Reproduce Simple QANet on SQuAD 2.0

Stanford CS224N {Default} Project

Zhengqi Zhu
Stanford Center for Performance Development
Stanford University
`stevezhu@stanford.edu`

Abstract

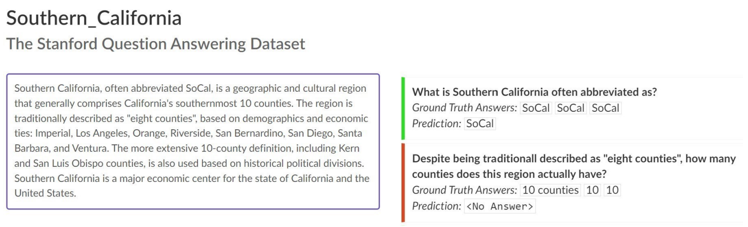
By the time QANet[1] paper was published, most end-to-end machine-reading and question answering models are primarily based on recurrent neural networks (RNNs) with attention. Due to the sequential nature of RNNs, these models are often slow for both training and inference despite their success on accuracy and reliability. This project reproduced a simple Q&A architecture called QANet. The major innovation of this model is that its encoder consists exclusively of convolution and self-attention, while recurrent networks are not required. On SQuAD 2.0 dataset, reproduced QANet model achieved a 63.855 EM score and 67.675 F1 score on the dev set, which outperformed the given baseline model on both EM and F1 scores with only half of the epoch number trained. However, since the QANet model size is significantly larger than the baseline model, the training time of each epoch is around 6x longer than the given baseline model.

1 Key Information to include

- Mentor: N/A
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Question Answering (QA) has been a hot topic in the natural language processing area for many years, it aims to answer a question about a given context or document which are posed in natural language. Given a context paragraph and a question, we would like to build a QA system that outputs a contiguous span from the context paragraph that answers the question, or mark that the given question is unanswerable.



Southern_California

The Stanford Question Answering Dataset

Southern California, often abbreviated SoCal, is a geographic and cultural region that generally comprises California's southernmost 10 counties. The region is traditionally described as "eight counties", based on demographics and economic ties: Imperial, Los Angeles, Orange, Riverside, San Bernardino, San Diego, Santa Barbara, and Ventura. The more extensive 10-county definition, including Kern and San Luis Obispo counties, is also used based on historical political divisions. Southern California is a major economic center for the state of California and the United States.

What is Southern California often abbreviated as?
Ground Truth Answers: SoCal SoCal SoCal
Prediction: SoCal

Despite being traditionall described as "eight counties", how many counties does this region actually have?
Ground Truth Answers: 10 counties 10 10
Prediction: `<No Answer>`

Figure 1: Sample data of SQuAD 2.0

This task is interesting because QA systems have been widely used in many important applications such as search engines, customer supports, and educational applications. However, there are some difficulties with the system. Firstly, natural language is complex, and understanding the meaning of a sentence needs to consider various factors. A word could represent multiple meanings based on the subtle nuances of the context and a small change could lead to the opposite meaning of a sentence. Secondly, the system not only needs to find the answer from the given input, but it also needs to determine if the question is answerable. This asks the system to develop a mechanism that could figure out none of the facts can be inferred as an answer.

QANet is one of the top-level performance models on SQuAD 1.0 and my motivation is to explore its performance on SQuAD 2.0. Besides, there is a clear trend that more and more models are attention-based. Since the major innovation of QANet is to exclusively use convolution and self-attention in the encoder, without the recurrent nature, I chose QANet to re-implement an attention-based model.

Our tasks are as follows:

- Reproduce the original QANet architecture from scratch using PyTorch.
- Make sure the re-implemented QANet model achieves a higher EM and F1 score compared to the given baseline model.
- Explore extensions to the original architecture.
- Evaluate the performance of QANet on SQuAD 2.0 and analyze the limitations.

To explore the improvements on the original QANet architecture, I applied a mask on self-attention to remove unnecessary attention and focus on the attention of the keywords. As a result, the QANet model outperformed the given baseline model on both EM and F1 scores with only half of the epochs trained.

3 Related Work

- Transformer[2] model deeply influenced QANet. Transformer model based solely on attention mechanisms and achieved remarkable performance on translation task. QANet chose to use convolution and self-attention mechanisms exclusively and abandoned the recurrent mechanism entirely. Without the recurrent nature, the QANet model reached a 3x to 13x faster training time compared to the state-of-art recurrent models by the time the paper was published.

- Bi-Directional Attention Flow (BiDAF) network[3] introduced a multi-stage hierarchical process that represents the context at different levels of granularity and the attention not only flow from question to the context, but also flow from the context to the question, which builds question-aware representations of the context without early summarization. This model is also our baseline model. My goal for this project is to re-implement a QANet model that outperforms the BiDAF model.

- Dynamic Coattention Networks[4] (DCN) is also used in the original paper. In order to focus on relevant parts of both the question and the context, the DCN first fuses co-dependent representations of both. Then a dynamic pointing decoder iterates over potential answer spans. This iterative procedure enables the model to recover from initial local maxima corresponding to incorrect answers.

4 Approach

- **Baseline**:

  According to the default project handout, the baseline model is based on Bidirectional Attention Flow (BiDAF)[3], except the baseline model does not include a character-level embedding layer. The model contains an embedding layer, an encoder layer, an attention layer, a modeling layer and an output layer. After setting up, training, and testing on the dev set, I got an EM score of 56.02 and an F1 score of 59.57.

- **QANet**:

  Similar to the baseline model and most of existing models, QANet model contains five major components: an input embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer and an output layer. The major difference between QANet and other methods is that QANet discarded RNNs and only use convolutional and self-attention mechanism for both embedding and modeling encoders.

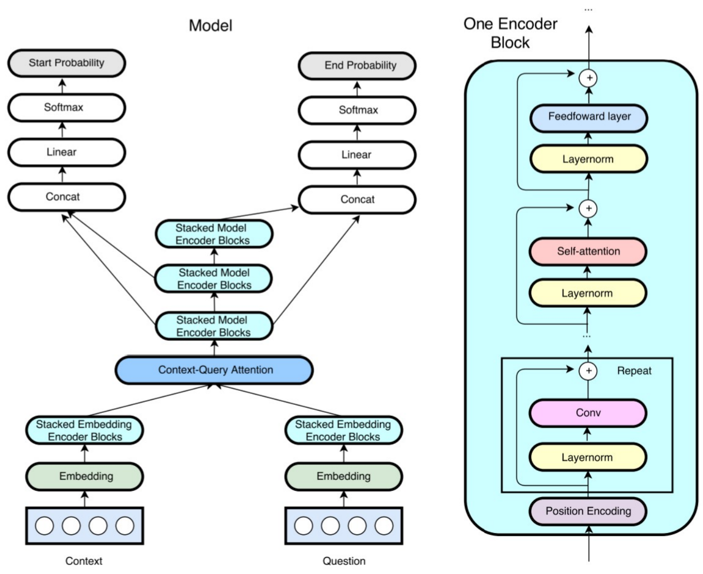


Figure 2: QANet Model architecture

1. Input Embedding Layer:

The embedding of each word w is obtained by concatenating its word embedding and character embedding. The word embedding is fixed during training and initialized from the $p_1 = 300$ dimensional pre-trained GloVe [5] word vectors. All the out-of-vocabulary words are mapped to an <UNK> token, whose embedding is trainable with random initialization. To obtain the character embedding, each character is represented as a trainable vector of $p_2 = 200$ dimension. The word length is truncated or padded to 16. The output from this layer of a word x is $[x_w; x_c] \in R^{p_1+p_2}$, x_w is the word embedding and x_c is the convolution output of character embedding of x.

2. Embedding Encoder Layer:

The structure of encoder layer is [n $\times$ convolution-layer + self-attention-layer + feed-forward-layer]. The convolution-layer has kernel size of 7, $d = 128$ filters and the each block has 3 convolution-layers. The self-attention-layer adopts the multi-head attention mechanism[2]. The number of heads is 8. The total number of encoder blocks is 1. The output of this layer is $d = 128$ dimensions.

3. Context-Query Attention Layer:

C denotes the encoded context and Q denotes the encoded query. First compute the similarities between each pair of context and query words, rendering a similarity matrix$S \in R^{n\times m}$. Then apply softmax function to each row of S to get a matrix $\bar{S}$. The context-to-query attention is computed as

$$A = \bar{S} \cdot Q^T \in R^{n\times d}$$

The similarity function used here is the trilinear function[3]:

$$f(q,c) = W_0[q, c, q \odot c]$$

The query-to-context attention is computed as

$$B = \bar{S} \cdot \bar{\bar{S}}^T \cdot C^T$$

where $\bar{\bar{S}}$ is column normalized matrix of S computed by softmax function.

4. Model Encoder Layer:

The input of this layer at each position is $[c, a, c \odot a, c \odot b]$[3], where a and b are respectively a row of attention matrix A and B. The number of convolution-layer is 2 and total number of blocks are 5. Other parameters are the same as the Embedding Encoder Layer.

5. Output Layer:

Each example in SQuAD is labeled with a span in the context containing the answer. The probabilities of the starting and ending position are modeled as

$$p^1 = softmax(W_1[M_0; M_1])$$

$$p^2 = softmax(W_2[M_0; M_2])$$

where W_1 and W_2 are two trainable variables and M_0, M_1, M_2 are the outputs of the three model encoders from button to top. Compute the product of its start position and end position probabilities to get the score. Finally, the objective function is defined as the negative sum of the log probabilities of the predicted distributions indexed by true start and end indices, averaged over all the training examples:

$$L(\theta) = -\frac{1}{N}\sum_{i}^{N}[log(p^1_{y^1_i}) + log(p^2_{y^2_i})]$$

where y^1_i and y^2_i are respectively the ground truth starting and ending position of example i. At inference time, the predicted span (s, e) is chosen such that $p^1_s p^2_e$ is maximized and $s < e$.

- **Extension**:

I adopted the attention mask technique to help improve the model's performance. The motivation of the attention mask is that not all keys should be included in the attention. Because in some cases, we will pad our queries for a practical purpose which doesn't mean these paddings have any relationships with our query. As a result, the attention score of the correct position will decrease. Attention mask is a binary indicator of whether the indices in the tensor are padded, so that model can ignore them.
The masked attention score is given by:

$$Score_{new} = Mask \cdot Score_{old} + (1 - Mask) \times -10^{30}$$

After applying the attention mask, the padding position of the dot product score is masked with an extremely small value. Then in the following softmax operation, these positions will receive a probability that is very close to 0, as the softmax function maps the value to [0, 1]. Therefore our correct position will have a higher attention score.

I apply attention mask on self-attention, context-query attention, and final output layer. However, I utilize log softmax instead of regular softmax at the output layer. As the loss function of the model is Negative Log-Likelihood which is given by:

$$l(x,y) = \begin{cases} \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n}} l_n, \ reduction = mean \\ \\ \sum_{n=1}^{N} l_n, \ reduction = sum \end{cases}$$

where $l_n = -w_{y_n} x_{n,y_n}$ and $w_c = weight[c] \cdot \mathbf{1}\{c \neq index_{ignored}\}$

And from the formula, we know that our loss would be negative as l_n is negative. As we want our final output positive, we need to turn our input x_n into negative. Hence we choose the log softmax, which takes the logarithm of regular softmax and so the output value of softmax would be negative, and the final output of our model would be positive.

5 Experiments

- **Data**:
  As described in the default project handout, the dataset is SQuAD 2.0[6]. This dataset was being pre-processed as given and split into three parts: train, dev, and test. The train and dev sets are publicly avaliable and the test set is entirely secret. The train set contains 129,941 examples, the dev set contains 6078 examples, and the test set contains 5915 examples.

- **Evaluation method**:
  As introduced in the default project handout, the evaluation metrics are EM and F1 scores, and no-answer(AvNA) scores. Exact Match is a binary measure (i.e.true/false) of whether the system output matches the ground truth answer exactly. F1 is a less strict metric –it is the harmonic mean of precision and recall.

- **Experimental details**:
  For hardware, I used Google CoLab NVIDIA Tesla P100 GPU with 16 GB memory. Due to the limitation of GPU memory, I have to decrease the number of blocks of each encoder layer.
  - Baseline:The epoch number is set to 30 and batch size of 64. The training time is around ten minutes per epoch. The learning rate is 0.5 as default, and the drop probability is set to 0.2.
  - QANet (first version): The epoch number is set to 30 and batch size of 32. But due to Google CoLab auto-disconnect mechanism, the training was stopped at 7. The training time is around one hour per epoch. The learning rate is 0.5 as default, and the drop probability is set to 0.2.
  - QANet (second version) + mask: The epoch number is set to 15 and batch size of 32. The training time is around one hour per epoch. The learning rate is 0.5 as default, and the drop probability is set to 0.2.

- **Results**:
  The results of the performance of the Baseline model and implemented model are shown below graph and table. Both the graph and the table are showing the results on the dev set. In the Figure 3, the blue line represents the QANet model implemented from scratch and the orange model represents the baseline model performance. It is clear to say that the QANet model outperforms the baseline model. On the dev leaderboard, the QANet + mask model got a 63.86 EM score and a 67.68 F1 score, whereas the baseline model has a 56.02 EM score and 59.57 F1 score. One thing to notice is that I improved the QANet model after the milestone so the improvement of the QANet + mask does not exclusively depend on the mask mechanism.

  On test leaderboard, QANet + mask model got 62.418 EM score and 65.749 F1 score.

Model	EM	F1	Loss	AvNA	Epochs
Baseline	56.02	59.57	3.30	63.54	30
QANet	57.08	60.42	2.58	68.68	7
QANet+mask	**63.86**	**67.68**	**2.47**	**73.84**	**15**

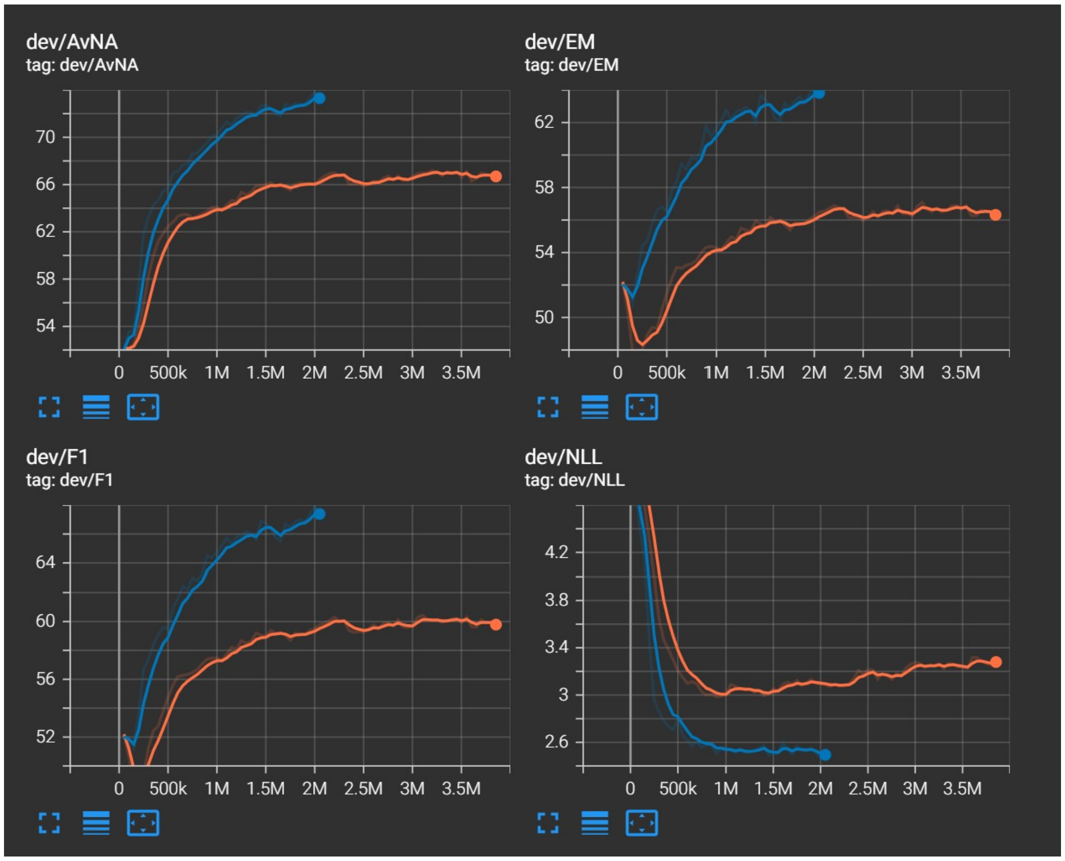


Figure 3: Result: Blue represents QANet and Orange represents the baseline

6 Analysis

- Dev/Test set score discrepancy:

  One observation from the result is that the QANet has an EM/F1 score of 63.86/67.68 on the dev set and 62.418/65.749 on the test set. Both scores on the test set are slightly lower than on the dev set. One possible reason could be the model has an over-fitting problem. To solve this, one possible way is to increase the drop probability value and another possible way is to use data augmentation techniques. The original paper introduced a data augmentation technique they used. Translate the original dataset to French and then translate it back. In this way, the amount of data would be doubled. Their result with 3x data augmentation increased around 1.5 on both EM/F1 scores. However, due to the limitation of time and I am doing this project individually, I chose to not apply data augmentation techniques at this time.

- Training and inference time is slow:

  The major goal of the original QANet paper was to discard the RNNs and make the model train faster. They did get a remarkable training time improvement. However, in my second version of QANet, the training time for each epoch is around 6x the baseline model. Even though the model achieved higher EM/F1 scores than the baseline model with only 15 epochs, which is half of the epochs trained in the baseline model, the overall training time is 3x the baseline model. One reason is that the QANet model is more complex and is significantly larger than the baseline model. As described in the experiment details, I have to shrink the model size by reducing the number of blocks and number of layers in order to make it run on the 16GB memory GPU. The model still occupied 14.58 GB of memory after the shrinking. Another possible reason the training time is long is that the

Google CoLab is notoriously slow. I chose to use Google CoLab is because by the time I started this project Azure was not available.

7 Conclusion

In conclusion, I successfully re-implemented QANet on SQuAD 2.0 and applied mask mechanism on self-attention, context-query attention, and final output layer. The QANet achieved a strong improvement compared to the given baseline model. It is my first time reproducing a paper and I learned that some of the results and performance may not be able to be reproduced exactly. In this project, QANet has both a higher EM/F1 score and slower training speed, although speeding is one of the major achievements of the original paper. Except for the training time, the original model size is reduced in this project due to GPU memory limitation.
One possible future work could be analyzing how QANet synergies with embedding from pre-trained language models. Another possible future work could test our QANet on other Question Answering datasets to see if the model has an over-fitting problem on SQuAD 2.0 dataset. Lastly, in this project, I do not have enough time to do data augmentation as mentioned in the original paper. Since the amount of data significantly influences the performance of a model, this task should also be included in future work.

References

[1] Minh-Thang Luong Rui Zhao Kai Chen Mohammad Norouzi Adams Wei Yu, David Dohan and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations (ICLR)*, 2018.

[2] Niki Parmar-Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Ashish Vaswani, Noam Shazeer and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

[3] Ali Farhadi-Hannaneh Hajishirzi Minjoon Seo, Aniruddha Kembhavi. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*, 2017.

[4] Richard Socher Caiming Xiong, Victor Zhong. Dynamic coattention networks for question answering. In *International Conference on Learning Representations (ICLR)*, 2017.

[5] Richard Socher Jeffrey Pennington and Christopher D. Manning. Glove: Global vectors for word//w representation. in empirical methods in natural language processing (emnlp). In *Association for Computational Linguistics (ACL)*, 2014.

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