

Extending QANet with Transformer-XL

Stanford CS224N {Default} Project

Sheng Li

Aero & Astro Engineering

Stanford University

lisheng@stanford.edu

Abstract

This default project tackles the machine reading comprehension (RC) problem on the SQuAD 2.0 dataset [1]. It involves inputting a context paragraph and a question into a model and outputting the span of the answer from the context paragraph. This project aims to extend the QANet [2], so that it can effectively perform RC on SQuAD 2.0. The segment-level recurrence with state reuse from Transformer-XL [3] is integrated into QANet to improve its ability of tackling long context paragraph (referred to as QANet-XL). In addition, character embeddings and a fusion layer after context-query attention are used to extend the BiDAF [4] baseline. Experiments show that QANet-XL underperforms the vanilla QANet and outperforms the extended BiDAF. *The segment-level recurrence mechanism from Transformer-XL is proven not a proper improvement for QANet on the SQuAD 2.0 dataset, since Transformer-XL is designed for extra-long continuous text.* For the dev set, The extended BiDAF achieved $EM/F1 = 62.16/65.98$, the vanilla QANet achieved $EM/F1 = 66.81/70.38$, and the QANet-XL achieved $EM/F1 = 63.12/66.67$. A majority voting ensemble model based on previous mentioned models achieved $EM/F1 = 66.85/69.97$ on the test set.

1 Introduction

Machine reading comprehension (RC) and automated question answering have been topics of interest in natural language processing. This project tackles a specific form of reading comprehension given by SQuAD 2.0 [1]: inputting a context paragraph to a model and outputting the span of the answer from the context paragraph. One of the challenges of SQuAD 2.0 is that a model must not only answer questions when possible, but also determine when no answer is supported by the context paragraph and abstain from answering [1].

Processing the long context paragraphs is the major difficulty in the RC tasks. One way is using a recurrent model to sequentially process the input tokens. A successful example is the bidirectional attention flow (BiDAF) model [4]. The drawbacks of such recurrent models are that it is slow for training and inference, and there is the forgetting issue, especially for long sequences. Another way is processing the input sequences using Transformer-like [5] encoders, which leverages multi-layer multi-head self-attention to deal with the interactions across long sequences. QANet [2] is a successful example in this approach. QANet achieved leading performance on the SQuAD 1.1 [6] dataset.

In this project, QANet is used as a backbone for extension. The Transformer-like [5] encoder of QANet suffers from the limit of scaling-up with long context paragraph. Borrowing ideas from Transformer-XL [3], its segment-level recurrence with state reuse could be helpful to improve the ability in encoding long context paragraphs. Therefore, QANet-XL is implemented with intra-context segmentation. In addition, BiDAF is used as a baseline. Character embeddings and a fusion layer after the context-query attention is added to improve the performance of BiDAF.

Experiment results show that the QANet-XL underperforms the vanilla QANet and outperforms the extended BiDAF. A reasonable explanation for QANet-XL’s worse performance than the vanilla version is that most of the lengths of the context paragraphs in the SQuAD 2.0 dataset are within 250 words. These lengths are within the representational capacity of the vanilla QANet. While, splitting the context paragraphs to shorter segments does not help improving the capacity of encoding, however in turn, harms the continuity of encoding. The RC task particularly requires a continuous understanding of the context paragraph, since the output is a span of the paragraph. QANet-XL might work better on datasets with much longer context paragraphs.

A majority voting ensemble model is built based on the extended BiDAF, the vanilla QANet and QANet-XL, producing the results outperforming all these single models. At the time of writing, the ensemble model achieved $\text{EM}/\text{F1} = 66.85/69.97$ on the test set.

2 Related Work

The existing work dealing with the RC task can be roughly divided into two kinds: (1) pre-trained contextual embeddings (PCE) based models, and (2) Non-PCE models.

PCE based models are those finetuned over large pre-trained languages models like ELMo [7], BERT [8] and ALBERT [9] for down-stream tasks. The state-of-the-art scores on the SQuAD 2.0 leaderboard are mostly achieved by these PCE based models, for example, the Retro-Reader [10] (finetuned on ALBERT [9]) and SG-Net Verifier [11] (finetuned on XLNet [12]). Undeniably, PCE based models have greatly boosted the performance in various natural language processing tasks.

Non-PCE models show inferior performance in the RC task comparing with PCE based models on both SQuAD 1.1 and 2.0 dataset. Although they have limited representational capacity and no more leading performance in many tasks including RC, Non-PCE models still have flexibility and novelty in their architectures, which can inspire developments in NLP research. QANet [2] was one of the top Non-PCE models for SQuAD 1.1. Other non-PCE models like BiDAF [4] and Threeway Attentive Networks (TriAN) [13] both employs bi-directional LSTM to solve the RC task.

3 Approach

This section introduces QANet [2], Transformer-XL [3] and how to extend QANet with it, and finally the BiDAF baseline.

QANet:

The vanilla QANet [2] is the backbone of this project. An overview for QANet architecture is in Fig. 1. There are five major modules in QANet:

1. **Input embedding** projects input word and character token indices to their embeddings. The word embeddings are initialized from the 300-dimensional pre-trained GloVe [14] and are fixed throughout training. The 64-dimensional character embeddings are randomly initialized from Gaussian distributions. A two-layer highway network [15] is used to integrate the word embeddings and the character embeddings.
2. **Embedding encoder** consists of one encoder block. An encoder block is a stack of positional encoder, depthwise separable convolution layers, dot product self-attention, and finally a feed-forward layer. Layer-normalization and residual connections are used in encoder blocks. See the right side of Fig. 1 for details.
3. **Context-Query attention** is standard in reading comprehension models. QANet use the context-query attention the same as the attention in BiDAF [4]. We use C and Q to denote the encoded hidden states of context and query. The context-to-query (C2Q) attention is constructed as $A = \bar{S} \cdot Q^\top$, where the similarity score $S_{ij} = W_0[Q_j; C_i; Q_j \odot C_i]$, $\bar{S}$ is a softmax row normalization of S , and W_0 is a trainable weight matrix. The query-to-context (Q2C) attention is constructed as $B = \bar{\bar{S}} \bar{\bar{S}}^\top C^\top$, where $\bar{\bar{S}}$ is a softmax column normalization of $\bar{S}$. The final output of context-query attention is a combination of C2Q and Q2C: $G = [C; A; C \odot A; C \odot B]$. Details can be found in the default project instruction, BiDAF paper [4] and the QANet paper [2].

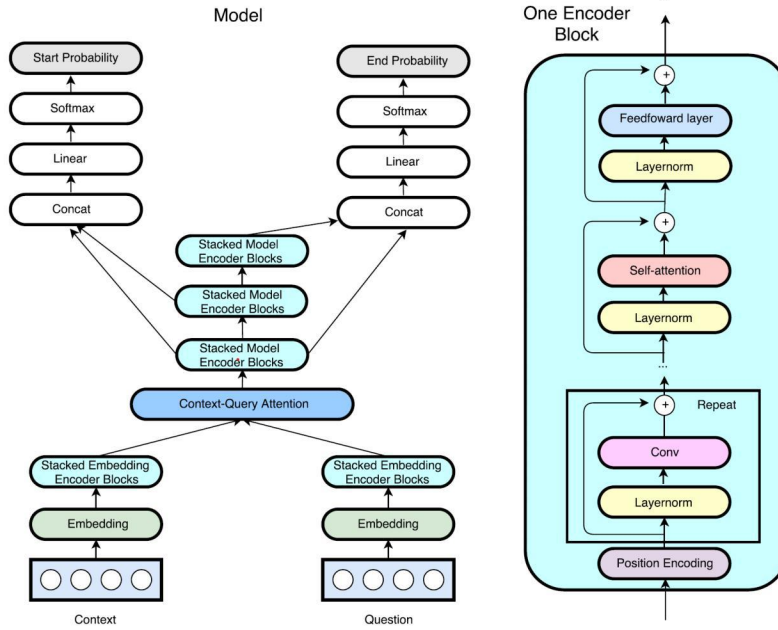


Figure 1: An overview of the QANet architecture [2]

4. **Model encoder** is very similar to the embedding encoder except for the number of encoder blocks and the number of the depthwise separable convolution layers in each block. All the three model encoders share their weights.
5. **Output pointer** takes in the outputs of the model encoders and outputs the log probabilities of the predictions for the start and end indices of answer spans: $p_s = \log_softmax(W_1[M_0; M_1])$ and $p_e = \log_softmax(W_2[M_0; M_2])$, where W_1 and W_2 are trainable variables and M_0, M_1, M_2 are the outputs of three model encoders respectively. The goal is to maximize the sum of log probability.

Transformer-XL: QANet suffers from its bottleneck in scaling up with long context paragraph. Transformer-XL [3] addresses this problem by segment-level recurrence with state reuse and relative positional encoding. These two differences enables Transformer-XL to encode longer sequences more effectively (adaptation in progress):

1. **Segment-level recurrence** reuses the hidden states obtained in previous segments as an extended context when processing the next new segment. Instead of computing the hidden states from scratch, the cached and reused hidden states serve as memory for the current state. Denoting the n -th layer hidden state sequence produced for the τ -th segment s_τ by h_τ^n . Then, the n -th layer hidden state for segment $s_{\tau+1}$ is calculated as follows:

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

where SG stands for stop gradient.

2. **Relative positional encoding** is an effective relative positional encoding to keep positional information coherent when the states are reused. The fundamental idea is to only encode the relative positional information in the hidden states. For an N -layer Transformer-XL, the segment-level recurrence equipped with relative positional encoding is calculated as follows

for $n = 1, \dots, N$:

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

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

QANet-XL: The main idea of integrating Transformer-XL with QANet is adding segment-level recurrence with state reuse into the Transformer-like encoder blocks of QANet. The resulting model is referred to as QANet-XL.

Transformer-XL encodes extra-long context by splitting a continuous text into segments and performing segment-level recurrence. While in the RC task given by SQuAD 2.0, consecutive context paragraphs are not necessarily related with each other, not to mention being continuous in terms of context. Therefore, it is improper to perform *inter-context* segment-level recurrence. A proper way is to divide a context paragraph into several segments, and then perform segment-level recurrence within each context paragraph. This is referred to as *intra-context* segment-level recurrence. This is different from the original purpose of Transformer-XL.

A major concern is that the context paragraphs from SQuAD 2.0 have limited lengths (95% of the lengths of context paragraph are within 250 words). Further segmentation on these context paragraphs could break its internal continuity in meanings. In addition, the RC task specifically requires outputting a span of context paragraph, splitting the context paragraph could be harmful for this goal, since it is possible a cut is made right across the span of the answer. Breaking context continuity is possibly harmful for the context-query attention mechanism. It is anticipated that QANet-XL will not be helpful in improving the performance of QANet on SQuAD 2.0. However, if there is a dataset with much longer context paragraphs (say 1000 words on average), QANet-XL might show its advantage.

In practice, each context paragraph is divided into $\lceil L_c/L_m \rceil$ segments, where L_c is the length of the context paragraph and L_m is the length of the memory sequence.

BiDAF and Extension: The baseline approach is a recurrent model: the Bidirectional Attention Flow (BiDAF) [4]. It is provided by the instructors for the default project. Two extensions are made to improve its performance:

1. Adding character embeddings to the inputs;
2. Adding a single layer fusion function with dropout after the concatenation output of the context-query attention.

Majority Voting Ensemble: This ensemble method directly reads the output predictions from various models, choosing the predictions with majority votes as final predictions. Tie-breaking is done by maximizing the F1 score.

4 Experiments

Implementation and Code References: The starter code for data preprocessing, data loading, training, testing and implementation of BiDAF baseline are provided by instructors (<https://github.com/minggg/squad>). The implementation of the vanilla QANet is based on the code from <https://github.com/andy840314/QANet-pytorch>. The implementation of Transformer-XL modules is mostly based on <https://github.com/kimiyoung/transformer-xl> and <https://github.com/inSam/QA-XL>.

I mainly (1) understood and implemented APIs of the referred codes (QANet and Transformer-XL) to properly integrate them into the framework of the starter code for convenient training and testing

on SQuAD 2.0, (2) implemented extensions on BiDAF, (3) implemented intra-context segment-level recurrence for QANet-XL, (4) implemented majority voting ensemble.

Dataset: The dataset is the SQuAD 2.0 [1] splits provided by the instructors (<https://github.com/minggg/squad/tree/master/data>). The train split has 129,941 examples. The dev split has 6078 examples.

Evaluation method: The metrics used by the RC task are exact match (EM) and F1 Score and AvNA.

- **EM** is a binary measure (i.e. 1/0) of whether the model prediction matches the ground truth answer exactly.
- **F1** is the harmonic mean of precision and recall. Precision refers to what percentage of words in predicted answer is in the ground truth answer, and recall refers to what percentage of words in the ground truth answer is in predicted answer.
- **AvNA** is a metric provided in the default project instruction. AvNA stands for Answer vs. No Answer and it measures the classification accuracy of the model when only considering its answer vs. no-answer predictions.

Experimental details: All the models are supposed to train for a maximum of 30 epochs on the **train** split of provided dataset. Early stop could happen if improvement stalls. The models use the pretrained word embeddings from the 300-dimensional GloVe [14], and the 64-dimensional randomized (Gaussian) character embeddings.

The model configuration and training settings of the BiDAF baseline completely follow the provided code. It is trained 9 hours on one-half Nvidia K80 GPU (12GB, Azure NC6). QANet and QANet-XL are trained on one Nvidia V100 GPU (16GB, Azure NC6s_v3). QANet and QANet-XL have variants as listed in Table 1.

Table 1: QANet variants and training configurations.

Variant	QANet-mid	QANet-large	QANet-XL-mid	QANet-XL-long
Model Connector Hidden Size	96	128	128	128
Self-Attention Head Dimension	16	16	16	16
# Self-Attention Heads	6	8	8	8
# Embedding Encoder Block Convolutions	4	4	4	4
# Model Encoder Block Convolutions	2	2	2	2
# Embedding Encoder Blocks	1	1	1	1
# Model Encoder Blocks	5	7	7	7
Memory Sequence Length	N/A	N/A	128	256
Batch Size	32	24	20	20
Training Time	9hr	15hr	21hr	22hr

Table 2: Results on **dev** split.

Model	AvNA	EM	F1
BiDAF	67.84	57.74	61.18
Extended BiDAF	72.59	62.16	65.98
QANet-mid	74.53	64.27	67.98
QANet-large	76.51	66.81	70.38
QANet-XL-mid	71.10	61.44	65.12
QANet-XL-long	73.07	63.12	66.67
Ensemble	N/A	68.75	71.81

Input sequence length limits are the same for the baseline, QANet and QANet-XL: (1) max number of words in a context paragraph: 400; (2) max number of words in a question: 50; (3) max number of characters to keep from a word: 16.

The optimizer for QANet and QANet-XL is the PyTorch [16] implementation of Adam [17], with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$, and weight decay 5×10^{-8} . A dropout rate of 0.1 is used for word embeddings, and a dropout rate of 0.05 is used for character embeddings.

The results on **dev** split are listed in Table 2. The extension for BiDAF significantly improves its performance. QANet-large performs the best as a single model. As anticipated, the intra-context segment-level recurrence borrowed from Transformer-XL does not effectively improve QANet. One finding is that QANet-XL with a longer memory sequence performs better. The majority voting ensemble that combines all the single models outperform its components on the dev set. The ensemble model achieved $EM/F1 = 66.85/69.97$ on the test set.

5 Analysis

Analysis for Prediction Length

Analysis is performed for the ensemble model.

One of the biggest challenges in RC tasks is long answer spans. A long span of answer requires a deep understanding of the context paragraph and query. Fig. 2 shows the distribution of the lengths of ground truth answers in the **blue bar plots** (length measured by words). Zero length indicates non-answerable. We can see that around a half of the ground truth answers are non-answerable. Most of the ground truth answers are within 5 words. The frequencies of long-span answers (> 10) are non-significant.

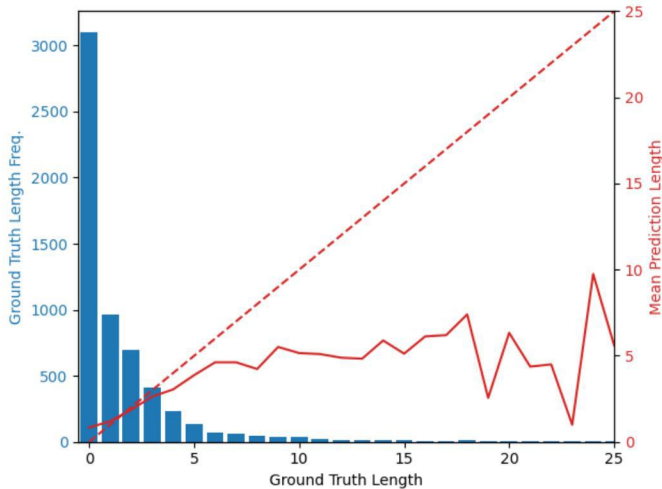


Figure 2: Analysis on the lengths of ground truth answers and the lengths of predictions. (Length measured by words)

Fig. 2 also shows the mean prediction length in the **red line** (the dashed line indicates 100% accuracy in length). The model tends to over estimate the length of the answer when the length of the ground truth answer is shorter than or equal to 1 word. The model is most accurate with the answers with 2 words. There is a clear plateau for the length of answers the model can produce. The mean prediction length stalls at around 5 words despite the increase of the length of ground truth answers. This can be explained by the lack of such data points in the dataset. Overall, the model is accurate in the length of answer for those with abundant data points ($\text{length} \leq 3$) in the dataset.

Error Example Analysis

Analysis is performed for best performing QANet-large.

A few examples of prediction error are listed in Table 3. In example (1), the model failed to match the description of “full size” from the question with the context. It output the first available passable answer from the context paragraph which does not fit with the description. Similarly in example (2), the model successfully matched “the 18th century” between the context and question, but failed to distinguish “snake” from “scorpion”. In example (3), the question does not directly coincide with the a continuous part of the context paragraph. Instead, it consists of two parts that is separated in the

context: “achieve a balance of between” and “speakers”. The context-query attention failed to relate separated parts in the context paragraph and reason about it.

These three examples demonstrate a deficiency of matching power of the context-query attention. If working properly, the attention mechanism should focus the context that fits the description in question and be able to relate tokens across relatively long spans. Improving the context-query attention should be promising for improving the performance in RC tasks.

Table 3: Error examples.

Truncated Context Paragraph	Question	Ground Truth	Prediction
(1) ... Instead of valves, the entire cylinder rocks, or oscillates, such that one or more holes in the cylinder line up with holes in a fixed port face or in the pivot mounting (trunnion). These engines are mainly used in toys and models, because of their simplicity, but have also been used in full size working engines, mainly on ships where their compactness is valued.	Full size working engines on what vehicles sometimes use oscillating cylinder steam engines?	ships	toys and models
(2) ... In the 18th century, Pierre-Louis Moreau de Maupertuis made experiments with scorpion venom and observed that certain dogs and mice were immune to this venom. This and other observations of acquired immunity were later exploited by Louis Pasteur in his development of vaccination and his proposed germ theory of disease. Pasteur’s theory was in direct opposition to contemporary theories of disease, such as the miasma theory ...	What scientist experimented with snake venom in the 18th century?	N/A	Pierre-Louis Moreau de Maupertuis
(3) The Presiding Officer (or Deputy Presiding Officer) decides who speaks in chamber debates and the amount of time for which they are allowed to speak. Normally, the Presiding Officer tries to achieve a balance between different viewpoints and political parties when selecting members to speak ...	What does the Presiding Officer try to achieve a balance of between speakers?	different viewpoints	N/A

6 Conclusion

This default project tackles the machine reading comprehension (RC) problem on the SQuAD 2.0 dataset [1]. This project extends QANet [2] with the segment-level recurrence with state reuse from Transformer-XL [3]. It also extends BiDAF [4] by adding character embeddings and a fusion layer after context-query attention. Experiments show that QANet-XL underperforms the vanilla QANet and outperforms the extended BiDAF. The segment-level recurrence mechanism from Transformer-XL is proven not a proper improvement for QANet on the SQuAD 2.0 dataset, since breaking context continuity is harmful for the context-query attention mechanism. A majority voting ensemble model achieved the best performance.

References

- [1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don’t know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.
- [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, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Language modeling with longer-term dependency. 2018.
- [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, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.
- [6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.
- [7] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.
- [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [9] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.
- [10] Zhuosheng Zhang, Junjie Yang, and Hai Zhao. Retrospective reader for machine reading comprehension. *arXiv preprint arXiv:2001.09694*, 2020.
- [11] Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. Sg-net: Syntax-guided machine reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9636–9643, 2020.
- [12] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.
- [13] Liang Wang, Meng Sun, Wei Zhao, Kewei Shen, and Jingming Liu. Yuanfudao at SemEval-2018 task 11: Three-way attention and relational knowledge for commonsense machine comprehension. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 758–762, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.
- [14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.
- [16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
- [17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.