QANet++: A QANet-Based Model for Question Answering

Final Project Report for ECE 449 Machine Learning

Hanyin Shao
Computer Engineering, ZJUI
3170111452
hanyin.17@intl.zju.edu.cn

Yiqing Du
Computer Engineering, ZJUI
3180112696
yiqing.18@intl.zju.edu.cn

Huili Tao
Computer Engineering, ZJUI
3180110990
Huili.18@intl.zju.edu.cn

Xinkai Yuan
Computer Engineering, ZJUI
3180110983
Xinkai.18@intl.zju.edu.cn

Shiqi Yu
Computer Engineering, ZJUI
3180110989
shiqiy.18@intl.zju.edu.cn

***Abstract*—Question Answering (QA) is an important area in Natural Language Processing. Among various Question Answering systems, QANet, which uses CNN and self-attention mechanism in encoder design, can process Question-Answering problems with good performance and short training time. In this project, we modify the original QANet by adding a novel connection from the start probability to the end probability through three different approaches to make further improvements to this model. We then apply those modified models to the SQuAD 2.0 (Stanford Question Answering Dataset 2.0) for evaluation. Comparing to the baseline model, our final model achieves shows better performance, with an EM score of 61.67 and F1 score of 74.14.**

I. Introduction

Question Answering (QA) is one of the important natural language processing (NLP) research area. The current QA system mostly focuses on factoid question, trying to solve the query and answer it with a segment from the provided context as answer. QA system has a wide application in reality, such as Question Answering website, chatbot and so on [1].

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset consisting of questions posed by crowdworkers on a set of Wikipedia articles [2]. It asks the model to answer the question with segments of context in the given article and figure out unanswerable questions.

- Input: Query and Context
- Output: Segment of the context (start index and end index)

The model performance is evaluated based on Exact Match (EM) and F1 score:

- Exact Match (EM): Measures the percentage of predictions that exactly match the ground truth answers.
- F1 Score: Measures the average overlap between the prediction and ground truth answer.

The development in deep learning in recent years strongly contributed to the study on QA systems. Most of the current designed QA models are based on the combination of recurrent neural networks (RNNs) and long term interactions , one of the popular models is Bidirectional Attention Flow (BiDAF) [3]. Though BiDAF has a strong ability in handling the language flexibility, returning a well performance on the SQuAD dataset, its recurrence computation leads to a expensive cost in computation and training time.

We choose QANet models as our baseline to make further improvement. QANet is a closed-domain machine learning model for QA task which composed with convolutions and self-attentions building blocks to encode query and context [1]. We modify the model by taking the start probability as an indicator for the end probability and use transformation with bias to capture the features.

In this paper, we report on the design, modification, and training result of our QANet model. The performance of our model is evaluated through Exact Match (EM) and F1 score with EM measuring the percentage of predictions that exactly match the ground truth answers and F1 score measure the average overlap between the prediction and ground truth answer.

II. Related Work

A. BiDAF

Bi-Directional Attention Flow (BiDAF) network is a multi-stage hierarchical process that represents the context at different levels of granularity and uses bidirectional attention flow mechanism to obtain a query-aware context representation without early summarization [4].

This model does not rely on preexisting knowledge and answer the query by returning the substring of the context and can answer questions whose answers are short factual statements.

BiDAF includes character-level, word-level, and contextual embeddings, and uses bi-directional attention flow to obtain a query-aware context representation. For improvements of

BiDAF, it attention layer is calculated for every time step. The vector computed at the current and previous layers is allowed to flow through to the subsequent modeling layer. This reduces the information loss caused by early summarization. Secondly, its attention mechanism is memoryless, which means the attention at each time step only depends on the query and the context paragraph at the current time step. In this way, attention at each time step to be unaffected from incorrectness at previous time steps and enable the modeling layer to focus on learning the interaction within the query-aware context representation. In addition, both directions, query-to-context and context-to-query, which provide complimentary information to each other.

However, as the BiDAF recurrent structure cannot be parallelly computed, its training process are quite slow. There had been proposed to avoid using RNNs, but it largely sacrifices the performance.

B. BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is inspired by the BiDAF model. BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers, and be fine-tuned with just one additional output layer to create various state-of-the-art models. Two main steps in this frame work is pre-training and fine-tuning [5].

The architecture of BERT is a stack of transformer's encoders. For the Text Processing process, Input representation is constructed by summing the corresponding token, segment, and position embeddings. The pre-training tasks contains Masked Language Modeling and Next Sentence Prediction. Masked Language Modeling randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. It enables the representation to fuse the left and the right context, which allows us to pretrain a deep bidirectional Transformer. The next sentence prediction task jointly pretrains text-pair representations.

C. QANet

QANet, aiming to make the machine comprehension fast, we propose to remove the recurrent nature of these models [1], and instead use convolutions and self-attentions in its feed-forward type architecture. Motivated by the knowledge that convolution captures the local structure of the text, while the self-attention learns the global interaction between each pair of words. It solves the speed problem of BiDAF by introducing convolutional network which can take advantage of GPU parallel computation.

III. Method

A. Baseline - QANet

QANet is the base model for our project. This model can be divided into 5 layers: Input Embedding Layer, Embedding Encoder Layer, Context-Query Attention Layer, Model Encoder Layer, and Output Layer as shown in Figure 1. We strictly follow the methods introduced by the QANet paper [1] to realize the base model. All of our modifications to QANet focus on the Output Layer (red-boxed part).

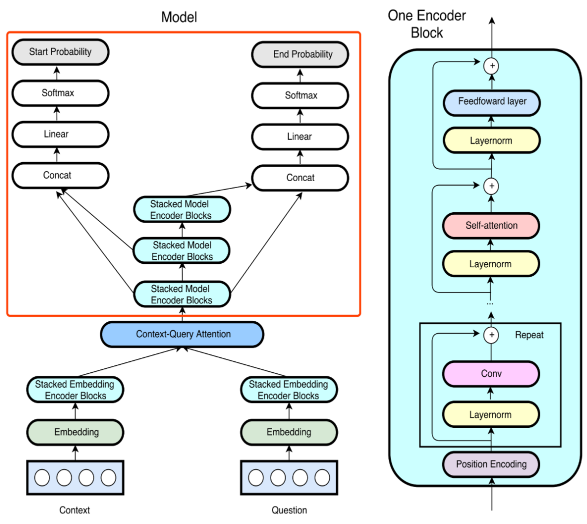

Fig. 1. QANet Structure [1]

Input Embedding Layer In this layer, we first use 300-dimensional pre-trained GloVe [6] word vectors for word and character embedding. For word embedding, all the words that are out of vocabulary are mapped to an <UNK> token. For character embedding, we assume that each word can be viewed as a concatenation of characters. Each character is converted into a 200-dimensional vector. All the words are truncated or padded to a length of 16. For a given input word x, we can get a word embedding output x_w and a character embedding output x_c.

Then, a highway network [7] is applied to the concatenated embedding output $[x_w;x_c]$, which is used to adjust the contribution of word embedding and character embedding. The highway network provides the final output of the whole Input Embdding Layer with a dimention of 500.

Embedding Encoder Layer The Embeeding Encoder Layer is a sequence of encoder blocks. Each encoder block contains four convolution layers followed by a self-attention layer and a feed forward layer. The convolution layer is the depthwise separable convolutions [8] with a kernel size of 7 and a filter number of 128. The self-attention layer follows the multi-head attention mechanism [9] with the number of heads to be 8 for all the layers. Also, after each operation, there is a layer-normalization [10]. For each word, the input dimension is 500; the output dimension is 128, which is reduced by the 1-D convolution.

Context-Query Attention Layer QANet uses the same context-query attention layer which is standard for almost all the previous reading comprehension models [4] [11].

Model Encoder Layer The Model Encoder Layer follows the similar structure as the Embedding Encoder Layer. The only difference is that there are 2 convolution layers in each block and there are 7 encoder blocks in total.

Output Layer This layer outputs the probability of each position in the context to be the start or the end of the answer span. Let M_0, M_1, and M_2 be the output of the three Stacked Model Encoder Blocks respectively, from bottom to top. The probability is calculated as

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

where W_0 and W_1 are two parameter matrices for linear transformation. Finally, as introduced in the QANet paper [1], 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:

$$L(\theta) = -\frac{1}{N}\sum_{i}^{N}[p_{start}(i;\theta) + p_{end}(j;\theta)]$$

where i and j denote the ground truth indices of starting and ending position and θ is the training parameter.

Inference At the inference step, we first generate two independent distributions of p_{start} and p_{end}. Then we search for the span that maximize the production of $p_{start}(s) \cdot p_{end}(e)$, where s and e are the target start index and end index.

B. Modification 1 - QANet+Addition

We notice that in the original QANet, the start probability and the end probability are predicted separately. But we suppose that the start probability should be a good indicator for the end probability. Therefore, we simply add [M_0;M_1] and the output of Softmax together and feed the result to the process of predicting the end probability. The new structure is shown as in Figure 2.

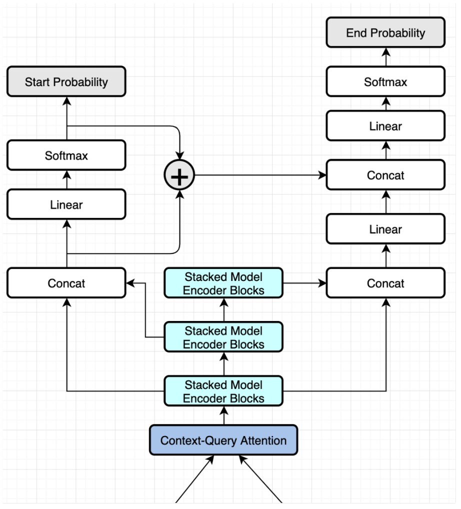

Fig. 2. Modification 1 Output Layer

We use the same notations as in the original QANet. The start and end probabilities are calculated as

$$X_1 = [M_0; M_1]$$

$$Y_1 = Softmax(W_0[M_0; M_1])$$

$$Z_1 = W_1[M_0; M_2]$$

$$p_{start} = Softmax(W_0[M_0; M_1])$$

$$p_{end} = Softmax(W_2[Z_1; X_1 + Y_1])$$

where W_0, W_1, and W_2 are three trainable weight matrices for linear transformation. The addition of X_1 and Y_1 is a point-wise operation.

C. Modification 2 - QANet+Addition+Bias

Then we realize that [M_0;M_1] and the output of softmax are highly dependent, therefore it is unnecessary to use both of them. In the second version of our modification, we only feed the linear output to the process of predicting the end probability.

On the other hand, we notice that in the original QANet, all the linear transformations are without bias. However, we suppose that bias might be helpful in capturing some features and making the model more robust. Therefore, in the last linear layer for predicting the end probability, we add a bias term for the linear transformation. The output layer structure is shown in Figure 3.

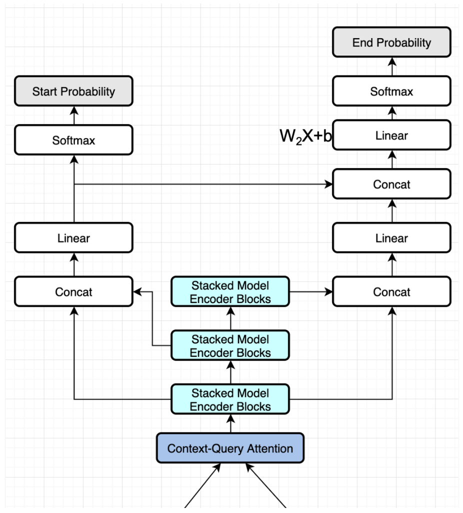

Fig. 3. Modification 2 Output Layer

We use the same notations as previous. The start and end probabilities are calculated as

$$Y_1 = W_1[M_0; M_1]$$

$$Y_2 = W_2[M_0; M_2]$$

$$p_{start} = Softmax(W_0[M_0; M_1])$$

$$p_{end} = Softmax(W_2[Y_1; Y_2] + b)$$

where b is the bias term we newly introduced.

D. Modification 3 - QANet+Multiplication+Bias

We also notice that positions in the context with higher probability of being the start will have higher probability to be the end. Therefore, we propose the third version of the modified QANet. The positions having higher probability to be the start will be more highly activated than others, and the model can utilize this information to predict the end position. The new structure is shown in Figure 4.

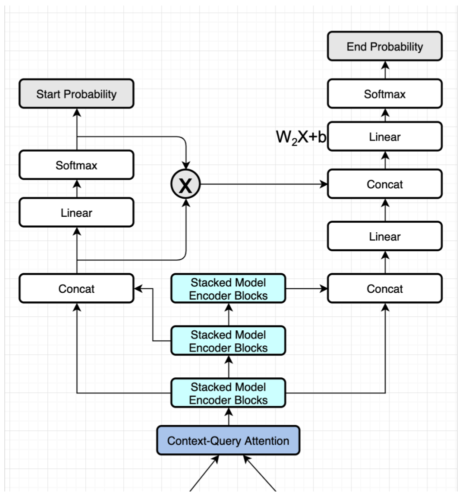

Fig. 4. Modification 3 Output Layer

We use the same notations as previous and calculate as follows.

$$X_1 = [M_0; M_1]$$

$$Y_1 = Softmax(W_0[M_0; M_1])$$

$$Z_1 = W_1[M_0; M_2]$$

$$p_{start} = Softmax(W_0[M_0; M_1])$$

$$p_{end} = Softmax(W_2[Z_1; X_1 * Y_1] + b)$$

where the multiplication is a point-wise operation.

IV. Experiments

A. Data

We train our QANet model version 2.0 of the SQuAD as our dataset. which contains both answerable and unanswerable questions, to train and test our models. The training set consists of the roughly 130K examples in the official SQuAD 2.0 training set. The development set and test set we use are roughly equally sized halves of the official SQuAD 2.0 development set, with about 6K examples each, since the official test set is hidden from the public.

B. Evaluation method

We primarily use the F1 score and the exact match (EM) score for evaluation, using these to compare the performance of different models.

$$F1 = 2/(Recall^{-1} + Precision^{-1})$$

where

$$Recall = TP/(TP + FN)$$

and

$$Precision = TP/(TP + FP)$$

where TP, FN and FP are True Positive (observations that are correctly classified into positive class), False Negative (observations that are incorrectly classified into negative class) and False Positive (observations that are incorrectly classified into positive class), respectively.

C. Experiment Details

Data Preprocessing We generate word embedding and character embedding according to the original QANet paper. We get 91591 tokens which have corresponding word embedding vector, and 1418 tokens which have corresponding character embedding vector. For training data, we build 129907 instances of features in total, and for dev samples, we build 11715 instances of features in total.

QANet We run our implementations of the base QANet, QANet with addition, QANet with addition and bias, and QANet with multiplication and bias in the Method section. For model configuration, we use the same as what is used in the original QANet paper [1]. We set the learning rate to 0.001, learning rate warm up number to 1000. We train QANet with addition and QANet with addition and bias 10 epochs. We then train base QANet and our last QANet model variant until convergence (roughly 26 epochs) within 13 hours on two Tesla V100 GPU. We set batch size to 32, which occupy about 25G GPU memories while training.

V. Results

TABLE I
Performance of modified model and baseline model on the SQuAD 2.0 development set

Model	F1	EM
QANet (baseline) (10 epochs)	73.11	59.50
QANet + Addition + Bias (10 epochs)	72.74	59.60
QANet + Multiplication + Bias (10 epochs)	73.67	61.14
QANet (baseline) (30 epochs)	73.89	61.60
QANet + Multiplication + Bias (30 epochs)	**74.14**	**61.67**

The performance of our modified QANet models and the origin QANet model experiments on the SQuAD 2.0 are shown in Table I. We start with 10 epochs for each model. It shows that the "QANet + Addition + Bias" modification returns a lower F1 score, which does not improve the baseline QANet model as expected. "QANet + multiplication + Bias" modification performs both a better F1 score and a better

EM, while its improvement on EM is also better than the "QANet + Addition + Bias". In addition, the model had not converged at the 10th epoch, indicating that it could give a better performance with further computation. Therefore, we consider the "QANet + multiplication + Bias" as a better modification and make a further training on it together with the baseline QANet model by 30 epochs.

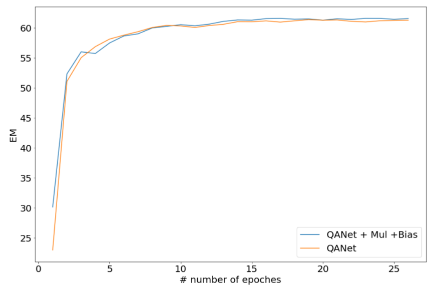

Fig. 5. EM Evaluation Result

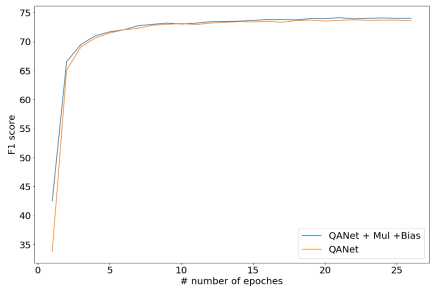

Fig. 6. F1 Score Evaluation Result

The evaluation result for "QANet + multiplication + Bias" and the original QANet model on F1 score and EM is shown in the Figure 5 and Figure 6. The modified model overall performs better than the baseline QANet model.

However, both F1 (74.14 vs 73.89) and EM (61.67 vs 61.60) are quite close in these two models. Our modified model does not make a obviously improvement on the baseline model. One of the possible reasons is that after epochs of training, the original structure of QANet is also able to capture the start probability information when predicting the end. As the outputs of the three Stacked Model Encoder Blocks are inferred one by one, the third one (M_2 which is used for predicting the end probability) also contains the information from the second output (M_1 which is used for predicting the start probability). Even though our model directly feeds the information of start probability to the process of predicting the end probability, such advantage would disappear after training for a long time.

VI. Conclusion

In summary, we present an implementation of QANet that achieves strong results on the SQUAD 2.0 dataset. We improve the original QANet by adding a novel connection from the start probability to the end probability, try three different ways, and finally find the fittest modified version and get a better result than the baseline. In the future, we are interested in the data augmentation. We will try to figure out how QANet synergizes with data using paraphrases. So far, we have only explored the viability of our modified model for a single dataset. It remains to be seen whether our performance improvement carries over to other tasks. We would like to find out whether our modified QANet works well for other QA datasets, such as NewsQA, as well. We will also consider the improvement of loss function. Finally, we can predict more than starting context and the ending context in QANet, and additional variables like the context span, and the core word can be added to further improve the model.

VII. Statement of Teamwork Contribution

Hanyin Shao is responsible for model design and also help on the model training for the first version of our modified model.

Xinkai Yuan is responsible for training models and the Experiments and Conclusion sections of the report, slides and presentation.

Tao HuiLi researches on the related model BiDAF, BERT and QANet. Then do analysis for different models to find their advantages and limitations. Do the presentation, paper and PPT of the related work part. Help run the code and get the result.

Shiqi Yu is responsible for the related model analysis and background research. Do the paper and PPT for part of introduction, related work and data embedding.

Yiqing Du is responsible for the related model analysis and the data processing of the QANet model. Do the PPT for part of the related part and paper for Introduction, part of Experiment and Result sections.

References

[1] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, "Qanet: Combining local convolution with global self-attention for reading comprehension," 2018.

[2] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," 2018.

[3] S. Quarteroni and S. Manandhar, "A chatbot-based interactive question answering system," 2007.

[4] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," 2018.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[6] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. [Online]. Available: https://www.aclweb.org/anthology/D14-1162 pp. 1532–1543.

[7] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," 2015.

[8] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2016.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[10] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[11] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading Wikipedia to answer open-domain questions," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017. [Online]. Available: https://www.aclweb.org/anthology/P17-1171 pp. 1870–1879.