respectively.

Output layers: We just used $\mathbf{WM_2}$ as logits before softmax, instead of the complex format used in BiDAF[2]. We also skipped the additional Bi-Directional LSTM between start and end indices prediction.

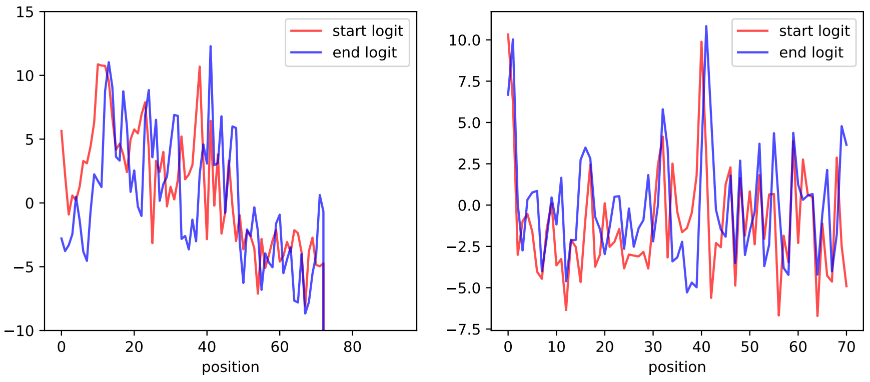

Figure 2: Example of start and end logits

Dynamic Programming: After getting the logits for start and end positions, we use dynamic programming strategies for the final prediction. In the baseline model, answer span (s, e) was selected with independent maximum probability p_s, p_e. As is shown in Figure 2, there might be multiple peaks of probabilities that are close to each other. Since the end position never lies before the start position, we should never choose an end position where start logits before are low. Therefore, we select (s, e) where $s \leq e$ with the maximum value of $p^1_s p^2_e$, which can be calculated in linear time using dynamic programming. In this way, we are able to select legal answer spans where the joint probability of start and end are the highest. After using this DP strategy, the F1 score was improved by 2 points. Since most the answers are short and lie in one sentence, we then introduced a sentence-level DP to find the maximum $p^1_s p^2_e$ where s and e are in same sentence. This improvement boosted the F1 score by another 0.5 point.

3.2 Training details

This sub-section is about hyper-parameter tuning and tricks to speed up training.

Padding Strategy: To deal with the sequence length differences of data in the same batch sent to the model, in the baseline method, each sequence was padded to the maximum length in the whole dataset, which is 766 and 60 for context and question respectively. The large proportion of redundant computation brought single epoch training over 3 hours runtime on the baseline model. To reduce

[1]https://github.com/yolandawww/QASystem

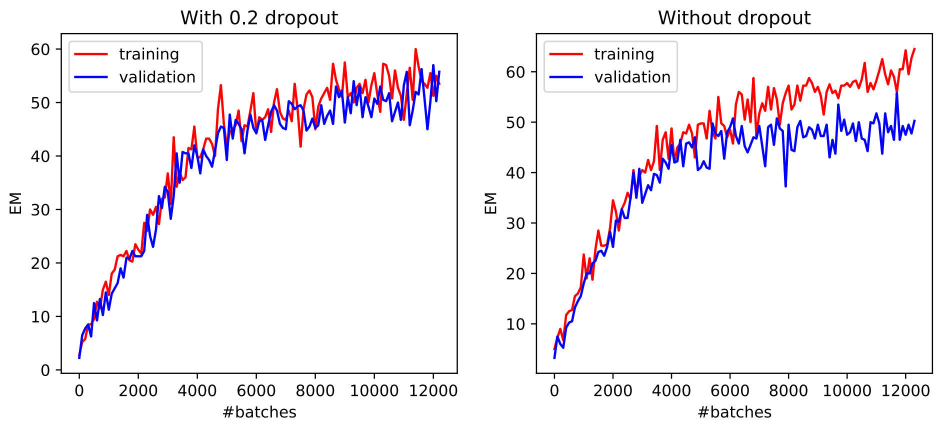

Figure 3: EM performance with and without dropout

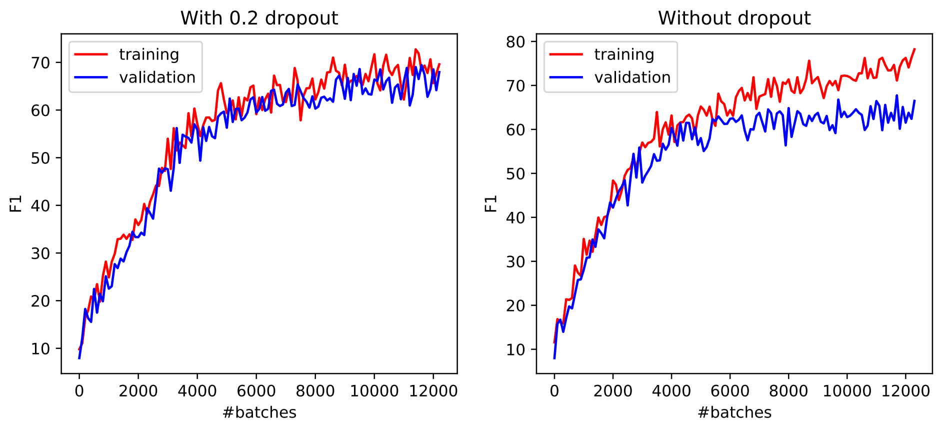

Figure 4: F1 performance with and without dropout

the redundancy, we sorted the training set by sequence lengths and then get each of our data batch randomly from a randomly chosen window in the sorted training set. The size of window is three times of the batch size. Then the input data are padded to the maximum length in the batch rather than the whole dataset. This trick fastened our training process by 3 times.

Optimizer: We used Adam optimizer with initial learning rate 0.048.

Avoid overfitting: We use a weight decay and dropout to avoid overfitting. The weight decay rate used is 0.9999. As in Fig 3 and Fig 4, even with weight decay, the model overfits quickly. So we applied a dropout to the input gate of all LSTM cells with dropout rate 0.8. Still, we are suffering over 10 points F1 score gap between training and validation set.

OOV(out of vocabulary) handling: OOV handling is important at test time. We use different OOV handling methods: 1) set embeddings of OOV to zeros. 2) set embeddings of OOV to a random vector, 3) set embeddings of OOV to glove embedding, if not in glove, set to random. The results can be found in Table 1.

Table 1: Performance comparison with different OOV handling

Model	F1 Score	EM
OOV set to glove/random	72.2	61.22
OOV set to random	71.9	60.8
OOV set to zero	71.2	60.2

Table 2: Performance comparison with other methods

	F1 Score	EM	F1 Score	EM
Model	**Dev set**		**Test set**	
BiDAF(our implementation, single)	72.1	61.0	-	-
BiDAF(our implementation, ensemble)	75.8	65.3	76.5	66.3
BiDAF[2](reference implementation, single model)	77.3	67.7	77.3	68.0
BiDAF(reference implementation, ensemble)	80.7	72.6	81.1	73.3
MPCM[3]	-	-	75.1	65.5
Dynamic Coattention	-	-	75.9	66.2
r-net	-	-	77.9	69.5

4 Results and Analysis

We achieved F1 score 75.8 and EM 65.3 on dev set, and F1 score 76.5 and EM 66.3 on test set. A detailed comparison with state-of-the-art models can be found in Table2. From the results, we can see that the performance is comparable to state of the art machine comprehension methods. The performance gap with the reference implementation of BiDAF can be explained by lack of character level embeddings.

4.1 Error Analysis

We analyze the error cases in 'dev-v1.1.json' where our model achieves less than 80% F1. We find out that there are 4 major problems with our model.

4.1.1 Imprecise answer boundaries

In most of the incorrect cases, our model gives an inaccurate position around the boundaries. 2-3 words around the answer span may be mistakenly omitted or included. It may be very hard to get rid of these mistakes completely.

Example:

- **Context**: Rugby is also a growing sport in southern California, particularly at the high school level, with increasing numbers of schools adding rugby as an official school sport.
- **Question**: At which level of education is this sport becoming more popular?
- **Prediction**: 'high school level'
- **Answer**: ['high school', 'high school', 'high school']

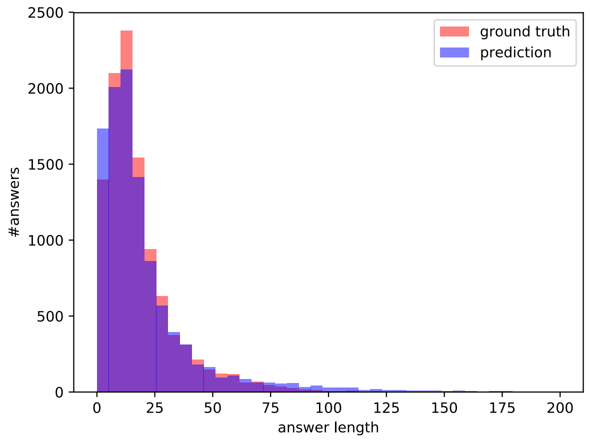

Figure 5: Distribution of ground truth answer length and predicted answer length

4.1.2 Long tail of predicted answers

By comparing the distribution of answer length (Figure 5), we find that our model tends to give a longer answer than ground-truth. Often times it's able to correctly predict the starting position but has a long tail of many redundant words.

Example:

- **Context**: The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott. The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott.
- **Question**: Where was the practice place the Panthers used for the Super Bowl?
- **Prediction**: 'San Jose State practice facility and stayed at the San Jose Marriott'
- **Answer**: ['San Jose', 'the San Jose State practice facility', 'San Jose State']

In those cases, although the model can pay attention to the correct starting position. It doesn't know where to stop. Since most answers are short and concise in the SQuAD dataset, so this problem may greatly impair our performance. This problem may due to the lack of interaction between the prediction of begin and end positions. We've tried different strategies to alleviate this problem, including the sentence-level dynamic programming mentioned previously. But the problem still exists after those improvements. We think that it may be beneficial to add an additional LSTM layer for the end position prediction.

4.1.3 Wrong position of attention

Sometimes the model is paying attention to the wrong region of the context and missing the real answer completely.

Example:

- **Context**: The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott. The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott.
- **Question**:At what university's facility did the Panthers practice?
- **Prediction**: Stanford University
- **Answer**: ['San Jose State', 'San Jose State', 'San Jose State']

In the above example, the model mistakenly focuses on 'Stanford University' because of the word 'practiced'. The real answer 'San Jose State' is also associated with 'practice' but the model failed to capture the word 'facility' in the context to distinguish it from the fake answer. The attention layer of our model is not strong enough to capture complex interactions between context and query. Since our model only used one layer of attention in two directions, we think that it may be helpful of adding deeper layers to recursively compute query-to-context and context-to-query attention. Intuitively, this will allow attention signals to flow through different regions with further comprehensive understandings so that the model can make multi-step complicated deductions.

4.1.4 Syntactic ambiguities

Example:

- **Context**: A piece of paper was later found on which Luther had written his last statement. The statement was in Latin, apart from "We are beggars," which was in German.
- **Question**: What was later discovered written by Luther?
- **Prediction**: A piece of paper
- **Answer**: ['his last statement', 'his last statement', 'last statement']

In those cases, there's nothing wrong with the model's answer syntactically but this response is not favored by humans. In the above example, the fact that Luther is writing on "A piece of paper" is of less importance. These are very hard cases because the model should be able to distinguish what's truly important from a human's perspective.

4.2 Attention Analysis

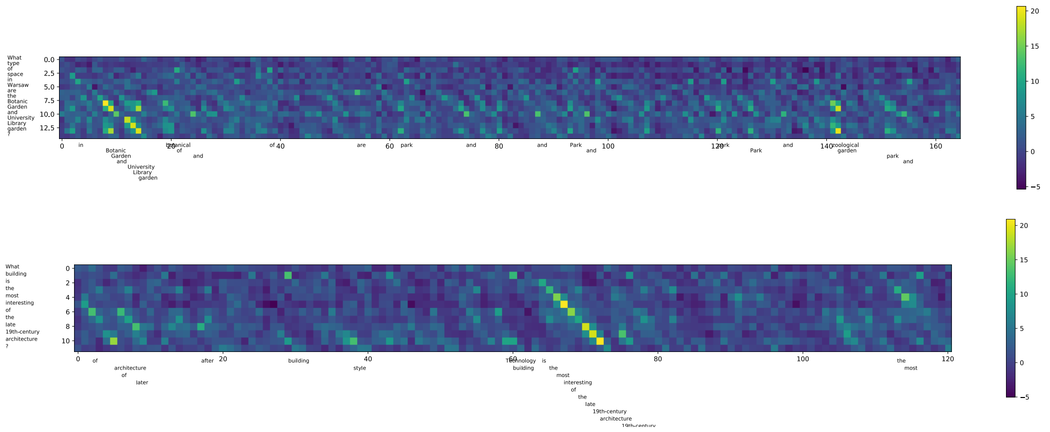

Figure 6: Attention matrices visualization

In this subsection we visualize the attention matrix(zoom in to see details in Fig 6 and Fig 7). Though in practice we used a more complicated method to fuse $\mathbf{u}$ and $\mathbf{h}$, here for simplicity, we visualize the heatmap of dot product of $\mathbf{u}$ and $\mathbf{h}$. The words with large values are displayed. In the first example, with the question mentioning "University Garden and Botanical garden", the attention is mainly on the start and the end of the sentence. In the start it mentions botanical garden and in the end it mentions zoological garden. In the second example, the question asks about which type of building is the most interesting. The network focuses on the part mentioning architecture and building, and get the answer "technology building" correctly. By doing this visualization, we find the attention mechanism is quite effective.

Note that even the dot product of $\mathbf{u}$ and $\mathbf{h}$ is not large, it does not mean the network cannot get the correct answer. Because later there are still modeling layers. In Fig 7, the attention from "year" to "2015" is higher compared with neighbors, but lower compared with identical matches, then after modeling layers the network is able to get the correct answer.

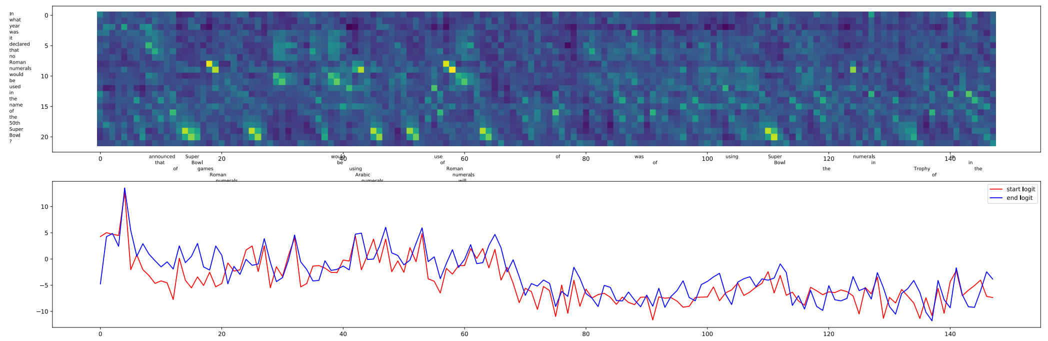

Figure 7: Attention matrices visualization aligned with logits

5 Discussion and Conclusion

In this paper, we reports our results and implementation details of the start-of-art Bi-Directional attention flow model which represents the context at different level and combined the context-to-query and query-to-context direction attention. The experimental evaluations show that our model achieves competitive results in Stanford Question Answering Dataset (SQuAD). Detail analysis and visualizations show that the model is able to attend to correct context locations based on the question. However, the attention region is mostly based on word meaning similarity and the model is incapable of doing complex deductions and reasoning. Future work can be done on incorporating deeper attention layers with multiple hops to allow deeper interaction between context and query.

Acknowledgments

The authors would like to thank TAs for insightful discussions and timely feedback about improving the models. They also want to thank the classmates for creating a competitive and exciting leader board.

References

[1] Rajpurkar P, Zhang J, Lopyrev K, et al. Squad: 100,000+ questions for machine comprehension of text[J]. *arXiv preprint* arXiv:1606.05250, 2016.

[2] Seo, Minjoon, et al. "Bidirectional Attention Flow for Machine Comprehension." *arXiv preprint* arXiv:1611.01603 (2016).

[3] Wang, Zhiguo, et al. "Multi-Perspective Context Matching for Machine Comprehension." *arXiv preprint* arXiv:1612.04211 (2016).