QA System with Transformer-XL

Stanford CS224N Default Project IID track

Yizhi Zhang
Department of Mechanical Engineering
Stanford University
zhangyz@stanford.edu

Zhuzhu Wang
Department of Mechanical Engineering
Stanford University
wangzz@stanford.edu

Abstract

The automated question-answering system has always been an interesting and challenging task. In this project, we implemented a baseline BiDAF model with character-level embedding, a QANet model, and a QANet with transformer-XL model to explore their performance on the SQuAD 2.0 question-answering dataset. From the result of our experiments, the QANet reaches EM of 52.86 and F1 of 55.74, and the QANet with the transformer-XL reaches EM of 50.92 and F1 of 51.01 on the dev set.

1 Key Information to include

- Mentor: Yian Zhang
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

The automated question-answering system has always been an interesting and challenging task and can be useful in many areas. For example, companies use chat-bots as virtual assistants, which needs the model to generate answers according to customers' specific questions. Our goal is to build a question answering system using the Stanford Question Answering Dataset 2.0 (SQuAD 2.0), and try to improve the performance using the QANet and Transformer-XLs.

The input to our model is a paragraph and a question about that paragraph, including both answerable and unanswerable questions. The output is the best answer to that question. For the answerable questions, the answers can be found directly from the original paragraph with no generation needed.

Our provided baseline for the question answering system is based on a Bidirectional Attention Flow (BiDAF) model, which uses a bidirectional LSTM as a RNN encoder and a bidirectional attention that flow in both directions - from the context to the question and from the question to the context. This is helpful as the model captures useful information from the context, but LSTMs may have the problem of vanishing and exploding gradients in long contexts. One potential improvement would be using transformer-XLs as it is able to capture the longer-term dependency in the context. In this project we made the following attempts to improve our model performance: using character-level embedding, using QANet, and using Transformer-XLs.

From the result of our experiments, the QANet reaches EM of 52.86 and F1 of 55.74 on the dev set, and the QANet with the transformer-XL reaches EM of 50.92 and F1 of 51.01 on the dev set.

3 Related Work

The provided baseline network was adapted from the BiDAF model [1], which is composed of five modules: an embedding layer used for generating the word-level embedding, an encoder layer which

uses bidirectional LSTM, an attention layer that captures information from both context to question and question to context, a modeling layer that uses two layers of bidirectional LSTM, and finally an output layer computing the possibilities of each word in the context.

With the growing interest in machine reading, the QANet [2] was then proposed to speed up the training process. Essentially the QANet replaced the recurrent structure in the BiDAF model with a bunch of encoder blocks that use convolutions and self-attentions. The model is also composed of five layers, and will be discussed in details in the next section.

The LSTMs are widely used, but potentially they may have optimization problems due to gradient vanishing and explosion between long-distance word pairs. Transformer-XL [3] is then proposed to address this issue. The main contribution of transformer-xl is a segment-level recurrence mechanism and a relative positional encoding scheme. The segment-level recurrence mechanism reuses the hidden states obtained in the previous segments. The relative positional encoding helps address the incoherence in the positional information gives the model a temporary clue about where to attend. As a result, modeling very long-term dependency becomes possible. In this project, we implemented transformer-xl for the attention layer in the encoder blocks.

4 Approach

4.1 Baseline

Our baseline model is the default starter code. This model only uses a word embedding layer as compared to the original BiDAF model [1], but it follows the same high-level structure. This model uses pretrained GloVe word embeddings [4]. All our other models are built upon this baseline model.

4.2 Character embedding layer

According to [1], they used a character-level embedding layer in addition to the word embedding. We recovered this model by adding this character-level embedding layer. Specifically, we first used an `nn.Embedding` layer to retrieve the 4-dimensional character embeddings, with the shape being (batch size, text length, max word length, char embed size).

Then we permute this tensor and apply a convolutional layer over the embedding. This conv2d layer treats the char embed size as the input channel and convolves over the max word length dimension, and the parameters of this layer is `Conv2d(out_channel=hidden_size, kernel_size=(1, 5))`, where size 5 is over the word length dimension, which is also the default value specified in [1]. We set the kernel size on the text length dimension to 1 to guarantee that the conv layer does not look at other words. Then we apply a maxpool to only the last dimension to yield the tensor with shape (batch size, text length, hidden size).

The word embeddings are still passed into the projection layer. Then we stack the char embeddings with word embeddings, and feed them into the rest of the network. Note that the highway encoder is expanded to twice of the original size since we now have two embeddings.

4.3 QANet

Our second improvement is to use the transformer model instead of the LSTM structure adopted in the baseline, since it has proved effective in many NLP tasks. Our implementation follows the QANet architecture [2]. The general architecture of our implementation is shown in Fig 1, which shares the same high-level structure as QANet.

Embedding layer In our implementation, we keep our word and char embedding layers as described in section 4.2. The dimensions of both word and character embeddings are 100, therefore our model dimension is set to 200 throughout all layers. We then pass the embeddings into the embedding encoder layer, which is one QANet encoder block shown in Fig 1b. A positional encoding constructed using sin and cos waves is added to the embeddings before they get passed into deeper layers in the block. The second part has 4 conv blocks stacked together, each including a layernorm and a depthwise separable convolution operation, as originally proposed in [5]. The next layer consists of a layernorm followed by a multi-head self-attention layer, which is adapted from the code in previous

course assignment. The outputs are finally passed into a two-layer MLP to generate inputs to the other parts of the network. Note that throughout the entire encoder block, we keep the dimensions of each output the same so as to realize the residual connections.

Context-query attention Different from the original QANet, we keep the bidirectional attention flow layer the same as the one used in the BiDAF model. This layer computes a similarity matrix between the context and question hidden states, and uses this matrix to distribute attention toward either context or question.

Modeling layer In the modeling layer, we implement the structure same as the QANet, which uses three stacked encoder blocks, with 7 encoder blocks in each stack. The three repetitive stacks share weights between them. The structure of each encoder block is the same as the one described in the embedding encoder except that it only has 2 conv layers instead of 4 in the embedding encoder, which is also implemented according to the original QANet. To regularize this deep layer and prevent overfitting, we follow the layer dropout method after each small layer (i.e. conv layer, self-attention layer, and the MLP) as mentioned in QANet. This mechanism varies the drop out probability of each layer according to Equation 1,

$$p = \text{dropout} \times \frac{l}{\text{total number of layers}} \tag{1}$$

Where l is the number of current layer in the entire stack. By scaling the dropout ratio according to the depth of layer, this method allows gradients from earlier layers to flow more easily into deeper layers, and prevents too many hidden units from being zeroed out due to frequent dropout operation.

Output layer The last layer takes the outputs from the modeling layer, and compute a probability distribution of the start and end positions of the answer in the context. The inputs from the first and the second modeling encoder stacks are concatenated together and used to generate the probability for the start position, and the second and third stack outputs are used to determine the end position. This layer is simply implemented using a two-layer MLP followed by a masked log softmax operation.

4.4 TransformerXL in QANet

Based on the QANet, we further improved our model by implementing the transformer-XL for the attention layer in the encoder blocks. This section will discuss the approach of building transformer-xl as well as the combination to our overall QANet architecture.

4.4.1 TransformerXL

According to [3], transformer-XL has the following features and may help learn longer dependencies and achieve better performance on both short and long dependencies:

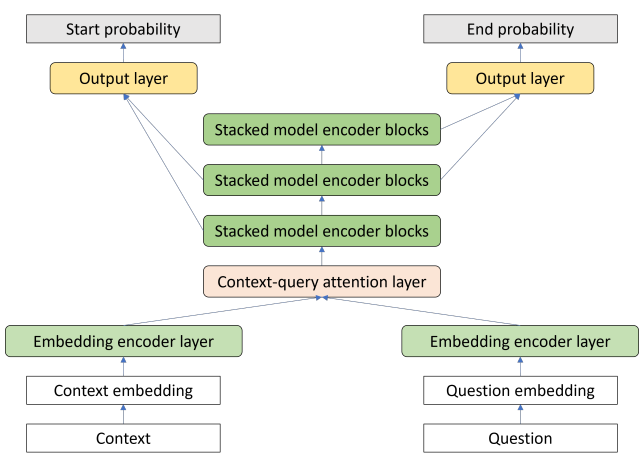
- **Segment-level recurrence:** The main idea is to reuse the hidden state sequence from the previous segment. The network now can process information in the history, and so it is able to model longer-term dependency. Specifically, the n-th layer hidden state for segment $s_{\tau+1}$, denoted as $\tilde{h}_{\tau+1}^{n-1}$, is:

  $$\tilde{h}_{\tau+1}^{n-1} = [SG(h_{\tau}^{n-1}), h_{\tau+1}^{n-1}]$$
  $$q_{\tau+1}^{n}, k_{\tau+1}^{n}, v_{\tau+1}^{n} = h_{\tau+1}^{n-1}W_{q}^{\top}, \tilde{h}_{\tau+1}^{n-1}W_{k}^{\top}, \tilde{h}_{\tau+1}^{n-1}W_{v}^{\top}$$
  $$h_{\tau+1}^{n} = \text{Transformer-Layer}(q_{\tau+1}^{n}, k_{\tau+1}^{n}, v_{\tau+1}^{n})$$

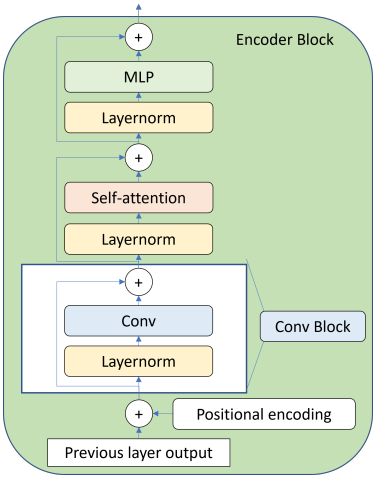
  where W is the model parameters, $h_{\tau+1}^{n-1}$ is the n-th layer hidden state produced for the τ-th segment, SG stands for stop-gradient, $[\cdot, \cdot]$ stands for the concatenation of the hidden sequences.
- **Relative positional encoding:** To successfully reuse the hidden states, transformer-XL encodes the relative positional information and injects into the attention score. Specifically, the attention score can be calculated with:

  $$A_{\tau,i,j}^{n} = q_{\tau,i}^{n}\top k_{\tau,i}^{n} + q_{\tau,i}^{n}\top W_{k,R}^{n} R_{i-j} + u^{\top} k_{\tau,j} + v^{\top} W_{k,R}^{n} R_{i-j}$$

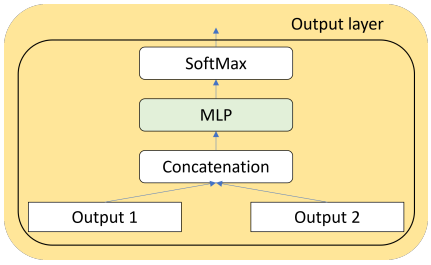
  where u and v are trainable parameters and $i - j$ represents the relative position.



(a) Architecture Overview



(b) Encoder Block



(c) Output Layer

Figure 1: QANet Architecture

4.4.2 Transformer-xl in the QANet

The overall architecture is very similar to the QANet model, except for the encoder blocks used in the encoding and modelling layers. That is to say, the embedding layer, attention layer, and the output layer are exactly the same as in the QANet.

For the embedding encoders (the encoding layer), we modified its encoder block. The encoder block for the QANet can be seen in Fig.1 (b). To implement the transformer-xl based on that, we made the following changes:

- We kept a memory that contains information of the previous l_mem hidden states spanning multiple segments and inputted it to the encoder block. l_mem is set to 128.
- We switched the positional encoding to a relative positional encoder with length 2*hidden_size, with hidden_size set to 200.
- We modified the self-attention to a multi-head attention with relative positional encoding and learnable bias parameters. We set n_head as 8, d_head as 16, and a dropout of 0.1.

For the modelling layer, which was concatenated with three encoder blocks in the QANet model, is now concatenated with seven modified encoder blocks.

5 Experiments

5.1 Data

We use the SQuAD 2.0 dataset [6]. We use the data splits generated using the provided setup script, making sure that we only trained with the train set and evaluated with the custom dev set following the rules specified in the handout. There are 129941 examples in the train set, 6078 examples in the dev set, and 5915 examples in the test set. Each example is composed of the context, question, and

answer. The task is to predict a span in the context that answers the question. When a question is not answerable with texts directly taken from the context, the question is marked as unanswerable.

5.2 Evaluation method

The evaluation metrics we use are F1, Exact Match (EM), and AvNA scores. We use F1 score to determine the best model.

5.3 Experimental details

Baseline and character embedding We use the Adadelta optimizer with a constant learning rate of 0.5, and exponential moving average (EMA) being 0.999. The drop out probability in the layers is 0.2. Both the word and character embedding have dimension of 100.

QANet We train this model using the Adadelta optimizer same as the baseline. We implement a 1000-step inverse exponential increase warm-up period, and after that we train using a constant learning rate of 0.5. We decrease the dropout rate to 0.1 when training this model. We train this model for 1.7M iterations (14 epochs).

Transformer-xl To train the transformer-xl model, we used a constant learning rate of 0.005. We train this model for 1.5M iterations with batch size 16.

5.4 Results

The results obtained from our baseline and the improved models tested on the dev set are shown in Table 1. The training curves of both models are shown in Fig 2.

It can be seen that in the baseline and the char embedding models, the metric scores continued improving across the training, and converged after around 3M iterations. The dev negative log likelihood curve shows that the loss starts to rise at around 1.5M iterations, which implies the model starts to overfit on the training set.

In the QANet training process, we see that the loss decreases slowly after 1.5M iterations. The metric scores are also improving much more slowly than the BiDAF model even if they share the same optimizer and learning rate. This could be due to the fact that QANet is a larger and also deeper network, so it would be harder to train. From this observation, we suspect that some potential issues to fix in the model include: 1) Size of the network. Our model currently uses 200 as model dimension throughout all layers, while the original model is using 128 as their model size and can perform well. Reducing the model dimension may help accelerate the convergence. 2) Regularization and drop out. Currently we adopt the layer dropout scheme which scales the dropout probability based on depth of each layer, but as we observe that there is a gap between the loss on the training data and validation data, we should consider allowing more dropout or seek another better way to scale it so as to close this gap and prevent overfitting. 3) Cross validation. We can consider doing cross validation to include the dev set as part of our training set, and hold out some training data to save as dev data. This may help also close the gap between the model behavior on the two datasets.

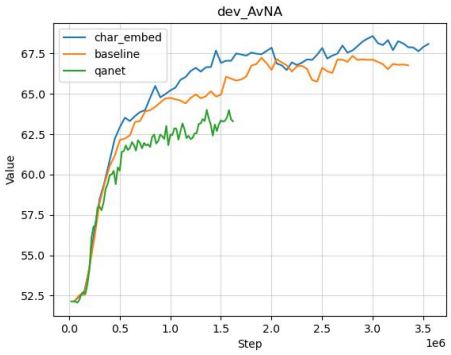
In the QANet with the transformer-xl training process, the loss decreases quickly in the first 500k iterations, but then starts to oscillating and increases after 1.1M iterations. From the observation, we think it might have the same overfitting problem as for the QANet, which potentially might be fixed with reducing the number of layers we use in the model, or adding regularization. We also think it might be helpful to tune the learning rate. In our experiments we found the QANet with the transformer-xl is sensitive to the leraning rate as a little modification from 0.005 to 0.01 would change the convergence. We may need to consider using adaptive learning rate for a better fitting.

6 Analysis

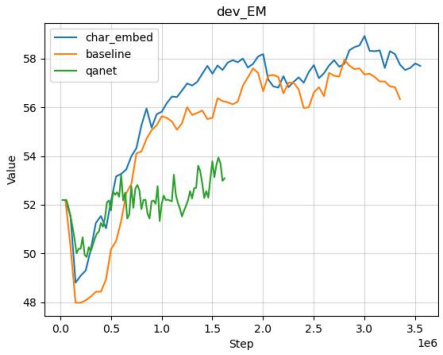
In this section we look at the output answer predictions to gain some insight about the characteristics of the models. We first provide some common successful and failure cases of the BiDAF model from the dev set, discuss what is observed from the pattern, and talk about the difference between the char

Model	Test Set		Dev Set		
	EM	F1	EM	F1	AvNA
Baseline	N/A	N/A	57.6	60.94	67.23
Char embedding	59.20	62.74	58.93	62.11	68.58
QANet	49.77	52.14	53.32	55.74	63.07
Transformer-XL	N/A	N/A	50.92	51.01	54.56

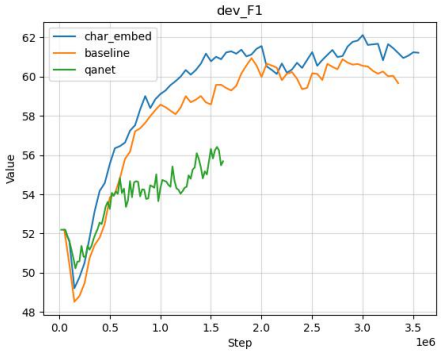
Table 1: Model Performance on Dev and Test Sets



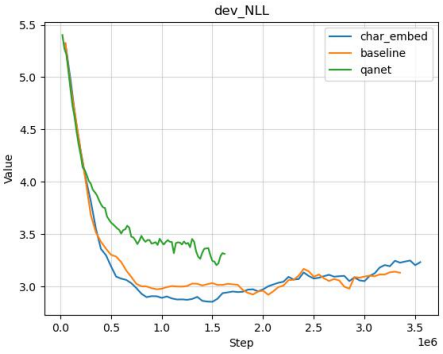
(a) AvNA



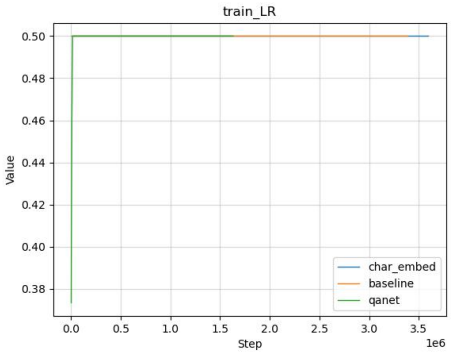
(b) EM



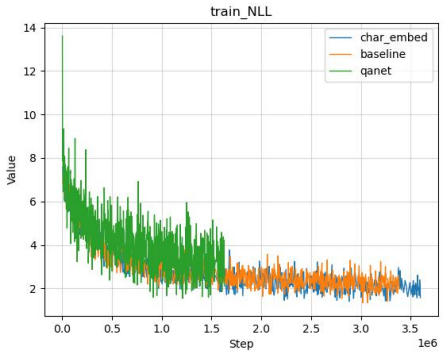
(c) F1



(d) dev NLL



(e) train LR



(f) train NLL

Figure 2: Model Training Curves

embedding model output and the QANet output. We omit some of the context to save space and only try to keep relevant information here.

(success - exact match) Question: When B cells and T cells begin to replicate, what do some of their offspring cells become?

Context: When B cells and T cells are activated and begin to replicate, some of their offspring become long-lived memory cells. ...(omitted)

Prediction: long-lived memory cells

(success - partial match) Question: The price of oil is usually a stable commodity until when?

Context: (omitted)... From 1947 to 1967, the dollar price of oil had risen by less than two percent per year. Until the oil shock, the price had also remained fairly stable versus other currencies and commodities. ...(omitted)

Prediction: the oil shock (**Answer:** Until the oil shock)

(failure - false answer) Question: Who ordered Loudoun to defend Louisbourg?

Context: Loudoun, a capable administrator but a cautious field commander, ...(omitted)... He was then ordered by William Pitt, the Secretary of State responsible for the colonies, to attack Louisbourg first. ...(omitted)

Prediction: William Pitt (**Answer:** N/A)

(failure - wrong range) Question: Thousands of madrasahs spawned what organization?

Context: The Taliban were spawned by the thousands of madrasahs the Deobandi movement established for impoverished Afghan refugees and supported by governmental and religious groups in neighboring Pakistan. ...(omitted)

Prediction: Deobandi (**Answer:** The Taliban)

(failure - no prediction) Question: Who founded Telnet

Context: Telenet was the first FCC-licensed public data network in the United States. It was founded by former ARPA IPTO director Larry Roberts as a means of making ARPANET technology public. ...(omitted)

Prediction: N/A (**Answer:** Larry Roberts)

In the successful examples shown above, we see that the model is learning to check for key words when predicting the answer, like the word "become" in the exact match example, and these examples seem relatively easier to answer because they do not include complicated sentence structure that requires high-level understanding. There are also many partially correct predictions that include a couple more or fewer words than the ground truth, but are still logically acceptable.

We find that there are three major categories of failure cases: 1) False answer, in which case the question is not answerable but the model gives a prediction, 2) Wrong range, when the model refers to a wrong part in the context, and 3) No prediction, when the model cannot extract any answer. As shown above, we see that if the ground truth answer is relatively far from the context that is relevant to the question, there is a higher probability that the model fails. Usually it either provides a false answer that is close to the question, or cannot find any answer. Interestingly, when there are two words that are of the same type and match what the question is asking (e.g. both are numbers, places, names, etc.), it could easily confuse the model.

These observations provide us some insight about what is being learned in our model and what is not. First, we see that the model can learn the short-range correlations more easily than words that span over a long sentence since short-term memories are easier to recover. Second, the model is good at paying attention to key words. Even though it sometimes predicts a wrong answer, it usually matches the question type and most times is highly relevant to part of the question. However, the model cannot distinguish different grammars very well, for example, it does not look backward when there is a passive voice, as shown in the wrong range failure case.

Now when we compare the above outputs from the BiDAF model to the QANet, though they still share similar types of failures, we notice that the there is a higher chance for the QANet to predict a wrong answer that is far away from the part relevant to the question. This happens even if the ground truth answer is very close to the question-related text and should be easier to locate. We think this is because the QANet treats the entire context as the input, so it has a wider receptive field when predicting the answer but may not be better at predicting answers that are right neighboring the question keywords than RNNs.

7 Conclusion

In this project, we explore the question answering problem using different NLP network structures. Starting from the baseline BiDAF model [1] without character embedding, we first recover the original work by adding the character embedding layer. Then we study the QANet [2] mechanism and adapt our model to it. Lastly, we try to combine the idea in Transformer-XL with QANet.

From the results, we see that adding the character embedding has improved the model performance already. With the other two improvements, QANet and Transformer-XL, though in our implementation we do not observe the improvement, we do observe that using transformers allow the model to better capture long-term information than the LSTMs used in BiDAF model.

Given the time span of this project, we expect that in the future we can try with different hyperparameters on the transformer models, such as a smaller network with fewer hidden units. We also want to try other techniques that may help with training a deep network, such as a different drop out mechanism that could help prevent overfitting or the Adam optimizer with a smaller learning rate.

Furthermore, from the qualitative results we see that although the models can capture many key words, they tend to miss many grammatical details that are critical to locating the answer. A lot of large models adopt pretraining with the NMT task before finetuning on the question answering dataset, and we also hope that we have a chance to train a model with pretraining and see how it helps.

References

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

[2] 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. *arXiv preprint arXiv:1804.09541*, 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. *arXiv preprint arXiv:1901.02860*, 2019.

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

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