Exploration of Attention in Question Answering

Anthony Perez
Department of Computer Science
Stanford University
Codalab Username: aperez8
aperez8@cs.stanford.edu

Abstract

Question answering is a complex and valuable task in natural language processing and artificial intelligence. Several deep learning models having already been proposed to solve it. In this work, we propose a deep learning model with an attention mechanism that is based on a previous work and a decoder that incorporates a wide summary of the context and question. That summary includes a condensed representation of the question, a context paragraph representation previous created by the model, as well as positional question summaries created by the attention mechanism. We demonstrate that a strong attention layer allows a deep learning model to do well even on long questions and context paragraphs in addition to contributing significantly to model performance.

1 Introduction

Question Answering or Machine Comprehension is an exciting and challenging task that requires complex reasoning across sentences. The task requires a machine to answer a given question from a given passage or context paragraph. This question can be in the form of a multiple choice answer, as it was in the MCTest dataset created by Richardson et al. [7], or it can be in the form of a series of words.

Rajpurkar et al. [6] recently released the Stanford Question Answering dataset (SQuAD). This dataset has become a widely used standard for evaluating models on the question answering task. The SQuAD dataset is both large enough to train deep learning models and has a constrained answer space, because the answers are spans of the context paragraph. Rajpurkar et al. [6] also show that the dataset contains a diverse set of answers and requires reasoning that incorporates information across multiple sentences. In this paper we restrict our attention exclusively to this dataset.

After a review of the literature, we have determined that an attention mechanism that relates the question to the context is one key to performing well on the question answering task. In this work, we introduce an end-to-end neural network model for question answering. The model includes an attention mechanism roughly based on the attention mechanism proposed by Xiong et al. [10] and a decoder that incorporates a variety of representations of the context paragraphs and questions. This model is illustrated in Figure 1. We perform experiments that evaluate how well the model performs given the lengths of the context, question, and answer because we expect that a strong attention mechanism will make the model robust to long contexts and questions. We also compare two of our models to those previous proposed using the F1 and exact match (EM) scores proposed by Rajpurkar et al. [6]. While our model does not improve on the state of the art in question answer, we hope that it is useful in illustrating the power of a strong attention mechanism in dealing with long context paragraphs and long questions. We also hope to point out a possible avenue for future work.

2 Related Work

Neural network models have been shown to do well on the question answering task. Many successful neural question answer models have a strong focus on an attention mechanism which is used find the relevant sections of the context paragraph that will be used to answer the question.

Xiong et al. [10] propose the Dynamic Coattention Model, which consists of a coattentive encoder that captures the interactions between the question and the context paragraph, as well as a dynamic pointing decoder that iteratively estimates the start and end positions of the answer span. In this work, the authors construct an "affinity matrix" that measures the pair-wise similarities of the encoded words in the context paragraph and the question. Our work takes inspiration from this as is described in the subsequent sections. Xiong et al. [10] also show that a strong attention model allows the model to perform consistently even on long context paragraphs and questions and we corroborate this result.

Wang et al. [9] propose the Multi-Perspective Context Matching (MPCM) model which identifies the answer by comparing each word in the context paragraph to the entire questions, and summarizing the similarity in a variety of ways, each of which are learned by the model. We draw inspiration from Wang et al. [9] in our decoder layer, and also create a simplified model using a version of the Multi-Perspective Context Matching layer in order to draw comparisons with a more complex model.

Seo et al. [8] introduce the Bi-Directional Attention Flow model, which introduces an attention mechanism that avoids summarizing the context paragraph into a fixed length vector or coupling attention temporally. The Bi-Directional Attention Flow model uses more than just a similarity score in layers after their attention layer, as does the model proposed in this paper.

3 Approach

3.1 Task Definition

The SQuAD question answering dataset can be defined as a set of triples (Q, P, A) where $Q = (q_1, ..., q_M)$ is the question with a length of M, $P = (p_1, ..., p_N)$ is the context paragraph with a length of N, and $A = (a_s, a_e)$ is the answer span which marks the beginning and end of the answer in the context paragraph with $1 \leq a_s \leq a_e \leq N$. The question answering task can be defined as estimating $P(A|Q, P)$. In order to simplify the answer space of the problem, we adopt the convention established by Wang et al. [9] and predict $A^* = \arg\max_{1 \leq a_s \leq a_e \leq N} P(a_s|Q, P)P(a_e|Q, P)$. In other words, we predict the start and end positions of the answer independently.

3.2 Model Definition

This section defines the model used to predict the answers given the question and context paragraph $A^* = \arg\max_{1 \leq a_s \leq a_e < N} P(a_s|Q, P)P(a_e|Q, P)$. The model is best explained as a series of layers, each of which builds incrementally upon the previous layers. A visual representation of the model is available in Figure 1.

Word Embedding Layer

Word vectors have proven to be useful across a variety of natural language processing tasks. The purpose of this layer is to replace the raw word inputs with d-dimensional word embeddings. We use the GloVe word vectors with $d = 100$ proposed by Pennington et al. [5]. Xiong et al. [10] mentions that they do not train the word vectors due to the tendency to overfit. Likewise, we do not modify the word vectors.
The output of this layer are two word vector sequences. One word vector sequence is for the question $Q = [\mathbf{q}_1, ..., \mathbf{q}_M], Q \in \mathbb{R}^{M \times d}$ and another word vector sequence corresponds to the context paragraph $P = [\mathbf{p}_1, ..., \mathbf{p}_N], P \in \mathbb{R}^{N \times d}$.

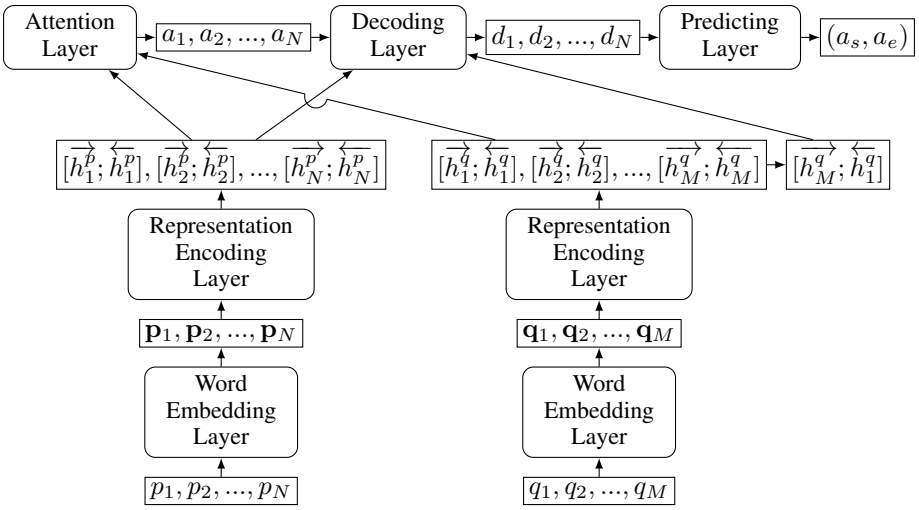

Figure 1: Model architecture.

Representation Encoding Layer

Word vectors are useful tools in general, but we would like to encode our word representation to be more specific to the question answer task and we would also like to contextualize them in the sentence.
To this end, this layer feeds the word embeddings of the previous layer into a bi-directional LSTM (BiLSTM). There are separate BiLSTMs for the question and context paragraph embeddings that do not share parameters. The outputs of the forward and backward LSTM are concatenated and that forms the output of this layer.

$$
\begin{aligned}
\overrightarrow{h_i^q} &= \overrightarrow{LSTM}(\overrightarrow{h_{i-1}^q}, \mathbf{q}_i) \\
\overleftarrow{h_i^q} &= \overleftarrow{LSTM}(\overleftarrow{h_{i+1}^q}, \mathbf{q}_i) \\
H^Q &= [[\overrightarrow{h_1^q}; \overleftarrow{h_1^q}], ..., [\overrightarrow{h_M^q}; \overleftarrow{h_M^q}]] \\
\overrightarrow{h_i^p} &= \overrightarrow{LSTM}(\overrightarrow{h_{i-1}^p}, \mathbf{p}_i) \\
\overleftarrow{h_i^p} &= \overleftarrow{LSTM}(\overleftarrow{h_{i+1}^p}, \mathbf{p}_i) \\
H^P &= [[\overrightarrow{h_1^p}; \overleftarrow{h_1^p}], ..., [\overrightarrow{h_N^p}; \overleftarrow{h_N^p}]]
\end{aligned}
$$

$H^Q \in \mathbb{R}^{M \times \ell}$ is the encoded output of this layer for the question and $H^P \in \mathbb{R}^{N \times \ell}$ is the encoded output of this layer for the context paragraph, where ℓ is twice the state size of each unidirectional LSTM.

Attention Layer

An attention model is key to being able to deal with long context paragraphs and questions. In our model this is done by computing a summary of the question for each word in the context paragraph. That summary relates to how relevant that context word is to the question. The operations performed in this layer can be seen below.

$$S = softmax(H^P(H^Q)^T) \qquad S \in \mathbb{R}^{N \times M}$$
$$C^P = [c_1^p, ..., c_N^p] = SH^Q$$
$$A = [C^P; H^P]W_A + b_a \qquad W_A \in \mathbb{R}^{2\ell \times \ell}$$

S_{ij} can be seen as the similarity between the ith word in the context and the jth word in the question. Note that the softmax is applied row-wise so that similarities are normalized with respect to the context paragraph position. S is equivalent to the affinity matrix in [10]. c_i^p can be seen as a normalized weighted average of the question vectors in H^Q. They are weighted by their similarity to the context paragraph vector at position i, which is $[\overrightarrow{h_i^p}; \overleftarrow{h_i^p}]$. C^P is then concatenated to the original context paragraph encoding and passed through a fully connected layer in order to incorporate information from the context as well as the question. The output of this layer is $A = [a_1, a_2, ..., a_N]$, $A \in \mathbb{R}^{N \times \ell}$.

Decoding Layer

Our decoding layer differs from similar layers found in works by Xiong et al. [10] and Wang et al. [9]. Rather than just use the output of the attention layer, this layer uses the output of both the attention layer and the representation encoding layer. We include more than just the attention outputs in order to allow the model to determine what is useful for question answering rather than do this by hand. The input to the decoding layer is the concatenation of A from the attention layer and H^P from the representation encoding layer, as well as the vector $[\overrightarrow{h_M^q}; \overleftarrow{h_1^q}]$, which is taken from H^Q and provides a fixed length summary of the question. The resulting input into the decoding layer is :

$$D_{in} = [[\overrightarrow{h_1^p}; \overleftarrow{h_1^p}; a_1; \overrightarrow{h_M^q}; \overleftarrow{h_1^q}], ..., [\overrightarrow{h_N^p}; \overleftarrow{h_N^p}; a_N; \overrightarrow{h_M^q}; \overleftarrow{h_1^q}]]$$

The input D_{in} is then passed through a dense or fully connected layer, a ReLU activation [4], and then a BiLSTM.

$$K = ReLU(D_{in}W_D + b_D) \qquad W_D \in \mathbb{R}^{3\ell \times \ell}$$

$$\overrightarrow{h_i^d} = \overrightarrow{LSTM}(\overrightarrow{h_{i-1}^d}, K_{i,:})$$
$$\overleftarrow{h_i^d} = \overleftarrow{LSTM}(\overleftarrow{h_{i+1}^d}, K_{i,:})$$
$$D_{out} = [[\overrightarrow{h_1^d}; \overleftarrow{h_1^d}], ..., [\overrightarrow{h_N^d}; \overleftarrow{h_N^d}]] \qquad D_{out} \in \mathbb{R}^{N \times \ell}$$

The fully connected layer is used to reduce the dimensionality of the input into the BiLSTM. This balances the number of parameters across layers in the network which forces the model to distribute labor across layers. It also reduces the capacity of the model, which helps prevent overfitting (along with dropout which will be discussed later). The BiLSTM is similar to the "Aggregation Layer" in [9]. It allows the model to include information from the surrounding positions in its encoding of the current position.

Prediction Layer

Predicting the start and end indices rather than predicting whether each position is part of the answer seems to be more successful in the literature and also in our exploration. To this end, our prediction layer is similar to the one proposed by Wang et al. [9]. We predict the start and end positions (a_s, a_e) independently by passing D_{out} through a fully connected layer then normalizing across the position dimension with a softmax.

$$P(a_s|Q, P) = softmax(D_{out}W_s + b_s) \qquad W_s \in \mathbb{R}^{\ell \times 1}$$
$$P(a_e|Q, P) = softmax(D_{out}W_e + b_e) \qquad W_e \in \mathbb{R}^{\ell \times 1}$$

4 Experiments

4.1 Implementation Details

We train the model by minimizing the sum of the cross entropy for predicting both a_s and a_e using TensorFlow [3]. We use the ADAM optimizer [2] with an initial learning rate of 0.0025 that decays by 75% every epoch and a batch size of 128. With higher learning rates, the loss would plateau early in the epoch until the learning rate decayed. All parameters are initialized using Xavier initalization [1]. The state size of all unidirectional LSTMs was 200 ($\ell = 400$), LSTM initial states are zero. Word embeddings were not train and were initialized from the GloVe word vectors [5], with out of vocabulary words initialized to zero. We do not train the word vectors to avoid overfitting, as was done by Xiong et al. [10] and Wang et al. [9]. The max context paragraph length used was 800 and the max question length used was 80 (generous with respect to the training data). At test time, if the end point is predicted to be before the start point, we set it to be equal to the start point which results in a one word answer. Word are tokenized using the NLTK python package.

4.2 Results

4.2.1 Details

We train and evaluate our model with the SQuAD dataset. This dataset includes 87,599 training examples, 10,570 dev examples and an unknown test set. We include all words in the training and dev set in our vocabulary, but do not otherwise use the dev set for training the model. This means our model may perform better on the dev set than the test set but does not otherwise affect the results because the additional vocabulary words in the dev set do not appear during training. We evaluate our model using the Exact Match (EM) score and the F1 score [6].

4.2.2 Performance on Test Set

In Table 1 we compare our model to other top performing models and to the baseline and oracle described by Rajpurkar et al. [6]. Note that all models in the table are single models and not ensembles. The model described in the Model section of this paper is labeled as "Affinity Matrix" in Table 1. We include results from another model labeled "Single Perspective", which is identical to the "Affinity Matrix" except for a difference in the attention layer. The "Single Perspective" draws inspiration from Wang et al. [9] and incorporates a variant of the Multi-Perspective Context Matching Layer (MPCM layer). The attention layer in the "Single Perspective" model is equivalent to an MPCM layer except that it shares the matrix $\mathbf{W}$ across the three types of matching, it concatenates the forward and backwards outputs from the previous layer rather than splitting them, and $\mathbf{W} \in \mathbb{R}^{1\times\ell}$ rather than $\mathbf{W} \in \mathbb{R}^{l\times\ell}$. Sadly, neither of our models out performs the other models presented in Table 1. However, the "Affinity Matrix" model out performs both the "Single Perspective" model and the baseline model. This indicates that the inclusion of a stronger attention model dramatically increased the performance of our model.

4.2.3 Attention Model Analysis

Figure 2 compares the model performance across the different context paragraph lengths, question lengths, and answer lengths on the held out dev set. One may expect that the model performance would decrease as either the context paragraph length or question length increases because the BiLSTMs in the model would be unable to capture the long range dependencies of language. However, Xiong et al. [10] have shown that with a powerful enough attention model, performance does not decrease as context or question length increases. We have demonstrated in Figure 2 that our model performance also does not decrease as context or question length increases which indicates that our attention layer contributes significantly to our performance.

5 Future Work

In both our analysis and the analysis done by Xiong et al. [10], it was found that performance decreases as answer length increases. Future work may be in improving the model decoder or including additional information in the model that would allow it to answer question with longer answers. Question with long answer may also require more complex forms of reasoning and this is something to explore.

Table 1: Model results and comparison. All models are single models, not ensembles. Scores are reported as they were by the model authors. An '-' indicates that the score was unavailable. The Affinity Matrix model is described in the model section, while the Single Perspective model is described in the Results section. We include the result of the Single Perspective model as a type of ablative analysis as it features a weaker attention mechanism.

Model	Dev EM	Dev F1	Test EM	Test F1
Baseline [6]	40.0	51.0	40.4	51.0
MPCM [9]	-	-	65.5	75.1
DCN [10]	65.4	75.6	66.2	75.9
BiDAF [8]	-	-	68.0	77.3
Single Perspective (Ours)	30.6	43.0	-	-
Affinity Matrix (Ours)	58.3	69.6	49.3	62.6
Human [6]	81.4	91.0	82.3	91.2

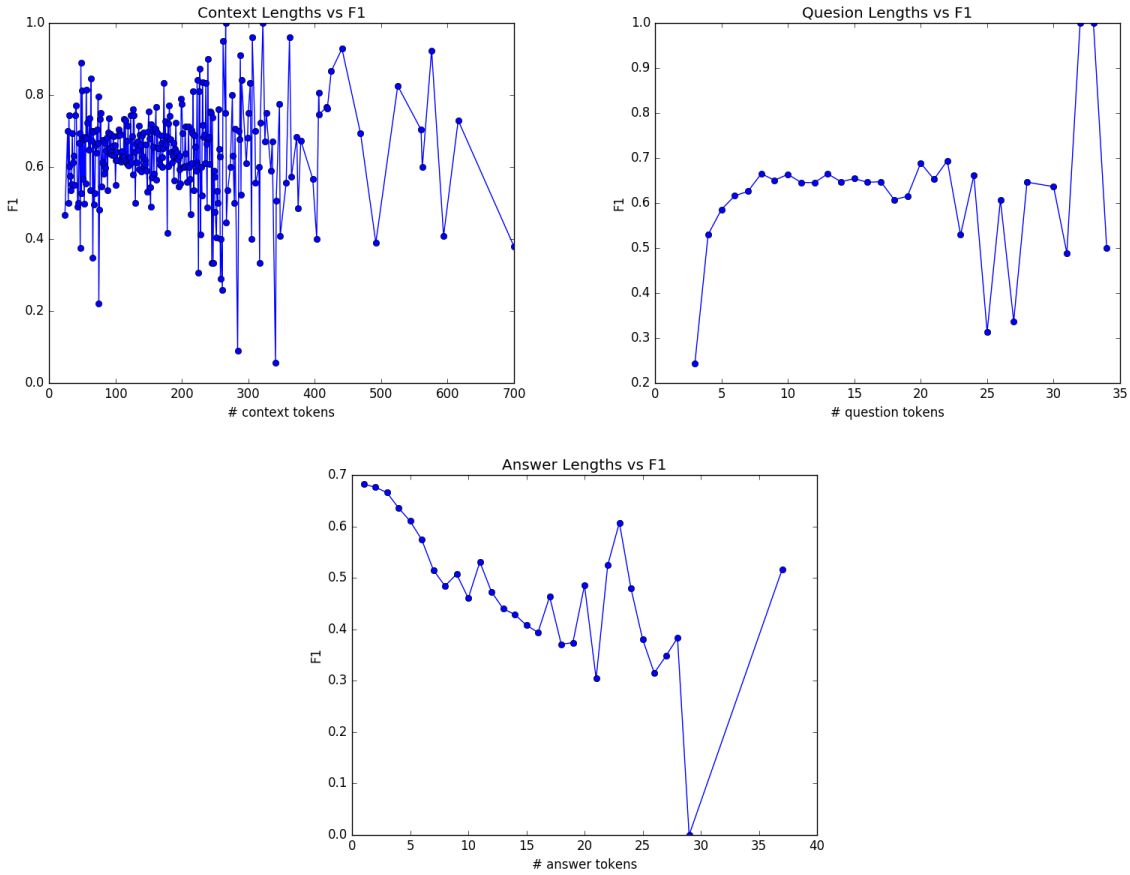

Figure 2: The lengths of the context paragraph, the question, and the answer, in number of tokens plotted against the F1 score. The results are averaged for each length and are taken from the dev set. Due to lack of number of examples, the F1 score fluctuates for longer lengths. The F1 score generally fluctuates around 0.7 across the context paragraph lengths and 0.67 across the question lengths. The F1 score does not decrease as the context and question lengths increase, but does decrease as answer lengths increase. This result indicates the attention model is doing well.

6 Conclusion

In this work, we propose a model for the question answering or machine comprehension task. Our model encodes the context and question sentences, then uses an attention model and decoder to predict the start and end points of the answer span. Analysis shows that our attention model was able to capture long term dependencies in the context paragraph and question and contributed significantly to the performance of our model.

References

[1] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*. 2010.

[2] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.

[3] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[4] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Ed. by Johannes Furnkranz and Thorsten Joachims. Omnipress, 2010, pp. 807–814. URL: http://www.icml2010.org/papers/432.pdf.

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[6] Pranav Rajpurkar et al. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250 (2016). URL: http://arxiv.org/abs/1606.05250.

[7] Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. *MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text*.

[8] Min Joon Seo et al. "Bidirectional Attention Flow for Machine Comprehension". In: *CoRR* abs/1611.01603 (2016). URL: http://arxiv.org/abs/1611.01603.

[9] Zhiguo Wang et al. "Multi-Perspective Context Matching for Machine Comprehension". In: *CoRR* abs/1612.04211 (2016). URL: http://arxiv.org/abs/1612.04211.

[10] Caiming Xiong, Victor Zhong, and Richard Socher. "Dynamic Coattention Networks For Question Answering". In: *CoRR* abs/1611.01604 (2016). URL: http://arxiv.org/abs/1611.01604.