Improving BiDAF Heuristically and with Self-Attention for SQuAD 2.0

Stanford CS224N Default IIDSQuAD Project

Nathaniel Goenawan
Department of Computer Science
Stanford University
nathgoh@stanford.edu

Christopher Wong
Department of Computer Science
Stanford University
cwong7@stanford.edu

Abstract

Our primary goal in this project was to build a QA system that improves upon a provided baseline's performance on the SQuAD 2.0 dataset, doing so through three major approaches. We first explore the performance of the provided baseline with and without a character-level embedding layer. Secondly, we explore the performance of the provdided baseline with the character-level embedding layer as well as an added simple context/type matching heuristic using a binary-like word-in-question feature (as discussed further later in this work). Thirdly, we add a self-attention layer on top of the existing BiDAF attention layer combined with character embeddings, and explore the performance of this model with and without the aforementioned binary-like word-in-question feature. We find that the model subsisting of the baseline with character embeddings yielded the best performance on the test set, yielding an EM score of 59.104 and an F1 score of 62.673.

1 Key Information

- Our TA mentor is Allan Yang Zhou; we have no external collaborators, no external mentors, and are not sharing this project with any other classes.

2 Introduction

Question answering (QA) is an important end-user task for natural language processing and information retrieval; as such, QA is one of the most studied machine comprehension tasks. The main goal with QA systems is to, given a question and associated context paragraph, correctly answer a question. QA systems are widely applicable and have many commercial applications, such as in popular search engines, where their usage is meant to serve as a relief for the otherwise existent need to read search results to get an answer to a question.

Specifically, this paper focuses on building a QA system for the Stanford Question Answering Dataset 2.0 (SQuAD 2.0) [1]. This dataset is an extension of SQuAD, a structured dataset containing over 100,000 triples consisting of a context paragraph, question, and answer derived from Wikipedia articles. The extension introduces over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones, with the intension being that to do well on the extended SQuAD 2.0 dataset, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering. While there have been many current state-of-the-art models whose performances exceed human performance in EM and F1 scores, it should be noted that due to the convenience and rigidity of the SQuAD and SQuAD 2.0 dataset structures, the performance state-of-the-art models exceeding human performance is not sufficient cause for concluding that the problem of question answering is solved.

This project aims to improve upon a provided, pre-established baseline QA model by incorporating combinations of character-level embeddings, a binary-like word-in-question feature (such as one

described in [2]), and a self-attention layer (similar to the one described in [3]). We found that the addition of character-level embeddings slightly improved on the performance of the baseline at the cost of a minimal increase in training time, while the addition of a self-attention layer to the existing BiDAF model with character-level embeddings performed marginally worse than the baseline. Combining character-level embeddings, a self-attention layer, and the binary-like word-in-question feature led to improved performance on the dev set but worse performance on the test set, likely due to overfitting.

3 Related Work

Many architectures have been developed in an attempt to provide solutions to the question-answering problem as posed by the SQuAD datatset. Here, we provide three models that inspired the work in this paper.

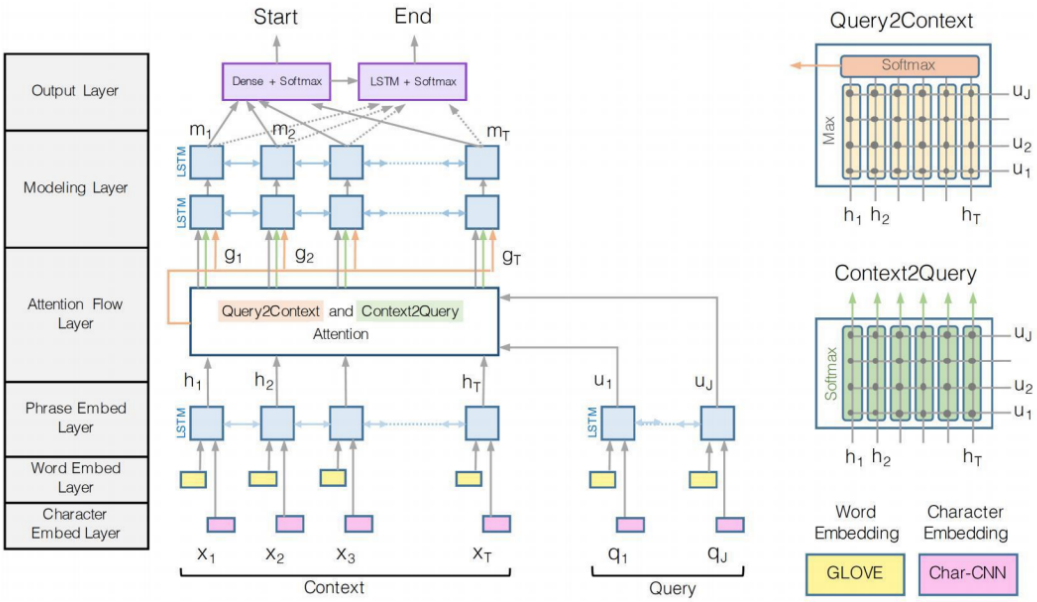

Figure 1: Full BiDAF Model Architecture

In 2016, the Bidirectional Attention Flow (BiDAF) model introduced a significant advancement in QA model architecture [4]. BiDAF utilized a Long Short-Term Memory Network (LSTM) with a bi-directional attention flow mechanism (having both context-to-query and query-to-context attention) to predict the starting and ending indices of the answer in the context paragraph. Our provided baseline for this project is very similar to the BiDAF model proposed [4], with the exception being that the provided baseline only contains word embeddings in its embedding layer, while the original BiDAF model's embedding layer contained both word and character-level embeddings. Hence, we also included character-level embeddings in the hopes of improving performance.

In 2017, Weissenborn et. al propose that to make an effective QA system, one key factor is having an understanding of the question being asked, specifically referring to an awareness of question words while processing the context of the question [2]. We implemented a binary-like word-in-question feature as described in the paper with the aim to be able to select answer spans that match the expected answer type and are close to important question words.

Also in 2017, R-Net built on BiDAF's architecture by combining BiDAF's context-to-query attention layer with a self-matching attention layer [3]. R-Net performed better than BiDAF on the SQuAD 1.1 test dataset, achieving an EM score 2.3 higher than BiDAF's EM score and an F1 score 2.4 higher than BiDAF's F1 score. In this project, we incorporate self-attention into the architecture in the hopes of obtaining similar performance improvements as seen with R-Net, recognizing that these performance improvements were on the SQuAD 1.1 test dataset rather than the SQuAD 2.0 dataset.

4 Approach

We extend the baseline model with character-level embedding, a context-matching feature as an additional input feature, and a self-attention layer.

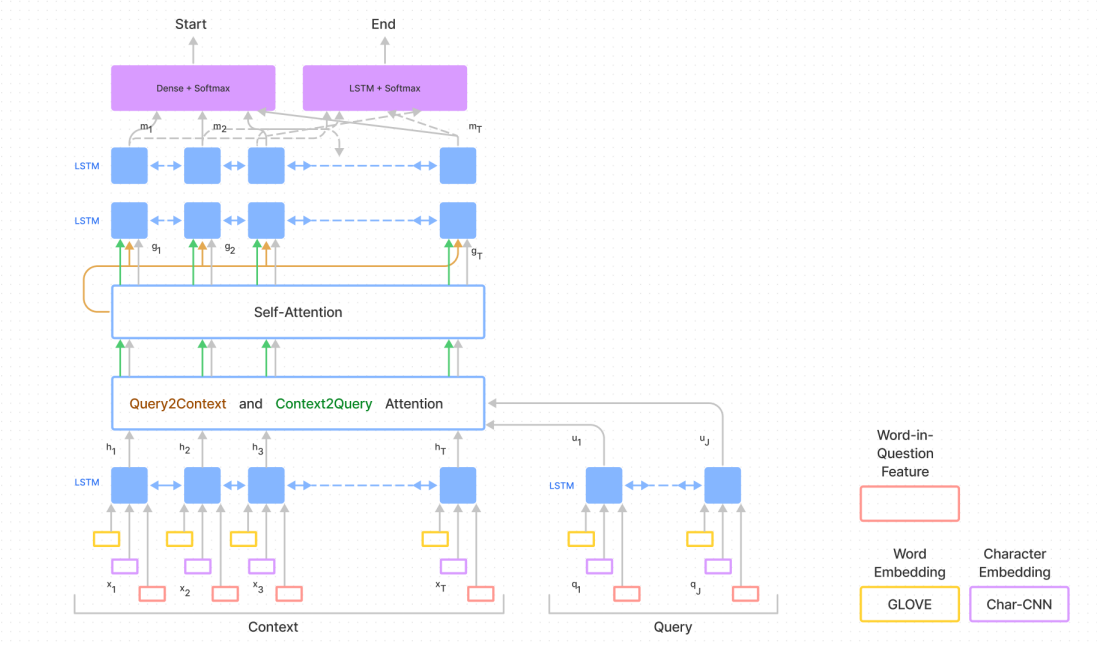

Figure 2: Extended BiDAF model with word-in-question feature and self-attention.

4.1 Baseline

Our baseline is the provided baseline model for the IIDSQuAD challenge by the CS224N staff, which is based on the Bi-Directional Attention Flow (BiDAF) model; the only difference is that we have included a character-level embedding layer [4]. More information about the provided baseline model can be found in the project handout.

4.2 Character-Level Embedding

The character-level embedding layer will map each word to a high-dimensional vector space. Let $\{w_1, w_2, \cdots, w_k\} \in \mathbb{N}$ and $\{c_1, c_2, \cdots, c_k\} \in \mathbb{N}$ for input word indices and character indices, respectively. In the baseline model, an embedding lookup is produced by converting word indices into word embeddings for both the context paragraph $\{c_1, c_2, \cdots, c_N\} \in \mathbb{R}^{\mathbb{D}}$ and query $\{q_1, q_2, \cdots, q_N\} \in \mathbb{R}^{\mathbb{D}}$. To apply the character-level embedding, we apply it to context paragraph and query. First, we perform the embedding lookup for the character indices to character embeddings. Then, a dropout is applied to the embeddings before passing the outputs through a 2D convolutional layer to finally obtained our learned embeddings. A 2D CNN layer was used as opposed to a 1D CNN layer because prior to applying the CNN layer, the output size of the character-level embeddings is 4D. Finally, the output of the CNN are max-pooled over the entire width before concatenating them onto the word embeddings.

4.3 Context-Matching Feature

The context-matching feature is a heuristic that aims to be able to select answer spans that match the expected answer type and are close to important question words. To accomplish this heuristic, we implemented a binary-like word-in-question (wiq^b) feature. This feature is computed for each context word x_j and explained as the following: j for tokens that are part of the question and otherwise 0. More formally,

$$wiq^b = I(\exists i : x_i = q_i) \tag{1}$$

where *I* denotes the indicator function.

We then get the embeddings for the wiq^b feature which is then concatenated onto the word embeddings as shown in Figure 2.

Informally, by providing this context-matching feature, we hope to provide our model with some "context clues" from the context paragraph that can be helpful in answering the question.

4.4 Self-Attention

For our self-attention layer, we implemented a multi-headed attention layer as described in Vaswani et al. [5]. Similar to the paper, we also employed 8 parallel attention layers or heads with each head using scaled dot-product attention. The self-attention layer is included after the BiDAF attention layer as shown in Figure 2.

5 Experiments

5.1 Data

- **Question:** What president eliminated the Christian position in the curriculum?
- **Context:** Charles W. Eliot, president 1869–1909, eliminated the favored position of Christianity from the curriculum while opening it to student self-direction. While Eliot was the most crucial figure in the secularization of American higher education, he was motivated not by a desire to secularize education, but by Transcendentalist Unitarian convictions. Derived from William Ellery Channing and Ralph Waldo Emerson, these convictions were focused on the dignity and worth of human nature, the right and ability of each person to perceive truth, and the indwelling God in each person.
- **Answer:** Charles W. Eliot

Figure 3: Example of (question, context, answer) triple from SQuAD

We will be using a portion of the SQuAD 2.0 dataset [1]. The dataset is split into train, dev, and test sets. The train set contains 129,941 examples, all taken from the SQuAD 2.0 dataset. The dev set contains 6078 examples (roughly half of the official dev set, randomly selected). Finally, the test set contains 5915 examples, which come from the remaining dev set and are supplemented by hand-labeled examples. Preprocessing is provided by the starter code, first processing the train set and obtaining word and character vocabularies. With the obtained word and character vocabularies, we processed the dev and test sets to obtain the context and question features.

In addition, from the CS224N staff, we are also provided with pre-trained GloVe word embeddings and pre-trained character-level embedddings.

5.2 Evaluation method

We will be using the official SQuAD evaluation metrics: Exact Match (EM) and F1 score metrics. EM is a strict binary metric that measures the percentage of predictions that match any one of the ground truth answers exactly. F1 score is a measure of a model's accuracy on a dataset, the harmonic mean of precision and recall.

5.3 Experimental details

We trained three different models: (1) baseline + character-level embeddings, baseline + character-level embeddings + wiq^b, and (3) baseline + character-level embeddings + wiq^b + self-attention. All models were trained over 30 epochs, a learning rate of 0.5, a drop probability of 0.2, an exponential decaying rate of 0.999, and a hidden size of 100. Though, it should be noted that while all models were trained over 30 epochs, we noted that past roughly 20-22 epochs, the model would begin to over-fit on the training data as indicated by the increase in the dev negative log-likelihood (NLL) as can be seen in Figure 4.

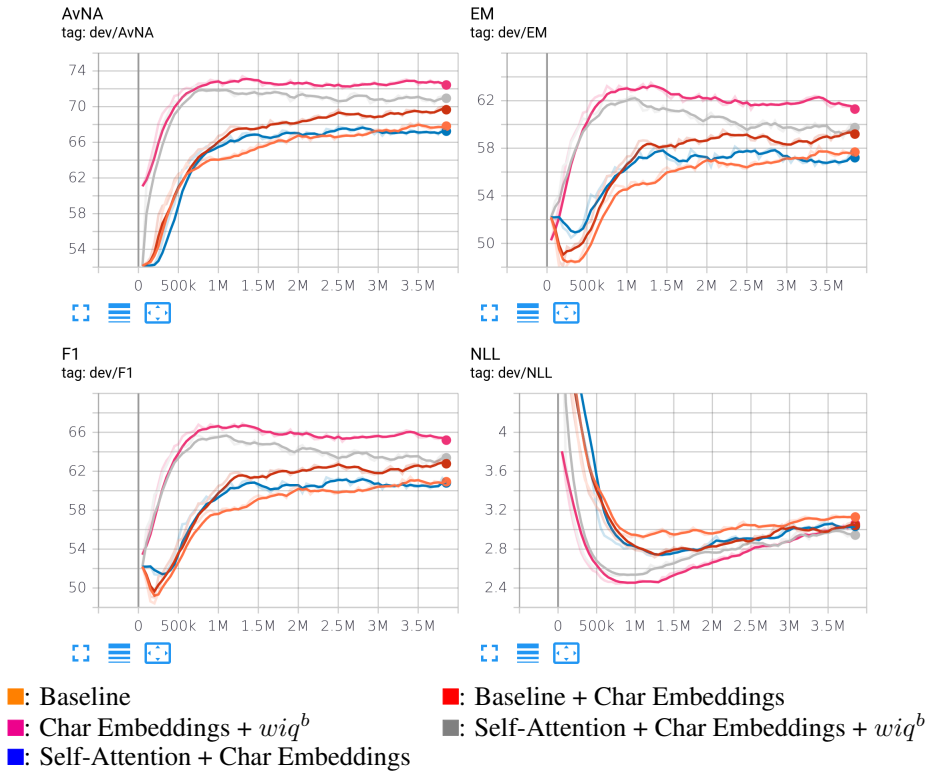

Figure 4: Plots captured from Tensorboard of various quantitative metrics over 30 epochs.

Model	Dev EM	Dev F1	Test EM	Test F1	Total Train Time
Baseline	58.175	61.440	-	-	3 hr 9 min
Baseline + Char Embeds	59.452	63.076	**59.104**	**62.673**	4 hr 4 min
Char Embeds + wiq^b	**63.519**	**66.926**	56.585	59.464	10 hr 18 min
Self-Att + Char Embeds	58.14	61.37	-	-	8 hr 47 min
Self-Att + Char Embeds + wiq^b	62.275	65.772	54.269	56.846	8 hr 48 min

Table 1: Model results on dev set and test set along with total train time over 30 epochs.

5.4 Results

Our results were rather underwhelming with none of our proposed models outperforming the baseline + character-level embedding model in the test set. The model with character-level embeddings and wiq^b showed promised in the dev set, showing a substantial 6.8% and 6.1% increase in F1 and EM scores respectively over the baseline + character-level embedding model. However, this unfortunately didn't translate to the test set as shown in Table 1. These observations also extend to the model with the self-attention layer as well which performed better than baseline + character-level embeddings on the dev set but not the test set.

6 Analysis

6.1 SQuAD 2.0 By Question Type

The SQuAD 2.0 dataset can be broken down by question type. Specifically, SQuAD 2.0 questions will have certain question words that we broke down to as such: 'what', 'who', 'when', 'where', 'why', 'which', and 'how'. Any question word types that didn't fit the aforementioned categories were categorized as 'other'.

As shown in Figure 5, we can see that the SQuAD 2.0 dev set is heavily skewed, with the majority of questions having the question word 'what'.

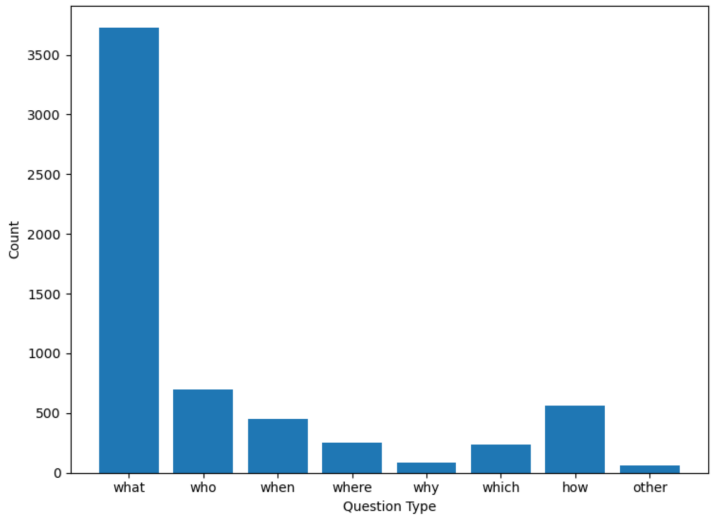

Figure 5: Distribution of Questions in Dev Set by Question Type

6.1.1 EM By Question Type

We were curious to see how the models performed on individually on each question type that we categorized the SQuAD 2.0 dataset into. We can see in Figure 6, that the char embeddings + wiq^b + self-attention model performed surprisingly well on 'why' and 'other' question types, while other models struggled on those two question types to various degrees. Though, these improvements could be attributed to the lack of data for the 'why' and 'other' question types as we see in Figure 5 which results in high variance and possibly misleading estimation of the model's performance. Overall, with the expection of the char embeddings + self-attention model, it was noted the other models all performed on par or better than the baseline model's performance.

6.2 SQuAD 1.1 vs SQuAD 2.0

The main difference between SQuAD 1.1 and SQuAD 2.0 is that SQuAD 2.0 has the addition of over 50,000 adversarial unanswerable questions. As such, we were curious to see how our trained models would perform on the SQuAD 1.1 dev set, which only contains answerable questions, as opposed to the SQuAD 2.0 dev set, which contains a mix of answerable and unanswerable questions.

Model	SQuAD 2.0 EM	SQuAD 1.1 EM
Baseline	58.18	68.12
Baseline + Char Embeds	59.45	69.59
Char Embeds + wiq^b	63.52	66.89
Self-Att + Char Embeds	58.14	69.14
Self-Att + Char Embeds + wiq^b	62.28	66.01

Table 2: Models' EM results on SQuAD 1.1 vs SQuAD 2.0

We can see from Table 2 that, as expected, without the unanswerable questions, all models performed significantly better on the SQuAD 1.1 dev set. However, despite the char embedding + wiq^b model and self-attention + char embedding + wiq^b model being the two top performers on the SQuAD 2.0 dev set, they were the worst performers on the SQuAD 1.1 dev set, even more so than the baseline model. This indicates that the performance gains on unanswerable questions are very likely attributed to the additional model features, mainly wiq^b. However, the improvements over other models come at the expense of answerable question performance.

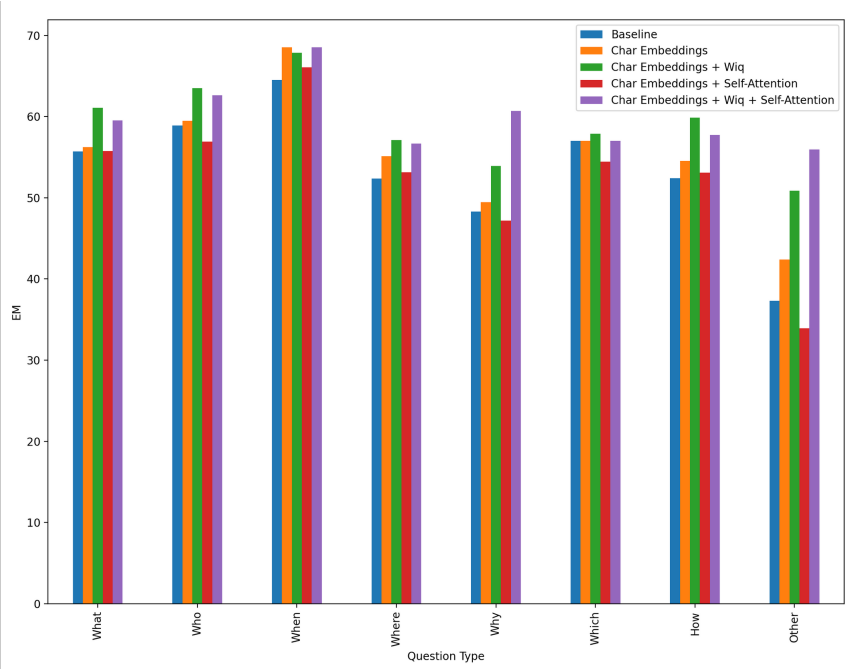

Figure 6: EM Breakdown by Question Type and Model

7 Conclusion

In conclusion, we found that the addition of character-level embeddings slightly improved on the performance of the baseline at the cost of a minimal increase in training time. On the other hand, adding a self-attention layer to the existing BiDAF model with character-level embeddings resulted in minimally worse performance than baseline. This could be because our implementation of self-attention isn't suited for a QA system. Combining character-level embeddings, a self-attention layer, and the binary-like word-in-question feature led to improved performance on the dev set but worse performance on the test set, likely due to overfitting. Overall, when evaluating all our models holistically, we see that our baseline with char-embeddings performs the best with minimal increase in training time and posting the best test set performance. However, we would like to acknowledge that our word-in-question feature has some potential, having posted a substantially better performance than baseline on the dev set.

In the future, we could do fine-tuning on our parameters rather than using the provided parameters across all the models. Having more specifically tuned parameters for each model could lead to improved results. In addition, we could also focus on implementing additional input features like a weighted word-in-question feature [2] or do ensemble training to improve performance by combining predictions from different models together.

References

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

[2] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Fastqa: A simple and efficient neural architecture for question answering. *CoRR*, abs/1703.04816, 2017.

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

[4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2018.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.