Exploring Question Answering on SQuAD 2.0 Using Character Embeddings, Self-Attention, and QANet

Stanford CS224N Default Project
Mentor: Kendrick Shen

Constance Horng
Department of Computer Science
Stanford University
constanh@stanford.edu

Daniel Ma
Department of Computer Science
Stanford University
dma1@stanford.edu

Jialin Zhuo
Department of Computer Science
Stanford University
zhuoj@stanford.edu

Abstract

In this paper, we explore the Question Answering task on the SQuAD 2.0 dataset using two models: BiDAF (Bi-Directional Attention Flow) and QANet, which leverages local convolution and self-attention. First, we added character embeddings and self-attention to the baseline BiDAF model, which improved our F1 and EM scores to 63.64 and 60.29. Then, we implemented the QANet model, which failed to improve performance beyond the baseline but was nonetheless rewarding to experiment with.

1 Introduction

Recently, we have seen a rise in end-to-end neural networks to perform question answering tasks. QA systems aim to answer the input question given the context paragraph, and therefore have many important applications such as search engines and dialogue systems. Designing an effective QA system is a complicated task that requires many considerations, such as linguistic nuances within different wordings and the possibility of unanswerable questions. Much of the past work done in the question answering field leverages pre-trained language models (PLMs), but recent work has seen a trend towards attention-based models rather than recurrent models (RNNs). Attention mechanisms allow models to establish connections among different network layers. As such, even though we focused the first part of our project on enhancing the BiDAF model using character embeddings and self-attention, we also attempted to implement QANet, an attention-based model that leverages local convolution.

Overall, we were able to achieve a notable performance increase with our implementation of character embeddings (F1 = 63.64, EM = 60.29). The addition of self-attention also improved scores beyond the baseline model, but unfortunately did not surpass the character embedding model. Additionally, our QANet implementation achieved linear scores of 52.19 due to some minor implementation issues that we were unable to debug within the time constraints of this course. However, it was rewarding to experiment with a complex model, learn how to implement an encoding system with convolution modeling local interactions and self-attention modeling global interactions, and explore question answering beyond RNNs.

2 Related Work

The first section of our project focuses on optimizing the BiDAF model's performance. In 2016, Seo [1] proposed the BiDAF model, which is a hierarchical multi-stage architecture for modeling context paragraph at different token sizes such as character-level, word-level, and contextual embeddings. This is an advancement to the existing work surrounding the attention mechanism. In this model, the attention layer is computed for every time step, allowing the attention vector from the previous layer to flow through to the next layer. Traditionally, the attention layer summarizes the context into a fixed size vector. The new model does not use early summarization, helping it retain more information. In addition, the attention is memory-less, which means each step's attention is independent from the previous step as it is a function of only the query and the current context paragraph. This is useful because incorrect attention will not carry through the layers, allowing for more distribution of work between the attention and model layer. Most importantly, the paper's method uses attention from both directions, query-to-context and content-to-query, which allows complimentary information to flow both ways.

Yoon Kim's [2] work inspired our implementation of character embeddings to the BiDAF model. At a high level, Kim's work used a single convolutional layer and max-pooling for sentence-level classification tasks.

We also experimented with self-attention, which relates different positions of a single sequence in order to compute a representation of the same sequence. Self-attention has been shown to be very useful in machine reading, abstractive summarization, and image description generation. Specifically, we examined the self-attention mechanism as outlined by the R-Net model introduced by Microsoft Natural Language Computing Group [3]. First, the network matches the question and passage with gated attention-based recurrent networks (RNNs) to obtain the question-aware passage representation. Then, it uses a self-matching attention mechanism to refine the representation by matching the passage against itself.

Finally, QANet differs from BiDAF in that the encoder block is made up of convolution and multi-head self attention with no recurrence [4]. The idea of only using self attention and feed forward blocks to model natural language was introduced by the network architecture Transformers [5]. Multi-head attention is also applied by both the Transformer and QANet models. By computing the attention numerous times, it was observed that the model could simultaneously attend to input from distinct representational sub-spaces at different places, resulting in higher scores.

3 Approach

3.1 BiDAF

1. **Baseline** We used the provided fully-functional BiDAF model with pre-trained 300-D GloVE word embeddings as a starting point for our project. The model is based on the BiDAF network [1].
2. **BiDAF with character embeddings** The baseline model only had word embeddings, and we improved it by adding a 200-D character-level embedding layer. We concatenated the character embeddings with the word vectors and fed that through an encoder layer, Context-query attention layer, modeling layer and output layer. More specifically, we implemented a CNN with one layer of 2D convolution followed by a kaiming normal of the character projection weights to account for the non-linearity of activation functions (i.e. ReLU activation). Then, we max-pooled the CNN's outputs to get a fixed-size vector for each word. The output of the CNN was inputted into the highway network. [6]. Our main motivation for adding the character embeddings was to capture meanings at a more granular level, which helps with predicting out-of-vocabulary and unseen words.
3. **BiDAF with self-attention** Next, we implemented self-attention as outlined in the "R-Net: Machine Reading Comprehension With Self-Matching Networks" paper, which uses the current passage word and its matching question information to extract evidence from the whole passage. The resulting passage representation h_t^P is as follows [3]:

$$h_t^P = BiRNN(h_{t-1}^P, [v_t^P, c_t])$$

where $c_t = att(v^P, v_t^P)$ is an attention-pooling vector of the whole passage (v^P):

$$s_j^t = v^T tanh(W_v^P v_j^P + W_v^{\tilde{P}} v_t^P)$$
$$a_i^t = exp(s_i^t) / \sum_{j=1}^{n} exp(s_j^t)$$
$$c_t = \sum_{i=1}^{n} a_i^t v_i^P$$

3.2 QANet

The QANet model follows the same high level implementation as the BiDAF model, and contains five major components: an embedding layer, an embedding encoder layer, a contex-query attention layer, a model encoder layer, and an output layer. This high level implementation is shown in Figure 1 on the left, which is taken directly from the QANet paper. Rather than RNN/LTSMs, the layers in the QANet model are built using transformer style encoder blocks, with the high level implementation being shown in Figure 1 on the right. Our implementation was inspired by Yu et al. [4] and the repository suggested on Ed (https://github.com/BangLiu/QANet-PyTorch).

1. **Embedding Layer**
   For our embedding layer, we reused the word and character embedding implementation we designed for BiDAF.
2. **Encoder Block**
   This is the main component of QANet. A sinusoidal positional encoding is added to the input to the Encoder Block, before being fed into a series of three residual blocks, which consists of a series of convolution layers, a self-attention layer with 8 heads, and a feed-forward layer. This is illustrated on the left of Figure 1. Each of the residual blocks performs the following operation:

   $$f(x) = x + Layer(Layernorm(x))$$

   The Layernorm function represents the layer-normalization that each residual block performs on the embedding dimension.

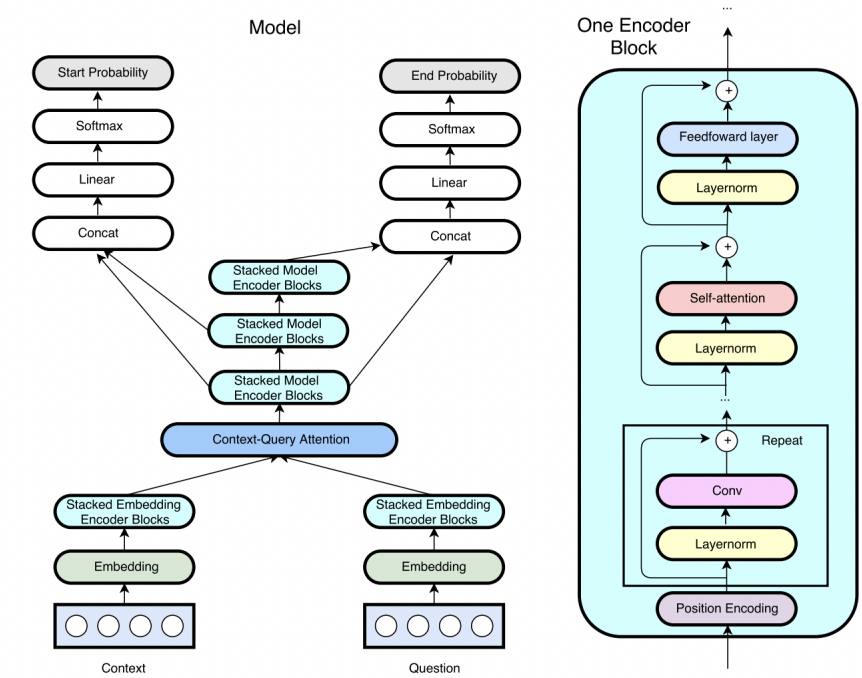

Figure 1: A visualization of the QANet architecture. **Left:** High level structure of the QANet Model. **Right:** High level structure of a single Encoder Block.

3. **Embedding Encoder Layer**
   This layer encodes the context and the question. It is made up of a single encoder block, with parameters defined in Yu et al [4]. For this layer, we used a kernel size of 7, and 4 convolution layers with filter size 128.
4. **Context-Query Attention**
   We applied the BiDAF context-query attention mechanism to QANet. [1]
5. **Model Encoder Layer**
   This layer consists of a series of 3 stacked model encoder blocks. Each one contains 7 encoder blocks, again with parameters defined in Yu et al [4]. For each encoder block, we used a kernel size of 5, and 2 convolution layers with filter size 128.
6. **Output**
   Our output layer computes the softmax using two different concatenations of encoder outputs. Let M_0, M_1, M_2 denote the output of the encoder block stack. The loss function is computed as the negative sum of log probabilities of the predicted distributions averaged over all training samples. Here, we predict the probability of each position in the context being the start or end of an answer span:

$$P_{start} = softmax(W_0[M_0; M_1])$$
$$P_{end} = softmax(W_1[M_0; M_2])$$

where W_0 and W_1 are learnable parameters.

4 Experiments

4.1 Data

We used a subset of the official SQuAD 2.0 dataset, which has already been split into the training, dev, and test sets, with each example in the dataset being split into (context, question, answer) triplets. The training set is 40 MB, consisting of 129,941 examples, the dev set is 4 MB, consisting of 6078 examples, and the test set is 4 MB, consisting of 5915 exmaples. We used the provided preprocessing code to retrieve the word tokens and populate our counters.

4.2 Evaluation method

To evaluate our model's performance, we used Exact Match (EM) and F1 scores. EM is a strict binary measure indicating whether the output matches ground truth, whereas F1 is the harmonic mean of precision and recall. As indicated in the spec, we averaged the EM and F1 scores across each set to get our final scores.

4.3 Experimental details

For every model, we trained on the SQuAD 2.0 training set provided using a learning rate of .01, a batch size of 64, a dropout probability of 0.1, and a maximum answer length of 15. The models were trained on NV6 Azure VM.

For the BiDAF model, we trained for 30 epochs in about 3 hours with the default hyper-parameters. After implementing character embeddings, training took over 4 hours and improved the scores by over 2 points (as shown below). We then changed our RNN type from bidirectional LSTM to GRU, which improved training efficiency to about 3 hours. Unfortunately, using GRU slightly decreased scores. We also experimented with regulation by using different dropout rates, but it did not cause a notable improvement.

For the QANet model, we trained for up to 5 epochs, with a batch size of 16 to avoid memory errors on the Azure VM, and a hidden size of 128 to match the number of filters in the Encoding Block that was suggested by Yu et al [4]. As will be explored in the Results and Analysis sections, due to the abnormal plateauing behavior of the QANet model, we deemed it unnecessary to waste additional resources training past 5 epochs to observe a longer score plateau. For this model, we experimented with learning rates of 0.5, 0.1, 0.005, and 0.001, as well as a learning rate warmup for 1000 steps

from 0 to 0.001. Unlike the original QANet implementation, we added a dropout after both the self-attention layer normalization and the feed-forward layer normalization in the Encoder Block, in addition to once every other layer for the convolution layers. Additionally, for the Encoder Block in the QANet model, we integrated the stochastic layer dropout suggested by Yu et al [4]. This method randomly drops individual sublayers within the Encoder Block with probability $p_l = 1 - \frac{1}{L}(1 - p_L)$, where L is the total number of sublayers and $p_L = 0.9$ is the survival probability of the last layer.

4.4 Results

Model	F1	EM
Baseline	61.72	58.37
BiDAF + Character Embeddings	**63.64**	**60.29**
BiDAF + Character Embeddings + GRU	61.10	57.57
BiDAF + Self-Attention	61.12	58.24
BiDAF + Character Embeddings + Self-Attention	62.43	58.42
QANet	52.19	52.19

Table 1: EM and F1 scores for 6 models we experimented with on the dev set.

After our explorations with character embeddings, self-attention, and QANet, we found that our BiDAF model with added character embeddings optimized performance the most. It achieved F1 and EM scores of 63.64 and 60.29 respectively on the dev set, and 63.24 and 59.81 on the test set, which marked a notable increase from the given baseline model.

However, our QANet model seemed to plateau at EM and F1 scores of 52.19, which is less than what the baseline BiDAF model achieved.

5 Analysis

The character embedding model optimized performance by conditioning on the internal structure of words, improving the handling of out-of-vocabulary words as compared to word-level embeddings.

Unfortunately, adding self-attention did not improve performance beyond the baseline. This is likely due to the limited nature of a single self-attention layer. In the future, we would like to explore more attention mechanisms like co-attention (as outlined in the R-Net paper) to hopefully improve F1 and EM scores.

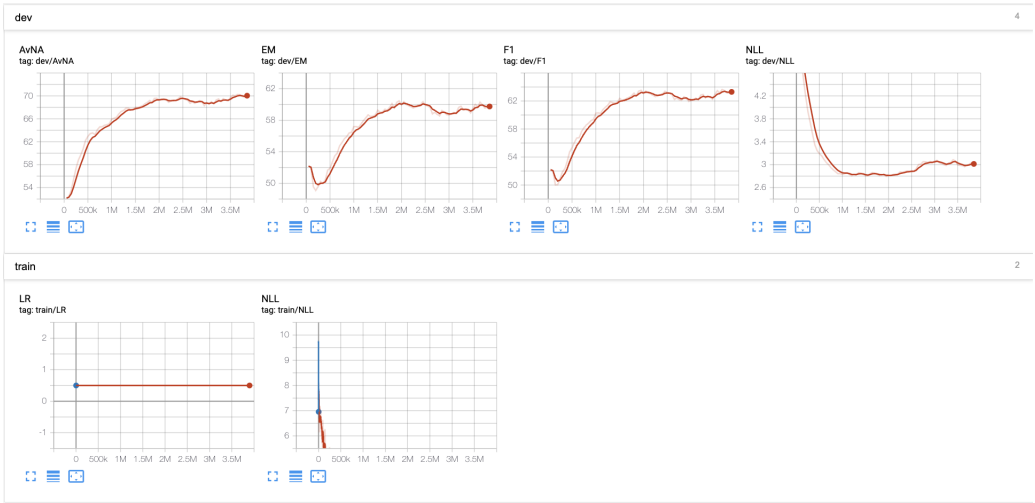

Figure 2: Tensorboard for BiDAF Model With Character Embeddings

Our QANet implementation also ran into some challenges as initially, our F1 and EM scores trained down towards 0, while NLL continued to decrease towards 0 as well. After observing the outputs of each of the layers in the Encoder Block, we noticed an error in the implementation of the position encoder that all inputs are run through. This lead to incorrect high probabilities being predicted, which lowered the NLL, but did not return the correct answer. After fixing the position encoder, our F1 and EM scores plateaued to a linear value of 52.19, and the AvNA score plateaued at 52.14 (shown in Figure 3). These trends occurred as the dev NLL was still decreasing, and train NLL was close to 0. After some investigation, we found that the model was predicting No Answer for all questions, which led to this perplexing result. We experimented with LeakyRelu using a negative slope of 0.01 and 0.1, lower learning rates, a learning rate warmup, and dropout rate reductions, but were unable to fix the model to achieve higher scores.

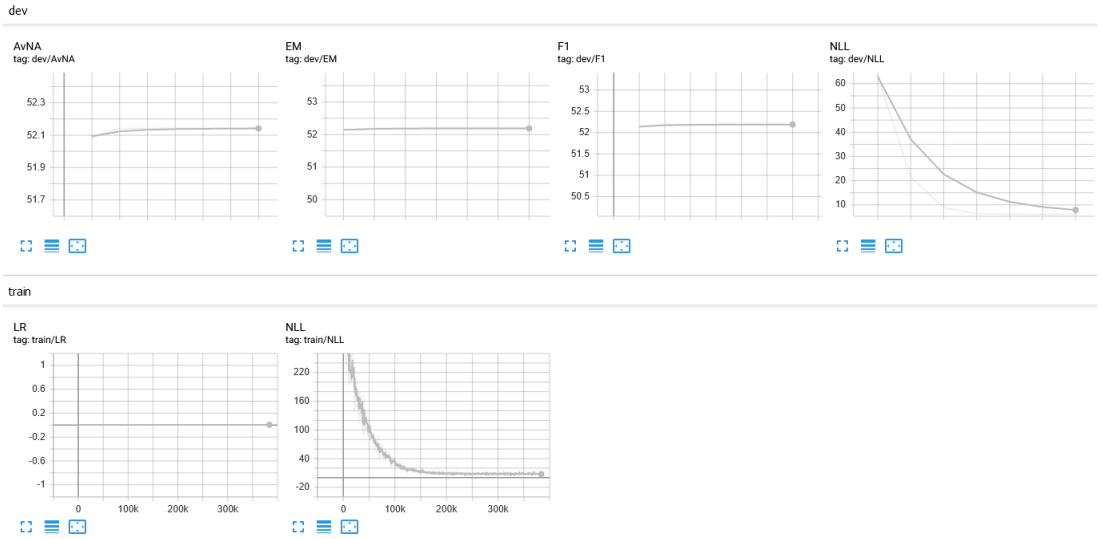

Figure 3: Tensorboard for QANet Model With Character Embeddings

6 Conclusion

In this project, we explored the question answering task using the BiDAF and QANet models. First, we implemented and improved the baseline BiDAF model with the addition of character embeddings and a self-attention layer. Then, we re-implemented the QANet model, which unfortunately resulted in plateauing scores. We trained these models on the SQuAD 2.0 dataset. According to our findings, the modifications in the embedding layers led to the greatest impact.

In the future, we would like to explore other attention mechanisms such as co-attention to hopefully optimize performance further.

References

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

[2] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

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

[4] 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.

[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.

[6] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

A Appendix (optional)