

QANet on SQUAD 2.0

Stanford CS224N Default Project

Siyuan Yuan

Department of Geophysics
Stanford University
syyuan@stanford.edu

Abstract

QANet achieved the state of the art prior to BERT on the SQUAD 2.0. The project aims to reimplement QANet based on a model from the Attention is All You Need paper. We also revised the provided BiDAF model by adding a character embedding layer. We find that with the character embedding layer, BiDAF model is significantly improved, and we show that ensembling the QANet and BiDAF model can evidently improve the performance on the SQUAD 2.0 dataset.

1 Key Information to include

- Mentor: None
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

Machine reading comprehension is an active research topic gaining more and more attention recently. Reading comprehension is an important task, because it can be used to assess how well computers understand natural languages, and many NLP tasks can be reduced to the reading comprehension problem. The current state-of-the-art model is dominated by large-scale pretrained language model, e.g. BERT and ELMo [1]. Prior to the BERT era, QANet [2] and BiDAF [3] models achieved the state-of-the-art. There were lots of original ideas behind the two models, which worth spending time to learn and understand. This project focus on revising the provided BiDAF model and reimplementing the QANet model.

To gain more understanding the QANet, we modified the 2D Convolutional layer in the character embedding to 1D depth separable layers. We change the number of heads in the attention layers, and we ensembled the BiDAF and QANet models.

The following sections provides an overview of the research context, the architecture of our model, details of implementation, results, analysis and summary.

3 Related Work

Our baseline model is Bi-directional Attention Flow (BiDAF) model, which combines word-level, character-level and contextual embeddings to obtain a query-aware context representation. Embeddings of the contexts and questions are merged through bi-directional attention flow. R-Net [4] is another popular model that leverages the idea of gated attention-based recurrent networks and self-matching attention technique to refine the question-aware passage representation.

Most of the models prior to QANet uses RNN to embed the contextual meanings of the questions and passages. QANet adopted attention convolutional mechanisms that are easy to parallel to avoid slow

RNN-based encoder. QANet achieves both fast speed and high accuracy. Transformer-XL introduces a concept of recurrence which reuses the historic representations to refine the standard Transformer. Transformer-XL is shown to be capable of capturing the long-term dependency, which evidently boosts the performance of the model.

4 Approach

4.1 Model architecture

The BiDAF model we used is based on the provided starter code. We added a character embedding layers to handle the Out-of-Vocabulary (OOV) issue. We implemented our QANet model by adapting the provided starter code and the attention model from the a pytorch code of the "Attention is all you need" paper. The QANet model consists of five layers: embedding layer, stacked embedding encoder blocks, an Context-Query attention layer, stacked model encoder layer and an output layer. Figure 1 shows the architecture of the QANet model adopted from the original paper.

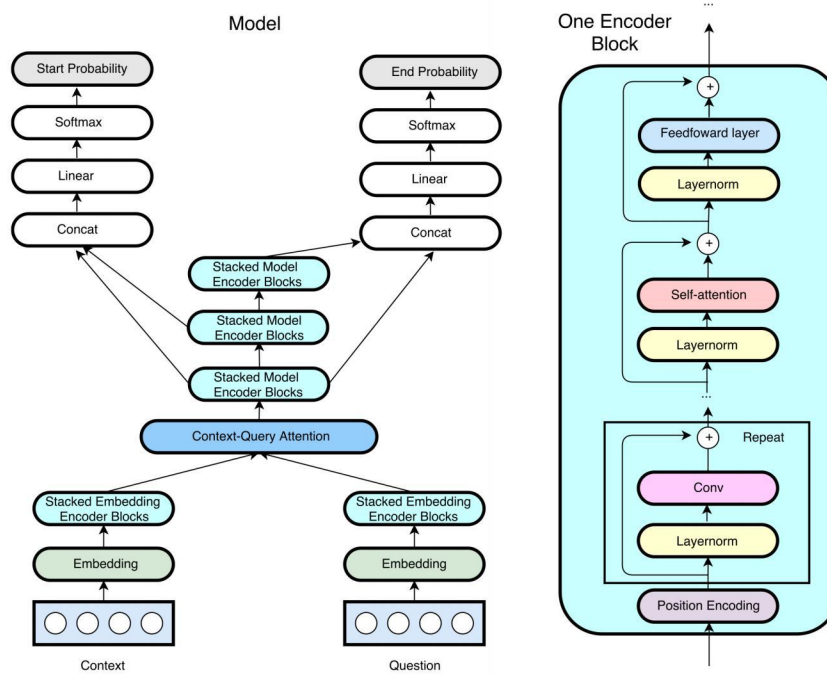


Figure 1: Model architecture of the QANet (adopted from the original paper).

4.2 Embedding layer

Both word and character are embedded and concatenated. Following the original paper, the dimension for the word and character embeddings are 300 and 200, respectively. The two embeddings are concatenated followed by a 1-D convolution layer to shrink the dimension from 500 to the hidden size, which is set to be 128, as used in the original paper. To save parameters and speed up, I used 1-D depth separable convolution layer instead of 2-D Conv to computer the character embeddings. I implemented the character embedding layer myself.

4.3 Embedding encoder layer

Both passage and question embeddings are processed by this layer. The first part is positional encoding, which aims to add word order information to the attention mechanism. My implementation

of the positional encoding is adapted from code of the "Attention is all you need" paper. The next part is repeated layernorm and conv layers. I repeated those layers only twice to avoid overfitting and saving memory. I adapted the multihead self-attention layer from the code of the "Attention is all you need" paper. The feedforward layer aims to add non-linearity. We implemented the feedforward layer as two stacked 1D Conv layers.

4.4 Context-Query Attention

Our Context-Query Attention is adopted from <https://github.com/andy840314/QANet-pytorch/blob/master/models.py>.

4.5 Model encoder blocks

The layer takes the output of Context-Query Attention as inputs. The implementation details is the same as the embedding encoder blocks.

4.6 Output layer

The output layers combines the outputs from the model encoder blocks to compute the log probability for the start and end position at each context word using softmax function. We implemented the output layer by ourselves. We adapted the

5 Experiments

This section contains the following.

5.1 Data

The dataset we used is SQuAD 2.0 which contains the unanswerable questions compared the version 1.0.

5.2 Evaluation method

The metrics we used to evaluate the models are F1 score and Exact Match (EM). F1 score is used during training to select the models with the best performance on dev set.

5.3 Experimental details

We made a few experiments. There are two experiments for the BiDAF models. We train the provided BiDAF model, and the model with the character embedding layers. For the QANet models, we train one with four heads and another one with six heads in the self-attention module. For the character embedding parts, we experimented with 1-D separable convolution layer and 2-D conv layer. We evaluated all trained models on the provided dev and test sets. We ensemble the modified BiDAF model and the four-head QANet model through majority vote on the test and dev set.

We run all of experiments with a learning rate of 0.5. The decay rate for exponential moving average of parameters is 0.999. For the BiDAF baseline model, we run 30 epochs. Due to the time constraints, for the other models, we only managed to run 22 epochs.

5.4 Results

The results of the experiments are shown in Figure 2. From the figure, we can see that the curve with the highest EM (59.79) and F1 (63.25) score comes from the BiDAF model with char embedding added. The second best model is the provided BiDAF model which gives the F1 of 60.69 and EM of 57.28. The red curve corresponds to QANet with four-head attention. Its best F1 and EM are 54.18 and 52.11. The blue curve refers to six-head QANet with its best F1 and EM of 54 and 51.44. We did model ensembling with the modified BiDAF and QANet with six heads. It achieves the best F1 and EM among all of the models. Its F1 and EM on the dev set are 65.11 and 62.31, respectively. Its leaderboard (IID SQuAD track) results are 61.48 and 64.05, respectively.

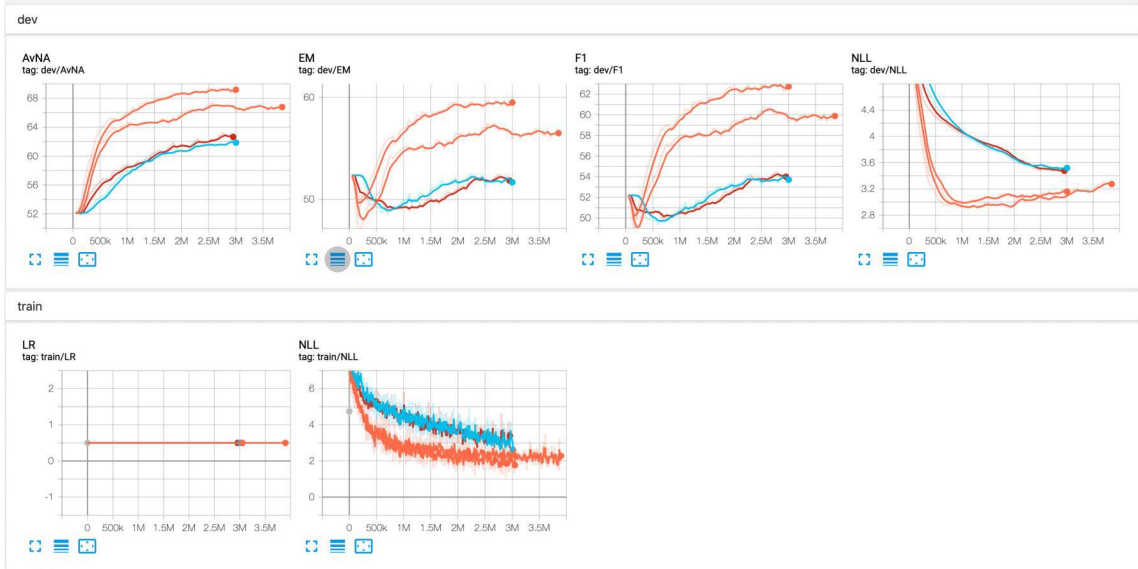


Figure 2: Experiment results. The curve with the highest EM (59.79) and F1 (63.25) score comes from the BiDAF model with char embedding added. The second best model is the provided BiDAF model which gives the F1 of 60.69 and EM of 57.28. The red curve corresponds to QANet with four-head attention. Its best F1 and EM are 54.18 and 52.11. The blue curve refers to six-head QANet with its best F1 and EM of 54 and 51.44.

6 Analysis

The results suggests that the BiDAF model with character embeddings performs the best. We can see from the fact that BiDAF with charEmb outperforms the original BiDAF even with fewer epochs, charEmb evidently improve the performance of BiDAF. The reason can be that CharEmb can learn the meanings of the OOV words in the context. In our case, QANet had worse results compared to BiDAF. The reason can be that for QANet, we runned much fewer iterations than needed for it to get converged. From the NLL plot in Figure 2, we can see that the blue and red curves referring to the QANets are still having decreasing trend. We think it's fair to believe that given more time to train the models, QANets can potentially get more accurate. We can also see that attention with four heads or six heads doesn't influence the results much.

7 Conclusion

We added character embedding layers to the provided BiDAF model, and we observe that character embedding evidently improve the performance. We reimplemented QANet based on the provided starter code and code from the paper, "Attention is all you need" (Link:<https://github.com/jadore801120/attention-is-all-you-need-pytorch/blob/132907dd272e2cc92e3c10e6c4e783a87ff8893d/transformer/>). CQ Attention module is adopted from <https://github.com/andy840314/QANet-pytorch/blob/master/models.py>. The QANet achieve worse results compared to BiDAF, which can be explained as less iterations to get it converged. We see an model ensemble of BiDAF and QANet improve the performance.

References

- [1] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.
- [2] Adams Wei Yu and Minh-Thang Luong David Dohan2. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*, 2018.

- [3] Ali Farhadi Hannaneh Hajishirzi Minjoon Seo, Aniruddha Kembhavi. Bidirectional attention flow for machine comprehension. In *ICLR*, 2017.
- [4] Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. May 2017.

A Appendix (optional)