Transformer Empowered BiDAF for SQuAD

Xin Chen xc1113@nyu.edu
Xinyi Liu xl2351@nyu.edu
Boran Wen bw1502@nyu.edu

May 12, 2019

Abstract

Question-Answering (QA) system has become one of the popular topics in the field of natural language processing. It aims to provide correct answers to users' questions from a given document, thus potentially saving humans a lot of time. This study demonstrates an improvement over the baseline BiDAF model provided by Stanford CS224n default project on the SQuAD 2.0 dataset [5]. The SQuAD 2.0 dataset extends the original SQuAD dataset with unanswerable questions, imposing an extra challenge for predicting if the question is answerable from the context. By adding a character embedding layer and combining the first bidirectional Long Short-Term Memory (LSTM) [2] layer with blocks of transformer encoders [9], this paper achieved an F1 score of 66.22 and an EM score of 62.86, which is better than the baseline 60.5 F1 score.

1 Introduction

The mastery of Question-Answering tasks is a strong indicator of achieving machine understanding and information retrieval. Similar to the baseline model, the model proposed by this study predicts the starting and ending position of the correct answer located in the context. Those contexts are extracted from Wikipedia articles. When the question is non-answerable from the context, the model should predict 0 as both starting position and ending position for the answer in the context.

Recently, a model relying entirely on an attention mechanism named Transformer [8] was developed by Vaswani. It can draw dependencies between input and output thus inspired us to replace or combine bidirectional LSTM with it. The Transformer allows more parallelization and has been shown to reach a new state of the art in machine translation quality with less training time compared to recurrent neural network.

2 Model Architecture

The baseline model is based on the Bidirectional Attention Flow (BiDAF) network proposed by Seo, M. et al. in 2016 [7]. This model consists of an input embedding layer, a contextual embed layer, an attention flow layer, a modeling layer and a output layer. This study experimented on the adding a character embedding layer into the input embedding layer, substitute Encoder

Blocks for the bidirectional LSTM structure or concatenate the Encoder Blocks and bidirectional LSTM structure together in the contextual layer.

Input Embedding Layer

We first added a character embedding layer to the input embedding layer of the baseline model. The character embedding of each word is generated by training a Convolutional Neural Network (CNN) on a pre-trained word embedding model named Word2Vec [4], following Y. Kim's experiment in 2014 [3]. This could act as a great complement to another pre-trained word embedding model GloVe, which is used for the word embedding layer in the input embedding layer. Here the dimension of a character is the same as the dimension of a word. As we set the maximal length of each word is 16, the character embedding of each word is the concatenation of 16 character embeddings with padding when it is necessary. We projected the word embedding and the character embedding onto the hidden layer with same size before concatenated them together to feed into the contextual embed layer. Therefore, the word embedding and the character embedding have the same effect on the final result.

Given input vectors $v_w \in \mathbb{R}^(n)$ denoting word embedding and $v_c \in \mathbb{R}^(n*16)$ denoting character embedding after reshape, we have the output as

$$
\begin{aligned}
v_{projw} &= [W_1(v_w); ...; W_h(v_w)] \in \mathbb{R}^n \\
v_{projc} &= [W_1(v_c); ...; W_h(v_c)] \in \mathbb{R}^n \\
v_{wc} &= [v_{projw}; v_{projc}] \in \mathbb{R}^{2n} \\
v_{hw} &= Highway(v_{wc}) \in \mathbb{R}^{2n}
\end{aligned}
$$

Contextual Embed Layer

Contextual embed layer takes the output of the input embedding layer as input. The baseline utilizes bidirectional LSTM to incorporate the surrounding words to generate the refined embedding of each word for both the context and the query. This study first replaced the Bi-LSTM with Blocks of Encoder, which it is introduced next. Moreover, this study further attempted to concatenate the output of original Bi-LSTM and the output of Encoder Blocks together.

This study chose the blocks of transformer encoders following the QANet developed by Adams Wei Yu in 2018 [9]. Inspired by the ideas of transformer, QANet combines self-attentions, feed-forward networks with convolutions. Layersnorm and residual connection are used between layers of the block encoder. While convolution helps take advantage of the local structure of the context, the self-attention captures the global interaction between words. The nature of feed-forward network reduces the training time in comparison with the recurrent neural network. In this study, we used 4 blocks of encoder.

The formula for the structure that concatenates both bidirectional LSTM and Encoder Blocks are shown below. Noted that since the LSTM is computed in two directions, the hidden size for each word is the double of the hidden size of input for this layer.

$$
\begin{aligned}
E_{i_{rnn}} &= BiLSTM(X)_i \in \mathbb{R}^{2n} \\
E_{i_{blk}} &= EncBlock(X)_i \in \mathbb{R}^n \\
E_i &= [E_{i_{rnn}}; E_{i_{blk}}] \in \mathbb{R}^{3n}
\end{aligned}
$$

Attention Flow Layer

Attention flow layer, which is the key part of the model, also calculates the attentions in two ways, which are from context to query and from query to context. The attention is derived from calculating the similarity matrix between the embedding of the context and the embedding of the query from contextual embed layer. From context to query, the row-wise softmax of the similarity matrix is used as weights to sum all the refined word embeddings from the query. From question to query, the column-wise softmax of the similarity matrix is multiplied with the row-wise softmax of the similarity matrix from before. Then the results is used as weights to sum all the refined word embeddings from the context. The refined word embeddings from context, output from context to query attention and output from query to context attention together form the final output of the attention flow layer.

Modeling Layer

Modeling layer utilizes the bidirectional LSTM network again. It takes the output from the attention flow layer as input. Similar to the contextual embed layer, this structure allows the system to incorporate the surrounding attention outputs. The only difference from the contextual embed layer is that modeling layer has two-layer LSTM structure.

Output Layer

Finally, the output layer is responsible for generating the probabilities vector of each position to act as starting position as well as the probabilities vector of each position to act as ending position. The starting probabilities vector is calculated by taking the softmax of concatenation between the output from the attention flow layer and the modeling layer. The ending probabilities vector, on the other hand, replace the output from the modeling layer by its output through a bidirectional LSTM network. The answer predicted by the model therefore will be span from the position with highest probability in starting probabilities vectors to the position with highest probability in ending probabilities vectors in the context.

Final Architecture

The final architecture is shown in Figure 1.

3 Experiments

3.1 Datasets

To train our BiDAF model with transformer, we used the SQuAD 2.0 data set[6]. It has three splits **train**, **dev** and **test**. The train set has 129,941 examples, the dev set has 6,078 examples and the test set has 5,915 examples. Since we do not have access to submit our results on test set to get the evaluation metrics for our model, here we only presented the metrics on the dev set.

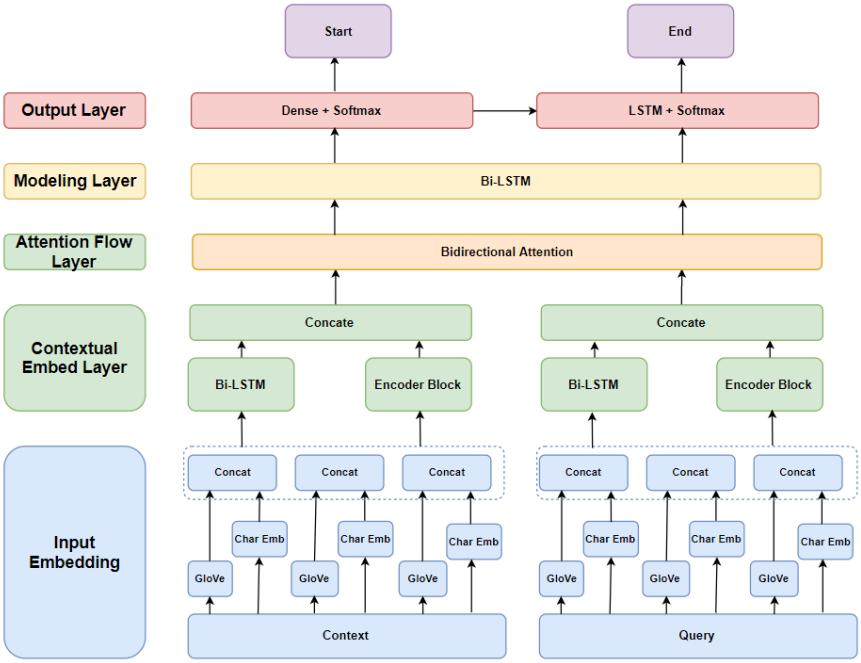

Figure 1: BiDAF Model with Transformers

3.2 Evaluation

This study mainly used two metrics to evaluate the performance of our model, F1 score and EM score. EM score tests the accuracy of producing exactly same result as ground truth. F1 score, on the other hand, uses a less strict metric, involving both the precision and recall rate.

3.3 Training Details

The training and development set were all ran on NYU HPC clusters with GPU. Each experiment ran for 30 epochs, which approxiately took around 12 hours in the final model . All models are implemented using PyTorch. We fine-tuned the parameters for the optimization method and the batch size.

4 Results and Observations

As shown in Table 1, the character embedding layer can significantly improve the performance of original baseline model by around 5 F1 score. The substitution of encoder blocks for bidirectional LSTM in the contextual embed layer also help improve the performance slightly. The improvement is around 2 F1 score. The required training time for the baseline + Encoder Blocks model significantly decreased, while the required training time for the baseline + Character Embedding model increased. For the baseline + Character embedding model, it took around 6 hours to finish 30 epochs in training. For the baseline + Encoder

Blocks model, it took around 5 hours to finish 30 epochs.

However, when the adding of the character embedding layer and the substitution of Encoder Blocks for bi-LSTM were both applied to the baseline model, the performance of the model did not further increase compared to the simple baseline + Character Embedding model. Therefore, this study further concatenated the bi-LSTM structure with the Encoder Blocks together in the contextual embed layer. With this final model structure, we achieved an F1 score of 66.22 and an EM score of 62.86.

In Figure 2, we presented the change of F1 score and EM score evaluated in the development set for the final model along with the number of epochs in training.

Table 1: Results From Experiments

Model	**EM**	**F1**
Baseline	57.13	60.50
Baseline + Character Embedding(CE)	62.24	65.24
Baseline + Encoder Blocks(EB)	58.70	62.00
Baseline + CE + EB	62.24	65.30
Baseline + CE + concate(EB, Bi-LSTM)	62.86	66.22

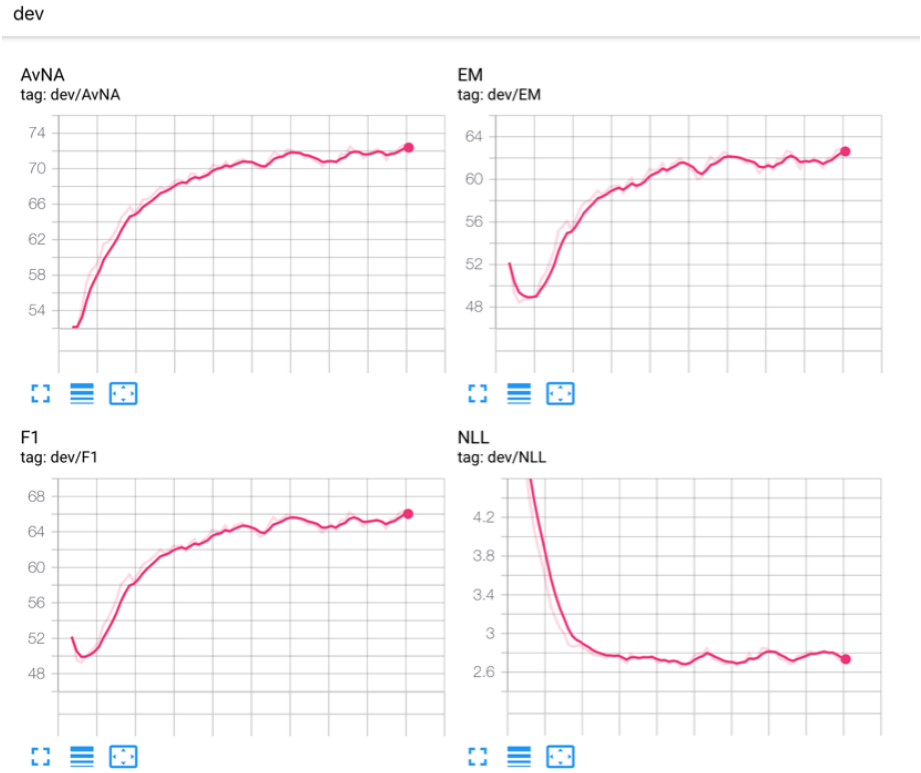

Figure 2: Evaluation of The Best Model in The Development Set

5 Conclusion

In this study, we combined character embeddings and a transformer into the baseline model, which is proved to be effective. The final result we gained is far better than the baseline model. For the future work, there are still many ways to further enhance the performance. For example, data can be augmented to give the model more information, or more features can be combined to better explore the potential of the model [1]. In addition, concatenating the bi-LSTM with the Encoder Blocks in the modeling layer as we did in the contextual embed layer also have a possibility to further improve the performance.

6 Acknowledgement

Our NLP term project team gratefully thanks professor Ralph Grishman, who gave an entirely amazing and excellent NLP course in the passing semester. Through all the written homework and programming assignments, the team members has gained a solid theoretical foundation for NLP basics as well as a systematic way of how to analyze and apply NLP algorithms, which we believe will pave a solid foundation in our future study and work in related fields.

References

[1] D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[5] P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

[6] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR, abs/1606.05250*, 2016.

[7] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[9] 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. *arXiv preprint arXiv:1804.09541*, 2018.