The Impact of Attention Mechanisms on Question Answering Performance

Ben Parks
Department of Computer Science
Stanford University

bparks@stanford.edu

Joseph Paggi
Department of Computer Science
Stanford University

jpaggi@stanford.edu

Abstract

We developed a machine learning agent that answers questions about a short passage of text. We implemented a variation of the Bi-Directional Attention Flow model and were able to achieve single model performance within 2% of the original model (dev set 75.84% F1 and 66.23% EM). We found that, while most of the performance improvements came from adding additional RNN layers, the question-to-context attention led to additional improvements. Analysis of the question-to-context attention showed that it generally attended to words near the correct answer, that appear in the question, and that are the subject of their sentence.

1 Introduction

In this project, we develop a machine learning algorithm to answer questions similar to, albeit much simpler than, those you might expect to see in an SAT Critical Reading test. Specifically, we address the task of question answering, in which one is given a short passage and an associated question and asked to find the answer to the question in the text. Question answering is a difficult problem because it requires the agent to jointly consider the question and answer, both of which are expressed in unstructured natural language. How to encode dependencies between the question and context passage is the foremost challenge in question answering.

For many natural language modeling tasks, the state-of-the-art models are recurrent neural networks (RNNs) [1], a class of models that take an ordered series of inputs and model the data by maintaining a "hidden state" which is passed forwards through the series and updated as each new data point is encountered. In theory, these models could capture dependencies between the question and answer by encoding all relevant information about the question and answer in their hidden states and directly answering the question from there. However, despite representing a major advance over other methods, they still struggle to retain information in their hidden states over long distances, precluding such trivial solutions.

In end-to-end, deep learning systems, the dependencies between questions and answers are most commonly addressed through "attention-layers". These layers provide direct connections between similar parts of the question and context passage. Specifically, a standard attention layer combines each context hidden state with a weighted average of question hidden states. In this way, they inform each part of the context about relevant characteristics of the question, preventing the entirety of the question and answer from needing to be encoded in a single hidden state. Attention mechanisms principally differ by how they quantify similarity between the question and context (many of which allow the model to learn a similarity function). In this paper, we explore the performance of various attention layers in the context of question answering. Additionally, we explore the effects of two non-standard attention mechanisms: self-attention and question-to-context attention.

2 Dataset and Representations

We trained and evaluated our models on the Stanford Question Answering Dataset (SQuAD), a reading comprehension dataset consisting of questions posed by crowdworkers on a set of Wikipedia articles [2]. For each example we will refer to the article as the "context". The answers to all questions appear verbatim in the context, so the task is to predict the start and end point of the answer. The contexts, questions, and answers are generally short with median lengths of 127, 11, and 2, respectively (Figure 1). The SQuAD data set contains over 100,000 examples, which are split into training, development, and test sets with an approximately 80%, 10%, and 10% split.

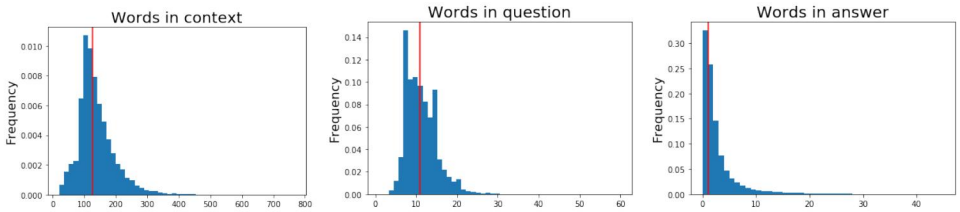

Figure 1: Number of words in contexts, questions, and answers in SQuAD training set.

Each word in the context and question were represented using 100-dimensional GloVe word vectors [3]. Rare words for which no word vector exists were all assigned the same randomly initialized embedding. In some experiments, we added an additional 'exact match' feature appended to the word vector, which was 1 if the word appears in the question and 0 otherwise. This was based on our observation that there is a strong relationship between the position of exact matches and the starting position of the answer span. As shown in Figure 2, there is a sharp spike of exact word matches with the question just before the start of the answer span, with nearly 25% of words directly before the answer span exactly matching a word from the question.

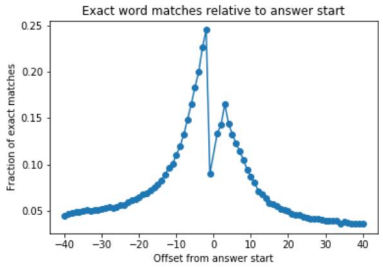

Figure 2: Frequency of exact word matches relative to answer start position

3 Model Architectures

Our models consist of stacked recurrent and attention layers, with variations in which layers are included as well as the makeup of each layer. Figure 3 shows the full sequence of model layers, although not every model we tested included every layer. At a minimum, however, each model consisted of a word embedding and RNN encoding layer to encode both the context and question. This was followed by an attention layer that combined information from the question into the context representations. After optional RNN and self-attention layers operating on the context representations, the model ran a prediction layer to output start and end positions for the answer span. Each layer and the variations implemented are described in detail below.

3.1 Embedding Layer

As described above, our embedding layer used pretrained 100-dimensional GloVe word vectors, with optional exact-match flags. In a small departure from the baseline starter code, we used a

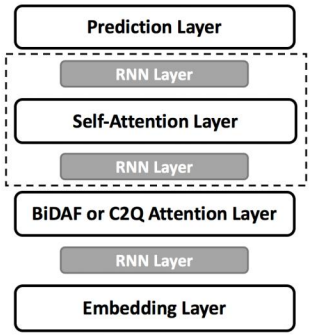

Figure 3: Model architecture overview. Dashed box indicates optional layers

constant random seed to initialize the out-of-vocabulary vector, so that it would remain constant between training and test time. For the encoding RNN after the embedding layer, we used the same weights to encode both the question and the context. Although we tested untying the weights for the encoding RNN, it just increased the number of model parameters without significantly improving performance.

3.2 Attention Layers

Since its introduction on machine translation tasks by Badahanu et al. [4], attention has become a widely used feature in neural networks for NLP. For context-to-query (C2Q) attention in our models, the attention layer takes as input a sequence of keys (the context representations) and values (the query representations). Each input key vector $\mathbf{k}_i$ is then compared to the full set of value vectors $\mathbf{v}_1 \dots \mathbf{v}_n$, and an attention vector is composed as a weighted average of the value vectors scored most similar to the input key. For C2Q attention, this provides us with a representation of the most relevant parts of the question for each position in the context. In the soft-attention model introduced by Badahanu et al., we compute a similarity matrix representing how similar each key is to each value, then use the softmax of those similarity scores to compute a weighted sum of the value vectors to obtain the attention output $\mathbf{a}_i$:

$$\mathbf{S}_{ij} = F(\mathbf{k}_i, \mathbf{v}_j)$$
$$\alpha^i = softmax(S_{i,:})$$
$$\mathbf{a}_i = \sum_{j=1}^{n} \alpha_j^i \mathbf{v}_j$$

In the context of SQuAD, attention can be used as a mechanism to combine meanings between the context and question, and self-attention has also been used successfully in models such as R-Net [5] to improve the context representations independently of the question as well. In this project, we considered context-to-question attention, self-attention, and bidirectional attention flow, as well as several options for the similarity function F.

In the baseline implementation, we used a simple context-to-question attention with dot-product similarity: $F(\mathbf{k}, \mathbf{v}) = \mathbf{k}^T\mathbf{v}$, which prioritizes relationships between close representation vectors. We also tested multiplicative attention: $F(\mathbf{k}, \mathbf{v}) = \mathbf{k}^T W \mathbf{v}$ used by Luong et al. [6], as well as additive attention: $F(\mathbf{k}, \mathbf{v}) = \mathbf{x}^T tanh(W_1\mathbf{k} + W_2\mathbf{v})$ used in the original formulation by Badahanu et al. [4]. (Unfortunately, our additive attention module used memory proportional to the product of the batch size, context length, question length, and hidden dimension, which caused out-of-memory errors for many of our experimental setups.) After each attention layer, the attention output is concatenated to the corresponding value input vector and passed to the following network layer.

3.3 Bi-Directional Attention Flow

In question answering, we want more than the access to relevant question hidden states provided through context-to-question attention—we want to know where in the context is most relevant to the

question. The question-to-context attention included in the Bi-Directional Attention Flow algorithm [7] attempts to provide this through the following procedure:

1. A similarity matrix S between the question and context hidden states is computed as described above.
2. The maximum similarity $\mathbf{m}$ between each context hidden state and any question hidden state is computed ($\mathbf{m}_i = \max_j S_{ij}$).
3. A softmax is taken over these maxima, in effect providing a distribution over the similarity of each part of the context with the question, $\beta = softmax(\mathbf{m})$.
4. A sum of context hidden states weighted by the above β distribution is computed.
5. The attention output is obtained by element-wise multiplication of the above sum and each context hidden state.

In this way, the attention output will tend to have positive elements at components of the context that are highly similar to part of the question.

3.4 Prediction Layer

In our prediction layer, we used a feed-forward ReLU layer over the context representations followed by a softmax layer to compute probability distributions for the start and end positions of the answer span. At test-time, we found the answer span maximizing $p(pos_{start}) \times p(pos_{end})$, subject to the constraint that $pos_{start} \leq pos_{end} < pos_{start} + 15$. 97.6% of answer spans in the training set fit these constraints. We also implemented a test-time prediction method that took into account the observed distribution of answer lengths from the training set by maximizing $p(pos_{start}) \times p(pos_{end}) \times p(len = pos_{end} - pos_{start})$. However, we found that this performed worse in practice than the simple length-based cutoff, and in larger models often failed to improve performance compared to naively taking the maximum of our start and end softmax distributions independently. In all but our baseline models, we used the length-based answer cutoff for evaluation.

3.5 Implementation Notes

Unless otherwise noted, each of our models used the following parameters: for RNN layers we used gated recurrent units (GRUs) as introduced by Cho et al. [8]; 200 dimensions for hidden layers; 100 dimensions for word vectors; Adam optimizer with the default parameters from the starter-code baseline model; dropout probability of 0.15, except for the models using LSTM layers where it was set to 0.2. Our minibatch size was set to 100 examples, and all models were run for at least 15k training iterations.

For improved training speed, we limited the maximum context length to 300, although we increased this to 600 at test time. We observed that the RNN layers were a major performance bottleneck, so this change nearly doubled our training speed while retaining 98.3% of our training data and resulted in no detectable decrease of eventual test-time performance.

4 Results and Discussion

4.1 Baseline Experiments

To better understand the interplay of RNN and attention layers in the SQuAD context, we conducted several experiments with variations on the baseline model. The results of these experiments are shown in Table 1. The baseline model consisted of the embedding layer, followed by an RNN and C2Q attention leading directly to the prediction layer. Because the C2Q attention used the dot-product similarity function and our encoding RNN weights were shared between the context and question, we knew it would likely score exact word matches very highly. And based on our observations detailed in Figure 2, we knew this exact match should be very prominent just before the start of the answer. However, because there were no further recurrent layers in the baseline model, it was unable to propagate this information forward from the exact match positions to the correct answer start positions.

Table 1: Baseline experiment results

Model	Dev F1	Dev F1
Original Baseline (RNN before C2Q attention)	42.42	33.30
Baseline (no attention + exact match flag)	49.12	39.39
Baseline (RNN after C2Q attention)	55.71	45.36
Double Baseline (RNN before + after C2Q attention)	68.08	57.35
Triple Baseline (1 RNN before + 2 RNNs after C2Q attention)	71.49	61.24

(Note: these predictions were taken without the answer length cutoff as described in our model architecture)

To validate this theory, we removed attention from the baseline model entirely, instead replacing it with our exact match word flags which were the only information on the question the model received. Astoundingly, this model – which was never given the full question sequence – managed to outperform the original baseline model by nearly 7 F1 points. We were able to improve on the baseline further with a less-handicapped version of the baseline in which the RNN layer was placed after rather than before the attention layer. (In this model the attention operated directly on the input word vectors). These results show the critical importance of including RNN layers after attention layers in order to distribute information from attention outputs to nearby sequence positions. Finally, we experimented with two models, termed "double baseline" and "triple baseline", which stacked additional RNNs after the original baseline's C2Q attention layer. We saw a sharp increase in performance in the "double baseline" model, with rapidly diminishing returns in the "triple baseline" model, which was ultimately outperformed by our models with more advanced attention mechanisms.

4.2 Main Model Performance

For our main model development, we initially tested several combinations of attention and recurrent layers, eventually settling on a pair of models with many similarities to the BiDAF model published by Seo et al. [7]. Our best dev-set F1 performance across all models was 76.22, and our best dev-set EM performance was 66.23, and a selection of model results is presented in Table 2.

Table 2: Selected model performance results

Model	Dev F1	Dev F1
Double Baseline (RNN before + after C2Q attention)	70.47	59.37
Double Baseline + Exact Match flag	72.86	62.40
Basic BiDAF (Double Baseline using BiDAF attention)	72.22	61.48
Basic BiDAF + Exact Match flag	73.28	62.69
LSTM BiDAF (Basic BiDAF using LSTM for RNN layers)	74.25	63.77
LSTM BiDAF + Exact Match flag	72.92	62.64
LSTM BiDAF + Multiplicative self-attention	75.31	65.07
LSTM BiDAF + 2 RNN Layers after BiDAF attention	75.34	65.04
LSTM BiDAF + Multiplicative self-attention (ensemble)	**76.22**	66.05
LSTM BiDAF + 2 RNN Layers after BiDAF attention (ensemble)	75.84	**66.23**
Seo et al. BiDAF (single model)	*77.3*	*67.7*
Seo et al. BiDAF (ensemble)	*80.7*	*72.6*

We initially started our development from the double-baseline model described in the previous section, as a lightweight and high-performance model. We then substituted the dot-product context-to-query attention layer with the more sophisticated bi-directional attention flow mechanism described in section 3.3, which provided about 2 points in F1 performance over the double baseline. We then switched our RNN layers from GRUs to LSTMs, which yielded an additional point in F1 performance.

On each of these models, we also tried adding our exact match flag. Although an exact match flag was very helpful to the simpler models, on our more complex models it actually decreased performance slightly. We hypothesize that once models reach a certain level of complexity, they can essentially recompute the exact match locations independently, so the flag just provided an extra signal to overfit onto.

In our final round of model improvements, we added an additional LSTM layer on top of our previous best model, and also experimented with inserting a multiplicative self-attention layer. (These two models are equivalent except for the optional inclusion of a self-attention layer between the final two LSTM layers). These improvements added an additional percentage point of improvement. We then extended the training runs of these models to create a majority-vote ensemble model composed of 5 checkpoints taken 500 training iterations apart. This simple ensembling approach was able to add an additional single point of improvement, although it likely suffers from incomplete independence between voting models. Our final model was very similar to the BiDAF model published by Seo et al., although our model lacked a character-CNN in the embedding layer and used a slightly simpler prediction layer. Nonetheless, we were able to come within two points of their F1 performance on a single model.

Surprisingly, the addition of self-attention layers had little impact on our model performance, and in all of our tested models it failed to give a significant performance increase compared to equivalent models with the self-attention removed. In theory, self-attention should help provide the model with a better understanding of the relationships within the context passage, and in practice the R-Net model [5] is able to achieve good SQuAD performance using self-attention. Notably, we were unable to replicate the additive similarity function used in R-Net with full-scale models due to out-of-memory errors. However, our observations may indicate that understanding the context-query relationship is a more critical function for SQuAD models than using self-attention to gain a more sophisticated understanding of the context passage.

4.3 Error Analysis

Where does our model perform best? Characteristics of the true answers had significant effects on model performance. First, we found that our model performs significantly worse when the answer contains an out-of-vocabulary word (0.51 EM) than when there are no out-of-vocabulary words (0.60 EM) (p = 2.17e-13 by chi-square test). Furthermore, model performance drastically fell with increasing length of the true answer with EM scores under 0.5 for answers longer than just 4 words.

Characteristics of the question also impacted model performance. We analyzed the dependence of performance on the presence of the words ‘what’, ‘where’, ‘when’, ‘why’, ‘how’, ‘which’, ‘who’, ‘whose’, or ‘whom’ in the question. Interestingly, we found that questions beginning with ‘what’ had lower than average performance (53.3 EM), despite being the most common type of question in the training set. This likely corresponds to ambiguity in the type of answer that is desired in a ’what’ question making them fundamentally harder questions, i.e. in the question “what year did X happen?”, we need to understand that the presence of the word ‘year’ indicates that we desire a date as an answer. Additionally, we noticed that about 1% of the questions contained none of these typical question words. These mostly consisted of questions with misspellings or questions that were actually phrased as a sentence fragment such as “ableine was retired and the new platform is called”. Our model did the worst on these atypical questions (36% EM).

4.4 Bi-Directional Attention Flow

After observing that Bi-Directional Attention Flow increased performance, we sought to understand what it was learning. Specifically, we performed analyses to uncover what the question-to-context attention was attending to. First, we asked how many positions in the context were on average being taken to account: is most of the weight on a few words or close to uniform? We found that on average 35% of the attention score is based on a single context position and the top 10 positions account for over 78% of the distribution.

Seeing that most of the attention was placed on a few words, we sought to determine properties of the highly attended to positions. We found that we attended to components of the context close to the answer, as quantified by the average attention score as a function of position relative to the

answer. Second, as is not surprising, it attended to components of the context that are exact matches to words in the question. The word with the highest attention score appeared in the question close to 90% of the time, much higher than the 15% average (Figure 4).

To test whether the attention was more dependent on the word at each position or their context, we analyzed the attention distributions at out-of-vocabulary words. Since in our representations out-of-vocabulary words are all assigned the same word vector, there is no way that out-of-vocabulary words appearing in the question could be attended to over out-of-vocabulary words not appearing in the question based solely on their word vectors. We found that out-of-vocabulary words that were assigned maximum attention weight were 4.72-fold more likely to appear in the question than other out-of-vocabulary words (p = 4.915e-4 by two-sided chi-squared test). However, this still appears to be less than optimal; of the out-of-vocabulary words that were assigned maximum attention weight only half appeared in the question. For example, when given the question "what have peridinin-type chloroplasts lost ?", the highest attention weight is placed on the word 'dinophyte' in the context phrase "the most common dinophyte chloroplast", despite "peridinin-type chloroplast" appearing several times in the context. It appears that the model was attempting to attend to an out-of-vocabulary word followed by 'chloroplast', but was unable to attend to the correct out-of-vocabulary word.

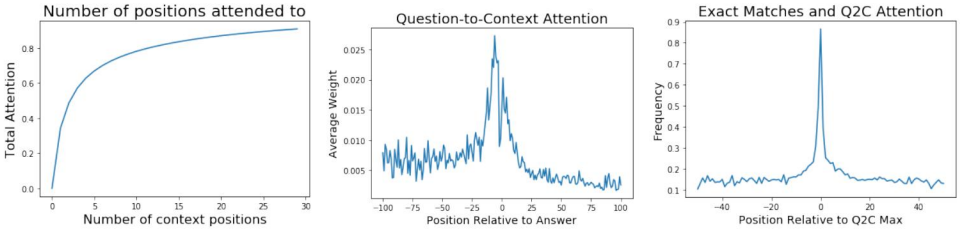

Figure 4: Analysis of question-to-context (Q2C) attention. (Left) The fraction of the total Q2C attention (y-axis) accounted for by the top k attended to positions (x-axis). (Center) The average Q2C attention value as a function of position relative to the true answer. (Right) The frequency of words appearing in the question as a function of position relative to the word receiving maximum Q2C attention.

So we know that exact matches are important, but there are a lot of exact matches and the model generally only attends to a few—how does the model choose? We observed that the model tends to attend to words in the beginning of a sentence (Figure 5). Furthermore, we found that this trend cannot be explained by the positioning of answers in sentences, which show no skew, nor positioning of exact matches in sentences, which are similarly skewed towards the beginning of sentences, but significantly more weakly. To quantify this observation, we considered whether maximum attention words appear before or after the verb in sentences containing a form of 'to be', i.e. 'is', 'are', etc.. We found that maximum attention words are 4.40-fold more likely to appear before than after the verb (p = 1.32e-70 by chi-squared test), whereas exact matches (excluding the 100 most common words) were only 1.13-fold more likely to appear before than after the verb. This suggests that the model has learned to recognize rearrangements of questions into associated sentences that could answer them, i.e. the sentence "What is X?" is likely to be answered by a sentence of the form "X is Y.".

5 Conclusion and Future Work

We found that attention, along with the arrangement of RNN layers surrounding them, was critical to our model's performance. Simply adding an RNN layer after the attention layer in the baseline model accounted for most of our performance gains. This is likely due to the fact that the position of the correct answer is indicated by words near the answer, but not in it (Figure 2). Our top performing models made use of question-to-context attention in addition to the standard context-to-question attention. We found that the question-to-context attention improved performance by attending to relatively few parts of the context that are generally close to the true answer.

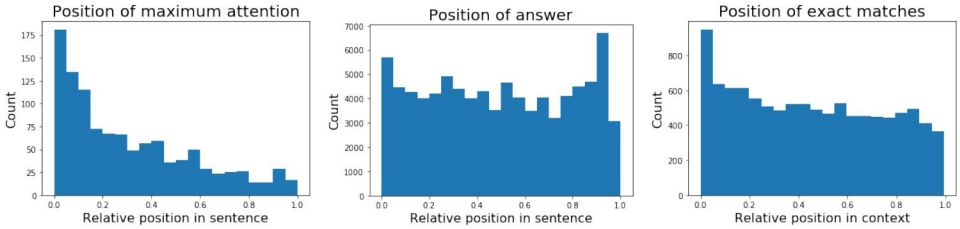

Figure 5: Where in a sentence does question-to-context (Q2C) attention attend to? (Left) The relative position of maximum attention words in their sentence. (Center) The relative position of answers in their sentence. (Right) The relative position of exact matches in their sentence.

Moving forward, the biggest issue with our models seem to be performance on questions with long answers and questions with abnormal forms. We found that even our best models were ineffective on questions whose answer was over 4 words long (EM $< 50\%$). We found that our model had trouble answering questions that were unusually worded (EM $< 40\%$).

References

[1] Zachary Chase Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015.

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[5] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[6] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

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

[8] Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, October 2014.