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

Constance Horng Daniel Ma Jialin Zhuo

Department of Computer Science, Stanford University

Introduction

We explored two end-to-end models to optimize performance for the Question Answering task on the SQuAD dataset: BiDAF (Bi-Directional Attention Flow) and QANet. First, we added character embeddings and self-attention to the baseline BiDAF model, and we found that character embeddings improved the EM and F1 scores by a considerable amount. Then, we implemented QANet, which leverages local convolution and self-attention.

BiDAF Model

The original baseline model contains an embedding layer, encoder layer, bidirectional attention layer, modeling layer, and output layer. The provided baseline model only included word embeddings, so we added character embeddings as outlined in the Seo 2016 paper to improve performance. We chose to implement a 2D convolution neural network for this model.

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 [1]:

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

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

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

Character Embeddings Tensorboard

We were able to optimize performance the most using the BiDAF model with character embeddings, as shown in the below Tensorboard:

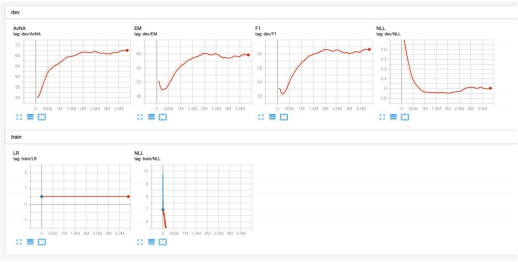

Figure 1. Tensorboard for BiDAF Model With Character Embeddings

QANet Model

1. **Embedding Layer**
   We broke down the question strings into sequences of word and character tokens to prepare our input. The first 16 characters of the word were concatenated, then each embedding was expanded to 128 characters through convolution with 1-D kernels of size 5. Then, to get a vector representation of each word, we took the maximum across the character dimension. Finally, we applied a two-layer highway network on the embedding vector. We performed this process for every word in the context and question.
2. **Encoder Block**
   This is the main component of QANet. First, we have the positional encoding, which consists of both sinusoidal and cosinusodial encodings. Then, we have several convolution layers, a multi-head attention layer with 8 heads, and a feed-forward layer. All of these layers are followed by a normalization layer to stabilize the hidden state dynamics. We use the encoder block to process the input embeddings and bi-directional attention outputs (Figure 1).

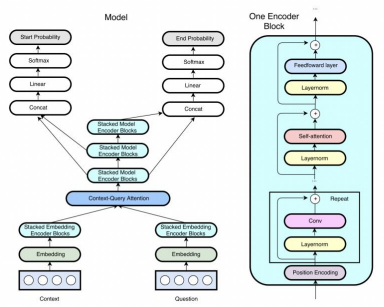

Figure 2. A visualisation of the QANet architecture

3. **Context-Query Attention**
   We applied the BiDAF context-query attention mechanism to QANet. [2]
4. **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.

Experiments

Data

We used the SQuAD dataset, which has already been split into the training, dev, and test sets. The training set is 40 MB, and the dev set is 4 MB. We used the provided preprocessing code to retrieve the word tokens and populate our counters, then added code to also retrieve character tokens.

Model Evaluation

To evaluate the models we explored, we looked at 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.

Results

Model	EM	F1
Baseline	61.72	58.37
BiDAF + Character Embeddings	63.64	60.29
BiDAF + Character Embeddings + GRU	61.10	57.57
BiDAF + Self-Attention	61.17	58.74
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.

Analysis

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 EM and F1 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.

Unfortunately, adding self-attention did not improve performance beyond the baseline. In the future, we would like to explore more attention mechanisms like co-attention (as outlined in the R-Net paper) to hopefully improve EM and F1 scores.

Our QANet implementation also ran into some challnges as our F1 and EM scores plateaued to a linear value of 52.19. We experimented with LeakyRelu, lower learning rates, and dropout rate reductions, but were unable to achieve higher scores.

References

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

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

[3] 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.
2018.