Question Answering on SQuAD 2.0 using Transformer-XL

Julien Hedou
Department of Biomedical Informatics
Stanford University
Stanford, CA 94305
jhedou@stanford.edu

Ianis Bougdal-Lambert
Department of Aeronautics and Astronautics
Stanford University
Stanford, CA 94305
ianisbl@stanford.edu

Abstract

We propose to investigate the use of the novel Transformer-XL architecture for the SQuAD2.0 challenge, so as to evaluate the benefits of using longer-term dependencies for the specific task of question-answering. This approach seems promising to us since many complex questions can only be answered by capturing information from a large context, and/or by combining information coming from different parts of the input texts. We integrate the Transformer-XL into a model similar to the QANet. We show that we can achieve good results for a non-PCE model.

1 Introduction

For this default project, we are interested in addressing the SQuAD2.0 reading comprehension challenge. Given a paragraph – the context – and a question – the query –, the goal is to find the extract in the paragraph that best answers the question. The novelty in the new SQuAD2.0 version of the dataset is that some questions cannot be answered from the provided context.

There has been a growing interest for question answering in recent years, as end-to-end models have become more performant. Solving this problem can be useful in every-day life and also seems like a better way to measure how well a model can really understand a text, rather than through simple translation. Most current approaches in NLP in general and in Question Answering in particular are based on Recurrent Neural Networks (RNNs) with attention. RNNs make use of the naturally sequential nature of human language, while attention modules help to cope with long-term dependencies in the inputs. Some of these methods have proved relatively efficient for that task. It is the case in particular of the BiDAF model (for Bidirectional Attention Flow), developed by [4], which was shown to perform particularly well on the SQuAD dataset (see [3]). However, due to the sequential nature of RNNs, these models tend to be slow to train and to test. Moreover, it was shown (see [2]) that they can only handle short to medium-term dependencies and cannot handle dependencies longer than a few hundred words. This is an important limitation when processing long texts for instance, as it is difficult to achieve convincing real-time performance. Consequently, a lot of effort has been put lately to find a replacement to RNNs.

In [6], the authors tackle this issue by replacing the RNNs by stacks of convolutions and self-attentions. In their model, named QANet, the self-attention layer is implemented as a Transformer, an architecture developed in [5]. We build here on the BiDAF and QANet models by replacing some of the recurrent layers by encoder blocks similar to QANet, but where self-attention layers are implemented by a Transformer-XL, a new type of attention mechanism specifically designed to handle long-term dependencies. We show that by doing so, we can achieve good performances on SQuAD2.0, with a relatively small training and testing time.

2 Related Work

As stated above, we build on the BiDAF model developed by [4]. Its architecture is essentially composed of five modules. A first Embedding layer is used to compute the word-level embeddings of both the context and the question separately, which are then fed as input to two bidirectional LSTM encoders. The next module is the core part of the BiDAF model. A Bidirectional Attention Flow layer captures two-ways dependencies between context and question through the use of a trained similarity matrix. The output is a sequence of length identical to the context length. It is finally fed to a modelling layer consisting of another bidirectional LSTM, and a feed-forward output layer that predicts the position of the start and end index of the answer in the original paragraph.

To replace RNNs, an interesting candidate is the Transformer. Transformers are self-attention mechanism that were first introduced in [5]. Through the use of positional encoding, skip-connections, self-attentions and feed-forwards sub-layers, they avoid the pitfalls due to the recurrent nature of RNNs, and can therefore be trained and tested faster. Transformers have notably been used for the task of question answering in [6]. In their paper, the authors get rid of the RNNs present in the BiDAF model, and replace them with Encoder Blocks. These blocks consist of stacked convolutional neural networks to capture local dependencies in the input, and Transformers for medium-term dependency.

One limitation of Transformers, however, is their incapacity to handle long-term dependencies due to the limited scope of their self-attention heads, which treat the input segment by segment. A solution was recently proposed in [1], where the authors present the Transformer-XL. To handle long-term dependencies, the model reintroduces recurrence by reusing hidden states from previous segments. They serve as a kind of memory for the current segment, which can thus make use of the information gathered previously. In order to account for the introduction of this memory mechanism, a relative positional encoding needs to be introduced to keep the positional information coherent when reusing previous states.

We take inspiration from all of these models and propose to test the use of Transformer-XL for the SQuAD2.0 challenge. We replace the two RNNs used for encoding in the BiDAF model by Encoder Blocks similar to those presented in [6]. In our case, the Transformers are replaced by instances of Transformer-XL. We show that we can achieve good results with this new architecture while keeping relatively low training and test time.

3 Approach

3.1 Baselines

For this project, our baselines are variations on the BiDAF model. The most basic version of this baseline – as provided in the starter code – only makes use of word embeddings. For this project, we improve this baseline by incorporating character embeddings. The pre-trained character embeddings of both the context and the query are concatenated to the corresponding word embeddings before being fed to the encoder. As can be seen in the results section further down, this simple approach already yields a significant improvement over the original baseline.

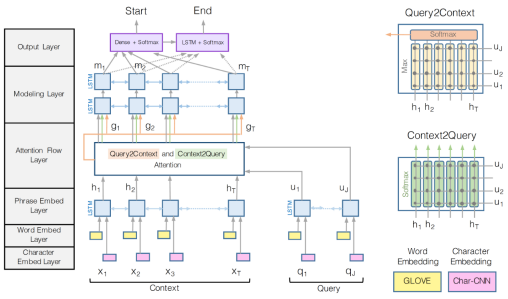


Figure 1: Overview of the BiDAF architecture. The key element is the the Query2Context and Context2Query Attention module in the center.

3.2 Our Model

Our main improvement here is to incorporate the Transformer-XL model to this architecture in a similar fashion to what was done in QANet. Essentially, Transformer-XL is a self-attention network that aims at improving on the vanilla Transformer by taking into account a much larger context. As illustrated below in Figure 2, the Transformer processes segments of its input sequentially one at a time. For a given segment, each index of the output is connected to several of the previous input indexes in a diagonal way. There are no connections between segments, however, which implies that the longest dependency that can be modeled is the length of a segment at most (about a few hundred words, according to [1]). This is an important limitation for who wishes to model longer-term dependencies.

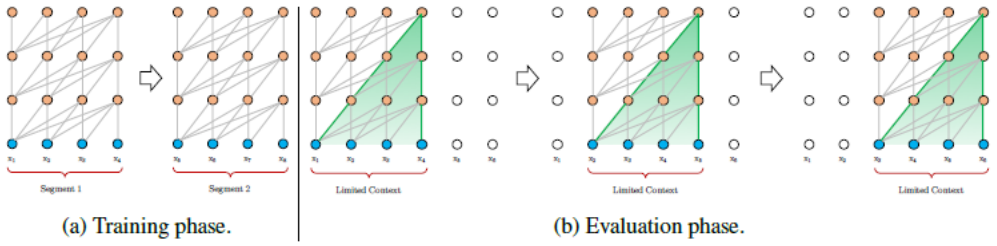


Figure 2: Illustration of the vanilla transformer architecture

The Transformer-XL addresses this issue by introducing a memory mechanism. Instead of treating each segment of input separately, connections are added between the hidden units of the previous and the current segment. Usually, the weights of the previous hidden units are frozen so as to not re-train them. This allows information to be passed along the segments, effectively introducing a recurrence mechanism at the segment-level.

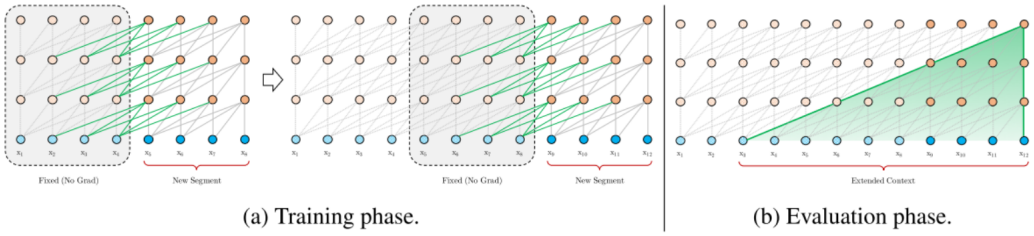


Figure 3: Transformer-XL is able to model longer-term dependencies by adding connections between the hidden layers of the current and the previous segment.

More precisely, let s_τ and $s_{\tau+1}$ be two consecutive segments. Additionally, let h_τ^n be the n^{th} hidden layer of segment s_τ. The first step of Transformer-XL is to concatenate the previous and current hidden units of each layer, taking care of freezing the previous ones:

$$\tilde{h}_{\tau+1}^n = \left[SG(h_\tau^n) \circ h_{\tau+1}^n\right]$$

where SG stands for "Stop Gradient". Then the key and value vectors of layer $n+1$ for the segment $\tau+1$ are computed the same way as in the vanilla transformer, but using this extended hidden state. Only the query uses the original, non-extended hidden state:

$$q_{\tau+1}^{n+1},\, k_{\tau+1}^{n+1},\, v_{\tau+1}^{n+1} = h_{\tau+1}^n W_q^T,\, \tilde{h}_{\tau+1}^n W_k^T,\, \tilde{h}_{\tau+1}^n W_v^T$$

$$h_{\tau+1}^{n+1} = \textit{\{Self attention + FF\}}(q_{\tau+1}^{n+1},\, k_{\tau+1}^{n+1},\, v_{\tau+1}^{n+1})$$

We take inspiration from the QANet architecture to adapt our original baseline. In the BiDAF model, the encoder and modeling layers are implemented using RNNs. The idea here is to replace the

encoder layers by two Transformer-XL. For simplicity, we only implement the "Relative Partial Learnable" version of the Transformer-XL. The majority of the code we added was borrowed from the paper's Github repository, although it had to be slightly modified to match our BiDAF baseline.

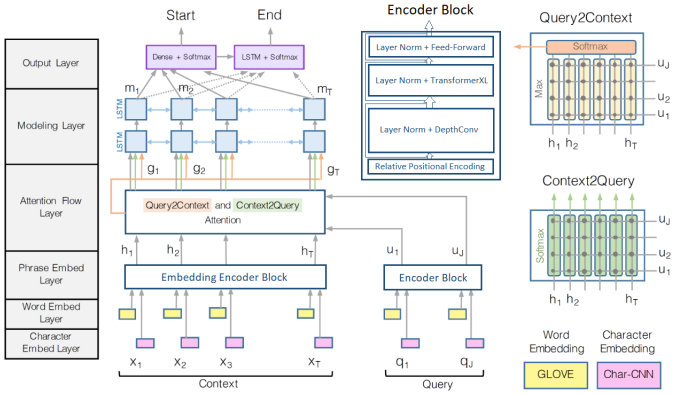


Figure 4: Modified BiDAF architecture for our model.

The overall architecture is illustrated in Figure 4. It contains the following elements:

Embedding Layer The embedding layer remains unchanged from the original BiDAF baseline. Fixed word embeddings are computed using GLoVE. When training the model with word vectors only, the embedding dimension is chosen to be $d_w = 500$. When character embeddings are added, we choose $d_w = 500$ and $d_c = 64$.. Word embeddings are further refined through a Linear layer followed by a Highway Encoder. Character embeddings are passed to a Convolutional layer, followed by another Highway Encoder. The resulting word and character embeddings are then concatenated to form a 1000-dimensional embedding for each word in the context and in the query.

Embedding Encoder Block The embedding encoder block is directly borrowed from the QANet architecture. There is one encoder block for each of the context and query. Each block consists of a stack of several depth-wise convolutional layers, followed by a self-attention layer and a feed-forward layer. In the original paper, the self-attention layer consisted of a vanilla Transformer. Here we modify this layer and replace it by a Transformer-XL attention. The number of convolutional layers is fixed to 4 and their parameters are kept the same as in [6]. The number of hidden layers in the Transformer-XL layer is set to 6, with $n_{heads} = 8$, $d_{model} = 128$ as in the original implementation of Transformer-XL. Due to memory constraints, the length of the memory (i.e. the number of previous heads to retain) is reduced to $l_{mem} = 128$.

Context-Query Attention Layer This layer is identical to the one used in the BiDAF baseline. Essentially, a similarity matrix is trained to capture two-ways dependencies between the encoded embeddings of the context and the query.

Model Encoder Layer For lack of time, we stick here with the Model Encoder used in BiDAF, that is to say a bi-directional, two-layer LSTM. In the future, it would be interesting to replace these LSTMs by Encoder Blocks as was done in QANet.

Output Layer Here again we follow the architecture of the BiDAF model. The output of the Encoder layer and of the Attention layer are fed through a succession of Feed-Forward and Convolutional networks with skip-connections to produce the final outputs, two vectors of length c_{len} representing the probability of each word in the context being the initial and final word of the answer, respectively.

4 Experiments

4.1 Data

We test all our baselines and models on the SQuAD2.0 dataset that was provided with the baseline. As opposed to SQuAD1.0, this dataset contains context/query pairs that do not have any answer, making it a more challenging task.

4.2 Evaluation Method

In order to match the SQuAD challenge metrics, we stick to the default evaluation that consists of the Exact Match (EM) score and the F1 score on the provided test set. During training, we also monitor the evolution of negative log-likelihood (NLL) to stop training when the model starts to show signs of overfitting.

4.3 Experimental Details

In addition to the two baselines mentioned above, we train two versions of our model, one without and one with character embeddings. The model configurations and training hyper-parameters are identical in both cases.

Model Configurations Due to time and memory constraints, we reduced the number of hidden layers in the Transformer-XL attention to 6 instead of 12. The length of the memory was also reduced from 512 to 128. However we kept the number of heads and hidden dimension to their original values of 8 and 128 respectively. The configuration of the other layers in the model are kept identical to the original BiDAF architecture, at the exception of the dimension of the word and character embeddings, chosen to be 500 and 64 when both are used, or 500 when only word embeddings are used.

Training Configuration To train our baseline, we stick with the Adadelta optimizer with a constant learning rate of 0.5. Evaluation is performed every 20000 iterations. For our own models, we take some inspiration from QANet and use Adam optimizer with a constant learning rate of 0.00025. This is the learning rate used in the original implementation of the Transformer-XL. We tried different values of learning rate ranging from 0.001 to 0.0001 and found this value to be the most adequate for our task. As mentioned in [1], warm-up has not been found to be useful when training the Transformer-XL, as opposed to the vanilla version of Transformer.

Additionally, we experimented with the β_1 and β_2 parameters of Adam, since [6] had somewhat modified the default values, but we didn't find any improvement over the standard values $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

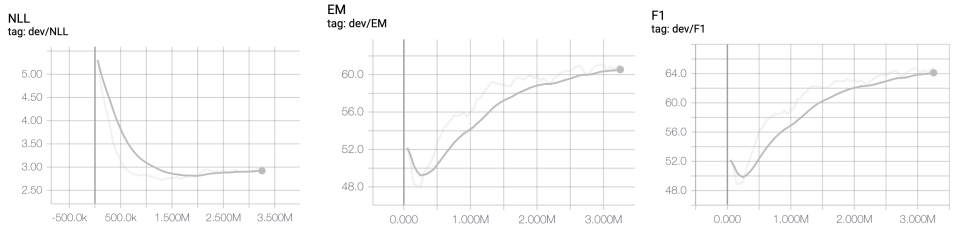
Finally, we used a decay rate of 0.999 for the exponential moving average of the parameters.

Infrastructure and Computation We trained all our models on an NV6 Microsoft Azure virtual machine with 6 vcpus and 56 GB memory.

For the sake of comparison, we report the following results for a batch size of 8. For our baselines, training time was of about 40 minutes per epoch. For our personal models, training time was of about 50 minutes per epoch. For these models, training was stopped after 2 million iterations, which makes up for a total training time of about $12h$.

4.4 Results

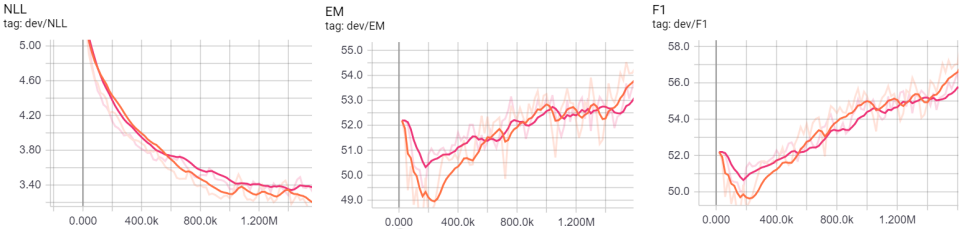
Below are the evolution of the NLL as well as the EM and F1 scores for our improved baseline. We see that the model starts to over-fit after about 2 million iterations. Final EM and F1 scores on the test set are reported in Table 1. We see that incorporating character embeddings considerably improves on the baseline score.



(a) NLL on dev set (b) EM score on dev set (c) F1 score on dev set

Figure 5: Dev metrics for the improved baseline

We now present in Figure 6 the same curves for our two models where encoding LSTMs have been replaced by Encoder Blocks with Transformer-XL attentions. In orange is the basic model and in red is the model that incorporates character embeddings. Once again, the model over-fits after around 1.5M iterations. The final scores on the test set are presented in the table. We see that our models achieve good scores on the SQuAD dataset, although they do not improve on their respective baseline. This is not too surprising given that the main advantage of Transformers is speed and not performance, and given that we somewhat scaled down the configuration of the Transformer-XL attention from its original implementation.



(a) NLL on dev set (b) EM score on dev set (c) F1 score on dev set

Figure 6: Dev metrics for our own models

Model	**EM score**	**F1 score**	**Time per epoch**
Baseline	55.99	59.29	$\approx$ 40 min
Baseline + char emb.	60.30	64.19	$\approx$ 40 min
Transformer-XL	52.7	55.6	$\approx$ 50 min
Transformer-XL + char emb.	51.11	55.12	$\approx$ 50 min

Table 1: EM and F1 scores on the test set

5 Analysis

The first thing to be noted is that none of our models can do as good as PCE models like BERT or ELMo. This was expected, as Contextual Embedding models represent the state-of-the-art for these kind of task. However our models still achieve good results in the non-PCE leaderboard, despite having been simplified.

Looking at the SQuAD2.0 leaderboard, it appears that our models do not achieve better scores than QANet, which uses Transformers. We think that this is due to the relatively small length of the inputs. The maximum number of words in a paragraph is 400. But, as stated in [1], transformers can process segments of "a few hundred" words. This is exactly the maximum length of our inputs, meaning that using Transformer-XL should not improve much on implementations that use the vanilla Transformer. Given that Transformer-XL is slightly more complex, this could partly account for the fact that we cannot seem to achieve better results. Below is a typical example of a context/query pair and its predicted answer for our improved baseline and our model without word embeddings:

Context *The biodiversity of plant species is the highest on Earth with one 2001 study finding a quarter square kilometer (62 acres) of Ecuadorian rainforest supports more than 1,100 tree species. A study in 1999 found one square kilometer (247 acres) of Amazon rainforest can contain about 90,790 tonnes of living plants. The average plant biomass is estimated at 356 47 tonnes per hectare. To date, an estimated 438,000 species of plants of economic and social interest have been registered in the region with many more remaining to be discovered or catalogued. The total number of tree species in the region is estimated at 16,000.*

Question *How many plant species are of interest to society and manufacturers exist in the amazon rainforest?*

Baseline answer $438,000$

Transformer-XL answer $1,100$

This example is typical of the limitations of our model. On many occasions, it seems to be able to look further in the text than the baseline model, at the cost of a slightly worse understanding of the meaning of both the question and the context. As a result, our model prefers to find a passage further in the text that matches more closely the question's wording (here "rainforest" and "species"), while the baseline seems to have a narrower view at the text, but to understand it better. Another consequence of this is that our model is more easily fooled by questions that do not have an answer. This matches our understanding that the encoding might be a limiting factor. Although the Transformer-XL attention can look further in the text, the encoding is of lower quality, meaning that it has a harder time grasping the meaning of complex questions and contexts.

Surprisingly, we note that models based on Transformer-XL do not reduce training time as we expected it. One reason could be that the reduced training time advertised in [1] was achieved on a model consisting only of a Transformer-XL attention. Here, we also add several layers of convolution that make the model more complex and likely slow it down. Moreover, it is surprising that the model that incorporates character embeddings cannot perform better than the one that does not. Although its performance is slightly better early on in the training, it quickly starts to follow the performance of the model with word embeddings only. This seems to also hint that there is some sort of bottleneck at the Encoder Block level that prevents the performance from improving past a certain amount.

6 Future Work

A lot of work remain to do to test the potential of using Transformer-XL for question answering. The first step would be to replace all the LSTMs with Transformer-XL Encoder Blocks, so as to follow the QANet architecture more closely. Hopefully, with the original parameters of the Transformer-XL and a better hyper-parameter tuning, it should be possible to achieve better results that the ones achieved by QANet.

7 Conclusion

For this project, we draw inspiration from BiDAF and the new Transformer-XL architecture to modify the QANet model in the aim of tackling the SQuAD2.0 challenge. By replacing the RNN encoders by Encoder Blocks with a Transformer-XL self-attention mechanism, we show that we can obtain good results while reducing training and testing time. This is an encouraging result as it could potentially allow to develop real-time question-answering systems, which is hard to achieved with traditional models based on RNNs like BiDAF. Further improvement of our models and hyper-parameter tuning should allow to surpass scores obtained by state-of-the-art non-PCE models such as QANet.

References

[1] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Language modeling with longer-term dependency. 2018.

[2] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*, 2018.

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. pages 5998–6008, 2017.

[6] 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.