Building a QA system (IID SQuAD track)

Stanford CS224N Default Project

I-Chien Lai
Department of Electrical Engineering
Stanford University
lai1998@stanford.edu

Pei-Wei Kao
Department of Electrical Engineering
Stanford University
pwkao@stanford.edu

Ta-Hsuan Chao
Department of Electrical Engineering
Stanford University
thchao@stanford.edu

Abstract

Our final project aims to implement a question answering system that works well on SQuAD 2.0 by capturing long-term dependency. We implemented a Transformer-based model on QA task, QANet to improve the performance compared to the baseline model, BiDAF. In addition, we add character embedding and applied the Transformer-XL to the original QANet to model long-term dependency. We modified the starter code and trained our QANet according to the original paper, and yielded better results than the baseline.

1 Key Information to include

- Mentor: Fenglu Hong
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

The advent of artificial intelligence and machine learning have greatly eased many tasks of our daily lives. The most successful tasks are those related to images, namely, computer vision, and also, the spoken language. In this final project, we focus on one very interesting task, which is reading comprehension. The task is difficult because even humans have difficulty doing reading comprehension asks. One has to fully understand the passage and the question in order to get the right answer, thus the task would even more difficult to teach machines how to answer reading comprehension problems. The automation of this process would bring great convenience in various applications related to the spoken language.

Thus, the goal of this project is to implement a question answering system that works well on SQuAD 2.0 by capturing long-term dependency. As we believe that Transformer-based model is a trending structure in recent years, we would like to implement a Transformer-based model on QA task, namely, the QANet [1], to see how it improve the performance on SQuAD 2.0, and compare it to the baseline model BiDAF [2].

To achieve better performance on longer contexts, we want to further adapt the ideas from Transformer-XL [3] to QANet to see if it could learn longer-term dependencies of the texts. If we can consider the entire context at once by applying Transformer-XL, we might see the improvement on SQuAD dataset or other longer context QA dataset. Thus, we added the key points of Transformer-XL

(Segment-Level Recurrence with State Reuse and Relative Positional Encoding) to our QANet, so as to solve longer context problems.

In this final project, we would show our results by comparing the EM/F1 scores with our baselines. Besides comparing the performance of several models, we will discuss some comparisons and ablation studies of the components in the QANet model to see which part is more beneficial to the task. In addition, we would make experiments on whether adding the character embedding on top of word embedding would improve the performance on the QA task.

3 Related Work

Various end-to-end models of machine learning or deep learning aim to solve this problem. As the Recurrent Neural Network (RNN) [4] is one the first successful networks to deal with natural language processing tasks. The RNet [5] is the one of the first successful approaches to solve the Question Answering task which uses the RNN network. This method adds the concept of self-attention to the RNN on the task. It adds two addition attention layers to the RNN, namely the Context-to-Question attention layer and a Self-Matching Attention layer.

In addition to the RNet, the Bidirectional Attention Flow (BiDAF) [2] furthermore introduces the Bidirectional Attention on top of the RNN. It is a very powerful concept added to the original RNN, and achieved very promising results. Ths method is also the default baseline provided by the TA in the handouts. It is a meaningful baseline , and thus we would compare it with the other models in this final report.

Since adding more seen knowledge of the model can also improve the model, the DrQA [6] further improves the Question answer task by adding additional information. This method based on the RNN, but adds additional semantic features learned from other web language sources, namely, Wikipedia. The added tags of words include features representing its frequency, part-of-speech tag, named entity type, etc. This added tagging would greatly improve the Question Answering task.

The most successful end-to-end models solving these problems generally have two key ingredients: (1) recurrent model to process sequential inputs, and (2) attention component. Although these models already lead to strong results on many challenging datasets like the SQuAD dataset, the QANet [1] points out that one crucial weakness is that these models are often too slow on both training and inference. This could make the models hard to train on larger dataset due to the training speed, and could prevent the models being deployed in real-time applications due to slow inference. Hence, we choose to implement the QANet, and we aim to make the machine comprehension fast by removing the recurrent nature of the past models, and exclusively use convolutions and self-attentions instead.

To further improve the task to answer questions with longer contexts, the Transformer-XL [3] solves the problem of QANet by introducing the notion of recurrence into the self-attention network by reusing the hidden states obtained in previous segments. While reusing the hidden states, it is necessary to use relative positional encodings rather than absolute ones. The method introduces two components, the Segment-Level Recurrence with State Reuse and Relative Positional Encoding. The Segment-Level Recurrence with State Reuse method states that, while training, the hidden state sequence computed for the previous segment is fixed and cached to be reused as an extended context when the model processes the next new segment. The fundamental idea of Positional Encoding is to only encode the relative positional information in the hidden states. We can create a set of relative positional encodings $\mathbf{R} \in^{L_{max} \times d}$, where the i-th row $\mathbf{R}_i$ indicates a relative distance of i between two positions.

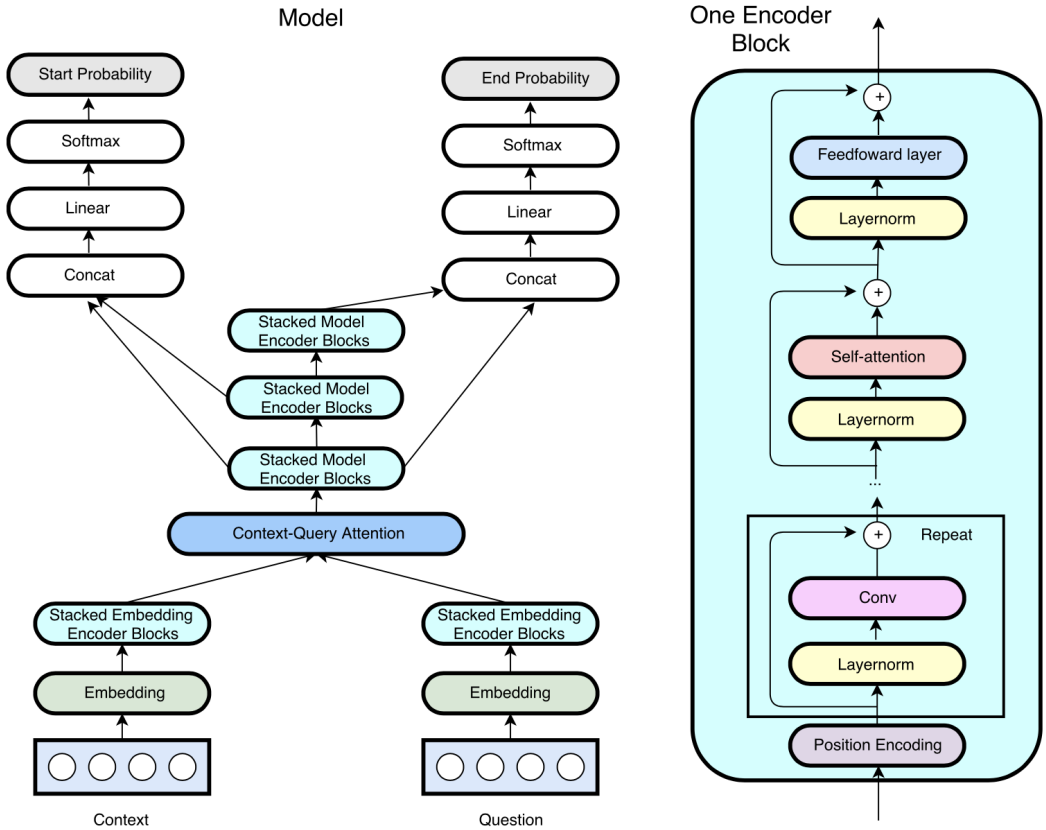
4 Approach

We choose to implement QANet [1], a Transformer-based model that combines local convolution with global self-attention on QA task. We also applied Transformer-XL [3] to the original QANet to model longer-term dependency. We adopted the key points of Transformer-XL (Segment-Level Recurrence with State Reuse and Relative Positional Encoding) to our QANet to see if it can improve the performance, we called this new model QANet-XL.

For embedding, we added character embedding in addition to word embedding. Each character will be represented as a trainable vector, and each word can be viewed as the concatenation of the

embedding vectors for each of its characters. So the output of the embedding layer would be the combination of word embedding and character embedding.

In this report, we will describe how we implement our Pytorch version of QANet-XL, and report the validation and test set scores of the model to see it's performance compared to the baseline.



- **Input Embedding Layer**: We used the standard method to obtain the embedding of each word by concatenating its word embedding and character embedding, which used 300-dimensional pre-trained word embeddings from GloVe [7] and 200-dimension trainable vector for each character. As the QANet paper mentioned, we also adopted a two-layer highway network [8] on the embeddings.
- **Embedding Encoder Layer**: At first, we implemented our encoder layer as the QANet paper mentioned, which is a stack of the following building block: [positional-encoding + convolution-layer × # + self-attention-layer + feed-forward-layer] with residual connection and layer normalization in between. We also adopted the layer dropout regularization method mentioned in the origin paper, where sublayer l has survival probability $p_l = 1 - \frac{l}{L}(1 - p_L)$ where L is the last layer and $p_L = 0.9$. In addition, we added the recurrence mechanism mentioned in the Transformer-XL [3] [1]. We stored the output of the previous hidden layer from the previous segment that allows the model to create long-term dependencies. At the same time, we used the relative distance between tokens instead of their absolute position to do encoding. More details about these two key factors are introduced below.
  - **Segment-Level Recurrence with State Reuse**: During training, the hidden state sequence computed for the previous segment is fixed and cached to be reused as an extended context when the model processes the next new segment.
  - **Relative Positional Encodings**: : The fundamental idea is to only encode the relative positional information in the hidden states. We can create a set of relative positional encodings $\mathbf{R} \in^{L_{max} \times d}$, where the i-th row $\mathbf{R}_i$ indicates a relative distance of i between two positions.

[1] Transformer-XL official github: `https://github.com/kimiyoung/transformer-xl`

- **Context-Query Attention Layer**: We implemented our context-query attention layer as the standard module used in almost every previous reading comprehension models, which is the same as BiDAF's starter code.
- **Model Encoder Layer**: This layer used the same block as the Embedding Encoder Layer, except that the convolution layer number is 2 within a block and the total number of blocks are 7.
- **Output layer**: We concatenated the outputs of Model Encoder Layer M_0, M_1, and M_2, and passed them through a convolutional layer referring to BangLiu's implementation [2]. At last, passed the outputs to masked-softmax layers to get the start probability and end probability.

5 Experiments

5.1 Data

We used the SQuAD 2.0 [9] provided by the starter repository. The dataset has three splits: train (129,941 examples), dev (6078 examples), and test (5915 examples).

5.2 Evaluation method

We used the F1 score and Exact Match (EM) score. The F1 score measures the portion of overlap tokens between the answer predicted by our model and the ground truth. Thus, even if the answer predicted is not totally correct, it would still gain some scores in the F1 score. The EM score on the other hand, is 1 if the prediction of our model is exactly the same as ground truth, and 0 otherwise. This metric is stricter than that of the F1 score.

5.3 Experimental details

For experiment, we implemented three different models on this task and all reached good results, including BiDAF as the baseline, QANet, and QANet-XL. All three models integrated the pre-trained word embedding and character embedding in the Input Embedding layer. We trained these models on Azure's virtual machine (Standard NC6 v3), with the following settings. The dropout rate, number of heads, and the hidden size are set to 0.1, 8, and 128, respectively. For QANet-XL, the memory length is set to 256.

5.4 Results

Table 1 and 2 showed that our QANet has successfully exceeded the baseline model BiDAF in both the EM and F1 score. This indicated that removing the recurrent networks in the encoder could help improve the performance on this task. And our QANet model was not only accurate but also fast, because the model was fully feed forward, composed entirely of separable convolutions, attention, linear layers, and layer normalization, which was suitable for parallel computation.

Although the results of QANet on validation set outperformed BiDAF a lot, the improvement on test set was slight, which didn't match the results in the QANet paper. We thought it was because we didn't implement the data augmentation trick mentioned in the paper on our QANet model . Referring to the paper, they achieved significant gains by utilizing data augmentation consisting of translating context and passage pairs to and from another language as a way of paraphrasing the questions and contexts. We implemented most of the tricks and innovation methods mentioned in the paper instead of data augmentation. We might reach a higher performance if we applied the data augmentation method to our model in the future.

On the other hand, QANet-XL should improve due to the recurrence mechanism and the relative position encoding, but the results on both validation set and test set showed that QANet-XL didn't achieve a higher score compared to the original QANet. One of the reasons we thought was that the paragraphs in SQuAD dataset are not that long (maximum context length approximately 400 tokens), they don't need to rely on Transformer-XL to achieve a remarkable result. And the other reason we thought was the lack of training time. From the training loss visualization, it showed that

[2] `https://github.com/BangLiu/QANet-PyTorch`

the learning rate of QANet-XL was slower than QANet, which implied that the model needed more training epoch to reach the same performance as QANet. Due to the limitation of our computing resources, we couldn't train that long to see if QANet-XL could outperform QANet.

	EM	F1
BiDAF	62.141	65.606
QANet	**64.191**	**67.677**
QANet-XL	60.561	63.430

Table 1: Validation Set Results

	EM	F1
BiDAF	60.575	64.235
QANet	**60.727**	**64.087**
QANet-XL	57.853	61.022

Table 2: Test Set Results

6 Analysis

6.1 Ablation Study

We conducted ablation studies to exam the components of QANet model: the embedding, the self attention mechanism, the layer dropout regularization method, and the feed forward layer. We report the results on validation set in Table 3.

At first, we compared the performance of using only word embeddings with that of adding character embeddings on top of word embeddings as the input. We found that in both single-headed and multi-headed attention model, using character embeddings plus word embeddings achieved better score on validation set than only using word embeddings. It could improve approximately 2~3% EM and 1~2% F1 scores. Then, we compared the results of using single-headed and multi-headed attention. Although multi-headed attention required 2x training time, it could achieve better score than single-headed attention. We believe that it's because multi-headed attention could better generalize the different token-token interactions than single-headed attention using multiple heads, just like the filters in CNN model.

In addition, we examined the robustness of using the layer dropout regularization method in the Encoder Block and the convolutional feed forward layer in both the Encoder Block and output module. Surprisingly, we found that discarding the layer dropout method and using a constant dropout in the Encoder Block improved the performance and achieved a comparable score with the multi-headed attention model. We thought that it might because it reduced the over-fitting on training set, hence, it appears to raise some improvement on validation set. As for the convolutional feed forward layer, using convolution layer as feed forward layer yielded higher results than using linear layer, which shows that convolution layer could better capture the local structure of the context.

	Embedding	**Layer Dropout**	**Attention**	**Feed Forward**	EM	F1	Difference
QANet	word+char	✓	single-head	convolution	62.83	66.4	
	word	✓	single-head	convolution	60.46	64.19	-2.37 / -2.21
	word+char	✗	single-head	convolution	64.85	68.48	**2.02 / 2.08**
	word+char	✓	single-head	linear	61.35	64.86	-1.48 / -1.54
	word+char	✗	single-head	linear	60.61	64.04	-2.22 / -2.36
	word+char	✓	multi-head	convolution	64.19	67.68	**1.36 / 1.28**
	word	✓	multi-head	convolution	62.39	65.77	-0.44 / -0.63

Table 3: Ablation studies on different components of QANet (validation score)

6.2 Error Analysis

We analyzed the outputs of three models: BiDAF, QANet, and QANet-XL (all used character and word embeddings as input), to see how they failed or succeeded on the task. Here we use the results on validation set for analysis.

Table 4 shows an example where all the models successfully output the correct answer. In this case, all the models have the ability to recognize the named-entity 'King George III' in the context to answer the question. This example is considered as an easy question.

Table 5 shows an example where all the models failed to output an non-answerable question. In this example, all the models could not identify the difference between 'input' and 'output'. So they answered the type of input graph in a decision problem as the answer. This example suggests that our models still have room for improvement on distinguishing synonyms and antonyms.

Table 6 shows an example that our QANet model outperforms the other two models: BiDAF and QANet-XL. The BiDAF model couldn't correctly tag the answer span, which just predicted the starting position right after the word 'nucleons'. The QANet-XL model didn't even detect the answer. In contrast, the QANet model successfully predicted the correct start and end positions, which shows that it can better identify the proper answer span and achieve higher exact match score.

uuid	0e6b64edaeab88a3bee8d3ec8
Question	Who issued the Royal Proclamation of 1763?
Context	Following the treaty, **King George III** issued the Royal Proclamation of 1763 on October 7, 1763, which outlined .....
Gold	King George III
BiDAF	King George III
QANet	King George III
QANet-XL	King George III

Table 4: Example for error analysis that all models succeed.

uuid	04ab24c55b6067b4f35e8746c
Question	What type of graph is an example of an **output** used in a decision problem?
Context	An example of a decision problem is the following. **The input is an arbitrary graph.** The problem consists in deciding whether the given graph is connected, or not .....
Gold	
BiDAF	arbitrary graph
QANet	arbitrary
QANet-XL	arbitrary

Table 5: Example for error analysis that all models fail.

uuid	075d3fdf07ba291acc81d747d
Question	What is the force between nucleons?
Context	The strong force only acts directly upon elementary particles. However, a residual of the force is observed between hadrons (the best known example being the force that acts between nucleons in atomic nuclei) as the **nuclear force** .....
Gold	nuclear force
BiDAF	in atomic nuclei) as the nuclear force
QANet	nuclear force
QANet-XL	

Table 6: Example for error analysis that QANet outperforms the others.

7 Conclusion

In this project, we successfully implemented the Question Answering system along with several models and ablation studies. We concluded that Transformer-based model, namely, the QANet, performed better than RNN-based model, namely, the baseline BiDAF. In addition to that, adding character embeddings would also greatly improve the performance of the model compared to that with only word embeddings. However, we also found out that adding the two key points of the Transformer-XL into our model would not improve the performance on SQuAD 2.0. The ablation study shows the robustness of the character embeddings and convolution layers, and surprisingly found that the layer dropout method isn't really beneficial to the QANet model in this dataset. The error analysis supports that the QANet model performs better in proper span tagging hence yields higher exact match score.

Acknowledgement

We would like to thank the course CS224N to give us a chance to learn about the interesting concepts of natural language processing, and to do a final project together on such an interesting problem. We would also like to thank the instructor, Chris Manning, for his diligent teaching, and our mentor, Fenglu Hong, for her informative feedback on our proposal and milestone report.

References

[1] 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.

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2018.

[3] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[4] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

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

[6] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions, 2017.

[7] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[8] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks, 2015.

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