Improved Question Answering on the SQuAD Dataset Using Attention Mechanisms

Vincent Sheu
Stanford Computer Science Department
Stanford University Law School
Stanford, CA 94305
vsheu@stanford.edu

Kelly Shen
Stanford Computer Science Department
Stanford, CA 94305
kshen21@stanford.edu

Abstract

Machine comprehension (MC) and question answering (QA) are NLP tasks which have seen increased interest and development with the release of the Stanford Question Answering Dataset (SQuAD). In this paper, we explore neural architectures for the QA task focused around the attention mechanisms; specifically, Bi-Directional Attention Flow and Self-Matching Attention. Our best single model achieves an F1 score of 69% and EM score of 54% on the SQuAD dev dataset.

1 Introduction

In Machine Comprehension (MC) and Question Answering (QA), machines are given a query from a corresponding piece of writing and attempts to predict a particular span of text representing the correct answer. Trained QA systems have a variety of theoretical and real-world applications-most readily apparent to the public in information retrieval via personal assistants like Alexa or Siri. This area of research has remained challenging despite an upsurge in popularity, as proposed systems must model complex question-context relations while excelling at coreference resolution, common sense reasoning, causal relations, spatiotemporal relationships, and other complex skills [1].

To motivate further research in the area, researchers in 2016 released the Stanford Question Answering Dataset (SQuAD): a training set composed of over 100,000 triples (question, context, answer) drawn from 536 Wikipedia articles, with 10,000 examples held in secret to be used as a test set [2]. SQuAD also includes a public leaderboard, on which teams can post results as compared to a crowdsourced human dataset.

Since the release of SQuAD, QA systems have improved quickly, incorporating developments from recurrent neural network architectures to develop systems that have matched (and exceeded, in recent submissions by teams at Alibaba and Microsoft) human response. We focus our project on one burgeoning topic of research: the attention mechanism.

2 Background/Related Work

The original SQuAD paper [2] proposed two basic approaches: a sliding window baseline, as well as a logistic regression model. These models achieve 20% and 51% F1 scores, respectively (compare to an F1 score of 86.8% in the human dataset). In later work, Wang et al. provided a Match-LSTM model with Answer Point architecture [3], using stacked LSTM-RNNs to encode question and context, and another RNN for probability distributions on each additional word in a proposed answer. In their work, they also determined that a boundary model (finding the first and last words of an answer) is superior to a sequence model (selecting words for an answer sequentially).

Later came Xiong et al.s work on Dynamic Coattention, providing a major foray into attention while encoding context and question [5]. Dynamic Coattention computed an affinity matrix between pairwise comparisons of words in context and question; those weights were later used to weight continuous representations of the two documents. Decoding alternatingly optimized estimates of start and end locations of answers (building on Wang et al.s work) to better handle any local minima in the search space. This paper achieved an 80.4% F1 score on the SQuAD dataset.

More recently, Seo et al. published the Bi-Directional Attention Flow (BiDAF model) [3]. This paper used an affinity matrix similar to that of Dynamic Coattention for context-to-query (C2Q) and query-to-context (Q2C) attention; then, the paper simplified previous work by passing the final context paragraph representation to an RNN and converting the resulting vectors to a score via dot product. This paper achieved an F1 score of 81.5% (approaching human performance, and third-highest submission to the public leaderboard at time of publicaion). Finally, Microsofts Natural Language Computing Group released a paper describing the R-NET model [7]. Among other features, R-NET introduced the Self-Matching Attention architecture amidst combining a BiLSTM encoding layer, C2Q self-attention layer, BiLSTM modeling layer, and fully connected output layer. The Self-Matching Attention architecture allowed for question-aware contexts to simultaneously be context aware. We explore concepts presented in these last two papers in our work.

3 Approach

We begin with the outlined baseline framework: single-layer BiGRU encoder + context-to-query attention layer + fully connected output layer. As suggested by BiDAF, we convert GRUCells to LSTMCells, and extend the baseline framework with two additional attention architectures: Bi-Directional Attention Flow (BiDAF) and Self-Matching Attention with partial R-NET output. We incorporate these architectures in various combinations outlined below. All models feed into a softmax prediction layer trained with an Adam optimizer minimizing cross-entropy loss.

3.1 BiDAF architecture

We begin by implementing the BiDAF architecture outlined in Seo et. al. [3] excluding character embeddings. This involves a BiLSTM encoding layer + context-to-query (C2Q) / query-to-context (Q2C) attention layer + two stacked BiLSTM modeling layers.

The attention layer involves:

1. Creating a similarity matrix S, which is the concatenation of contexts C, queries Q, and $C \cdot Q$ weighted by similarity vector w:

$$S_{ij} = w^T[c_i; q_j; c_i \cdot q_j]$$

2. Computing C2Q: for each context c_i, determining weights α^i_j indicating how similar c_i is to each component q_j of the question, followed by a sum over Q weighting each q_j by similarities α^i. Thus, each C2Q attention a_i reflects the similarity of context c_i across Q.

$$\alpha^i = softmax(S_{i,:})$$
$$a_i = \sum_j \alpha^i_j q_j$$

3. Computing Q2C: for each context c_i, determining weight m_i indicating the most similar component q_j of the question, followed by a sum over C weighting each c_i by its highest similarity to Q. Thus, Q2C attention c' reflects C weighted by similarity to Q.

$$m_i = \max_j S_{ij}$$
$$\beta = softmax(m)$$
$$c' = \sum_i \beta_i c_i$$

4. Concatenating the hidden context h outputted from the BiLSTM encoder, C2Q, $h\cdot$ C2Q, and $h\cdot$ Q2C, then feeding into two BiLSTM modeling layers.

The BiDAF architecture improves the baseline's attention mechanism by allowing for attention in both C2Q and Q2C directions, compared to only C2Q in the baseline.

3.2 Self-Matching Attention architecture

We continue by implementing the Self-Matching Attention architecture outlined by Natural Language Computing Group [7]. This involves a BiLSTM encoding layer + C2Q self-attention layer + BiLSTM modeling layer + fully connected output layer.

The attention layer involves:

1. Creating a similarity matrix S, which measures the similarity of each C2Q attention v_t to each other attention v_j.

$$s_j^t = w^T tanh(W_v^P v_j^P + W_v^{P^\sim} v_t^P)$$

2. For each v_t, determining weights a_i^t indicating relative similarity of v_t to attention v_i compared to all other attention v_j.

$$a_i^t = exp(s_i^t) / \sum_j exp(s_j^t)$$

3. Summing over V weighting each v_i by its similarities a_i^t. Thus, each self-similarity attention c_t reflects the relative similarity of attention v_t across V.

$$c_i = \sum_i a_i^t v_i^P$$

4. Concatenating the hidden context H outputted from the BiLSTM encoder, C2Q, and self-attention V, then feeding into the BiLSTM modeling layer.

The Self-Matching Attention architecture improves the baseline's attention mechanism by allowing for question-aware contexts to simultaneously be context aware.

3.3 R-NET output extension of Self-Matching Attention (R-NET)

We extend Self-Matching Attention by implementing the output layer outlined by Natural Language Computing Group [4]. This involves combining the BiLSTM modeling layer from Self-Matching Attention with an additional LSTM modeling layer on the hidden answer. The joint representation is then fed into a fully connected output layer.

The output layer involves:

1. Summing over hidden context H^p outputted by Self-Matching Attention, weighting each h_i^p by relative attention self-similarities a_i^t. Thus, each context-similarity attention c_t reflects the attention-context similarity of context h_i^p across H^p.

$$c_i = \sum_i a_i^t h_i^P$$

2. Passing in context-similarity attentions c_t into a LSTM trained on hidden answer representations.

$$h_t^a = RNN(h_{t-1}^a, c_t)$$

3. Adding the hidden answer LSTM H^a to the self-attention BiLSTM H^p, then feeding into a fully connected layer.

$$s_j^t = w^T tanh(W_h^P h_j^P + W_h^a h_{t-1}^a)$$

R-NET output extends Self-Matching Attention by incorporating similarity to the answer representation.

3.4 BiDAF + Self-Matching Attention feed-in architecture

We combine BiDAF with Self-Matching Attention by:

1. Feeding the BiDAF blended representation as input to the Self-Matching Attention layer.
2. Feeding the Self-Matching Attention blended representation into two BiLSTM modeling layers, followed by a fully connected output layer.

The feed-in structure overloads the self-attention layer with too many representations to attend to at once, leading to empirically worse performance.

3.5 BiDAF + Self-Matching Attention / R-NET piped architecture

We combine BiDAF with Self-Matching Attention by:

1. Running the entirety of BiDAF: Acquiring the blended representation, then feeding into two BiLSTM modeling layers.
2. Feeding the BiDAF blended representation as input to the Self-Matching Attention layer.
3. Feeding the stacked output of the BiDAF BiLSTM modeling layers and Self-Matching Attention blended representation into another BiLSTM modeling layer (+ hidden answer LSTM modeling layer for R-NET), followed by a full connected output layer.

While the piped structure exhibited superior performance to feed-in, its linear combination similarly led to sub-par performance.

3.6 BiDAF + Self-Matching Attention / R-NET parallel architecture

We combine BiDAF with Self-Matching Attention by:

1. Running the entirety of BiDAF: Acquiring the blended representation, then feeding into two BiLSTM modeling layers.
2. Running Self-Matching Attention without the final BiLSTM modeling layer to obtain the blended representation.
3. Feeding the stacked output of BiDAF and Self-Matching Attention blended representation into a BiLSTM modeling layer (+ hidden answer LSTM modeling layer for R-NET), followed by a full connected output layer.

The parallel structure effectively creates an ensemble modeling effect, combining the predictive power of two architectures.

3.7 BiDAF + R-NET combined structure

We combine BiDAF with Self-Matching Attention by:

1. Acquiring the blended representations from BiDAF and Self-Matching Attention.
2. Feeding the stacked blended representations into two BiLSTM modeling layers (+ hidden answer LSTM modeling layer for R-NET), followed by a fully connected output layer.

The combined structure allows for the predictive power of the BiLSTM to be trained on all possible attention representations simultaneously.

4 Experiments and Results

4.1 Dataset

The SQuAD dataset is composed of 107,785 training set triples (question, context, answer) drawn from 536 Wikipedia articles, with 10,000 examples held in secret to be used as a test set [2]. Answers

are crowdsourced and span a portion of the context. The SQuAD leaderboard considers two metrics for submissions: Exact Match (EM) and F1 score. EM provides the percentage of predictions that match one of the three ground truth answers exactly. F1 measures the harmonic mean between the prediction and ground truth answer (overlap). We use pre-trained 100-dimension GloVe embeddings to represent words in each of question, context, answer.

4.2 Hyperparameter Tuning

In order to better tune parameters, we ran histogram analyses on lengths, relative lengths, indices, and relative indices of questions, contexts, and answers. Salient takeaways include:

- Almost all context lengths are <400 words - this allowed us to reduce hyperparameter `context_len` from 600 to 400 without losing context representation.
- Almost all question lengths are <30 words - this allowed us to confirm that hyperparameter `question_len` set at the default 30 was reasonable.
- Almost all answer lengths are <20 words, with the majority <10 words. Almost all answers start and end before the 200th index. In addition, almost all answers are <10% of their context lengths, and <2 times their question lengths. This allowed us to improve upon finding start and end positions of the answers, further descried in Evaluation.

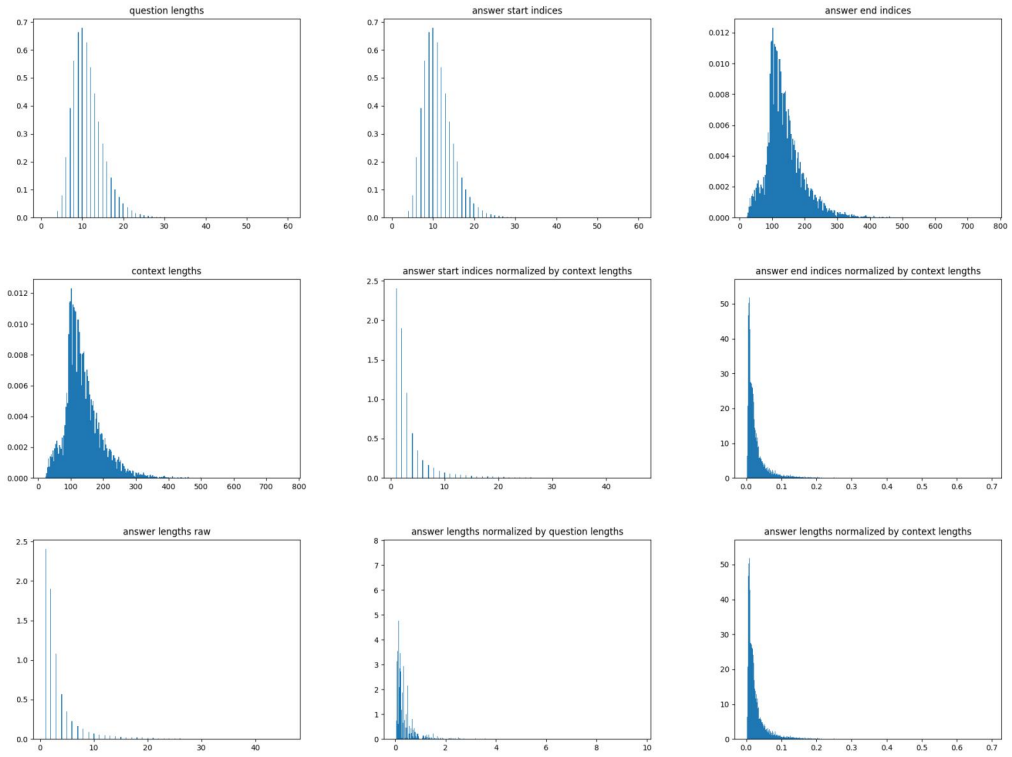

Figure 1: Histogram analyses of various hyperparameters

In addition, we ran experiments on `hidden_size` and `learning_rate`. We discovered a hidden size of 100 was optimal (compared to the default 200). We experimented with increasing learning rate to 0.005, which performed sub-optimally compared to the default 0.001. We added one new hyperparameter `self_attn_dim` that we discovered to be optimal when set to 1, representing shared weights between the various context attentions.

4.3 Experiments

We ran the following 12 experiments modeled after the various architectures described in Approach. Experiments were ran across four Mirosoft Azure servers until dev/loss began increasing which indicated overfitting. This typically occurs around 14,000 iterations into training.

1. Baseline
2. BiDAF
3. Self-Matching Attention
4. Self-Matching Attention + R-NET output
5. BiDAF feed-in Self-Matching Attention
6. BiDAF piped Self-Matching Attention
7. BiDAF parallel Self-Matching Attention (+R-NET output) (+ 0.005 learning rate)
8. BiDAF combined Self-Matching Attention + R-NET output (+ 0.005 learning rate)
9. BiDAF* combined Self-Matching Attention* + R-NET output

BiDAF* and Self-Matching Attention* indicate slight modifications to prior implementations that more closely reflect the architectures described in Approach.

4.4 Answer Window Evaluation

Both scoring metrics rely on accurately identifying the start and end positions of answers. The default implementation of `get_start_end_pos` is a simple argmax across all start and end position probabilities. We achieve 1-2% improvement on EM and F1 scores through the following method:

1. Consider the top 10 highest probability start and end positions as joint pairs (100 possible start-end pairs).
2. Check that end position comes $>=$ start position in each pair.
3. Based on empirical evidence, answers $>$15 words or starting $>$200th index or ending $>$250th index are rare. Down-weight the joint probability of a pair by 0.8 in any of these cases.
4. Choose the start-end pair with highest joint probability after possible penalization.

4.5 Results

We provide the EM and F1 scores for 8 of our 12 experiments. The scores reported are prior to improvements made to `get_start_end_pos`. We denote Self-Matching Attention as SM-A and Self-Matching Attention + R-NET output as R-NET. Four experiments were stopped early due to clearly inferior performance relative to their peers: BiDAF + SM-A feed-in, BiDAF + SM-A piped, and both experiments ran with 0.005 learning rate.

Table 1: EM and F1 dev scores across architectures

Architecture	EM	F1
Baseline	.29	.40
BiDAF	.46	.60
SM-A	.50	.65
R-NET	.50	.65
BiDAF + SM-A parallel	.50	.66
BiDAF + R-NET parallel	.52	.67
BiDAF + R-NET combined	.52	.67
BiDAF* + R-NET* combined	.54	.69

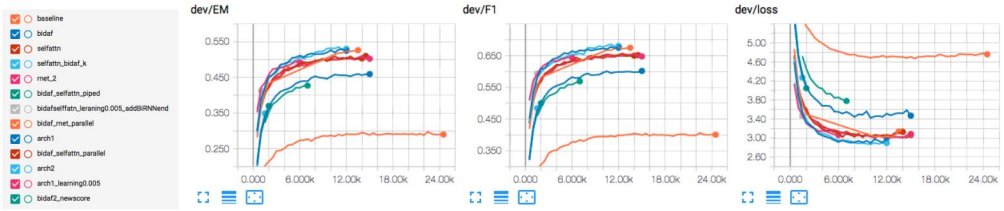

Figure 2: Tensorboard visualization of model performance on dev set

4.6 Results: Error Analysis

To further explore our models, we also randomly selected 50 examples and analyzed model errors. We classify our most common errors into the following broad categories, and provide an example of an incorrect prediction for each type of error:

4.6.1 Imprecise Answer Boundaries

- **Example: Imprecise answer boundary (predicted answer too narrow):**
  Question: despite being traditionally described as " eight counties ", how many counties does this region actually have?
  Truth: 10 counties
  Incorrect Prediction: 10
- **Example: Imprecise answer boundary (predicted answer too wide):**
  Question: What did itv increase their yearly offer for control of the rights to broadcast the primer[sic] league to ?
  Truth: 34m
  Incorrect Prediction: 34m per year

Here, the predicted answer span is slightly mismatched from ground truth. While in some cases, the prediction is not specific enough, in other cases, the prediction may be effectively equivalent to ground truth, but still be penalized by EM.

In all examples of this error, our models have found the right general answer span, but give imprecise answer boundaries to the tune of 2-3 mistakenly-included/omitted words. This distribution matches Seo et al.'s findings, which show that a plurality (50%) of their model's mistakes come from imprecise answer boundaries. These mistakes are difficult to remedy, though in some cases, they may not necessarily be mistakes at all and merely present alternate ways of presenting data (e.g. with or without an attached article; for further discussion, see 4.6.4 False Negatives Discussion).

4.6.2 Syntactic Complications and Ambiguities

- **Example: Predicted answer span has no overlap with ground truth**
  Truth: Organisation for Economic Co-opeation[sic] and Development
  Incorrect Prediction: OECD
  In cases of syntactic complications or ambiguities, the model's answer is effectively another way of answering the question; however, none of the three crowdsourced answers match the model's. Here, the predictions are effectively equivalent. This is often true, for example, for acronyms. These are also difficult problems to fix, as they stem from many possible problems (e.g. external information required to link an organization's acronym to itself). Finally, this specific example may have suffered from another issue as well: human mispellings.

- **Ground truth does not provide all possible permutations of correct answers:**
  Question: Why was NBC unable to broadcast the coronation of Queen Elizabeth II?
  Truth: flight delays, technical problems and flight delays, technical problems and flight delays

Incorrect Prediction: technical problems

In this specific case, the text makes no distinction between technical problems and flight delays as being more or less important than the other. Furthermore, the three answers indicate that giving one of the two provided causes is seen as a sufficiently reasonable answer to the question. However, the EM score would fail a response of technical problems; the F1 score would heavily punish that response as well (despite it not being any more or less correct than flight delays).

4.6.3 Complete Answer Context Mismatch

- **Example: Wrong Attention Position**

  Context: a resurgence came in the late 19th century , with the scramble for africa and major additions in asia and the middle east . the british spirit of imperialism was expressed by joseph chamberlain and lord _rosebury_ , and implemented in africa by cecil rhodes . the _pseudo-sciences_ of social darwinism and theories of race formed an ideological underpinning during this time . other influential spokesmen included lord cromer , lord curzon , general _kitchner_ , lord milner , and the writer rudyard kipling . the british empire was the largest empire that the world has ever seen both in terms of landmass and population . its power , both military and economic , remained unmatched .

  Question: by the late 19th century , which country had the largest empire ever to exist in the world ?

  Truth: the british empire

  Incorrect Prediction: africa

Originally, we found this to be one of the more common errors in the baseline (as well as one of the more fixable ones, with our focus on the attention mechanism). In these errors, an incorrect region is selected entirely. In this example (with a particularly wide context), a focus on "19th century" probably pulls the predicted answer context into the earlier passage's local minima, and the model thus has little hope of finding the right answer. We hypothesize that this is because a different region of the text entirely becomes the subject of attention; in these cases, no tuning of answer boundaries will save the model. Because our work primarily seeks to improve on the attention layer, we find many fewer of these errors in our test models (compared, e.g., to imprecise answer boundary errors).

4.6.4 False Negatives Discussion

While each of our models will miss and provide incorrect answers that are clearly wrong (by informed human intuition), we also conjecture that our models are scored as making errors that informed human intuition would not consider as incorrect in an information retrieval sense. The SQuAD dataset was created via a crowd-sourced poll, with little attention likely paid to selecting experts likely to provide optimal answers to each question. In addition, mechanical Turk-type platforms tend to provide an incentive structure that lends to its users optimizing for speed, rather than accuracy. As a result, we posit that the dataset may not be pure; that is, there are some questions for which the ground truth as determined by survey is suboptimal (e.g. multiple possible answers could be equally valid, but the dataset only considers one of those answers as correct).

5 Conclusion

Machine comprehension (MC) and question answering (QA) are NLP tasks which have seen increased interest and development with the release of the Stanford Question Answering Dataset (SQuAD). In this paper, we propose a combination of (and adjustments for) attention mechanism-focused neural architecture implementations (specifically, elements from Bi-Directional Attention Flow and R-NET/Self-Matching Attention). Our results show a marked improvement in F1 (69%) and EM scores (54%), in part due to an improvement in context window targeting. The largest remaining source of errors to tackle arises from imprecise answer matches, where the machine has located the approximate answer but fails to match the human answer exactly. We propose that this is due to a combination of fuzziness in responses, as well as a possible false negative phenomenom from a strong, but imperfect, dataset.

Acknowledgments

The authors would like to acknowledge and thank Professor Richard Socher and the teaching staff of CS224N (Winter 2018) for the invigorating and informative course. In particular, the authors thank the TA staff (especially head TAs Abigail See and Kevin Clark for their willingness to go above and beyond with (sometimes unorthodox) office hours, as well as their enthusiasm and drive to improve the class.

References

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

[2] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, Squad: 100,000+ questions for machine comprehension of text, arXiv preprint arXiv:1606.05250, 2016.

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

[4] S. Wang and J. Jiang, Machine comprehension using match-lstm and answer pointer, arXiv preprint arXiv:1608.07905, 2016.

[5] C. Xiong, V. Zhong, and R. Socher, Dynamic coattention networks for question answering, arXiv preprint arXiv:1611.01604, 2016.

[7] Natural Language Computing Group. R-NET: Machine Reading Comprehension with Self-Matching Networks, technical report, 2017.