Transformer-XL Architecture For Question Answering

Stanford CS224N {Default} Project

Chenkai Mao
Department of Electrical Engineering
Stanford University
chenkaim@stanford.edu

Qinghong Zheng
Department of Mechanical Engineering
Stanford University
colinz07@stanford.edu

Abstract

In this project, we aim to incorporate the Transformer-XL model[1] into the SQuAD QA system and determine if the model is better performing than vanilla transformer in QA tasks. To approach this final goal of our project, we completed two intermediate goals: the first goal was to add character embedding to the BiDAF model using character-level convnets, as described in[2]. The second goal we completed was to adapt the transformer model and apply to the QA model. This method was previously implemented in QANet[3], which we used as a reference in developing our QANet. Because the Transformer-XL model uses segment-level recursion and is mainly developed for text generation tasks, we point out some possible limitations to apply such model to reading comprehension tasks, and that it might not be an improvement over QANet. The results for the Transformer-XL model were F1: 66.13, EM: 63.06 whereas the QANet scored F1: 67.92, EM: 64.31. Because of different model configurations and less training time for the Transformer-XL model, it can only be concluded that Transformer-XL had similar performance to vanilla transformer QANet for the RC task. Lastly, some error modes in the model prediction are discussed.

1 Key Information to include

- Mentor: Kendrick Shen
- External Collaborators (if you have any): None
- Sharing project: No

2 Introduction

Machine reading comprehension (RC) and question answering (QA) tasks have gained major popularity in the past few years. This project focuses on such task on the SQuAD 2.0 dataset. Given a question and a context paragraph, the objective of SQuAD 2.0 QA system is to output a span of the context paragraph as the answer. A number of models were used and tested to tackle such task, and our group focuses on two specific models: QANet [3], which uses the vanilla transformer model, and Transformer-XL [1] that would yield a possible performance increse over the vanilla transformer model. Improvements over the baseline BiDAF model was also discussed in this project by including convnet character-level embedding[2][4].

The major challenge of the SQuAD QA task is to process long paragraphs, with the context consisting of over 200 words. Two models are discussed in our project. One of which is using the recurrent neural network (RNN), as exampled by the bi-directional attention flow (BiDAF)[2]. However, the potential drawback of RNN models is the issue of vanishing gradient, which poses a challenge in preserving information over long context. Another method is to use the transformer model to process the input. QANet is an example of such model[3]. The tranformer model uses multi-head attention and multi-layer to process long sequences of text. It is claimed to be both faster and able to achieve

higher scores[3]. The Transformer-XL model is an extension over the vanilla transformer model that allows segment-level recurrence with state reuse[1]. In text-generation tasks, vanilla transformers are unable to achieve long-term dependencies over its context length, while it is achievable with Transformer-XL models by caching and reusing the previous hidden states[1]. The ability to preserve long-term context would also potentially increase performance of processing long context paragraphs for QA tasks. However, the segment-level recurrence mechanism of the Transformer-XL model also means that the context paragraph would need to be segmented into fixed-length context, which has two potential drawbacks: breaking the context as a result of segmentation, and limiting the span of the answer to be inside each segmentation. In our project, the QANet would be the backbone to apply the Transformer-XL model.

3 Related Work

3.1 Transformer

Recurrent Neural Networks (RNNs) have dominated for years in sequence-to-sequence problems, but the architecture has a great limitation that when working with long sequences, their ability to retain information from earlier elements was lost. It has been found that LSTM language models use 200 context words on average[5]. Even with Long-Short-Term-Memory (LSTM), the recurrent nature still limits its performance. In the famous paper "Attention is all you need" [6], a new attention mechanism was introduced which extracts information from the whole sequence in the form of a weighted sum of all the past encoder states. This allows the decoder to assign greater weight or importance to a certain element of the input for each element of the output.

3.2 QANet

QANet [3] is the first work that fully utilize transformer architecture in the Question Answering problem setup on the SQuAD dataset. They replaced the traditional RNN components in embedding and encoder layers with transformers and obtained a significant performance boost. On top of the transformers which captures global relation, convolution operations are also utilized for modeling local relations.

3.3 Transformer-XL

Transformer-XL is a neural architecture proposed by Dai et al. [1] to better leverage long-term dependencies in text generation tasks. As discussed before and in [7], RNNs, in particular LSTM networks, are difficult to optimize in processing long context because of gradient vanishing and explosion. By using a transformer model, the results outperformed LSTM models by a large margin[8]. However, the model in [8] is performed on fixed-length segments without any information flow between the segments. It could not capture longer-term dependencies beyond the designated context length. Transformer-XL introduced the algorithm of segment-level recurrence that enables using the cached hidden states from the previous segment in processing the new segment[1]. The authors report significantly longer dependency than both RNN and transformer models, and also improvements in calculation speed in both short and long sequences. Although the model was not initially developed for the RC task, performance increase might be seen over QANet.

4 Approach

4.1 Character-Level Embedding

The first model we implemented was adding character embeddings to the BiDAF model. The code is developed on our own with the architecture referenced from [4]. The character embedding layer was a 1D convnet. It is shown that simple CNN with little hyperparameter tuning and static vectors would achieve good results in sentence classification benchmarks[4], hence we propose to use convnet for word embeddings as well for the development of the transformer model. The architecture of the model is similar to one shown in 1 in the Appendix. The figure shows 2-channel convnet embedding on the word level, whereas our model applies a one-channel convnet on the character level. Before applying convnet, a dropout layer is applied on the input. Let $\mathbf{x}_i$ be the k-dimensional character

vector for the i-th character in a word. A word of length n is represented as

$$\mathbf{x}_{1:n} = [\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_n] \tag{1}$$

A convolution operation uses a filter $\mathbf{w} \in \mathbb{R}^{hk}$ which is applied to a window of h characters to produce a new feature. Then, a feature c_i is generated from a window of characters $x_{i:i+h-1}$ by

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b) \tag{2}$$

where b is a bias term and f is a non-linear function, which we used ReLU. The filter is applied to each window of words to produce a feature map $\mathbf{c} = \{c_1, c_2, \ldots, c_{n-h+1}\}$. Then, a maxpool is applied over the feature map and take the maximum value $\hat{c} = max\{\mathbf{c}\}$ as the feature corresponding to this filter. The window size we used for this model was 3. [2] used a hidden layer size of 100, but because of our computational resources are very limited, our output layer size for the convnet was set to 20. The character output was then concatenated with the word output $\mathbf{h}_i = [\mathbf{h}_{ci}; \mathbf{h}_{wi}]$, and then passed to the highway network.

4.2 Transformer

In order to incorporate ideas from transformer-xl into QA probelm, we first implemented a vanilla transformer model with reference to QANet[3]. Here we'll talk about implementation details of vanilla transformer.

Although the bi-directional LSTM model used in the RNN encoders in the baseline model captures information flow in both directions with long-term dependency, the transformer architecture introduces the self-attention mechanism which can directly access all positions in the sequence, equivalent to having full random access memory of the sequence during encoding and decoding. This advantage not only enables easier learning of long-term dependency, but also make the model easily parallelizable and more efficient to train and evaluate (although in practice this benefit seems mostly work with multiple gpus).

Our implementation of the transformer, for which we take QANet as a refernece, could be summarized in the following manner:
(1) Input layer: both the pre-trained word embedding and character embedding are taken as model input and processed with convnet. A two-layer highway network was adopted at the end.
(2) Encoder layer: The encoder layer consists of 4 convolution layers, a multi-head self-attention layer and a feed-forward layer. Residual connections and layer-norms are also utilized for stabler training.
(3) Context-Query Attention Layer: First, a trilinear similarity function f is defined for a context word c and query word q: $f(q, c) = W_0(q, c, q \bigodot c)$, with $\bigodot$ being element-wise multiplication and W_0 a learnable weight. Then the similarity matrix is computed for each context, query pair $S = f(Q, C)$. And the context-to-query attention A and the query-to-context attention B are defined as:

$$A = \overline{S} \cdot Q^T$$
$$B = \overline{S} \cdot \overline{\overline{S}}^T \cdot C^T$$

in which $\overline{S}$ is row-normalized S and $\overline{\overline{S}}$ is column-normalized S, both using softmax function.

(4) Model Layer: After the CQ-attention is calculated, for a context c, we take the rows a, b of A, B and form the input to the encoder layers as $[c, a, c \bigodot a, c \bigodot b]$. The encoder is the same as in the encoder layer, and 3 repetitions of the encoder is used, which share the same weights.

(5) Output Layer: Let the output of the 3 encoder layers from model layer to be M_0, M_1 and M_2, the output probability for the starting and ending positions are:

$$p_{start} = softmax(W_1[M_0; M_1])$$
$$p_{end} = softmax(W_2[M_0; M_2])$$

with W_1 and W_2 trainable weights.

The weights of the context and question encoder, and of the three output encoders are shared. For each encoder layer, a positional encoding using sin and cos functions with varying frequency is added to the input. The model architecture and one encoder block is shown in 2 in Appendix.

4.3 Transformer-XL

The main limitation of the vanilla transformer model is its ability to scale up with the length of the context length. There are two main components of the Transformer-XL model to address this issue to effectively process long paragraphs:

(1) Segment-level recurrence: This reuses the previously-calculated hidden state when processing the next segment. It caches the previous state designated by its memory sequence length and its information is reused in calculating the current state. Let h_τ^n be the n-th layer hidden state of the τ-th segment, then, the n-th layer hidden state for the next segment of $\tau + 1$ would be:

$$\tilde{h}_{\tau+1}^{n-1} = [SG(h_\tau^{n-1}); h_{\tau+1}^{n-1}]$$
$$q_{\tau+1}^n, k_{\tau+1}^n, v_{\tau+1}^n = h_{\tau+1}^{n-1} W_q^\top, \tilde{h}_{\tau+1}^{n-1} W_k^\top, \tilde{h}_{\tau+1}^{n-1} W_v^\top$$
$$h_{\tau+1}^n = Transformer - Layer(q_{\tau+1}^n, k_{\tau+1}^n, v_{\tau+1}^n)$$

Where $SG(\cdot)$ stands for stop-gradient, and W stands for model parameters.

(2) Relative positional encoding: To apply segment-level recurrence, the positional information should be coherent when the states are reused. Using absolute positional encoding would result in same positional encodings in different states. Hence, relative positional encoding should be used. Equipping the recurrence model with relative-positional encoding, the final N-layer Transformer-XL model with a single attention head can be summarized below:

$$For\ n = 1, \ldots, N:$$
$$\tilde{h}_\tau^{n-1} = [SG(m_\tau^{n-1}); h_\tau^{n-1}]$$
$$q_\tau^n, k_\tau^n, v_\tau^n = h_\tau^{n-1} W_q^{n\top}, \tilde{h}_\tau^{n-1} W_{k,E}^{n\top}, \tilde{h}_\tau^{n-1} W_v^{n\top}$$
$$A_{\tau,i,j}^n = q_{\tau,i}^{n\top} k_{\tau,j}^n + q_{\tau,i}^{n\top} W_{k,R}^n R_{i-j} + u^\top k_{\tau,j} + v^\top W_{k,R}^n R_{i-j}$$
$$a_\tau^n = Masked - Softmax(A_\tau^n) v_\tau^n$$
$$o_\tau^n = LayerNorm(Linear(a_\tau^n) + h_\tau^{n-1})$$
$$h_\tau^n = Positionwise - Feed - Forward(o_\tau^n)$$

where m is the memory sequence from the previous segment, and R is the sinusoidal relative encoding matrix.

(3) Transformer-XL in QA: To apply Transformer-XL in QA tasks, the model is incorporated to the QANet model. As discussed above, the core component of Transformer-XL is to segment the context for recurrence. There are two ways to apply segment-level recurrence: one is to treat batches as segments and use the memory sequence from the previous batch as input to the next batch. However, because different questions do not have clear contextual relations to each other, we instead choose to segment the context paragraph to each question and perform segment-level recurrence on each context sequence. The number of segments would be dividing the context length over the memory sequence length. The main drawback of this approach is that breaking a whole context into segments would potentially break the continuity in the context, and the previous segments would have no information about the later segments. In our approach, only the context paragraph is segmented, while the question paragraph is processed in whole. Because of the drawback mentioned above, we anticipate that the Transformer-XL model would not perform as well as the QANet model, but would still outperform the baseline model.

4.4 Implementation and Code References

The starter code for data loading, BiDAF model, training and testing was provided by the course instructors (https://github.com/michiyasunaga/squad). The implementation of the vanilla transformer QANet is referenced from the code from https://github.com/heliumsea/QANet-pytorch. The implementation of the Transformer-XL model is referenced from the code from https://github.com/kimiyoung/transformer-xl.

Our team mainly (1) implemented the character-level embedding extension for BiDAF, (2) understood and implemented the models of QANet for testing and for the Transformer-XL backbone, and (3) incorporated the Transformer-XL segment-level recurrence to the QANet model.

5 Experiments

5.1 Data

The dataset is SQuAD 2.0 [9] provided by the instructors of this course (https://github.com/michiyasunaga/squad). The train set contains 129,941 examples, the dev set contains 6078 examples, and the test set contains 5915 examples.

5.2 Evaluation method

The metrics for evaluation mainly include F1 score, exact match (EM), answer vs no answer (AvNA), and negative log likelihood (NLL). Exact Match is a binary measure (i.e. true/false) of whether the system output matches the ground truth answer exactly. F1 is the harmonic mean of precision and recall. Because the SQuAD task sometimes contains questions that no answer could be extracted from the context paragraph, the model should determine if an answer exists for a certain question-context pair, or otherwise should produce "N/A" for no answer. AvNA is a metric that measures its answer vs no answer predictions. The NLL loss of the model is determined by the sum of NLL loss of the predicted start and end positions of the answer segment in the context paragraph.

5.3 Experimental details

All models were trained for 30 epochs, with the exception of Transformer-XL, which was trained for 17 epochs because of 24-hour time limit set on the VM was reached. The models use pretrained word embeddings from the GloVe model, and Gaussian randomized character embeddings. Configuration of the BiDAF model with character-level embeddings is the same as the provided in the starter code. Model and training configurations for QANet and Transformer-XL are shown in table 1.

Configuration	Char-Embed	QANet-mid	QANet-large	Transformer-XL
Epochs	30	30	30	17
Batch Size	64	64	64	32
Dropout Rate	0.2	0.1	0.1	0.1
Hidden Layer Size	100	128	256	128
Num. Self-Attention Heads	-	4	8	8
Self-Attention Head Dimention	-	32	32	16
Memory Sequence Length	-	-	-	256
Training time (hrs)	5	10	14	24 (capped)

Table 1: Model configurations.

The optimizer used for both BiDAF and transformer models was Adadelta, and the learning rate for all models was set at 0.5. β_1 and β_2 for all transformer models were set to be 0.8 and 0.999 respectively.

5.4 Results

Metric	Baseline	Char Embed	QANet small	QANet large	Transformer-XL
F1	60.86	63.43	65.31	67.92	66.13
EM	57.69	60.14	62.76	64.31	63.06
AvNA	67.13	70.01	72.19	74.27	71.49
NLL	3.05	3.00	2.97	3.07	2.52

Table 2: Results on dev set.

Table 2 shows the dev set results of the different models we have developed, and table 3 shows the test set results for the Transformer-XL model. From table 2, including character-level embedding

F1	EM
64.575	60.913

Table 3: Test set results of Transformer-XL

and using QANet increases scores for all metrics except for NLL loss, which was similar for all models mentioned above. Also, by incorporating the Transformer-XL model, the performance slightly decreased compared to vanilla transformer QANet. However, because the Transformer-XL model did not finish all 30 epochs, we can only conclude that Transformer-XL had similar performance to the vanilla transformer model with the hyperparameters we set. The training time, however, was significantly larger for the Transformer-XL model.

6 Analysis

Table 4 in the Appendix section shows the examples of errors occurred when performing the Transformer-XL model. From the table, we can see that the errors fall in several categories:

1. Failure in understanding the sentence context and provide an answer. Although the context did contain certain keywords that overlap with the question such as "Economy, Energy and Tourism", it did not explicitly contain certain words in the question such as "one of the". Hence, the model would not be able to locate the answer span in the context.
2. Ambiguities in context-question matching. The question mentioned "directive" and "1994", and had clear overlap of the same words in the context. However, the prediction was another directive created in 1996.
3. Imprecise answer boundaries. The prediction answered the time period correctly, but it did not include the definite article "the" in front of the time period. There are two possible reasons: one is that it requires external knowledge that it is more natural to include the definite article when describing a decade, and the other is that the character-level embedding could be tuned to recognize the postfix "s" that the model could produce better predictions.
4. Possible OOV word. "Romanesque" in the answer is a possible OOV word that the model could not properly embed.

7 Conclusion

This project tackles the question answering (QA) or machine reading comprehension (RC) problem on the SQuAD 2.0 dataset. Multiple models, including extending the baseline BiDAF model with character-level embeddings, QANet with vanilla transformer architecture, and Transformer-XL are implemented and evaluated for performance. Including character-embedding in the BiDAF model increases its performance over baseline, but better results are seen from the QANet model. Our current conclusion on the Transformer-XL model is that it does not have a significant increase over the vanilla transformer model, and that the training time is more than twice as long as the QANet model that doubles hidden layer length. Segment-level recurrence, the main architecture of the Transformer-XL model, does not fit well with the QA task, because the input data is processed differently in text generation and question answering tasks. The main limitation for this project is that the Transformer-XL model was not trained fully to 30 epochs, and that the comparison between QANet and Transformer-XL was indirect because of different hyperparameters. If time permits, both the Transformer-XL and QANet can be better tuned to determine whether segment-level recurrence could improve over vanilla transformer QANet.

References

[1] 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.

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. Bi-directional attention flow for machine comprehension. *ICLR conference 2017*, 2017.

[3] Adams W. 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.

[4] Yoon Kim. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751*, 2014.

[5] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. *CoRR*, abs/1805.04623, 2018.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[7] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. 2001.

[8] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. *CoRR*, abs/1808.04444, 2018.

[9] Pranad Rajpurkar, Jian Zhang, Constantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250*, 2016.

A Appendix

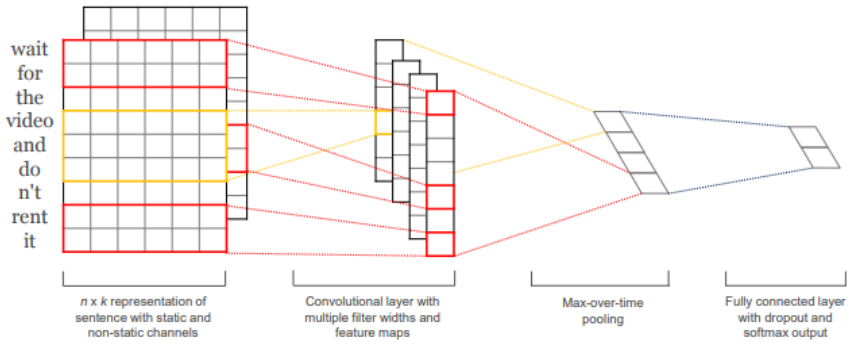

Figure 1: Sample convnet architecture. Image is taken from [4].

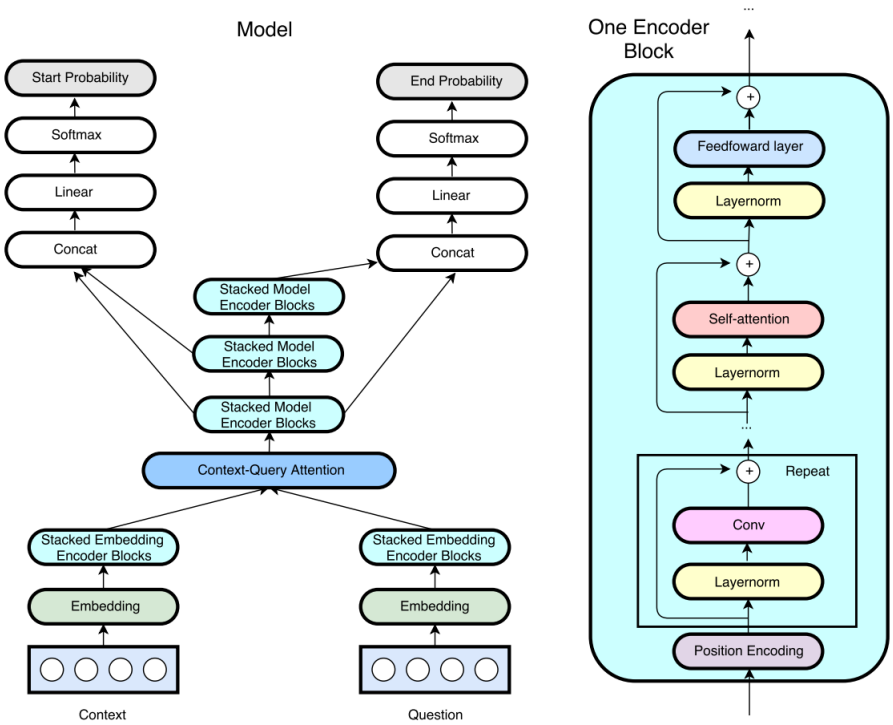

Figure 2: The implemented QANet architecture, served as our transformer backbone for incorporating transfermer-xl ideas, also as a transformer baseline. Shown on the left is the model architecture and one the right one encoder block. Image is taken from [3].

Context	Question	Answer	Prediction
...Typically each committee corresponds with one (or more) of the departments (or ministries) of the Scottish Government. The current Subject Committees in the fourth Session are: Economy, Energy and Tourism; Education and Culture; Health and Sport; Justice; Local Government and Regeneration; Rural Affairs, Climate Change and Environment; Welfare Reform; and Infrastructure and Capital Investment.	Economy, Energy and Tourism is one of the what?	current Subject Committees	N/A
...The UK subsequently adopted the main legislation previously agreed under the Agreement on Social Policy, the 1994 Works Council Directive, which required workforce consultation in businesses, and the 1996 Parental Leave Directive. ...	Which directive mentioned was created in 1994?	Works Council Directive	Parental Leave Directive
...Their combined work informed the study of imperialism and it's impact on Europe, as well as contributed to reflections on the rise of the military-political complex in the United States from the 1950s. ...	When was the military-political complex reflected upon within the scope of understanding imperialism?	the 1950s	1950s
Norman architecture typically stands out as a new stage in the architectural history of the regions they subdued. They spread a unique Romanesque idiom to England and Italy, and the encastellation of these regions with keeps in their north French style fundamentally altered the military landscape....	What is the Norman architecture idiom?	Romanesque	England and Italy

Table 4: Sample errors of SQuAD.