# Recurrent Neural Networks with Attention for Question Answering

Benjamin Hannel
Department of Computer Science
Stanford University
Stanford, CA, 94305
bhannel@stanford.edu

March 21, 2018

**Abstract**

Effective reading comprehension systems have great promise to make vast databases of text more accessible. I have created a model that finds the answer to a question within a passage using recurrent neural networks with long short term memory, and a basic attention layer to place more weight on sections of the question which are relevant to each part of the context. With a single model, I achieved an F1 score of 77.235 and an exact match score of 67.985 on the Stanford question answering development dataset (SQuAD). With an ensemble model, I reached an F1 of 78.984 and an exact match score of 69.612.

## 1 Introduction

Massive amounts of unstructured real world knowledge are difficult to search when trying to answer a specific question. It is likely that the answer is available, but the shear volume of material makes it difficult to search. Extractive question answering promises to provide a tool that can process a question and select not just to a page or passage, but an individual span of text containing the answer.

In formal terms, a machine reading algorithm is presented with a question composed of tokens $q_0, q_1, ..., q_n$ and a context also composed of a sequence of tokens $c_0, c_1, ..., c_m$ which contains the answer to the given question. The goal of the algorithm is to correctly identify a contiguous subset of the context $c_k, c_{k+1}, ...c_{k+l}$ which answers the question.

I approached the problem using a neural architecture composed of a bidirectional LSTM encoder layer, followed by a basic attention mechanism to adaptively summarize the question, and two more bidirectional LSTMs to take advantage of contextual and attentional information. The output layer is a two simple independent softmaxes over the context, one to identify the start position of the answer, the other to identify the end. The model performs remarkably well with a very basic attention mechanism, so I have dubbed it basic attention flow, or BasicAF.

## 2 Related Work

There have been a series of state of the art models for question answering published in 2015 and 2016, enabled by SQuAD. Often, these models begin with an RNN encoder layer of the question and passage, followed by some kind of attention mechanism, followed by more RNN layers and a final prediction layer.

### 2.1 Bidirectional Attention Flow

The bidirectional attention flow (BiDAF) model achieved state of the art results on SQuAD through the use of a novel attention mechanism (Seo et al., 2015). BiDAF first extracts character level, word level, and then contextual level (multiple words). The layered extraction at different levels of granularity is reminiscent of convolutional techniques used in image processing. Next comes the bidirectional flow layer for which the model is named, which adaptively summarizes the question and context at each time step and blends them into a representation of that context token. The blended representation are passed through two more biLSTMs, the modeling layers. The start position is predicted by a fully connected layer and a softmax while the end position is conditioned on the start via another biLSTM.

### 2.2 DrQA

The DrQA question answering system achieves state of the art performance with a relatively simple model (Chen et al., 2017). The following engineered features are concatenated to each context token's GLoVe word embedding (Pennington et al., 2014).

1. Does that word also appear in the question, in exact or lemmatized form?
2. What is the part of speech of the context word?
3. Is the context word a named entity?
4. What is the normalized term frequency of the token?
5. How similar is this context word's embedding to the embedding of the question words?

They apply a stacked biLSTM to arrive at a representation of each context word. The question is summarized into a single vector. The unnormalized probability distribution of the start and end of the answer span is computed via a multiplicative attention layer with exponential nonlinearity.

## 3 Approach

The BasicAF model consists of 7 steps to process an input (a question and context) and produce a span of text as the answer.

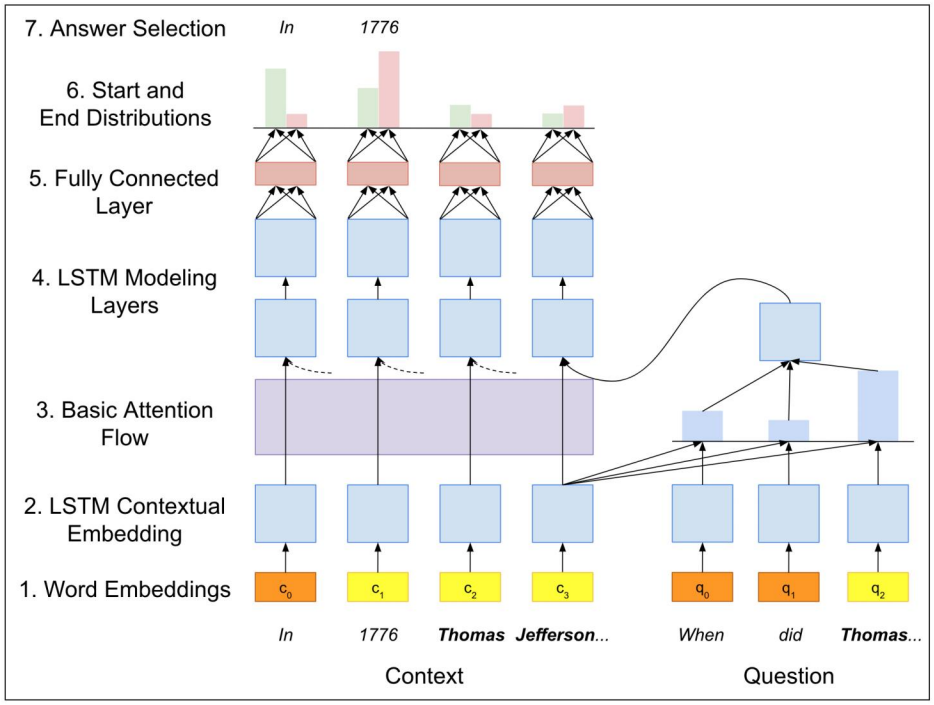


Figure 1: The architecture of the BasicAF model.

## 3.1 Word Embedding

The word embedding layer uses GLoVe vectors to extract word meaning (Pennington et al., 2014). The word vectors of the 1000 most common tokens are fine tuned during training (orange), while the others are held fixed (yellow). Like DrQA I add two new features to each token in the question and context; $f_{exact\_match}$ and $f_{lemma\_match}$. Exact match is the number of tokens in the question which are identical (neglecting case) to the given token in the context. Lemma match is defined likewise, except it counts number of words with equivalent lemmatized forms.

$$\tilde{\mathbf{q}}_\mathbf{i} \in \mathbb{R}^{d+2} = \mathbf{E}(\mathbf{q_i})||f_{exact\_match}||f_{lemma\_match}$$

$$\tilde{\mathbf{c}}_\mathbf{i} \in \mathbb{R}^{d+2} = \mathbf{E}(\mathbf{c_i})||f_{exact\_match}||f_{lemma\_match}$$

$\mathbf{E}(\mathbf{c_i})$ represents looking up the embedding of word $i$, and $||$ represents concatenation. $d$ is the dimension of the word embeddings, for which I used 100.

### 3.1.1 Engineered Features

While feature engineering is generally unscalable in natural language processing, it has the power to convey priors to the model which it may be slow or unable to learn on

its own. In this case, it produced a substantial increase in performance, especially with limited training time. After 2000 iterations, the model including the exact match feature had a 14 percentage point higher F1 score than the model without. When training had stabilized at 20,000 iterations, the difference narrowed to 7 percentage points.

## 3.2 Contextual Embedding

The contextual layer applies a biLSTM to the word embeddings for the question ($\tilde{\mathbf{q}}$) and context ($\tilde{\mathbf{c}}$) to produce more precise meanings of words in the question and answer based on their context. While the context and question are processed independently by the biLSTM, the weights are shared. The output of the forward and backward directions of the LSTM are concatenated to form vectors $\hat{\mathbf{q}}_\mathbf{i}, \hat{\mathbf{c}}_\mathbf{i} \in \mathbb{R}^{2h}$. $h$ is the hidden state size, for which I used 100.

## 3.3 Basic Attention Flow

The each context token attends to each word in the question via a basic dot product attention ($\mathbf{e}_i$) and softmax normalization ($\alpha_i$). The question is adaptively summarized with respect to each context token via a weighted average ($\mathbf{a}_i$). This summary is concatenated with the context token representation to form the blended representations $\mathbf{b}_i$, which form the columns of $\mathbf{B}$. I used this layer directly from the project handout. I experimented extensively with a bidirectional attention flow layer, but found the basic attention layer to perform better.

$$\mathbf{e}_i = [\mathbf{c}_i^T, \mathbf{q}_1, ..., \mathbf{c}_i^T, \mathbf{q}_M]$$

$$\alpha_i = softmax(\mathbf{e}_i)$$

$$\mathbf{a}_i = \sum_{j=1}^{M} \alpha_i^j \mathbf{q}_j$$

## 3.4 LSTM Modeling Layers

The modeling layers consist of two more biLSTMs, which further refine the blended representations to determine which words are most pertinent to the question.

$$\mathbf{H} \in \mathbb{R}^{2h \times l} = biLSTM(biLSTM(\mathbf{B}))$$

## 3.5 Fully Connected Layer

The output of the modeling layer is concatenated with the output of the attention layer and goes through a fully connected layer with ReLU nonlinearity. This layer was in the project handout but not the BiDAF model nor the DrQA model, but I found it increased performance in this case.

$$\mathbf{F} \in \mathbb{R}^{h \times l} = ReLU(W_F(\mathbf{H}||\mathbf{G}))$$

$$W_F \in \mathbb{R}^{h \times 6h}$$

### 3.6 Start and End Distribution

The output of the fully connected layer for each context word is multiplied by a trainable weight vector $\mathbf{w_{start}} \in R^h$ to compress each word in the context to a single value. I normalize the values to a distribution with a softmax.

$$\mathbf{p_{start}} = softmax(\mathbf{w_{start}}\mathbf{F})$$

$$\mathbf{p_{end}} = softmax(\mathbf{w_{end}}\mathbf{F})$$

For training, I take the cross entropy loss between $\mathbf{p_{start}}$ and the start position of the human answer and add that to the cross entropy for the end position.

### 3.7 Answer Selection

At test time, I pick the (start, end) pair which maximizes

$$\mathbf{p_{start}}(i)\mathbf{p_{end}}(j)\mathbf{p_{length}}(j-i)$$

where $\mathbf{p_{length}}$ is the empirical distribution of lengths in the training data, raised to the 0.35 power[1]. This encourages the model to pick answers with reasonable length, generally 1 to 10.

### 3.8 Ensemble

I created an ensemble model using four of the models described above, trained with slightly different hyperparameters to improve diversity. I also used one model in the ensemble which used bidirectional attention flow instead of basic attention flow. While bidirectional attention flow performed worse by about 2 points as a single model, it caused the ensemble to perform about 1 point better, likely because it added diversity. The ensembles each cast one vote, and the majority is used. In the event of a tie, the candidate span with the highest degree of overlap with other spans is used.

## 4 Failed Approaches[2]

Developing a neural architecture is very much a trial and error process. Here is a brief summary of approaches that did not work well.

1. I thought that it would help the model to understand that the start and end were endpoints of a range. To this end, I trained it to predict an output distribution for the interior of the span, and added the cross entropy loss of this distribution to the start and end loss. The resulting model performed no better or worse than the original.

[1]The exponent "softens" the distribution, because the conditional probability the given length is more uniform given that the model is predicting that start and end.

[2]While it is not standard to discuss failed approaches in a paper about a new neural architecture, I think it should be. Trying to learn from other people's experiences while only seeing their successes is like trying to learn the boundary for a decision problem with only positive examples. Machine learning experts, of all people, should understand why negative results are important.

2. In the BiDAF paper, they used a final LSTM layer for only the prediction of the end token of the answer. I added this layer but saw no improvement in scores and slower training.

3. When I increased the number of modeling layers from 2 to 4, performance decreased substantially.

4. I tried making the length distribution $\mathbf{p_{length}}$ trainable so the model was end-to-end, but the performance dropped.

# 5 Experiments

## 5.1 Dataset

The examples for BiDAF came from the Stanford question answering dataset (SQuAD), which contains over 100,000 question-answer pairs gathered from Wikipedia (Rajpurkar et al., 2016). The data was partitioned into a training set of 86,326 question-answer pairs, a development set of 10,391 pairs for hyperparameter tuning, and a test set against which I only ran the model once for the final result. The development and test sets have multiple human generated answers per question so that performance can be measured more consistently when there are multiple correct answers.

## 5.2 Preprocessing

All contexts and questions are initialized with the default Python NLTK word tokenizer. However, if a token is found to be outside of the vocabulary, I first try to standardize unicode characters, remove quotes, and perform other text wrangling trick to try to determine the word really intended. This reduced unknown tokens substantially and produced a 0.4 improvement in scores.

## 5.3 Training Details

I trained BasicAF using TensorFlow running on the NV12[3] in the Microsoft Azure cloud. It took approximately 20 hours of training for performance to converge. It used the Adam optimizer with a learning rate of 0.001 and 25 samples per batch. Once performance stabilized, I continued training the network at learning rate 0.0003 and 50 samples per batch. In order to allow for unrolling of the LSTMs in the computational graph, questions were clipped after 25 words and the contexts were clipped after 300 words during training. At test time, questions were clipped to 30 words and contexts to 600 words. This accommodated virtually all inputs without clipping.

## 5.4 Results

I measured the success of BasicAF's by two metrics, the exact match (EM) score and the F1 score. Exact match simply calculates the percentage of all answers which the

[3]Tesla M60 GPU.

| | Single Model | | Ensemble | |
|---|---|---|---|---|
| Model | F1 | EM | F1 | EM |
| Human | 91.2 | 82.3 | N/A | N/A |
| R-net | 80.6 | 72.3 | 83.7 | 76.7 |
| DrQA | 78.8 | 69.5 | N/A | N/A |
| BiDAF | 77.3 | 68.0 | 81.5 | 73.7 |
| BasicAF | 77.2 | 68.0 | 79.0 | 69.6 |

Figure 2: A comparison of BasicAF's performance to the state of the art models on the SQuAD dev set. Metrics for DrQA and the BiDAF ensemble are on the test set, because dev set metrics were unavailable.

model produced the exact same span as the human answer. The F1 score gives partial credit for answers which overlap the correct answer and penalizes for answers which contain words not in the correct answer ($F_1 = \frac{2PR}{P+R}$). For both metrics, the human answer most similar to the model answer is used for comparison.

| Question Type | F1 | EM | Count |
|---|---|---|---|
| who | 80.8 | 74.5 | 1368 |
| what | 77.3 | 67.1 | 6065 |
| when | 87.9 | 82.8 | 862 |
| which | 79.3 | 69.9 | 747 |
| how | 80.2 | 70.8 | 1241 |
| other | 74.8 | 60.6 | 716 |

(a) Breakdown by type of question.

| Answer Length | F1 | EM | Count |
|---|---|---|---|
| 0-3 | 80.6 | 73.5 | 8850 |
| 4-7 | 74.7 | 59.0 | 1994 |
| 8-11 | 68.0 | 43.5 | 474 |
| 12-15 | 62.5 | 36.7 | 196 |
| 16-19 | 64.2 | 29.2 | 72 |
| 20+ | 55.7 | 31.4 | 51 |

(b) Breakdown by length of the true answer

Figure 3: An analysis of the performance of BasicAF across different subsets of the SQuAD dev set.

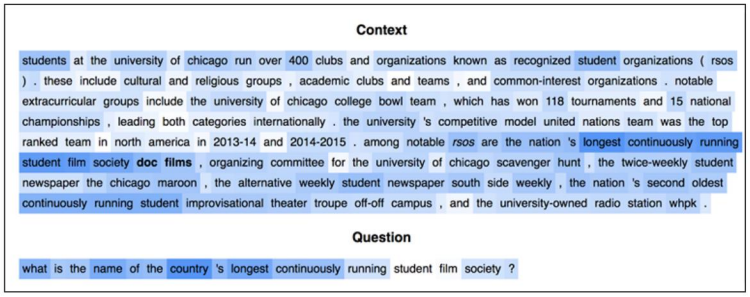


Figure 4: This is the attention distribution over a sample question from BasicAF with bidirectional attention. The correct answer is "doc films" but BasicAF predicted "rsos". Colors are derived from row-wise and column-wise averaging of the attention matrix. This class of error, where the model picks a span just on the wrong side of an overlapping phrase, is common.

# 6 Conclusion

In the BasicAF model, I synthesized the results of earlier work to produce a high performing model with a relatively simple architecture. Attention mechanisms have the potential to allow information to pass through a neural network more easily, even after a large number of intervening tokens. They are clearly a key component of any high performing model on this task. However, the impact of engineered features like exact match and lemma match should not be underestimated. Particularly in natural language processing, humans understand powerful prior patterns which can provide substantial gains for models without increasing the parameter space or training time. It is worth noting that BasicAF achieves competitive performance with less than 1 million parameters.

While I tried many of the modifications to BasicAF that I could think of, there are many which I did not have a chance to test. Future researchers could take advantage of semi-supervised approaches by replacing the contextual embedding layer with CoVe (McCann et al., 2017) or ELMo (Peters et al., 2018) to take advantage of richer language understanding than word vectors can capture. While I chose not to implement the character embedding layer of BiDAF, adding it to the model would likely yield a benefit of about 2 percentage points in the F1 score, as shown in the ablation analysis of the BiDAF model.

# 7 References

[1] Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017). *Reading wikipedia to answer open-domain questions.* arXiv preprint arXiv:1704.00051.

[2] McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). *Learned in translation: Contextualized word vectors.* In Advances in Neural Information Processing Systems (pp. 6297-6308).

[3] Pennington, J., Socher, R., & Manning, C. (2014). *Glove: Global vectors for word representation.* In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[4] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep contextualized word representations.* arXiv preprint arXiv:1802.05365.

[5] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). *Squad: 100,000+ questions for machine comprehension of text.* arXiv preprint arXiv:1606.05250.

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