Improving BiDAF: Character Embedding, Self-Attention, and Answer Pointer

Stanford CS224N Default Project

Edward Gao
Stanford University
yxgao19@stanford.edu

Abstract

Question answering (QA) is one of the most important challenges in machine comprehension as an increasingly diverse array of natural language processing tasks are being framed as QA problems. This work explores several different strategies for enhancing the performance of the popular Bi-Directional Attention Flow (BiDAF) QA model. Adding a character embedding layer to better extract word meaning and a self-attention layer to capture structures within the context passage successfully led to an improved BiDAF model. In contrast, an Answer Pointer output layer, designed to produce end point predictions conditioned upon start point predictions, did not offer performance benefits. The final model (with hyperparameter tuning) achieved **67.40 F1/64.36 EM** on the dev set and **65.84 F1/62.30 EM** on the test set.

1 Key Information to include

- Mentor: Sarthak Kanodia
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Question answering (QA) is one of the classic tasks in the fields of machine comprehension and natural language processing (NLP). In a typical QA problem, a model is presented with some context passage and a question (or "query"), and the model has to answer the question using information found within the context. A robust QA system is highly sought after for three reasons. First, it has direct applications in areas such as search engines and virtual assistants. Second, it can serve as a benchmark for evaluating and understanding machine comprehension in general. Finally, in recent years, many different NLP tasks such as information extraction [1] and semantic labeling [2] have been framed as QA problems, so a high-performing QA model has wider applications in NLP as well.

A popular benchmark dataset for evaluating QA models is SQuAD 2.0 [3]. This dataset poses two distinct challenges. First, answers to questions in this dataset can be arbitrary in length, and a model has to locate a contiguous sequence of tokens within the context to answer the question. Earlier examples of QA models were usually limited to single-word answers or multiple choice questions, and more open-ended questions are significantly harder to address. Second, roughly half of the questions in SQuAD 2.0 cannot be answered using the given context, which further stresses a QA model's ability to synthesize information.

Recently, end-to-end deep learning QA models have gained popularity, in contrast to earlier techniques that relied heavily on feature engineering or linguistic analysis. One notable example is the RNN-based Bi-Directional Attention Flow (BiDAF) network [4], which uses two-way attention between the context and the query to identify the answer. While transformer-based models such as QANet [5]

have shown higher performance, BiDAF offers the advantage of lower memory usage and remains a classic example that demonstrates the utility of recurrent architectures.

In this project, I aim to improve the performance of the baseline BiDAF model by implementing three additional layers. The character embedding layer is meant to better extract word meaning at the sub-word level and can help the model process new vocabulary that was not seen during training. The self-attention layer captures structure within the context in order to better synthesize and understand the information presented. Finally, the Answer Pointer layer predicts the end point of the answer span conditioned upon the start point prediction, in the hopes of capturing any patterns shared by correct answers and thereby providing better predictions. I examine the effects of introducing these three modifications and combine the beneficial changes to arrive at a better BiDAF model.

3 Related Work

3.1 Bi-Directional Attention Flow (BiDAF)

The BiDAF model [4] serves as the baseline for this project. It uses pre-trained word vectors and LSTMs to encode information in both the context and the query. Its most prominent feature is the bi-directional attention layer, which uses both context-to-query (C2Q) and query-to-context (Q2C) attentions to generate query-aware context representations. These attention output vectors are then used for various downstream tasks such as predicting answer start and end points.

3.2 Character Embedding

Sub-word or character embedding is a useful strategy in NLP for helping a model better capture word meaning and process rare or otherwise unseen vocabulary. Both the original BiDAF model [4] and transformer-based models such as QANet [5] use character embedding in addition to word vectors. A convolution layer is usually responsible for pooling over the character vectors of a given word and producing a character-level word representation. The original BiDAF model showed that removing character embedding indeeds leads to a drop in F1 score by a few points.

3.3 Self-Attention

While BiDAF employs bi-directional attention between context and query, it does not take advantage of semantic structures within the context itself. Self-attention is a popular method for extracting this information. This strategy is demonstrated in the R-Net architecture [6] as well as other related reading comprehension models [7]. Transformer-based models also utilize self-attention extensively [5].

3.4 Answer Pointer

The Answer Pointer [8] network has been proposed as an alternative to the existing output layer in BiDAF. The key difference is that it uses an LSTM to predict the end point of the answer span based on the calculated probability distribution of the start point. This approach takes advantage of any patterns that may be present within correct answers. The Answer Pointer layer is a generalizable strategy that has been applied to many different areas beyond question answering.

4 Approach

The architecture of the modified BiDAF model is depicted in Figure 1. The baseline model is described in the original BiDAF report [4]. The three modified layers **that I implemented myself** are discussed in detail below.

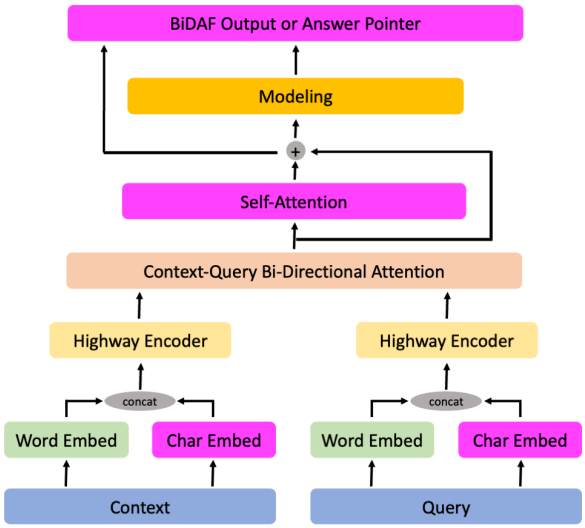

Figure 1. Architecture of the improved BiDAF model. Modified layers are highlighted in magenta.

4.1 Character Embedding

In the character embedding layer, each word is converted into a 2D array using pre-trained character vectors. This array is then processed using a one-dimensional convolution along the word length axis. The height of the filter is a hyperparameter to be set, and the number of output channels is equal to the dimension of pre-trained word vectors used in the word embedding layer. Each output channel is then separately max-pooled to produce a character-level word representation that is of the same dimension as pre-trained word vectors. The outputs of the word and character embedding layers are concatenated before being fed into a highway encoder. This structure is summarized in Figure 2 below.

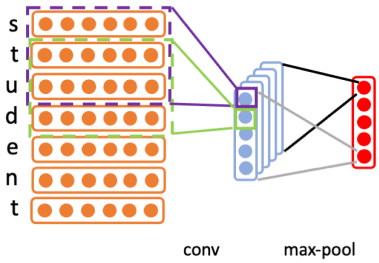

Figure 2. Architecture of the character embedding layer.

4.2 Self-Attention

Although additive self-attention is the most popular mechanism, I chose to implement the self-attention layer using multiplicative attention due to memory constraints. The attention score between query-aware context representations (as produced by the bi-directional attention layer) v_i and v_j is calculated as

$$s_j^i = v_i^\top W v_j,$$

where W is a learnable weight matrix. The self attention output c_i for word i is then

$$a_j^i = \exp(s_j^i) / \sum_{n=1}^{P} \exp(s_n^i)$$

$$c_i = \sum_{n=1}^{P} a_n^i v_i$$

where P is the length of the context. Note that s_i^i is masked to $-\infty$. From here I experimented with two different strategies. In "simple" self attention, the resulting vectors c is simply passed through a linear layer followed by a ReLU activation. In "gated" self attention, the vectors c are concatenated with the input vectors v, passed through an element-wise multiplicative gate with a sigmoid activation, and further processed by a bi-directional GRU network:

$$h = \text{BiGRU}([v; c] \odot \text{Sigmoid}(G[v; c])),$$

where G is a learnable weight matrix and $\odot$ denotes element-wise multiplication. For both strategies, a residual connection is added, meaning that the self attention output is added to the input vectors.

4.2.1 Answer Pointer

The Answer Pointer layer optionally replaces the current BiDAF output layer. Let H be the concatenation of the outputs of the attention and modeling layers. The probability distribution of the answer start point β_{start} is calculated as

$$F_{\text{start}} = \tanh(VH + (Wh_{\text{start}}))$$
$$\beta_{\text{start}} = \text{softmax}(v^\top F_{\text{start}} + c),$$

where V, W, h_{start}, v, and c are trainable parameters. The end point distribution β_{end} is calculated similarly, except h_{end} is generated by running an LSTM for one time step on the start point prediction:

$$h_{\text{end}} = \text{LSTM}(H\beta_{\text{start}}^\top, h_{\text{start}})$$

5 Experiments

5.1 Data

All models were trained and evaluated on SQuAD 2.0 [3]. There are 129,941 examples in the train set, 6,078 in the dev set, and 5,915 in the test set. Each example contains a context passage, a question, and the ground-truth answer (Figure 3). The answer must always be a contiguous segment of tokens found in the passage. Roughly half of the questions cannot be answered. Concretely, to answer a question, the model outputs start and end indices that define a span within the context. If a question cannot be answered, the model outputs (-1, -1).

Example

Passage:
European Union law is a body of treaties and legislation, such as Regulations and Directives, which have direct effect or indirect effect on the laws of European Union member states. The three sources of European Union law are primary law, secondary law and supplementary law. The main sources of primary law are the Treaties establishing the European Union. Secondary sources include regulations and directives which are based on the Treaties. The legislature of the European Union is principally composed of the European Parliament and the Council of the European Union, which under the Treaties may establish secondary law to pursue the objective set out in the Treaties.

Question: What are the main sources of primary law?
Answer: Treaties establishing the European Union

Question: What is the last source of European Union law?
Answer: (Cannot be answered)

Figure 3. Example entry in the dataset.

5.2 Evaluation method

Models are primarily evaluated based on the F1 scores achieved, although EM scores are also reported. The binary answer vs. no-answer (AvNA) accuracy is also included for the dev set. For each question, the dataset contains three human-labeled ground-truth answers. F1 and EM scores are based on the maximum value achievable across these three labels.

5.3 Experimental details

Unless otherwise noted, all models were trained on an NVIDIA Tesla V100 PCIe GPU (16 GB) for 30 epochs with batch size = 64, learning rate = 0.5, dropout probability = 0.2, highway encoder RNN hidden size of 100, no L2 regularization, and a random seed of 224. Hidden sizes of subsequent RNN networks are based on the output dimensions of upstream layers. The model was optimized using an AdaDelta optimizer and negative log loss as the loss function. The model was evaluated roughly every 50,000 iterations on the dev set, and the configuration that gave the highest F1 score was kept. Training took roughly 3-4 hours for the baseline model and 6-7 hours for the largest models.

5.4 Results

Performance of the various models on the dev set as well as results from the **non-PCE** leaderboard are summarized in Table 1.

Table 1. Summary of Model Performance

Model	dev set			test set	
	F1	**EM**	**AvNA**	**F1**	**EM**
baseline	60.58	57.25	67.74	60.91	57.13
+ char embed	63.48	60.36	69.67	-	-
+ simple self-attention	62.05	58.49	68.48	-	-
+ gated self-attention	63.07	59.84	69.25	-	-
+ Answer Pointer	60.55	56.98	67.25	-	-
+ char embed + gated self-attention	65.93	62.71	71.79	64.24	60.78
+ char embed + gated self-attention + dropout 0.1	**67.40**	**64.36**	**73.06**	**65.84**	**62.30**

The character embedding layer improved the F1 score by roughly 3 percentage points, similar to what was reported in the original BiDAF paper [4]. This is unsurprising, as modeling at the sub-word level is known to help with meaning extraction and model performance, especially when rare or unseen words are likely present in the dev or test sets.

Self-attention also successfully enhanced model performance, with gated self-attention offering greater improvements than simple self-attention, likely due to the additional expressive power afforded by the extra RNN layer. It is worth noting that the performance boost in this case is smaller than that offered by character embedding. This may suggest that semantic structure within the context is not very helpful for question answering. Of course, this result may also speak to the limitations of the current model architecture, and other potential self-attention setups were not explored in this project. For example, it may be worthwhile to investigate placing the self-attention layer before the context-query attention layer. This alternative arrangement will allow the model to better capture structures within the context without being distracted by the query.

The Answer Pointer layer did not offer any performance benefits. There are two potential explanations for this result. First, it may be possible that the probability distributions of the start and end points of the answer span are relatively independent, so an output layer that produces conditional probabilities performs similarly to one that does not. Alternatively, it may be the case that the Answer Pointer layer is not sufficiently different from the original BiDAF output layer. In the original output layer, the start point prediction is generated using the outputs of the attention and modeling layers. The modeling layer output is then processed using an LSTM before being used to generate the end point prediction. The Answer Pointer layer similarly uses an LSTM to map start point probabilities to an end point distribution. Although the BiDAF output layer does not explicitly generate conditional probabilities, it is possible that these two layers are actually mechanistically similar enough that no significant difference in performance is seen.

Combining the beneficial character embedding and gated self-attention layers indeed had an additive effect. After a brief hyperparameter search, it was revealed that reducing the dropout probability to 0.1 further enhanced model performance on both the dev and test sets. It appeared that the model did not significantly overfit under the default hyperparameter settings, and adding any L2 regularization led to poor learning (data not shown). Conversely, learning rate decay caused significant overfit, with a very low loss on the training set but poor F1 and EM scores on the dev set (data not shown). Reducing the

dropout probability therefore lowered the regularization strength slightly, thereby improving learning but without overfitting extensively.

6 Analysis

Inspection on the dev set data revealed that the examples are not balanced in terms of question type[1] (Figure 4). "What" questions accounted for the majority of the dataset. A few examples fell into the "other" category, which mainly contain Yes/No questions or questions phrased as commands (e.g. "Name the person that invented the microwave").

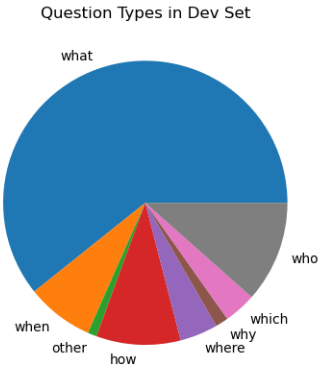

Figure 4. Breakdown of dev set data by question type.[2]

Performance of the optimized model varied by question type (Figure 5). "When" and "Who" questions had the highest accuracy. This is unsurprising, as answers to these types of questions are usually short and have unique characteristics. It is possible that the model simply learned to extract dates when presented with a "when" question and person names when presented with a "who" question. In contrast, the model had a harder time answering "why" and "how" questions, as they usually require longer answers that may encompass a wide array of different words.

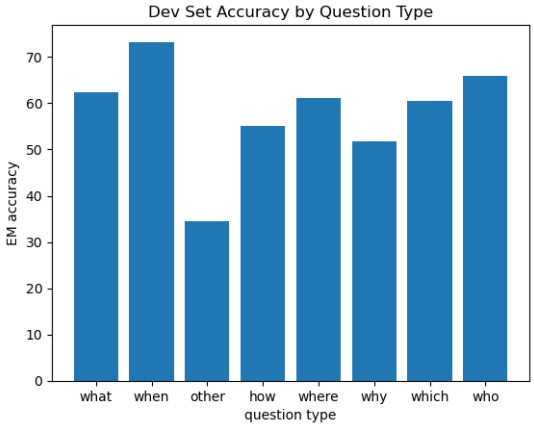

Figure 5. Model performance by question type.

Interestingly, the model had very poor performance on "other" questions. Yes/No questions are a little bit special. Even when the context contains the relevant information, these questions usually cannot be answered in the current setup because the context most likely does not contain the words "Yes"

[1]This distribution is only approximate. When multiple question words appear in a sentence, it is not always straightforward to determine the exact question type. Consider the (contrived) example "Out of all presidents who had cats when they were in office, which one also had dogs?"

[2]Both "whom" and "whose" questions count as "who."

or "No." Since the model can simply produce "no answer" whenever it detects a Yes/No question, and given its high performance on questions that cannot be answered (see below), it is a little bit surprising that accuracy in this category is so poor. It is possible that the poor performance is mainly caused by imperative questions, as they are phrased quite differently from most other questions in the training set and are harder to process.

Model performance also decreased as answer length increased (Figure 6). The model had the highest accuracy on questions that cannot be answered. Questions whose answers are shorter than five words were addressed with reasonably high accuracy. However, the model had significantly more difficulties generating longer answers.

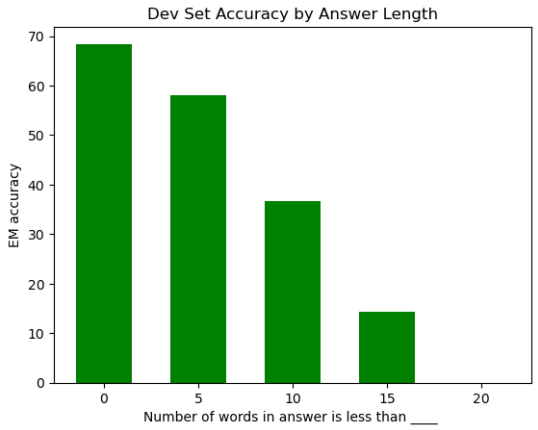

Figure 6. Model performance by answer length.

Taken together, these observations suggest that the current model has learned reasonably well to identify key words (names, dates, etc.) and to process simpler questions, while more nuanced questions with more complicated answers still pose a significant challenge.

7 Conclusion

In this project, I aimed to improve the baseline BiDAF model by implementing a character embedding layer, a self-attention layer, and an Answer Pointer layer. Both the character embedding and self-attention layers led to performance improvements as expected, but the Answer Pointer layer had minimal effect. A final model combining beneficial modifications and with hyperparameter tuning achieved +4.93 F1 and +5.17 EM on the test set. It appears that the model is much better at processing simpler questions with well defined answers. It remains an open challenge to further improve the model to better handle more complicated text.

References

[1] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *CoRR*, abs/1706.04115, 2017.

[2] Luheng He, Mike Lewis, and Luke Zettlemoyer. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[3] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

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

[5] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018.

[6] Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. May 2017.

[7] Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. *CoRR*, abs/1710.10723, 2017.

[8] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016.