Reading Comprehension on SQuAD: An Insight into BiDAF

Vivekkumar Patel
Department of Computer Science
Stanford University
vivek14@stanford.edu

Shreyash Pandey
Department of Electrical Engineering
Stanford University
shreyash@stanford.edu

Abstract

Application of end-to-end Deep Learning has been very successful in a lot of tasks. Here, we apply this technique for the problem of reading comprehension and answering questions based on them. Specifically, we re-implement the Bi-directional Attention Flow technique and sutdy the benefits of various components of the model. We also suggest improvements to the modeling and output layers, and obtain final F1 and EM scores of **77.08** and **66.80** respectively on the dev set, and **77.77** and **68.006** on the test set.

1 Introduction

With the advances in deep learning and availability of large computational resources and data, end-to-end deep learning models have become very successful. In various problems today, the state of the art models have out-performed humans. Another reason for their attractiveness is that the problem solver no longer needs in-depth domain knowledge and there is no need for hand-crafted features. These benefits are valuable when one works in the field of NLP and especially Machine Comprehension (MC).

In this task, the algorithm is given a context and a question, and the algorithm is expected to provide the answer to that question based on the context. Hence, the algorithm must be able to see through the complex relationships existing between the question and the context to figure out the right answer. We use the SQuAD dataset for training and testing of our model. It is a reading comprehension dataset which has more than 100,000 triples of context,question and answers.

The rest of the paper is as follows: Section 2 goes through some of the existing work on this problem. In Section 3, we describe our approach. Section 4 consists of the Experiments, Results and our observations. Finally, we conclude in Section 5 with our experiences.

2 Related Work

Since the release of SQuAD dataset in 2016, a multitude of deep learning approaches have achieved near human F1 accuracy, with some of them surpassing the human Exact Matching (EM) accuracy. RNN encoders and attention mechanisms are known to be particularly successful at the reading comprehension task. **BiDAF**[3] is a very well known and studied approach that takes advantage of two-way attention, from question to context and from context to question, by constructing a Bidirectional Attention Flow layer. Models trained on variants of BiDAF are still competitive with the best models on the SQuAD leaderboard. Other attention mechanisms such as **Coattention** and **self-attention** have been parts of successful approaches such as Dynamic Coattention Network[4] and R-Net[5]. The current state-of-the-art model uses Attention-over-Attention Neural Networks [2] that estimates the answer from the document-level attention instead of calculating blended representations of the document by placing another attention over the primary attentions, to model the "importance" of

each attention. Other competitive approaches include Reinforced Mnemonic Reader[1] that drifts away from the above "encoder-interaction-pointer" approaches by incorporating both lexical and syntactic features such as POS, NER with the embedding of each word to enhance the capacity of the encoder. The modeling is based on a memory-based answer pointing mechanism, and they directly optimize both the EM metric and the F1 score, by introducing a new objective function which combines the maximum-likelihood cross-entropy loss with rewards from reinforcement learning.

In this work, we re-implement Bidirectional Attention Flow, and experiment with and analyze the effect of each modeling detail. We implement character-level CNN to augment the word embeddings and experiment with different modeling layers such as LSTMs to improve upon the fully connected layers. We also employ dynamic programming in the output layer to make sure that our model always outputs valid answers. Lastly, we construct an ensemble of our models to achieve a final dev F1 of **77.08** and test F1 of **77.77**.

3 Approach

3.1 Dataset Analysis

We use the SQuAD dataset for training and testing of our models. It consists of more than 100,000 question-answer pairs on more than 500 articles, asked by crowdworkers on wikipedia articles. The answers to the questions always lie in the context paragraph. Below is a distribution of lengths of contexts, questions and answers in the training set. These help us fix values of certain hyperparameters, such as the max context length and question lengths to be considered.

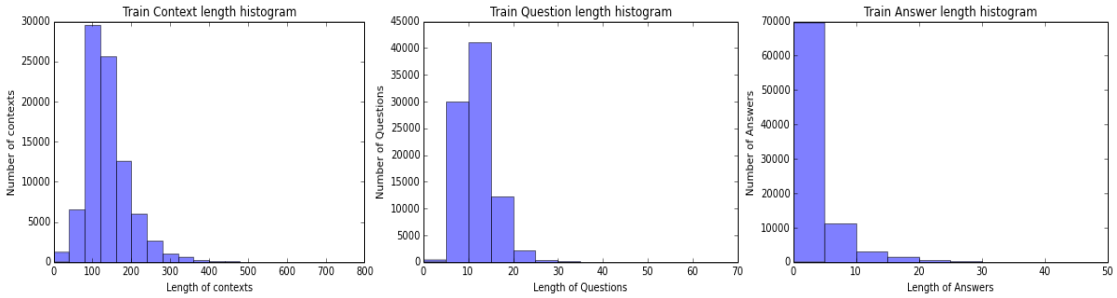

Figure 1: Distributions of Context, Question and Answer Lengths respectively

We chose **`context_len`** to be **600** and **`question_len`** to be **30** even though smaller values would have been fine. This was because we did not want to lose out on information in longer contexts or questions. We used a **`batch_size`** of **32** so that we did not face memory errors during run time.

3.2 Model details

The model we implement is very similar to the original BiDAF model.

The following are the details for each layer.

- **Glove and character embeddings**: To represent each word, we use pre-trained Glove[7] embeddings of dimension **100** and have trainable character embeddings of size **20** for each character. We use 1-D CNN followed by max pooling over the character embeddings to get an embedding for the whole word of dimension **100**. This embedding obtained from characters is concatenated with the Glove embedding and is passed on to the next layer. We denote the output of this layer as $x_1, x_2, x_3 \dots x_N \in \mathbb{R}^{200}$ for the context and $y_1, y_2, y_3 \dots y_M \in \mathbb{R}^{200}$ for the question. Here, N is the **`context_len`** and M is the **`question_len`**.
- **Contextual Layer**: This layer consists of a bi-directional LSTM, which takes in the word embeddings and outputs the hidden-states. We denote these hidden states as $c_1, c_2, \dots c_N \in$

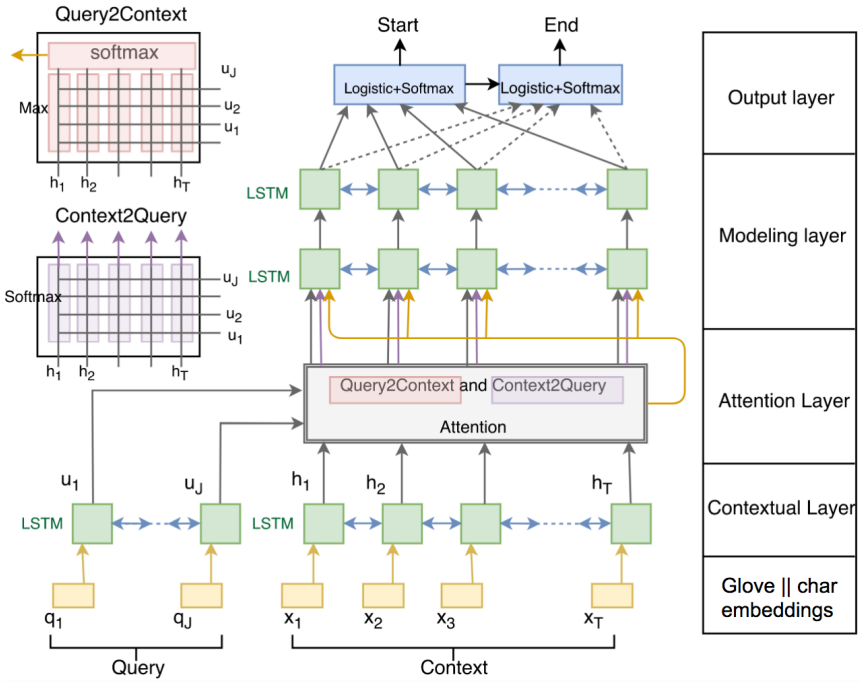

Figure 2: A picture of the architecture we implement. Image borrowed from Fei Xia et al.'s report from last year.

$\mathbb{R}^{200}$ for context and $q_1, q_2, \ldots q_M \in \mathbb{R}^{200}$ for question, where each c_i, q_i are of the form $[\overleftarrow{x}_i; \overrightarrow{x}_i]$, the concatenation of hidden layers of LSTMs of both directions.

- **Attention Layer**: This layer is implemented in the same way as done in the original BiDAF paper. The C2Q attention is calculated as $a_i = \sum_{j=1}^{N} softmax(S_{i,:})_j q_j \in \mathbb{R}^{200}$, where S is the similarity function given by $S_{ij} = w^T[c_i; q_j; c_i \circ q_j]$, for $i \in [1, n], j \in [1, m]$. The Q2C attention is calculated as $c' = \sum_{i=1}^{N} softmax(\theta)_i c_i \in \mathbb{R}^{200}$ where $\theta_i = \max_j S_{ij} \in \mathbb{R}$ and so $\theta \in \mathbb{R}^N$.
- **Modeling Layer**: From the previous layer, we make a blended representation $b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c'] \in \mathbb{R}^{800}$ for $i \in [1, n]$. Denote $[b_1, b_2, \ldots b_N]$ as $G \in \mathbb{R}^{N,800}$. These representations are passed through two layers of LSTMs(as shown in the figure above). The final output of these layers is denoted as $M \in \mathbb{R}^{N,200}$. This makes our **Full_bidaf** model. We then add two more bi-directional LSTM layers, which gives us our **Modif_bidaf** model.
- **Output Layer**: In this layer, we calculate the starting point and ending point. To calculate the start point, we simply take the softmax of $w_1^T[G; M]$. To calculate the end-point, we first pass M through another LSTM layer to obtain $M_2 \in \mathbb{R}^{N,200}$. Then, for the end-point, we take the softmax of $w_2^T[G; M_2]$.
- **Dynamic Programming**: In the baseline model, answer spans (start, end) were chosen with maximum probability p_{start}, p_{end}, independent of each other. But as pointed out by Junjie et al. [6], there are generally several peaks of probabilities that are very close to each other. To generate a valid answer, we have to make sure that the end position always follows the start position. Hence, we choose (start, end) where start $\leq$ end with the maximum value of $p^1_{start} p^2_{end}$, and this can be calculated in linear time using dynamic programming. By doing this, we choose legal answer spans where the joint probability of start and end locations are the highest.
- **Ensembling**: As a final step, we combine the predictions of three models trained with the above architecture. For this, we do a form of majority voting. The predicted start and end distribution is generated by a weighted average of the three models' predictions, weighted

by their F1 scores on the Dev set. Dynamic Programming on this distribution gives us the predicted answer span.

3.3 Training Details

Following are some of the details that describe our training process:

- **Padding Strategy**: Padding is required to make all examples of same size for batch processing. Based on our analysis of the dataset, very few contexts had length greater than 600, and very few questions had length greater than 30. For a consistenty's sake, and for a fair comparison of various components of the model, we kept the padding same for all our experiments.
- **Optimizer**: We use Adam optimizer with an initial learning rate of 0.001.
- **Dropout**: To avoid overfitting on the training set, we use a dropout probability of 0.15.
- **Model Sizes**: All our bi-directional LSTMs have a hidden size of 100 (total 200). Our CNN based character embeddings use a kernel "window" size of 5 with 100 such filters.

4 Experiments, Results and Observations

4.1 Evaluation of different stages

As we implemented features one at a time, this allowed us to analyze and figure out the benefits of these features individually. The following table lists the **F1** and **EM** and **no-answer** scores of the models. All these values are on the dev set.

Table 1: Performance on the Dev set

Model	**F1 Score**	**EM Score**	**No-answer**
Baseline	43.93	34.58	16.05
Bidaf_attn	49.99	39.66	14.07
Bidaf_attn+cnn	51.76	41.67	13.08
Full_bidaf	73.16	62.89	3.44
Modif_bidaf	73.72	63.83	3.45
Modif_bidaf+dp	75.13	64.02	**0.0**
Ensemble	**77.08**	**66.80**	**0.0**

4.2 Evaluation based on Question Types

The first analysis that we carried out was to see how these models performed over different kinds of questions. The most common questions generally have one or more of the question words: What, When, Why, Where, How, Whom, Who and Which. Many questions in the dev-set had two question words, and some even had three. The following table summarizes the metric scores on the questions based on the question word they contain. For cleanliness, we only mention statistics of 3 models.

Table 2: Performance on Different Question types

Question Word	#Questions	**Baseline**			**Modif_bidaf**			**Ensemble**		
		F1	**EM**	**NA**	**F1**	**EM**	**NA**	**F1**	**EM**	**NA**
What	5981	40.48	30.13	17.72	71.38	60.61	4.31	**75.25**	**63.97**	**0.0**
How	1338	49.27	40.10	13.75	74.50	64.36	2.88	**75.99**	**65.73**	**0.0**
When	840	55.52	49.65	10.41	83.75	78.24	1.27	**85.43**	**79.43**	**0.0**
Whom	39	57.71	51.28	12.82	**82.05**	**82.05**	**0.0**	**82.05**	**82.05**	**0.0**

The following observations can be made from the above table:

- **Modif_bidaf** performs best on most types of questions.
- On the "Whom" questions, **Modified_bidaf** never predicts the end point before the start point. Also, it has same **F1** and **EM** score in this category, which means whenever it has a nearly correct answer, the answer is exact.

4.3 Examples to analyse benefits of different components of BiDAF

We now present the advantages of different layers that have been used in the **BiDAF** model and support them through examples.

4.3.1 Advantage of Bidaf Attention over Simple Attention

In the Simple attention method, we only made use of Context2Question (C2Q) attention, whereas in BiDAF, we have a similar Context2Question (C2Q) as well as Question2Context(Q2C) attention. In the following example, the baseline fails to answer whereas using only the BiDAF attention, the model answers correctly.

Context: `The Panthers finished the regular season with a 15-1 record, and quarterback Cam Newton was named the NFL Most Valuable Player (MVP). They defeated the Arizona Cardinals 49-15 in the NFC Championship Game and advanced to their second Super Bowl appearance since the franchise was founded in 1995. The Broncos finished the regular season with a 12-4 record, and denied the New England Patriots a chance to defend their title from Super Bowl XLIX by defeating them 20-18 in the AFC Championship Game. They joined the Patriots, Dallas Cowboys, and Pittsburgh Steelers as one of four teams that have made eight appearances in the Super Bowl.`

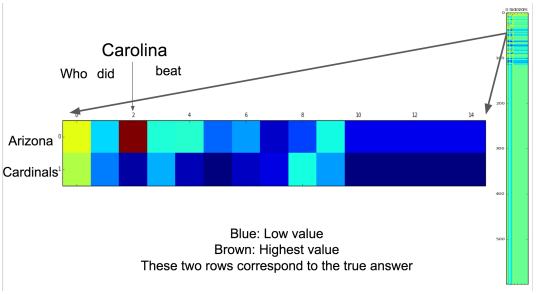

Figure 3: A visualization of similarity matrix used in BiDAF

Question: `Who did Carolina beat in the NFC championship game?`

Answer: `Arizona Cardinals`

Answers from our models:

1. Baseline: (No answer)
2. Bidaf_attn: Arizona Cardinals
3. Bidaf_attn+cnn: Arizona Cardinals 49-15
4. Full_bidaf: Arizona Cardinals 49-15
5. Modif_bidaf: Arizona Cardinals 49-15

Explanation: In the above example, after going through the context and the question, there remains a confusion. The question asks "Who did Carolina beat" whereas the context does not have any mention of "Carolina". Hence the answer can be found only after finding other relevant features from the question that relate to context. This needs an attention mechanism going from question to context. As can be seen from the answers of the models, only after using the Bi-directional attention, do we see that the model is able to produce correct answer in such cases. Figure 3 also captures this information.

4.3.2 Benefit of Character Level Embeddings

One of the shortcomings of using only pre-trained word-embeddings is that the model will fail if the answer is **numerical** or has some other **special characters**. The below example is one of the many that we found, which demonstrates this effect.

Question: `How many receptions did Cotchery get for the 2015 season?`

Context: `The Panthers offense, which led the NFL in scoring (500 points), was loaded with talent, boasting six Pro Bowl selections ...Ginn also rushed for 60 yards and returned 27 punts for 277 yards. Other key receivers included veteran Jerricho Cotchery (39 receptions for 485 yards), rookie Devin Funchess (31 receptions for 473 yards and five touchdowns), and second-year receiver Corey Brown (31 receptions for 447 yards) ...Carolina's offensive line also featured two Pro Bowl selections: center Ryan Kalil and guard Trai Turner.`

Answer: `39`

Answers from our models:

1. Baseline: (No answer)
2. Bidaf_attn: (No answer)
3. Bidaf_attn+cnn: 39 receptions for 485 yards
4. Full_bidaf: 39
5. Modif_bidaf: 39

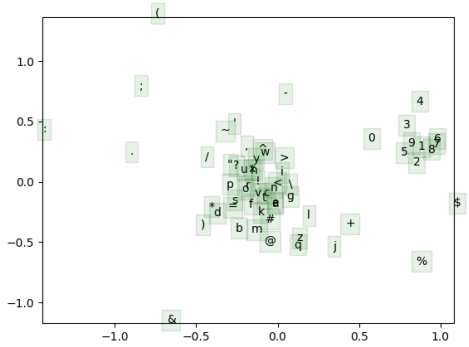

Figure 4: A visualization of the character level embeddings. Notice that the digits and alphabets are grouped in different clusters.

Explanation: When we use only pre-trained word embeddings, numbers will generally get replaced by <UNK> and so the model will fail to answer correctly in cases where the answer is numerical. If we supplement the word embeddings with character embeddings, then the model has some information to learn features about these numbers (see Figure 4, the digits are in a separate cluster) and hence, a shot at predicting the correct answer. The answers from our models strongly support this observation. Infact, even the answers in previous example demonstrate this (Bidaf_attn+cnn predicts "Arizona Cardinals 49-15").

4.3.3 Benefit of Modeling and Output Layer and the Modification

The most common problems in the **baseline** and **bidaf_attn** model were that the models would either predict longer answers than necessary or would predict the end before the start. Using RNNs (LSTM to be specific) in the modelling and output layers gives the largest improvement on the evaluation metrics by partially solving these problems. These are following observations that can be made from the results:

- The main benefit of using any RNN over a fully connected layer is that the weights are shared across words. This helps use the attention features in the blended representation more efficiently.
- Models with fully connected layer generally predict longer answers (the true answer is generally contained in the predicted answer). By using RNNs, we observe that the model is able to find the start and end points more accurately. With the modification suggested, start and end point prediction improve more. The example after the Figure 5 is a good indicator of this point.

- We introduce a dependency (even if it is very light) between the predicted start and end point when we add an LSTM layer between the the start and end computations.
- The predicted answer length distribution converges to the true answer length distribution with this change as evidenced by the following plot.

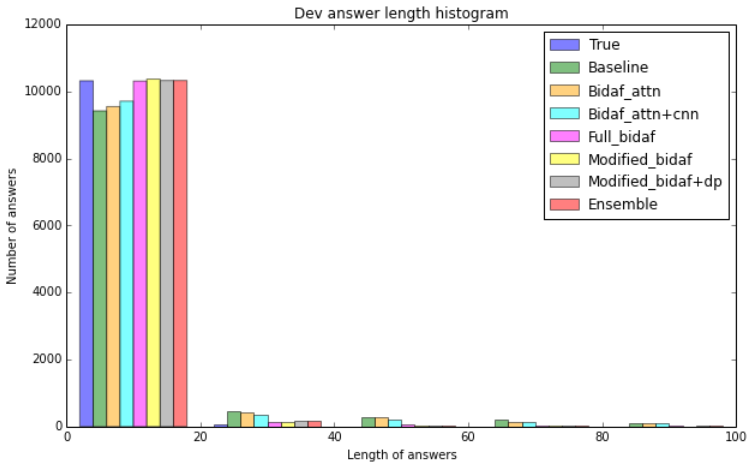

Figure 5: Answer Length Distribution

Context: `None of the original treaties establishing the European Union mention protection for fundamental rights. It was not envisaged for European Union measures, that is legislative and administrative actions by European Union institutions, to be subject to human rights. At the time the only concern was that member states should be prevented from violating human rights, hence the establishment of the European Convention on Human Rights in 1950 and the establishment of the European Court of Human Rights. The European Court of Justice recognised fundamental rights as general principle of European Union law as the need to ensure that ...`

Question: `What other entity was established at the same time as the European Convention on Human Rights?`

Answer: `European Court of Human Rights`

Answers from our models:

1. **Baseline**: european union institutions . . . in 1950 and the establishment of the european court of human rights. the european court of justice recognised fundamental rights as general principle of european union law as the need to ensure that european union measures are compatible with the human rights enshrined in member states' constitution became ever more apparent. in 1999 the european council . . . the declaration on fundamental rights produced by the european parliament
2. **Bidaf_attn**: european union institutions
3. **Bidaf_attn+cnn**: european council
4. **Full_bidaf**: european court of human rights
5. **Modif_bidaf**: european court of human rights

Explanation: The above example is a very complex one. There are multiple entities with the name "European" in them. As a result, we can see that simple models only using a fully connected layer fail on these. The models with RNNs are able to correctly identify the answer.

5 Conclusion

In this project, we reimplement and experiment with the Bi-directional Attention Flow mechanism to tackle the machine comprehension task. We compare the effects of each of the model aspects on the final performance, and analyze the predictions to obtain insights into the workings of the model. Our analysis shows that the BiDAF attention in the same baseline model improves the performance only marginally but is able to model the question to context attention in an effective manner, the character-level CNNs help model the out-of-vocabulary words and provide improved performance, especially on the numerical answers. Modeling the output layer with LSTMs provides significant boost in the performance and dynamic programming makes sure that all the generated answers are legal. With an ensemble of the above models, we are able to achieve a F1 score of **77.08** on the Dev set and **77.77** on the test set. Our model's strength lies in that it always predicts a legal answer, is robust to numerical answers, and captures word similarity between question and context very well. But it is still unable to predict the correct answer length many times. This could be a possible direction for future work. Incorporating more complex attention mechanisms such as Attention-over-Attention, or using the N-best re-ranking strategy proposed by [2] are some of the possible approaches to rectify this.

Acknowledgments

We would like to thank the CS224n course staff for making this a wonderful experience. The default project was structured in a great way and allowed us to learn much more than we could have if we had tried to do everything from scratch.

References

[1] Hu, Minghao, Yuxing Peng, and Xipeng Qiu. "Reinforced mnemonic reader for machine comprehension." CoRR, abs/1705.02798 (2017).

[2] Cui, Yiming, et al. "Attention-over-attention neural networks for reading comprehension." arXiv preprint arXiv:1607.04423 (2016).

[3] Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. "Bidirectional attention flow for machine comprehension." arXiv preprint arXiv:1611.01603 (2016).

[4] Xiong, Caiming, Victor Zhong, and Richard Socher. "Dynamic coattention networks for question answering." arXiv preprint arXiv:1611.01604 (2016).

[5] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. "Gated Self-Matching Networks for Reading Comprehension and Question Answering." ACL (2017).

[6] Junjie Ke, Yuanfang Wang, Fei Xia. "Question Answering System with Bi-Directional Attention Flow." CS224N Report (2017).

[7] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing. EMNLP (2014).