Applying Bi-Directional Attention Flow to SQuAD

Jialin Ding
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
jding09@stanford.edu

Jestin Ma
Department of Computer Science
Stanford University
Stanford, CA 94305
jestinm@stanford.edu

Abstract

Question answering is one of the core challenges in natural language processing. Efforts to improve question answering models have been aided by both the recent trend towards using deep neural networks and the release of extensive datasets such as SQuAD. SQuAD serves as a benchmark for reading comprehension: given a paragraph and a question about that paragraph, can a system 'understand' the text and provide a correct answer? In this paper, we present a neural network model that incorporates state-of-the-art techniques such as bi-directional attention, character CNN embeddings and highway networks. Our single model achieves competitive performance on the SQuAD dataset, yielding an F1 score of 77.023 and an EM score of 67.072 during testing.

1 Introduction

Question answering (QA) is a key challenge in the field of natural language processing (NLP). QA systems aim to answer questions posed in the form of natural language, often doing so by making using of some provided document collection. These systems can potentially serve as customer service chatbots or personal assistants.

Recently, two key developments have aided in the effort towards better QA systems. First, deep learning has revolutionized the field of computer science and has already been used to rapidly make significant progress on difficult problems in natural language processing without the need for painstaking feature engineering. Second, the release of large-scale labeled datasets, such as the Stanford Question Answering Dataset (SQuAD) [1], has served as a common benchmark for new QA models.

In this paper, we integrate state-of-the-art techniques in NLP and deep learning into a question answering model that performs competitively on the SQuAD dataset. Our model consists of a pipeline of layers: an embedding layer that represents tokens (words) at the word and character level; a filtering layer implementing a Highway Network [4]; a encoding layer; an attention layer implementing bidirectional attention flow (BiDAF) [2]; and an output layer.

The rest of this paper is structured as follows: Section 2 gives background on the task and introduces our pre-training analysis of the SQuAD dataset and resulting modelling decisions; Section 3 presents our model in depth; Section 4 evaluates our model on the SQuAD train/dev/test sets; Section 5 analyzes and visualizes our model's performance and errors; and Section 6 concludes with future work and gleaned takeaways.

2 Background

The SQuAD dataset is a reading comprehension dataset. The dataset contains over 100,000 entries. Each entry contains a context, which is a paragraph pulled from a Wikipedia page; a question, which

asks about something in the context; and an answer, which is pulled directly from the context, so that the QA system does not need to generate any new text. Figure 1 shows a sample context-question-answer triple. Since it was released by the Stanford NLP group in 2016, many state-of-the-art models have been submitted to the leaderboard [5]. The top models incorporate some complex attention mechanism to focus on certain portions of the question and context, and some models incorporate pointer networks to favorably permute inputs. These models include BiDAF [2], which inspires much of our model; Dynamic Coattention Network [6] and R-Net [7], which use more advanced forms of attention; and Match-LSTM [8], which conditions end prediction on start prediction. In early 2018, the first model to outperform humans appeared.

Question: Why was Tesla returned to Gospic?
Context paragraph: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke). During that year, Tesla taught a large class of students in his old school, Higher Real Gymnasium, in Gospic.
Answer: not having a residence permit

Figure 1: Sample SQuAD question.

2.1 Dataset characteristics

In order to better understand the dataset, we first analyzed some characteristics of the training set. Figure 2 shows the histograms of lengths, in words, of contexts, questions and answers. We also examined the distribution of lengths for words, using the training set contexts, as shown in Figure 3.

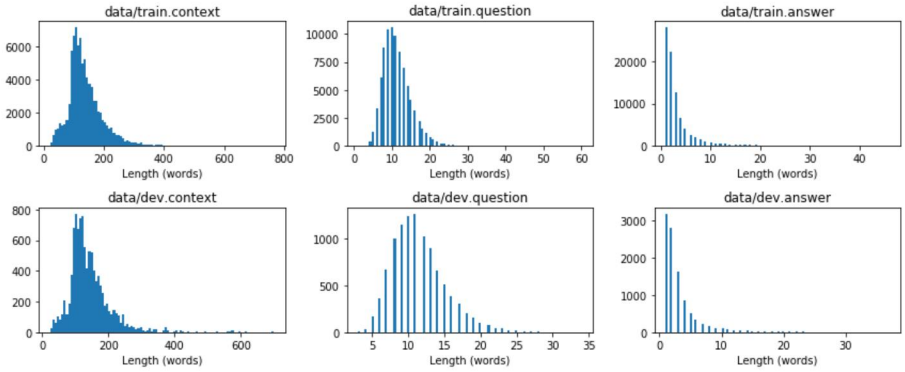


Figure 2: Distribution of lengths of contexts, questions and answers on the SQuAD training set.

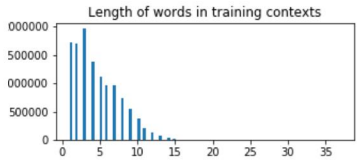


Figure 3: Distribution of lengths of words in the SQuAD training set contexts.

A cursory analysis of the training and dev set contexts, questions, and answers reveals several characteristics and decisions regarding the dataset:

- Nearly all contexts in both train and dev partitions are no more than 400 words long. We opted for a maximum context length of 420 to reduce training time and traded off some context tokens at the tail end of contexts.
- Nearly all questions in both train and dev partitions are no more than 30 words long. We kept the maximum question length of 30.

- Nearly all answers in both train and dev partitions are no more than 20 words long. When predicting the start and end points of an answer, we impose an answer length limit of 15 tokens to prevent our model from incorrectly predicting a lengthy answer.
- Nearly all words in contexts are no more than 15 words long. We opt for a maximum word length of 15 characters.

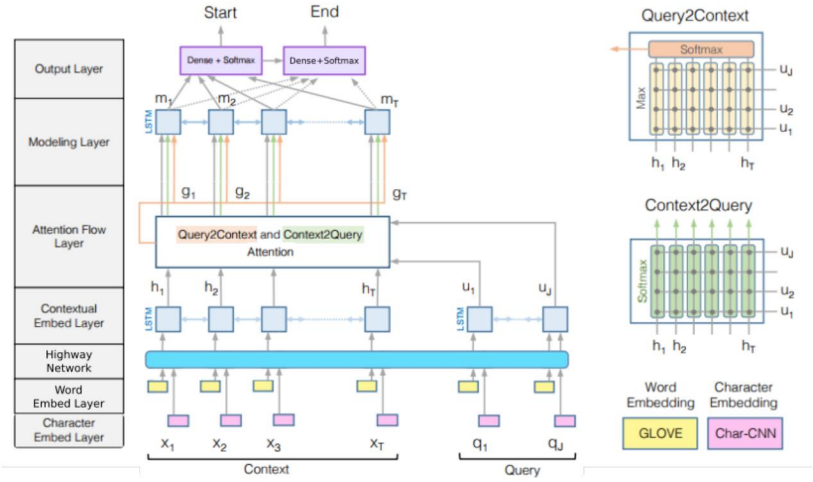


Figure 4: Final model architecture. Diagram adapted from original BiDAF architecture diagram.

3 Model

In this section, we first briefly describe the baseline model, and then go into detail about the incremental improvements that we added in the order we implemented them. The final model is shown in Figure 4. Table 1 shows the default parameters for our complete model.

3.1 Baseline

The baseline model, which was provided to us, contains:

- An RNN encoder layer that encodes the context and question into hidden states by feeding pre-trained GloVe embeddings into a 1-layer bi-directional GRU.
- A basic dot-product attention layer with context hidden states attending to question hidden states, which produces an attention output that is appended to the context hidden state to produce a blended representation.
- An output layer that feeds the blended representation through a RELU layer and a softmax layer to produce probability distributions for the start and end of the answer.

The loss is computed by summing the cross-entropy loss for the predicted start and end locations of the answer. The baseline model uses the Adam optimizer.

3.2 Bi-Directional Attention

Inspired by BiDAF [2], we change the basic attention layer of the baseline model in a bi-directional attention layer. In bi-direction attention, we compute both a context-to-question attention output, $c2q$ and a question-to-context attention output, $q2c$, both of which are computed using a similarity matrix S, where S_{ij} is the similar score of the encoded context-question pair (c_i, q_j) where $S_{ij} = {w_{\text{sim}}}^T[c_i; q_j; c_i \circ q_j]$, where w_{sim} is a trainable weight vector. We use the BiDAF attention output function, yielding the blended representations:

$$\mathbf{G} = [c_i, c2q, c_i \circ c2q, c_i \circ q2c]$$

Table 1: Default model parameters

Parameter	Value	Parameter	Value
Learning rate	0.001	Max context length	420
Max gradient norm	5.0	Max question length	30
Dropout	0.2	Max answer length	15
L2 loss weight	0.001	Max word length	15
Batch size	50	Word embedding size	300
Hidden size	200	Character embedding size	20
Highway depth	2	Character-level word embedding size	100
		CNN kernel size	5

3.3 Character CNN Embeddings

Also inspired by BiDAF [2], we supplement the GloVe-based embeddings with trained character-based embeddings by adding a character CNN to the embedding layer. In order to encode characters, we create a dictionary of characters that represent at least 0.01% of the characters in the train contexts. 51 such characters exist, which includes all letters and numerals as well as some common punctuation. Any characters that do not match these 51 are mapped to an UNK character. We represent these 52 characters (51 actual characters and UNK) using trainable character embeddings. We feed the character embeddings through a 1-dimensional convolutional layer in order to produce the character-level word embeddings, which we then concatenate to the GloVe embeddings to produce the overall embeddings.

3.4 Additional LSTM Layers

In place of the bidirection GRU encoder, we opt for a biLSTM encoder to increase the complexity of the encoder. Moreover, we replace the fully connected RELU layer with a biLSTM "modeler" in order to capture the relationship, $\mathbf{R}$ between question-aware context words, $\mathbf{G}$. The final blended representations that are fed through a softmax layer include both $\mathbf{R}$ as well as the pre-modeler blended representation $\mathbf{G}$ (G is like a residual connection). Note our modeling layer is not as complex as the original BiDAF modeling layer.

3.5 Smart Span Selection at Test Time

Inspired by DrQA [3], we added smart span selection, which does not affect training, but does improve performance at test time. Smart span selection is based on two observations. First, the end location for an answer should never be before the start location. Second, an analysis of answer lengths in the training and development sets shows that answers are rarely longer than 15 words, so we can reject any predicted answers of over 15 words. Based on these observations, we adopt a new method for answer prediction at test time: we return the start and end locations that maximize the joint probability $p^{\text{start}}p^{\text{end}}$, under the constraints that $\ell^{\text{start}} \leq \ell^{\text{end}} < \ell^{\text{start}} + 15$. This could be done through dynamic programming, but we found that in practice, brute-force searching is just as fast, if not faster, than fancy dynamic programming techniques.

3.6 Highway Networks

As we added additional layers of computation into our model, our model increased in depth. This prompted the addition of a 2-layer Highway Network [4] that learns gating mechanism to regulate information flow. We use the highway network layer in order to modify certain parts of the context and question embeddings before encoding.

4 Evaluation

In this section we evaluate our model on the SQuAD dataset. We use two scores:

Table 2: Impact of additional features on baseline.

Model	Dev F1	Dev EM
Baseline	43.226	34.305
+Bi-direction Attention	50.222	40.095
+Character CNN embeddings	51.94	42.157
+Additional LSTM layers	72.783	62.867
+Smart span selection	73.913	63.141
+Highway Networks (complete model)	74.248	64.04
Original BiDAF model [2]	77.3	67.7

- F1, which is the harmonic mean of word-level precision and recall.
- Exact Match (EM), which is a binary measure of whether our answer exactly matches the true answer.

Note that the SQuAD dataset actually provides three answers for every context-question pair, each from a different crowd-sourced human worker. Therefore, we took the highest F1 and EM score across the three human-provided answers. Each evaluated model was trained until the F1 and EM scores on the development set reached a plateau. This usually took around 7500 iterations, corresponding to around 6 hours when trained on a Microsoft Azure NV6 machine with an Nvidia M60 GPU.

4.1 Lesion Study

In Table 2, we present the development scores of our model after each incremental addition described in Section 4. We note that the additional LSTM layers gave the greatest improvement, bi-directional attention also had positive impact, and the other improvements each contributed one or two points to the F1 and EM scores. Our complete model achieved an F1 score of 74.2 and an EM score of 64.0 on the development set. This is only a few points lower than the original BiDAF model, which achieved and F1 score of 77.3 and an EM score of 67.7. After hyperparameter tuning, the gap became even smaller.

4.2 Hyperparameter Tuning

After completing our model, we tried varying hyperparameters for embedding size, regularization, and optimization in our model.

We found that increasing the GloVe embedding size to 300 did not have a noticeable impact. However, since making that change also did not noticeably increase training time, we decided to use 300-dimensional GloVe embeddings as our default.

Results for regularization are shown in Table 3. Note that the first row of the table corresponds to our complete model. In addition to tuning dropout, we also added average L2 loss computed over all non-bias trainable variables, multiplied by an L2 loss weight, to our overall loss function. We found that changing dropout had negative impact on performance; introducing L2 loss as an alternative regularizer had a negative impact; but combining dropout with L2 loss had a minor positive impact, giving an extra 0.6 F1 score and 0.3 EM score over our complete model.

Results for optimization are shown in Table 4. Note that these experiments use an L2 loss weight of 0.001. As before, the first row of the table corresponds to our complete model. We performed a brief experiment with using RMSProp instead of Adam. However, that model gave very poor performance. We therefore focused our attention on tuning the learning rate for Adam. In addition to training new models with lower learning rates, we also tried taking the learned weights of our plateaued complete model and continuing training using new learning rates. These continuing models are marked with * in the table. Overall, we found that decreasing the learning rate for a model whose development scores have plateaued contributes some incremental improvements, giving an

Table 3: Hyperparameter tuning for regularization.

Dropout	L2 weight	Dev F1	Dev EM
0.2	0.0	74.248	64.04
0.0	0.0	73.477	61.779
0.01	0.0	73.808	62.649
0.1	0.0	74.031	63.132
0.4	0.0	68.533	57.143
0.0	0.0001	73.819	62.800
0.0	0.001	65.798	54.551
0.0	0.01	61.927	50.700
0.2	0.0001	74.005	63.605
0.2	0.001	**74.812**	**64.267**
0.2	0.01	72.209	61.902

Table 4: Hyperparameter tuning for optimization.

Learning Rate	Dev F1	Dev EM
0.001	74.248	64.04
0.0005	74.543	63.756
0.00025	74.166	63.689
0.0005*	75.892	65.421
0.00025*	76.369	66.017
0.0001*	**76.430**	**66.102**

extra 2.2 F1 score and 2.1 EM score over our complete model. We decided not to keep decreasing the learning rate past 0.0001 because marginal improvements at that point were minimal.

Our final model used a dropout of 0.2, an L2 loss weight of 0.001, and the Adam optimizer with an initial learning rate of 0.001, followed by a secondary learning rate of 0.0001 when the model plateaus under the original learning rate.

With more time, we would have tried tuning other hyperparameters, such as: trying L1 or max-norm regularization; trying other optimizers, such as AdaGrad; and varying number of layers of our LSTMs and hidden size. The last possibility is especially limited by the memory cap of GPUs: we briefly attempted to increase the hidden size but ran into out-of-memory errors.

On the test set, this final tuned model achieves a 77.023 F1 score and a 67.072 EM score.

5 Analysis

In this section we perform analyses of our model performance and behavior in order to better understand the situations in which it performs well or poorly. We also do a deep dive into a few specific errors.

5.1 Breaking Down by Category

We differentiate the dataset by question type (determined by the first word in the question) as well as question length and average answer length, to see if our final model performs differently on each kind of question-answer pair. Figure 5 summarizes the results.

Questions that ask ”what” are the most prevalent, followed by questions that ask ”how,” ”who,” ”when,” ”which,” ”in,” ”where,” and ”why.” F1 and EM scores are roughly the same, though ”when” and ”in” questions tend to do better and ”why” questions tend do worse. This could be because ”when” and ”in [which]” questions often have obvious answers (a proper noun such as a date or location), whereas answers to ”why” questions are more abstract.

F1 and EM scores are also roughly comparable across question length and answer length, but scores begin to decrease as the length increases past 20 words for questions and 15 words for answers. This makes sense because longer questions and answers are difficult to parse correctly or predict exactly. The EM score for average answer lengths above 15 words is especially low because our smart span selector rejects answers longer than 15 words. Note that the EM score is not necessarily 0 when the average answer length is above 15, because we could still exactly match one of the gold answers which is not above 15 words.

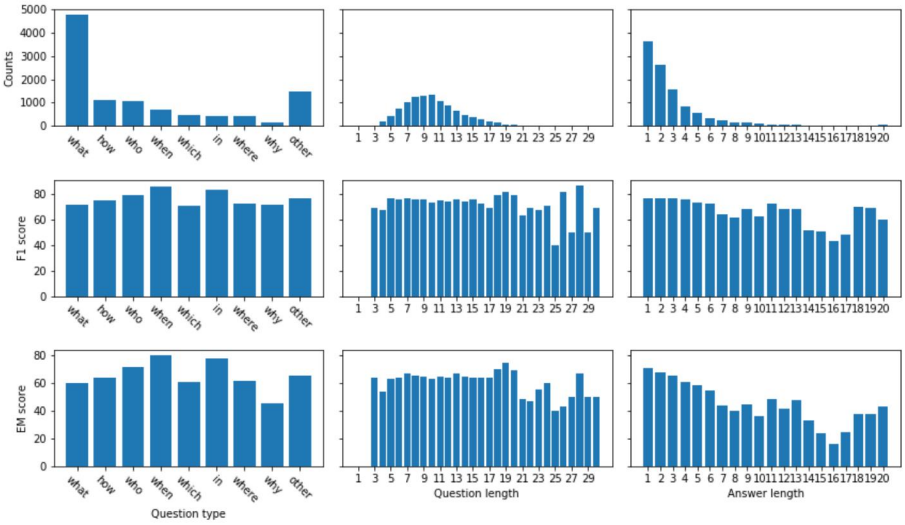


Figure 5: Scores broken down by question type (first column), question length (second column), and answer length (third column).

Table 5: Categories of errors, along with a sample.

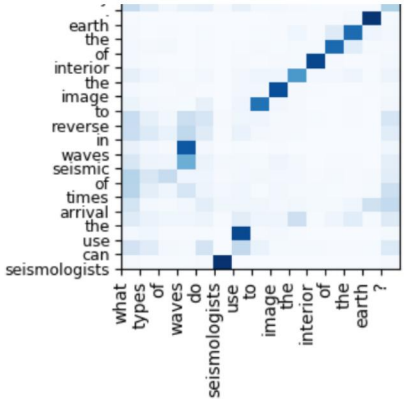
Error category	Example
Imprecise error boundaries	**Context:** johnson was known to the iroquois as warraghiggey, meaning "he who does great things." **Question:** what was william johnson's iroquois name? **True Answer:** warraghiggey, meaning "he who does great things." **Predicted Answer:** warraghiggey
Syntactic/semantic complications	**Context:** if the head of government of a country were to refuse to enforce a decision of that country's highest court, it would not be civil disobedience **Question:** what does not constitute as civil disobedience? **Predicted Answer:** refuse to enforce a decision **True Answer:** country's highest court
Incorrect word-matching	**Context:** atp synthase uses the energy from the flowing hydrogen ions to phosphorylate adenosine diphosphate into adenosine triphosphate, or atp. because chloroplast atp synthase projects out into the stroma, the atp is synthesized there **Question:** what does atp synthase change into atp? **True Answer:** phosphorylate adenosine diphosphate **Predicted Answer:** chloroplast atp synthase projects out into the stroma

5.2 Error Analysis

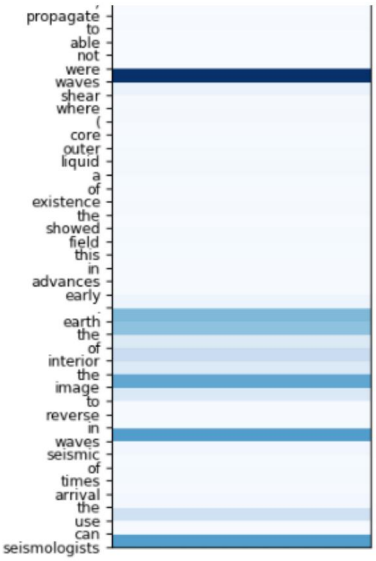
We manually inspected errors made by our final model (ie, questions with EM of 0). We found that most errors fell into a few different categories, as shown in Table 5. Imprecise error boundaries, where the predicted answer has a few more or fewer words than the true answer, accounted for most of the errors. Other errors were caused by syntactic/semantic complications, where the model identified the right region of text but did not understand the meaning of the text, as well as incorrect word-matching, where the model simply found the piece of text that matched key words in the question.

5.3 Attention Visualization

In order to understand how our attention layer contributed to our models' accuracy, we plot the context-to-question and question-to-context attention distributions. When inspecting some exam-



(a) Context-to-question attention distribution. Each row is a context word's probability mass function indicating which question words (columns) received the most attention.



(b) Question-to-context attention distribution. Each row represents how important that context word is to the question.

Figure 6: Attention distributions for a selected data point. The model answered 'seismic', while the correct answer was 'seismic waves'.

ples, as in Figure 6, context-to-question attention works as expected; context words attend to the same words in the question, and even *seismic* attends mostly to *waves*. We conjecture this is because *seismic* is associated with *waves* topically in the word embeddings and locally in the context.

However, the question-to-context attention distribution indicates some flaws: the question is simply attending to the same words present in the context, and believes *waves* in a different part of the context should be paid the most attention to. Although this attention mechanism was not enough to dissuade the model from predicting *seismic* (influence of other model layers), we believe question-to-context attention mechanisms should be revisited and improved.

6 Conclusion

In this paper, we presented a model that performs competitively on SQuAD, a machine reading and question answering dataset. We incorporated techniques such as bi-directional attention, character CNN embeddings, and highway networks into our model. We found that additional LSTM layers in the encoding and modeling layers yielded the greatest improvements, while other techniques and hyperparameter tuning each managed to supply a few extra points of performance. Thus, even though the original BiDAF model contains a more complex modeling layer, our final tuned model remains competitive.

For future work, we intend to (1) increase the complexity of the modeling layer which should eke out better performance; (2) include self-attention and/or revise question-to-context attention; (3) gather an ensemble of models for better decisionmaking; (4) perform further hyperparameter tuning to gauge the impact of varying hidden size and number of layers

Acknowledgements

We thank Professor Richard Socher and the CS224N teaching staff for their endless help on submitting to CodaLab and suggestions for model improvements, and for a great class.

References

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

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603, 2016.

[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051, 2017.

[4] Rupesh Kumar Srivastava, Klaus Greff, and Jurgen Schmidhuber. Highway networks. arXiv preprint arXiv:1505.00387, 2015.

[5] https://rajpurkar.github.io/SQuAD-explorer/

[6] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. arXiv preprint arXiv:1611.01604, 2016.

[7] https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf

[8] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. arXiv preprint arXiv:1608.07905, 2016.