BiDAF with Self-Attention for SQuAD 2.0

Stanford CS224N Default Project - IID SQuAD Track

Michelle Huang
Department of Computer Science
Stanford University
huangmjy@stanford.edu

Abstract

The primary goal of this work is to build a QA system that improves upon a baseline modified BiDAF model's performance on the SQuAD 2.0 dataset. To achieve this improvement, two approaches are explored. In the first one, the modified BiDAF model's embedding layer is extended with character-level embeddings. In the second approach, a self-attention layer is added on top of the existing BiDAF attention layer. The performance of these two approaches is evaluated separately and also when combined together into a single model. The model with character embeddings yielded the best performance on the test set, achieving an EM score of 56.872 and a F1 score of 60.652. The self-attention model performed below expectations overall, though it was the best model when it came to performance on unanswerable questions.

1 Introduction

Question answering (QA) is a widely researched topic in the field of natural language processing. The primary goal of QA systems is to answer a question correctly given a question and an associated context paragraph. This task provides us with a quantitative measure of how well machine learning models can "comprehend" text. QA systems have many commercial applications, with a key application being incorporation into popular search engines such as Google to better serve the information need of humans. These systems allow search engines to directly provide answers to more basic questions, alleviating the need for humans to spend time reading through search results to obtain an answer.

This paper focuses on building a QA system for the Stanford Question Answering Dataset (SQuAD), which was first introduced as part of Rajpurkar et al. [2016] [1]. SQuAD is a structured dataset that contains many (question, context paragraph, answer) triples derived from Wikipedia articles. A more recent iteration of the SQuAD dataset, SQuAD 2.0, introduced adversarial unanswerable questions, where the provided context paragraph does not contain an answer to the question [2]. Rapid progress has been made on the SQuAD challenge since release, with current state-of-the-art models exceeding human performance. Most of the high-performing QA systems leverage pre-trained contextual embeddings [3][4] to achieve their performance, but this project was intentionally limited to using non-contextual GloVe word embeddings. It is important to note that while the state-of-the-art models have exceeded human performance on the SQuAD dataset, the problem of question answering is far from solved because real world data is not structured nicely like SQuAD.

This project aims to improve upon a pre-established baseline QA model by first incorporating character-level embeddings and then adding in a self-attention layer similar to what is described in Wang et al. [2017] [5]. I found that the addition of character-level embeddings improved performance slightly at the cost of a significant increase in training time, while the addition of a self-attention layer to the existing BiDAF model performed worse than the baseline. Combining character-level embeddings with a self-attention layer led to improved performance on the dev set, but this variant performed worse on the test set, possibly due to overfitting and the added complexity.

2 Related Work

Numerous architectures have been developed over the years to tackle the question answering problem posed by the SQuAD dataset. In this section, I provide a brief overview of the two models that inspired this work. Please note that the terms "question" and "query" will be used interchangeably throughout this paper.

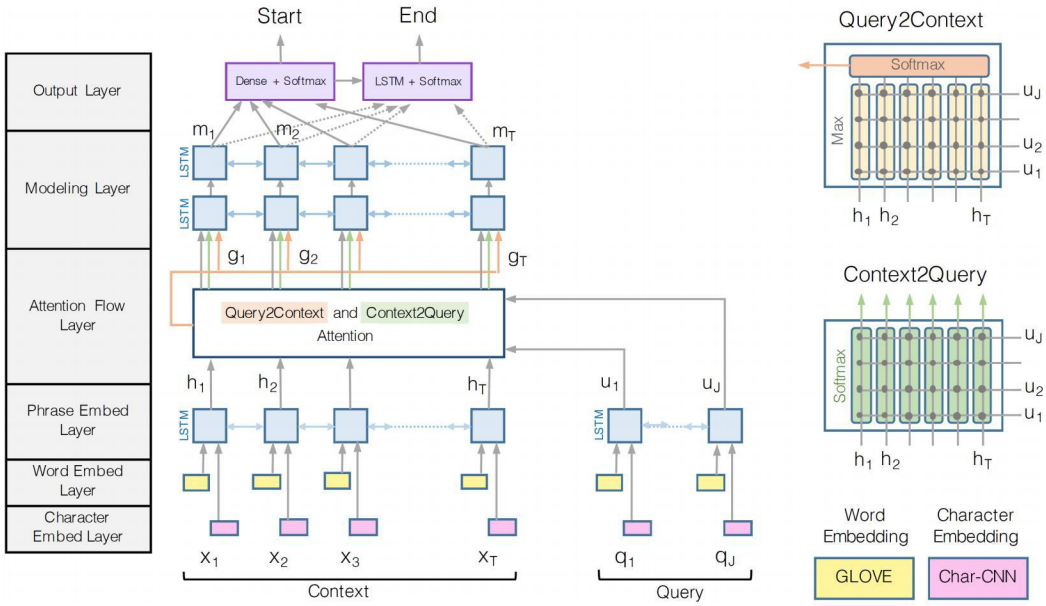

Figure 1: Full BiDAF Model Architecture

Back in 2016, before the rise of models using pre-trained contextual embeddings, the Bidirectional Attention Flow (BiDAF) model introduced one of the more significant advances in QA model architecture [6]. BiDAF utilized a bi-directional long short-term memory RNN (LSTM) with a bi-directional attention flow network to predict the starting and ending positions of the answer in the context paragraph. Here, "bi-directional" means there is both context-to-query and query-to-context attention. This BiDAF model is used as the baseline for this project, though the baseline model differs slightly in that it only contains word embeddings in its embedding layer while the original model's embedding layer contained both word and character-level embeddings.

R-Net, a later model introduced in 2017 [5], builds on the BiDAF architecture by combining BiDAF's context-to-query attention layer with a self-attention layer. R-Net performed better than BiDAF on the SQuAD 1.1 test dataset, achieving an EM score of 71.3 and F1 score of 79.7 compared to BiDAF's EM score of 68.0 and F1 score of 77.3 respectively. Similar to R-Net, this project attempts to improve upon a modified BiDAF model's performance on the newer SQuAD 2.0 dataset by incorporating self-attention into the model architecture.

3 Approach

I extended the baseline model with character-level embeddings and a self-attention layer.

3.1 Baseline

The baseline model used for this project is a BiDAF model without character-level embeddings provided by the CS224N course staff. More details can be found in the project handout.

3.2 Character Level Embeddings

Let $w_1, ..., w_k \in \mathbb{N}$ be input word indices, and let $c_1, ..., c_z \in \mathbb{N}$ be the input character indices corresponding to the word indices. The embedding layer of the baseline model performs an embedding

lookup to convert the word indices into word embeddings $v_1, ..., v_k \in \mathbb{R}^D$ for both the context paragraph and the query, producing embeddings $c_1, ..., c_N \in \mathbb{R}^D$ for the context and embeddings $q_1, ..., q_M \in \mathbb{R}^D$ for the query. To augment the baseline embedding layer, I added character-level embeddings for both the context paragraph and query. First, I performed the embedding lookup to convert the character indices into character embeddings, then I applied dropout and passed the output through a 2D convolutional layer to learn the embeddings. I decided to use a 2D CNN in my implementation because the character-level embedding returns a 4D tensor. After the CNN step, I applied max pooling to the character-level embeddings before concatenating them to the projected word embeddings.

3.3 Self-Attention Layer

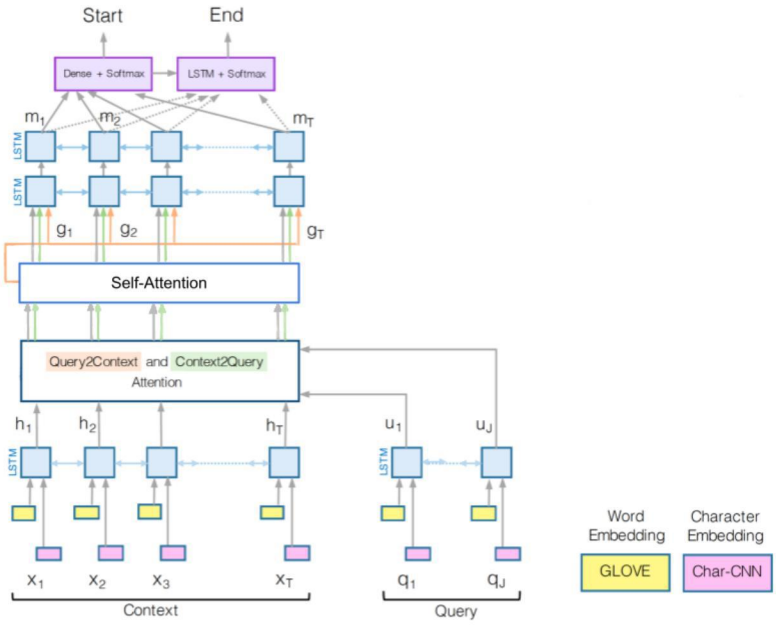

Figure 2: BiDAF Modified with Self-Attention Layer

For the self-attention layer, I used multi-headed attention with 8 attention heads, with each head using scaled dot-product attention, as described in Vaswani et al. [2017] [7]. I included this layer after the BiDAF attention layer.

4 Experiments

4.1 Data

- **Question:** What did the Edict do for Huguenots in France?
- **Context:** The pattern of warfare, followed by brief periods of peace, continued for nearly another quarter-century. The warfare was definitively quelled in 1598, when Henry of Navarre, having succeeded to the French throne as Henry IV, and having recanted Protestantism in favour of Roman Catholicism, issued the Edict of Nantes. The Edict reaffirmed Catholicism as the state religion of France, but granted the Protestants equality with Catholics under the throne and a degree of religious and political freedom within their domains. The Edict simultaneously protected Catholic interests by discouraging the founding of new Protestant churches in Catholic-controlled regions.[citation needed]
- **Answer:** granted the Protestants equality with Catholics

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

This project uses a portion of the SQuAD 2.0 question answering dataset [2], which contains approximately 141,000 (question, context paragraph, answer) triples derived from Wikipedia articles, with answers crowdsourced from Amazon Mechanical Turk. The data is split into training, dev, and test sets with about 130,000, 6,000 and 6,000 examples respectively.

Other data provided by the course staff include pretrained GloVe word embeddings and pretrained character-level embeddings.

4.2 Evaluation method

I will be using the official SQuAD evaluation metrics, which are Exact Match (EM) and F1 score. These two metrics are briefly described below.

- EM: a strict binary metric measuring whether the system output matches at least one of the ground truth answers provided word-for-word
- F1: the harmonic mean of precision and recall, $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

4.3 Experimental details

I trained three different models: (1) baseline with character-level embeddings, (2) baseline with self-attention, and (3) baseline with character-level embeddings and self-attention. Each model was trained for at least 25 epochs, though some ran a bit longer to 30 epochs. Note that there was not a gain in performance for training past 25 epochs, as the dev set negative log likelihood (NLL) would begin to go up due to the model over-fitting on the training data. All of the models were trained with a learning rate of 0.5 and a drop-out probability of 0.2.

4.4 Results

The following results are for the IID SQuAD track.

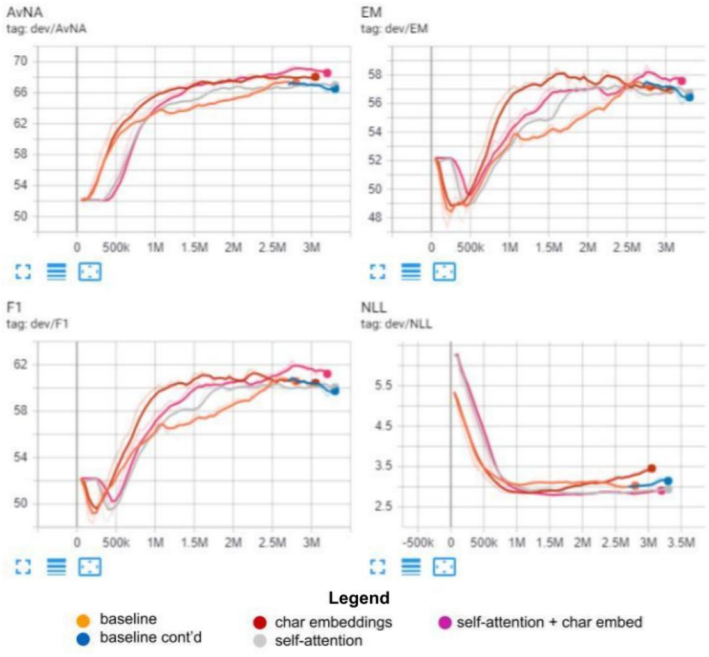

Figure 4: Plots from Tensorboard for various quantitative metrics. Note that the baseline appears in two colors because training was interrupted around epoch 20.

The results were rather disappointing. The model with character-level embeddings was the only one to outperform the baseline on both the dev and test sets, but it was a rather minimal improvement

Model	Dev EM	Dev F1	Test EM	Test F1	Time/Epoch	Total Train Time
No Answer For All	52.193	52.193	-	-	-	-
Baseline	57.520	60.771	56.703	60.348	20 min	8.3 hrs
Char Embeddings	58.310	61.583	**56.872**	**60.652**	38 min	15.8 hrs
Self-Attention	57.251	60.423	-	-	27 min	11.3 hrs
Char Embed + Self-Attn	**58.646**	**62.352**	56.348	60.322	57 min	23.8 hrs

considering the required training time nearly doubled. The models with self-attention performed poorly overall, with the self-attention model performing worse than baseline on the dev set, and the combined character embedding and self-attention model performing better than baseline on the dev set but worse than the baseline on the test set. It was interesting how the models with the self-attention layer followed a slightly different curve for the various quantitative metrics. It is likely that all of these models would have benefited from the finetuning of hyperparameters.

5 Analysis

5.1 Performance on Answerable vs. Unanswerable Questions

The primary difference between SQuAD 1.1 and SQuAD 2.0 is the addition of adversial unanswerable questions. The dev set used for this project contained 5,951 questions, with unanswerable questions making up 52.1% (3103) of those questions.

Model	EM, Answerable Qs	EM, Unanswerable Qs	Prop. N/A Prediction for Answerable Qs
Baseline	**57.690**	55.044	**0.185**
Char Embeddings	56.074	57.267	0.196
Self-Attention	52.110	**60.010**	0.252
Self-Attn + Char Emb	55.337	58.492	0.188

In the table above, we see that the models using self-attention performed better overall on unanswerable questions. For answerable questions, the baseline model actually performed the best. This shows that most of the performance gain from the added model features came from improved performance on unanswerable questions. It is worth noting that the self-attention model predicted "no answer" more frequently for both question types when compared to the other models, and this behavior dragged down its performance for answerable questions.

5.2 Length of Prediction

Model	Mean Prediction Len, Answerable Qs	Median Len, Answerable Qs	Mean Prediction Len, Unanswerable Qs	Median Len, Unanswerable Qs
Baseline	18.37	13	19.32	14
Char Embeddings	19.12	14	20.26	15
Self-Attention	17.96	13	18.14	14
Self-Attn + Char Emb	18.54	13	18.81	14

Note: These mean and median values are calculated after *excluding* "no answer" predictions.

For the length of prediction, even when we do not take into consideration "no answer" predictions, which would drag down the mean and median values across the board, but especially for the self-attention model, the self-attention model tends to predict slightly shorter answers, while the character embedding model tends to predict slightly longer answers. The combined model takes on characteristics of both the self-attention and character embedding models and ends up in the middle of the two. Something else to note is that when the models choose to predict an answer for unanswerable questions, these predictions actually tend to be slightly longer than the predictions made for answerable questions.

5.3 Performance by Question Word Type

In addition to answerable and unanswerable questions, the SQuAD questions can also be broken down by what question word they contain. For this analysis, I chose to keep it relatively simple and break the questions down into what, when, where, who, which, why, and how questions. Questions that did not fit any of these categories were labelled as "other."

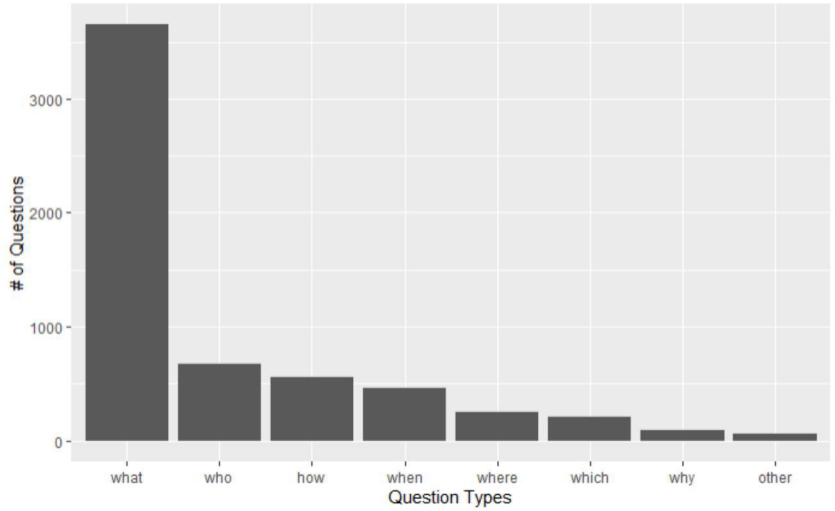

Figure 5: Distribution of Dev Set Questions by Question Word Type

We can see that the dev set used for this project is very imbalanced when it comes to question type, with "what" questions being much more common than the other types.

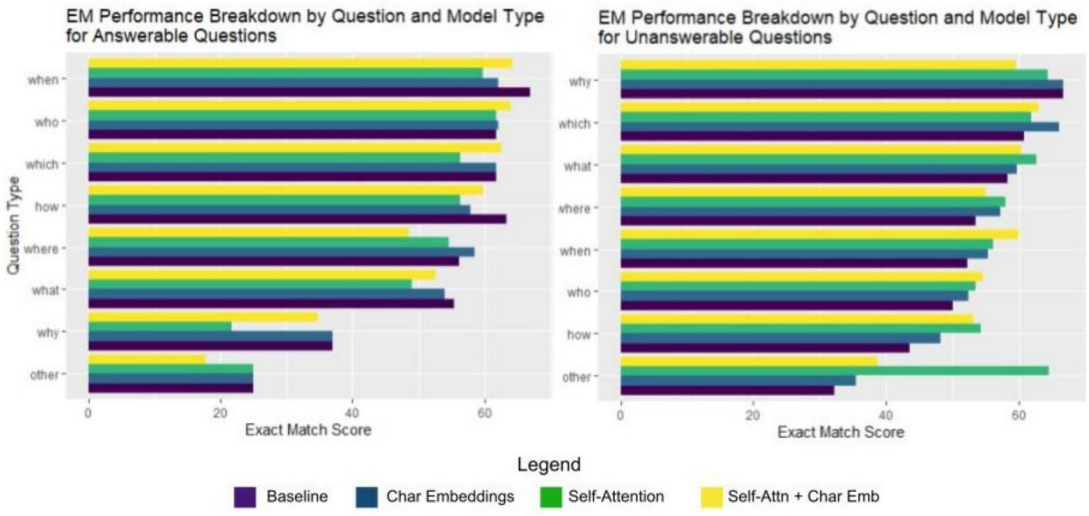

Figure 6: Breakdown of Performance by Question and Model Type

For answerable questions, the models generally perform the best on "when" questions, followed by "who" and "which" questions, though performance does vary by model. For example, the self-attention model performs better on "who" questions compared to other question types, and the combined model really struggles with "where" questions. All the models do poorly on "why" and "other" questions, and this may be due to a lack of training and dev set data for these two question types.

For unanswerable questions, "other" is still a weak point, but the models do surprisingly well on "why" questions. This may be because the models are more likely to predict "no answer" when they have less data and lower prediction confidence. Here, the self-attention model is a clear outlier in terms of performance on "other"-type questions, but this high success rate may be because of the small sample size of "other" questions in the dev set. It is also interesting how the combined model struggles with "why" questions compared to the other models, but it performs the best on "when" questions. The character embedding model performs quite consistently across the board and always matches or beats the baseline model's performance.

6 Conclusion

In conclusion, I found that adding character-level embeddings gave a small performance boost over the baseline model. Adding a self-attention layer, on the other hand, did not improve overall performance because adding the layer increased the model's tendency of predicting "no answer" on answerable questions. The combined model with both character-level embeddings and a self-attention layer performed better than the character-level embeddings model on the dev set, but this combined model performed poorly on the test set, possibly due to the added complexity. When evaluating all of my models holistically based on a combination of performance and required training time, the baseline actually performed the best. While incremental improvements can have impact, there is power in having a simpler model, especially when training time is a consideration.

Due to time constraints, I was not able to experiment with finetuning hyperparameters, which could have lead to improved results. Beyond finetuning, future work could focus on shifting to an entirely different model architecture like Transformer-XL [8] rather than simply adding a self-attention layer to the baseline BiDAF model.

References

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

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

[3] Jacob Devin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv preprint arXiv:1810.04805*, 2018.

[4] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *arXiv preprint arXiv:1802.05365*, 2018.

[5] Natural Language Computing Group and Microsoft Research Asia. R-net: Machine reading comprehension with self-matching networks. In *Microsoft Research*, 2017.

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

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *arXiv preprint arXiv:1706.03762*, 2017.

[8] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Rusian Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length content. In *arXiv preprint arXiv:1901.02860*, 2019.