Attention Predicts "Nothing"

Stanford CS224N Default Project
Track: SQuAD

Mayank Gupta
SCPD Student
Stanford University
gmayank@stanford.edu

Samidh Pratap Singh
SCPD Student
Stanford University
samidhps@stanford.edu

Abstract

In this project we experiment with architectural improvements (embedding, attention and output layers) on BiDAF [1] baseline architecture provided to us with an eye on the performance on SQuAD 2.0 dataset [2]. Character embedding provides a large jump (6%, F1 Score) in performance whereas attention layers provide only a marginal improvement (1.5%, F1 Score) on the bi-directional attention present in the baseline. Our single best model, achieves F1 score (68.81) and EM score (65.05), and ensemble model achieves **F1 score (70.62) and EM score(67.94)** at dev set. This puts us at **6th & 8th place at dev & test leaderboards**. We find that our model's performance is driven by an increased ability to detect "no-answer" examples.

1 Key Information to include

- Mentor: Kendrick Shen
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

Question-answering, where the goal is to answer a question based on a passage, requires an understanding of the passage as well as the question, to do well. In that sense, the performance of a model in this task serves as a proxy of its ability to understand text. Also, question-answering task is central to applications such as personal assistant, search engines, dialogue systems etc. So, question-answering task is useful both in practical applications as well as a tool to benchmark natural language understanding of our models.

Question-answering is a challenging task because it requires one to capture the meaning of the "whole" question and passage. Mapping the words to word-vectors alone doesn't work as word-vectors are learnt by marginalizing the context away. So, we need to figure out a way of capture the dynamic context around the word. Also, we need to capture long-run context to "comprehend" the passage and the question. Recurrent neural nets are limited in capturing long-run context. The fact that natural language displays complicated and often ambiguous syntactic structure and semantics exacerbates these challenges.

This project aims to develop a model that does well on the SQuAD 2.0 dataset. Attention has proven to be the model of choice in contextualizing word vectors (as opposed to older RNN based models). The "moderately"-sized models which do well on the SQuAD dataset , BiDAF [1], DCN [3], RNet [4] and QANet [5], all use some form of attention. We experiment with use of co-attention and self-attention described in these models to improve the performance of the baseline model provided to us a starting point. We hope our results shed some light on why these layers improve performance on SQuAD 2.0.

We experiment with other techniques like 1-d convolution, different decoder architectures, and different training hyper-parameters.

3 Related Work

With the release of SQUAD dataset [2] by Rajpurkar, training end to end neural network with large number of parameters became easier. Bidirectional attention flow (BiDAF) model was the first of its kind to bring together question-to-context and context-to-question attentions together to build an end to end neural question answering system. At the same time, a related attention mechanism (Co-Attention) was introduced in the paper [3]. Microsoft Research introduced a passage self-attention mechanism in their R-net model [4]. In the decoder side, some researchers have focused on improving the output layer for predicting the start and end of the answer in the given passage. LSTM encoder [6] and pointer network decoder introduced by Vinyals et.al. [7] are some of these. Yu et.al (2016) [8] also proposes a dynamic chunk reader model which extracts the set of candidate answers and then rank them to predict the answer.

DCN paper introduces a novel approach for decoding by introducing a dynamic iterative decoder. The neural QA models usually have an output stage/decode that generates an answer and then stops. Authors believe that because of the single pass nature of these models, they can get stuck in a local maxima resulting in prediction of wrong answers. To solve this problem, the authors proposed an iterative approach for prediction similar to iterative conditional modes algorithm by Besag, 1986. [9].

4 Approach

We started with the baseline implementation of BiDAF, provided to us in starter code. BiDAF model has six layers, Character embedding, word embedding, contextual embedding, attention flow, modelling and output layers, as shown in figure 1. The started code has implementation for all layers except character embedding layer.

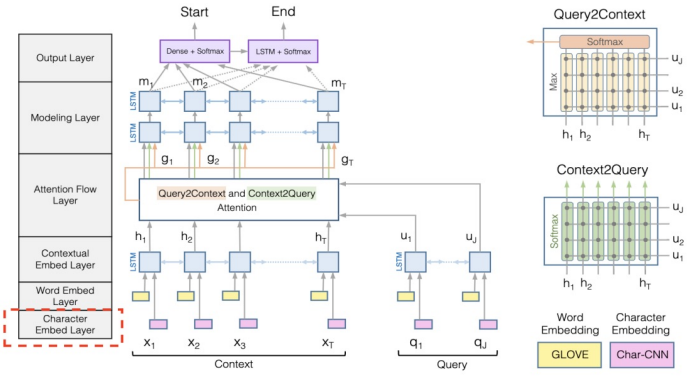

Figure 1: BiDirectional Attention Flow Model

Figure 1: Overview of BiDAF

We experimented with the following improvements on the baseline model. The code for the approaches described here, was written by us based on the descriptions provided in the papers.

Character embedding: Character embedding help with out of vocabulary words and capture the meaning of morphemes. We implemented a char embedding layer using 1d convolution filters and max pooling over convolutions output. First all characters are assigned some embedding, and then we learn the right conv-1d filters to get final character embedding of any given word. Convolutions are applied on top of character embedding of all characters of the given word. We first tried 100, 1d-filters with kernel size of 3, which gave us some improvement. After this we added

100 more filters of kernel size 5 to get even better results. This layer includes Conv1d layer, Relu, batchnorm and AdaptiveMaxPool. Batchnorm layer helped in stablizing the training and boosted the performance. We experimented with different ways of combining word and char embedding and found that fusing word and char embedding together and then projecting them in 100 dimension hidden size performs best.

Co-attention inspired by Dynamic Co-attention Network (DCN) [3]: Co-attention was described in our proposal. We implemented the co-attention with the same architecture and configuration as mentioned in DCN paper. The output of co-attention is appended to output of the bi-directional flow of attention. Unlike the original paper, we forego the use of sentinel vectors that represent a "no - answer" because BiDAF baseline already has a mechanism to generate a "no - answer " prediction.

Self-attention inspired by R-Net [4]:
Self attention helps refine the passage representation by applying attention to other words in the passage itself. We used a multiplicative attention.

Multi head attention: Taking inspiration from transformers success, we decided to experiment with multi head attentions. We experimented with four and eight headed co- and self-attention.

Iterative Decoder: BiDAF model predicts start and end probabilities once, by using the modelling and attention layer outputs. DCN paper claims that sometimes this kind of approach can result in sub-optimal answers in the passage, and if we can provide the model some more chances to iterate over previous generated answer probabilities to refine and come up with better answer, it can improve the decode/output quality. Using DCN paper results as motivation, we implemented Iterative decoder from the DCN paper, for our output layer.

5 Experiments

5.1 Data

We were provided a derived dataset of SQuAD 2.0 [2], where instead of directly using the original test data, dev set was split into roughly equal sizes of 2 datasets forming the dev and test data. This derived data, which we used for all our experiments, contains 129941 train, 6078 dev, and 5915 test examples. Each example or data point is a tuple of (c, q, a), i.e. context, question and answer. Answer is a span from the context. In squad 2.0, there is a possibility that there is no span for a given question, context pair, whenever question can not be answered using the information present in the context.

5.2 Evaluation method

We are using the same metrics, defined in the default project handout, i.e. F1, EM (Exact match) and AvNA (Answer vs No Answer). F1 and EM are the official metrics for the SQUAD 2.0 dataset leader-board as well. During training we optimize for F1 metric, and keep the checkpoints with the best F1 scores.

5.3 Experimental details

We experimented with different architecture and configurations at various layers of BiDAF.

Embedding Layer: We started with augmenting the baseline approach by adding char embedding layer along with word embedding. We used Conv-1d with filter sizes of kernel sizes 3 and 5. We used 100 filters for each kernel, and then concatenated the all filter outputs to get char embeddings. We also added BatchNorm layer inside Char embedding layer, which increased the performance by great margin.

Attention Layer: We implemented Co-attention mentioned in [3], Self passage attention [4], multi head attention mechanisms from scratch. We experimented with replacing BiDAF attention layer with these attention implementations and also experimented with different combinations of attentions used together. For all attentions we used same hidden size, and for multi attention, we tried 4, 8 heads, and found 8 heads give better performance. Overall, We experienced, BiDAF attention is really good and hard to beat, but when multiple attention architectures are combined together

they improve the performance, as each attention architecture brings its own benefits, for example multihead attention is particularly better at predicting long answers but is similar or little worse than simpler BiDAF attention on smaller answers.

Iterative Decoder: We used the modelling layer output and fed it to iterative decider layer. We kept max decode steps and max pool size to 4. For the initial state of the decoder, we first used start position as 0 and end as end of context. This didn't give better result compare to BiDAF decoder. Then, we used the older BiDAF decoder to get the seed (initial) start, end positions, and fed them to Iterative decoder in the hope, further iterations should refine the result and make predictions even better. But this didn't help with the performance.We experimented with fusing the attention and modelling layer outputs, projecting it to hidden size and then feeding it to the decoder. This too didn't result in any improvement.

Ensemble: We implemented majority voting and weighted average ensembling techniques at inference time in the hope of leveraging the complementary strengths of 10 of our best models of different architectures that we trained during our experiments. Both ensembling techniques, helped increase F1 scores by 2% points. For implementing weighted average ensembling, we calculated start and end pobabilities from all the models, and then took their weighted averages. These averaged probabilities were then passed to utils.discretize() function, to select the correct answer span. For majority voting, we first called the utils.discretize() to get start and end predictions from all the different models and then chose the (start, end) pair which is getting predicted by highest number of models. we broke the tie by randomly choosing from answer pairs, with the highest counts.

Hyper Parameters tuning: We experimented with exponential decay in learning rate and dropout of 0.15, 0.2, 0.3. We experimented with bigger hidden sizes of the network. We found that for our models the default learning rate 0.5, and dropout 0.2 works best in most cases. In one model, a lower dropout of 0.15 helped a bit, but mostly default hyper parameters were giving the best performance for our experiments.

Partial loading of parameters: Training a single model for our experiments from scratch was taking anywhere from 3hrs to 6hrs for 30 epochs of training. This was a big limiting factor in iterating quickly and validating different hypothesis's. Since we were training models that are a variation of the same architecture, we could load the layers which are previously trained rather than initializing them randomly. This enabled us to iterate fast and get better results quickly. With partially loading of previously trained models, we only need to train few epochs to converge.

5.4 Results

Here we present the F1, EM and AvNA scores achieved by different model and experiment. Baseline refers to the default BiDAF implementation provided to us. All the model names starting with plus sign (+) means, those are additional experiments/layers added on top of baseline model. All of the results presented here are on dev set, except the last Ensemble (test leader-board) model. As we had only 3 chances for test leader-board submission, we only present single best submission on test set.

As noted in the table: 5.4 char embedding gave a large (roughly 6%) F1 score improvement. Co-Attention and self-attention individually didn't improve the model, but Co-& self attention used together gave 1.5% jump in F1 score. Also, there seems to be an interaction effect of using char embedding with co-& self-attention. It gives us an added improvement over the sum of individual improvements from char-embedding and attention. Finally, using many heads further improves the attention layer, and using co- & self + multihead attention along with char embedding gives us our best single model. On adding iterative decoder on top of this best single model, doesn't result in any improvement, instead it degrade the F1 score by 2% point.
Ensembling significantly improved performance. We built an ensemble of 9 different models, trained during our experiments. The consensus answer was found by majority voting and weighted-average of probabilities. Majority voting gave a 1.7% jump while weighted average by F1 score gave a 1.8% jump on top of our single best model. On the test dataset, this model got a 69.2 F1 score, which is 1.4% less than what we got on the dev set.

Model	F1	EM	AvNA	Dev Loss
Baseline	61.29	57.84	68.01	3.08
+ char-emb	67.37	64.22	72.88	2.59
+ coattention	61.89	58.54	68.53	3.02
+ self-attention	61.49	58.36	68.19	3.04
+ co- & self-attention	62.84	59.27	69.53	2.98
+ char-emb + coattention	66.81	63.27	72.90	2.67
+ char-emb + co- & self-attention	68.23	65.17	74.09	2.51
+ char-emb + co- & self + **multihead-attention**	**68.81**	**65.05**	**74.69**	**2.57**
+ char-emb + all attentions + iterative-decoder	66.44	63.59	72.24	2.58
Ensemble (majority-voting)	70.51	**67.97**	74.95	2.63
Ensemble (weighted avg by F1)	**70.62**	67.94	**75.08**	2.63
Ensemble (test leader-board)	69.208	66.627	NA	NA

Table 1: Results of Experiments

6 Analysis

6.1 AvNA

This section analyses the model's ability to predict whether an answer to the question exists in the passage. A confusion matrix is helpful in shedding light on the performance of a classifier. Table 2 and 3 present the confusion matrices for the baseline and the ensemble model respectively.

		Truth		
		No Answer	Answer	Total
Prediction	No Answer	1733	534	2267
	Answer	1370	2314	3684
	Total	3103	2848	5951

Table 2: Confusion matrix of the baseline model

		Truth		
		No Answer	Answer	Total
Prediction	No Answer	2229	609	2838
	Answer	874	2239	3113
	Total	3103	2848	5951

Table 3: Confusion matrix of the ensemble model

We note that improvement of our model's accuracy has driven the "No Answer" column of the confusion matrix. That is, our model is much better at detecting negative i.e. the case where the passage doesn't contain the answer to the question.

6.2 Performance drill-down

We compare our model's performance by question-type, answer length and context length.

Question type is determined by the starting word of the question. For example, if the question starts with "how", it is deemed to be of type "how". If a question doesn't belong to any known category, it is grouped with "others". Our model does better than baseline on all questions except "which" questions (refer table 6.2).

Model	what	when	how	where	why	which	who	other
Baseline	0.58	0.64	0.54	0.56	0.54	0.62	0.60	0.65
Ensemble	0.68	0.73	0.63	0.65	0.60	0.59	0.68	0.75

Table 4: Performance by Question type

Model	0	1-5	6-10	11-20	>20
Baseline	0.55	0.77	0.67	0.67	0.58
Ensemble	0.71	0.79	0.71	0.70	0.58

Table 5: Performance by Answer length

In table 5, which presents the performance drill-down by answer length, we note that our model provides a large step-up in performance on answers of length 0 than other buckets. This is in line with our earlier observation that our model's improvement is primarily driven by its ability to detect "no-anwer" cases. We also note that ensemble model does-not improve on answers of length > 20.

In table 6.2, we note that the model does better at all passage lengths but the difference in performance drops in context length > 300. It seems that use of a single self-attention layer is not enough to lead to improvement over large contexts and answer lengths.

6.3 An novel ensemble

We have seen evidence that our model primarily does well at detecting cases where the question is not answered by the passage. We posit that this due to the fact that we choose the model that does the best on F1 score. Since F1 score changes the most (from 0 to 1) on correctly predicting the zero-length answers, choosing model with best F1 score chooses the model that does well on these questions.

To take the focus away from the "no-answer" examples in training, we define a loss function that assigns a zero loss when the passage doesn't answer the question. A model trained this way should complement the model trained before. An ensemble of the two should improve performance.

In figure 2, we compare the performance of two models: "Attention-4head" which is 4-head co-attention model that was trained with the conventional negative log-loss whereas the "NoAnswer" model was the same model trained with the new loss function that assigns zero to examples where the answer is not present in the passage. We note that this new model does better than the traditional model on answer of length 1-10. This evidence is reinforced by the plot of EM score by answer length (in figure 3 in Appendix).

Surprisingly, the new model does worse on answers of length > 10. We believe this due to the fact that we trained the new model for only 10 epochs (whereas the conventional model was trained for 30 epochs). Also, the effective samples on which the model is trained is reduced as the training "loss" skips a vast majority of examples (those which don't have an answer).

We build a "conditional" ensemble from the two-models. If the traditional model predicts that the answer is present in the passage, we use the modified-loss model to predict the answer. This ensemble improves the F1 score by 1.7% and EM score by 2.6% (6.3).

Model	'0-50'	'50-100'	'100-150'	'150-200'	'200-250'	'250-300'	'>300'
Baseline	0.58	0.60	0.61	0.60	0.60	0.64	0.72
Ensemble	0.69	0.69	0.70	0.69	0.71	0.74	0.76

Table 6: Performance by Context length

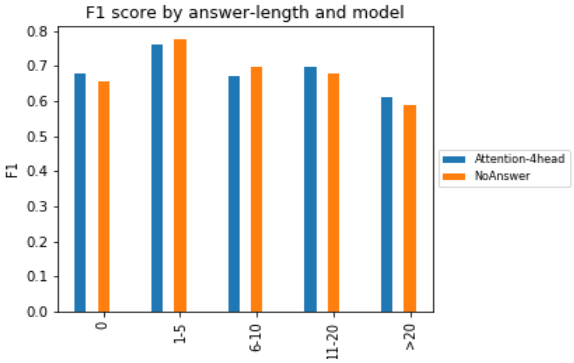

Figure 2: F1 comparison by Answer length of complementary models

Model	F1	EM	AvNA
Traditional	67.55	63.87	74.73
New-loss	66.00	63.00	72.34
Ensemble	69.22	66.54	74.63

Table 7: Answer No-answer ensemble model

6.4 Analysis of attention through examples

6.4.1 Multi-headed co-attention

In table 8, the most likely the explanation behind the four-headed co-attention model in choosing the right dynasty from the substring "the Yuan dynasty is usually considered to be the legitimate dynasty between the Song dynasty and the Ming dynasty" is because it can also focus on "including the government of the Ming dynasty which overthrew the Yuan dynasty". This alludes to multi-head attentions ability to focus on different parts of the passage when the pieces of answer are spread across the passage.

	Focus on two parts in the passage
Question	What legitimate dynasty came after the Yuan?
Partial Context	In traditional historiography of China, on the other hand, the **Yuan dynasty is usually considered to be the legitimate dynasty between the Song dynasty and the Ming dynasty.** Note, however, Yuan dynasty is traditionally often extended to cover the Mongol Empire before Kublai Khan's formal establishment of the Yuan in 1271, partly because Kublai had his grandfather Genghis Khan placed on the official record as the founder of the dynasty or Taizu (Chinese: å¤ªç¥–). Despite the traditional historiography as well as the official views (including **the government of the Ming dynasty which overthrew the Yuan dynasty**), there also exist Chinese people[who?] .. scientifically.
Gold Answers	'Ming', 'Ming dynasty', 'the Ming dynasty'
Model prediction	Ming
Baseline prediction	Song dynasty
uid	befbee0ae0a7892998e8bb798

Table 8: Multi-head attention captures different parts of the passage

6.4.2 Self-attention

In table 9, the model is able to capture the dependency between "Merit Network, Inc", "was formed in 1966", and "to help the state's educational and economic development". These strings span the breadth of a large sentence. This shows that self-attention helps capture long-run dependencies in the passage.

	Capture long-run dependencies
Question	WHy was the Merit network formed in Michigan
Partial Context	**Merit Network**, Inc., an independent non-profit 501(c)(3) corporation governed by Michigan's public universities, **was formed** in 1966 as the Michigan Educational Research Information Triad to explore computer networking between three of Michigan's public universities as a means to help **the state's educational and economic development**. With initial support from the State of Michigan ... in the mid-1980s.
Gold Answers	"as a means to help the state's educational and economic development", "to explore computer networking between three of Michigan's public universities", "explore computer networking"
Model prediction	to help the state's educational and economic development
Baseline prediction	''
uid	3a7ece7cee5d629d81afae3f0

Table 9: Self-attention captures long-run dependencies

7 Conclusion

In this project, we implemented and explored various architectural improvements in BiDAF model for SQUAD 2.0 question-answer task. After all the improvements, we achieved the best F1/EM scores of 70.62/67.94 on dev set and 69.2/66.6 on test set, which puts us in top 10 submissions on dev and test leader-boards. We learnt that character embeddings have very significant impact on performance, as they handle out of vocabulary words and capture morphemes meaning. From different attention layer experiments, we learnt that each attention has its own strength, and bringing them together can really help with the performance. From the analysis, we learnt that high portion of the improvements is coming from "No answer" type questions, because baseline model was not good at these questions to begin with and getting "No answer" question right, directly gives 0 to 1 jump in F1 score, compare to other question types where F1 score can be fractional. Also, its clear from the analysis that multi-attention is lot better at predicting long answers. Finally, we believe there is lot of room for improvement and future work for effectively utilizing attention mechanisms and effectively training models, to get better performance improvements on both "No answer" and other type of questions. Also, for future work, it might be interesting to explore better decoder/output layer architectures which might help our best model.

References

[1] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[2] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[3] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering, 2018.

[4] Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. May 2017.

[5] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018.

[6] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm, 2016.

[7] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[8] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk extraction and ranking for reading comprehension, 2016.

[9] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986.

A Appendix

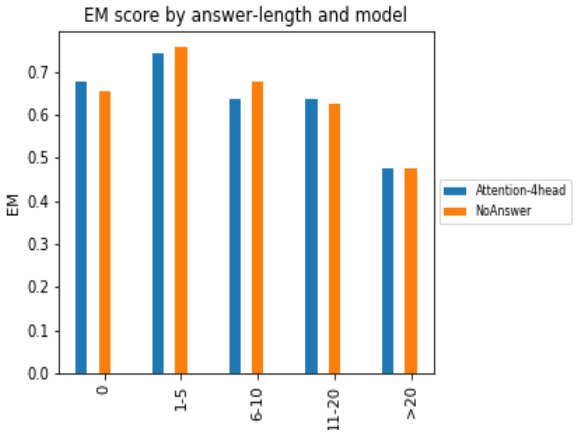

Figure 3: EM score comparison by Answer length of complementary models