Bidirectional Attention Flow Using Answer Pointer and QANet

Stanford CS224N Default Project
Track: SQuAD

Qingxi Meng
Department of Electrical Engineering
Stanford University
qingxi@stanford.edu

Yuxiao Chen
Department of Electrical Engineering
Stanford University
yuxiaoc@stanford.edu

Abstract

Machine comprehension and question answering have been very popular over the past few years in the natural language processing community. In this work, we choose the SQUAD default project and we improve the baseline BiDAF model with character level embedding. We also implement and improve the QANet [1] model, and explore both a lightweight model and a larger model with more parameters. In addition, we also implement the Answer Pointer model proposed by the work [2] and replace the regular output layers in both BiDAF and QANet with our specially designed Answer Pointer output layer, and perform experiments on the SQuAD 2.0 dataset. We observe that the large QANet with character embedding and Answer Pointer achieve the best results with 68.29 F1 score and 64.98 EM score on the development dataset, and 64.87 F1 score and 61.32 EM score on the test dataset.

1 Key Information to include

- TA mentor: Grace Lam
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Introduction

Machine comprehension and question answering have been very popular over the past few years in the natural language processing community. A lot of research work tries to find the best way for computers to understand a given context and output the correct answers to corresponding questions. Traditionally researchers tended to use lots of human-crafted features and relied on NLP pipelines that involve multiple steps of linguistic analysis and feature engineering, including syntactic parsing, named entity recognition, question classification, and semantic parsing. Recently, with the advances of deep learning, there has been much progress in building end-to-end neural architectures for various NLP tasks [3] [4] [5]. Specifically, with the help of emerging high-performance GPUs, deep learning methods, such as Birectional Attention Flow (BiDAF) [4], QANet [1], and Match-LSTM [2], have been very successful in the area of machine comprehension. In addition, the emergence of SQuAD 2.0 [6], a crowd-sourced version of SQuAD 1.0, further expedite the research of machine comprehension. Since the SQuAD 2.0 were created by humans through crowdsourcing, it makes the dataset more realistic with the addition of adversarial no-answer questions.

In this work, we choose the SQUAD default project and we improve the baseline BiDAF model with character level embedding. We also implement and improve the QANet [1] model, and explore both a lightweight model and a larger model with more parameters. In addition, we also implement

the Answer Pointer model proposed by the work [2] on SQuAD dataset. We replace the regular output layers in both BiDAF and QANet with our specially designed Answer Pointer output layer, and perform experiments on the SQuAD 2.0 dataset.

3 Related Work

There have been a number of studies proposing end-to-end neural network models for machine comprehension. Specifically, there are two main promising methods for the machine comprehension and question answering: RNN-based neural network and transformer-like neural network.

The first common approach is to use recurrent neural networks (RNNs) to process the given text and the question in order to predict or generate the answers. For example, the work [7] tries to process large amount of information using uni-directional RNN network structure. Later works also explore the attention mechanism used on top of RNNs in order to match the question with the given passage[8] [9] [10]. Recent works also explore other model structure and modification. For example, the Bi-Directional Attention Flow (BIDAF) network [4] propose a multi-stage hierarchical process and use bi-directional attention. The work [2] proposed to combine match-LSTM with the Answer Pointer layer, which selects the end conditioned on the start.

One big disadvantage of RNN-like structure is that it is very slow for large model [1]. In the contrast, for transformer-like neural network, they instead exclusively use convolutions and self-attentions as the building blocks of encoders that separately encodes the query and context. The QANet model[11] combined local interactions captured by convolution models and global interactions captured by self-attention models. However, one disadvantage of Transformer models is that they have fixed-length context in the setting of language modeling [9]. One improved version Transformer-XL [12] solves this problem by proposing a segment-level recurrence mechanism to learn long-distance dependencies among words.

4 Approach

In our final project, we explore six main structures on top of the baseline BiDAF model. Specifically, we explored BiDAF with Character Embedding, BiDAF with Character Embedding and Answer Pointer, QANet with Character Embedding, and QANet with Character Embedding and Answer Pointer, and a more complex QANet with Character Embedding and Answer Pointer. We adapt some utility functions from the work [1] and write the main structure of QANet model on our own. We implement the character-level embedding, various versions of Answer Pointer from scratch.

4.1 Baseline

For baseline, we used the provided Bidirectional Attention Flow (BiDAF) model in the default final project[4].

4.2 BiDAF with Character Embedding

Inspired by the the work [4], we add the character-level embedding to the baseline model. Each word in the context and question is mapped to a high dimensional vector space. Then, we feed this embedding vector as an input to a 2-dimensional CNN network. Then, we apply a ReLU function and a max-pooling layer over the output of the CNN network. We also apply a dropout layer with dropout rate of 0.1 for both the word embedding and the character embedding. In the end, we concatenate the character-level embedding vector with the word embedding vector.

4.3 BiDAF with Character Embedding and Answer Pointer

Inspired by work [2], we implemented the Answer Pointer for BiDAF. The purpose of the Answer Pointer layer is to condition the probability of selecting a word as the end token by the probability of selecting a word as the start token. For the Answer Pointer, it will generate two integer indexes indicating the start and end positions of the selected tokens in the original passage. As shown in equations below, the attention mechanism is used again to obtain an attention weight vector where

V, W^a, b^a, v and c are the parameters to be learned, h_i^a is the hidden vector of the answer-LSTM at position i, and $\tilde{H}$ is the concatenation of the hidden state from the attention layer and model layer.

$$
\begin{aligned}
F_{start} &= tanh(V\tilde{H}^T + b^a) \\
p_{start} &= softmax(v^T F_{start} + c) \\
h_i^a &= \overrightarrow{LSTM}(\tilde{H}^T p_{start}, h_{i-1}^a) \\
F_{end} &= tanh(V\tilde{H}^T + W^a h_i^a + b^a) \\
p_{end} &= softmax(v^T F_{end} + c)
\end{aligned}
$$

Figure 1 shows the overall model architecture for the BiDAF with character embedding and Answer Pointer. In our work, we made some modification to the Answer Pointer in the original work [2]. First of all, we concatenate the outputs from the attention layer and the modeling layer and feed them into a linear layer such that the new input hidden state context representations could keep the information from different scales and distances: both the attention of words at earlier layers and the hidden states from the latest layers. Another modification we made is to condition the probability of choosing a word as the end token on both the probability of choosing start token and the hidden state of start and end tokens. We observe that this modification works pretty well in our experiments, and one reason may be that the probability of start token may lose some useful information about the token itself, and complementing with hidden states of start and end token helps the model to better learn the correlations between them.

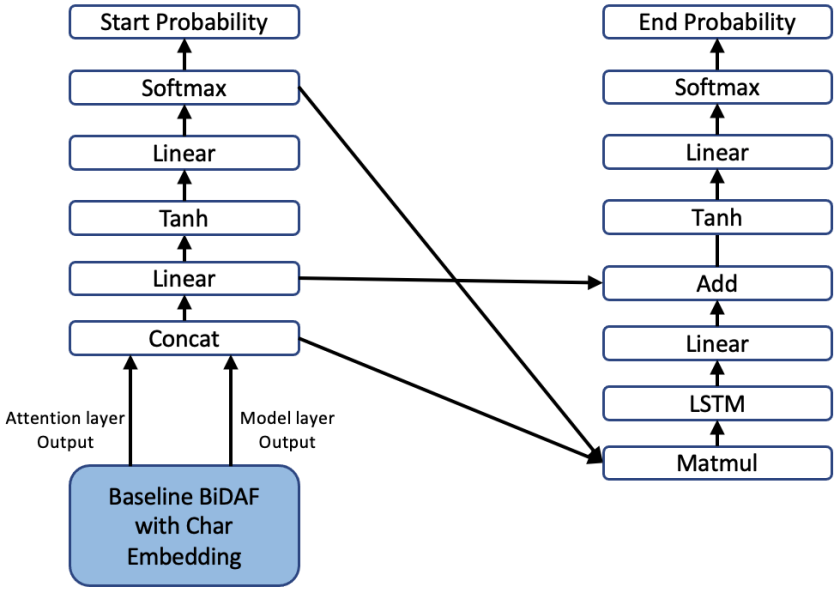

Figure 1: BiDAF with character embedding and answer pointer overall model architecture

4.4 QANet with Character Embedding

The second modification over the baseline is to explore the QANet [1]. The model structure of QANet is shown in figure 2. QANet is very similar to Transformer, and it does not have any RNN models. Its main component is called Encoder Block. The Encoder Block draws inspiration from the Transformer. Both have positional encoding, residual connections, layer normalization, self-attention sublayers, and feed-forward sublayers.

However, the Encoder Block differs from the Transformer in its use of stacked convolutional sublayers, which use depthwise-separable convolution to capture local dependencies in the input sequence. As

shown in figure 2, the QANet is consisted of the following modules: positional encoder, depthwise separable convolutions layer, self-attention layer, and feed-forward layer. The self-attention layer is multi-head attention instead of single attention. Queries, keys and values are linearly projected and then applied with scaled-dot product attention in parallel to get output values. The three result values are further projected and then concatenated to get the final output.

We also use the same embedding layer as BiDAF embedding layer, where we concatenate word embedding vector and character-level embedding vector after the dropout layers. After some parameter tuning, we choose the character embedding hidden size 128 in QANet.

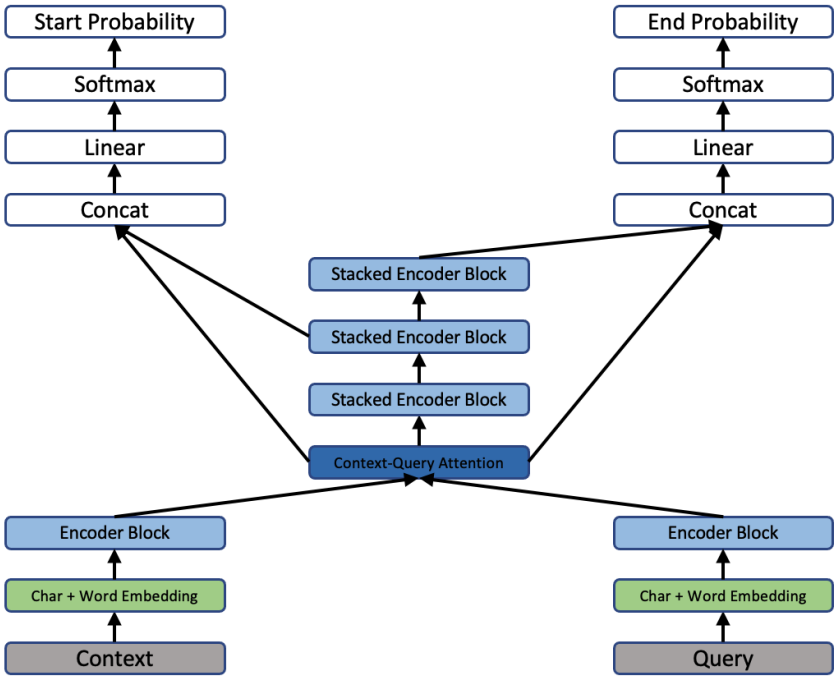

Figure 2: QANet with character embedding overall model architecture

4.5 QANet with Character Embedding and Answer Pointer

As shown in figure 3, we also try to add the answer pointer on top of the QANet. However, from our experiments, we observe that the LSTM structure in the original Answer Pointer [2] does not work well for QANet since one of the main purpose of QANet is to stay away from the RNN structure. Therefore, we made a modifciation to the Answer Pointer and change the LSTM layer to a fully-connected layer. We could also use multi-layer neural network for this part but we find that the overhead becomes very large.

We observe that our QANet with character-level embedding and answer pointer becomes quite large to train. Therefore, we also explore two different version of the model: a lightweight version and a large model.

5 Experiments

5.1 Data

We used the Stanford Question Answering Dataset (SQuAD) 2.0 [6], which combines 100,000 questions in the old dataset, SQuAD. This dataset provides over 150,000 answerable and non-answerable questions comprehension.

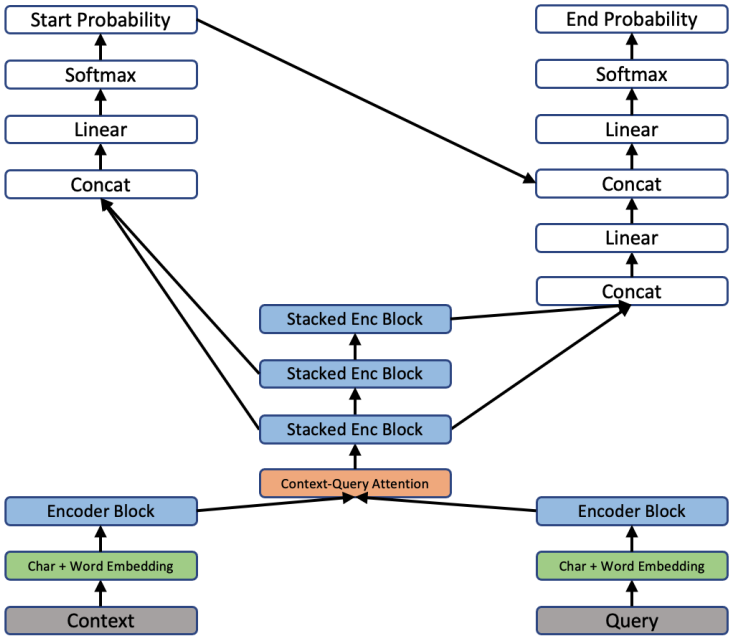

Figure 3: QANet with character embedding and answer pointer overall model architecture

5.2 Evaluation Method

We used the official SQuAD evaluation metrics EM and F1 scores to evaluate the performance of our model. We also used the AvNA (Answer vs. No Answer) which is a useful metric to measure the classification accuracy of our model.

5.3 Experimental Details

For the experiments, we run six different models on the SQuAD 2.0 dataset: the BiDAF baseline model, the BiDAF baseline model with character embedding, and the BiDAF baseline model with character embedding and answer pointer, a light-weight QANet model v1 with character embedding, the same light-weight QANet model v1 with character embedding and answer pointer, and a heavy-weight QANet model v2 with character embedding and answer pointer.

We run 30 epoches for all six models, and the detailed configurations of them are shown in table 1. hid is the number of hidden state, #enc-blk is the number of encoder blocks in each stacked encoder blocks. #head is the number of head in the attention layer of each encoder block. #conv-emb is the number of depthwise-separable convolution layers in the encoder block after char+word embedding. #conv-mod is the number of depthwise-separable convolution layers in each encoder block of the stacked encoder block after the context-query attention layer.

Model	LR	batch	hid	#enc-blk	#head	#conv-emb	#conv-mod
BiDAF Baseline	0.5	64	100	N/A	N/A	N/A	N/A
BiDAF Baseline + Char Emb + AP	0.5	64	100	N/A	N/A	N/A	N/A
BiDAF Baseline + Char Emb + AP	0.5	64	100	N/A	N/A	N/A	N/A
QANet v1 + Char Emb	0.001	32	64	4	4	3	3
QANet v1 + Char Emb + AP	0.001	32	64	4	4	3	3
QANet v2 + Char Emb + AP	0.001	8	128	7	8	4	2

Table 1: Model configurations

5.4 Results

Table 2 shows the performance of our six models. Figure 4 shows the AvNA accuracy, EM score, F1 score, and NLL loss our six models. The best performance we achieve is 68.29 F1 score and 64.98 EM score on the development dataset, and 64.87 F1 score and 61.32 EM score on the test dataset. We achieve the results on the non-PCE leaderboard.

Model	Dev F1	Dev EM
BiDAF Baseline	58.26	55.05
BiDAF Baseline + Char Embedding	61.52	58.21
BiDAF Baseline + Char Embedding + Answer Pointer	62.18	59.13
QANet v1 + Char Embedding	61.69	58.01
QANet v1 + Char Embedding + Answer Pointer	64.35	60.93
QANet v2 + Char Embedding + Answer Pointer	68.29	64.98

Table 2: Model Dev performances

5.4.1 BiDAF

From Table 2, we observe that the performances on F1 and EM metrics improve about 6% after adding the character-level embedding layer to the baseline. The loss curve of the second model also converges faster than the baseline. This is expected since the character embedding enables the model to learn more sophisticated internal structure of words.

As shown table 2, we could observe that third model achieves about 7% improvement for both F1 and EM scores over the baseline, yielding a better performance than the second model. The loss curve of third model converges at a very similar rate as the second model. The result meets our expectation that by introducing the selection of end token depending on the start token, we should reach a higher performance.

Hence for BiDAF model, character embedding significantly improves the model performance, and answer pointer also improves the model performance.

5.4.2 QANet

To compare the effectiveness of QANet versus BiDAF, we take a close look at the second and forth, third and fifth model in table 2. Although each pair of models has distinct number of hidden layers, batch size, and other configurations, each pair takes a similar amount of time to train, so we put them on the same scale to compare. From the result in table 2 we observe that within each pair QANet has a better performance than BiDAF. We confirm our observation from the training loss. Within each pair, the loss curve of QANet model also converges faster than the BiDAF model.

Compare the forth and fifth model, we confirm our conclusion about the effectiveness of Answer Pointer. Our QANet v1 with char embedding achieves 7% improvement for both F1 and EM scores over the baseline, while our QANet v1 with char embedding and answer pointer achieves 10% improvement for both F1 and EM scores over the baseline. The loss curve of fifth model converges faster than the forth model. We conclude that answer pointer is more effective to QANet than BiDAF.

Compare the fifth and sixth model, our QANet v2 with char embedding and answer pointer achieves 17% improvement for both F1 and EM scores over the baseline, resulting the best performance of all models. The loss curve of the sixth model converges the fastest among all models. Because QANet v2 is a more complex model with more hidden layers and more stacked encoder blocks, it is expected that QANet v2 would yield a better result than QANet v1. Hence, the trade-off is between model complexity and model performance.

6 Analysis

6.1 Answer Length

As shown in the figure 5, we perform analysis on the length of the ground truth answer and its relationship with the average length of predicted answer. We observe that the most of the questions in

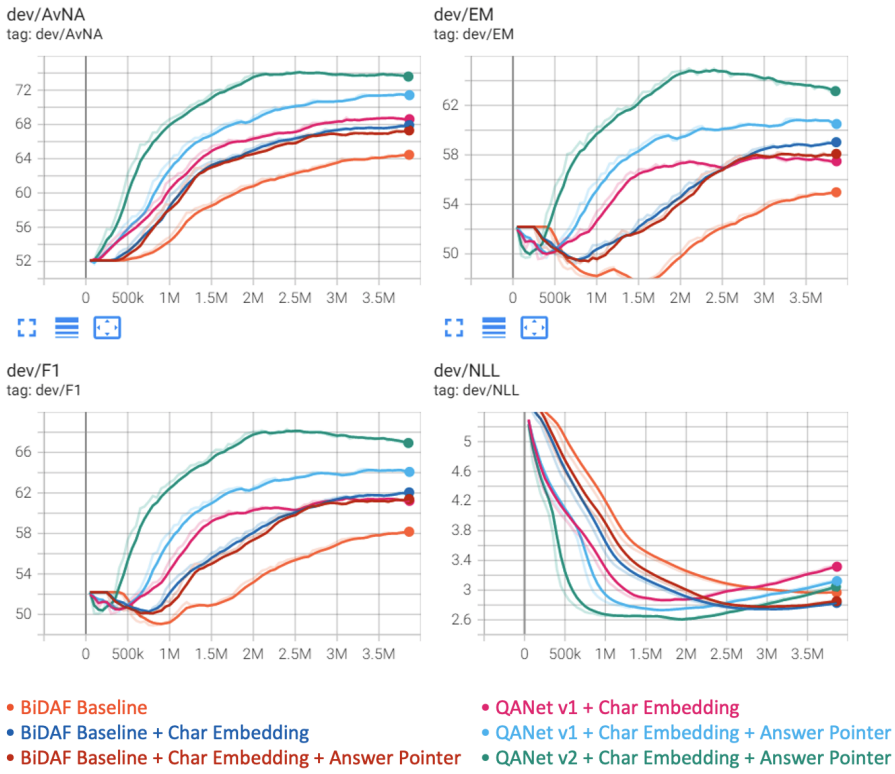

Figure 4: Dev AvNA accuracy, the EM score, the F1 score, and the NLL loss for each model

the dev dataset has ground truth answer with length 1. In general, the frequency drops very quickly as the length of the ground truth answer increases. We could also see that the mean prediction length is in general smaller than that of the ground truth. It is interesting to note that there is no consistent trend of the prediction length as the ground truth length increases. In addition, we could also observe that the mean prediction length is very close to that of the ground truth when the ground truth is short.

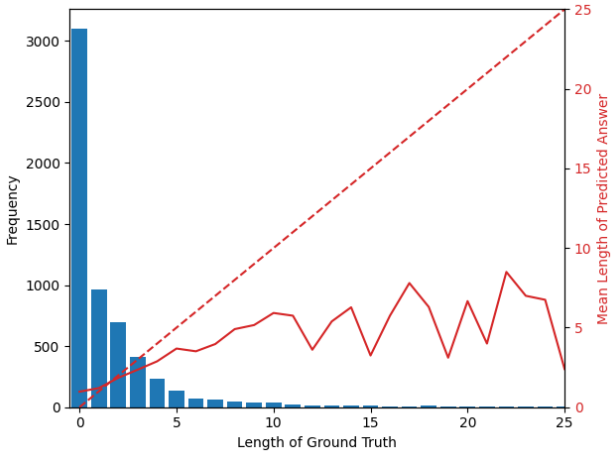

Figure 5: Ground truth length analysis with the QANet v2 model

As shown in figure 6, we also observe a negative relationship between our model performance and answer length. This similar phenomenon is also shown in previous studies [2] [9] [13] which reported diminishing model performance with increasing answer length.

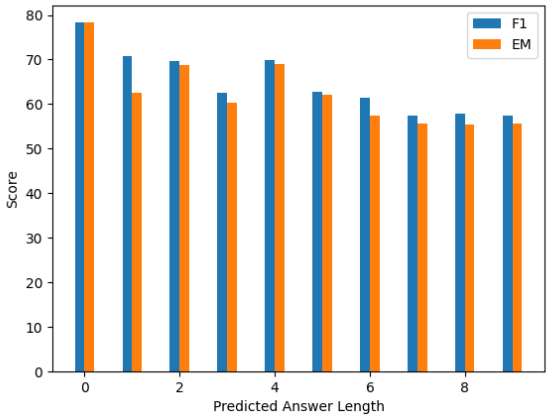

Figure 6: The EM and F1 scores of the the QANet v2 mode with increasing answer length.

6.2 Analysis on Selected Examples

When we looked at the outputs of our model on the dev dataset, we found that the model performs very poorly on the question that contains several similar options that are deceptively wrong. For example, as shown in figure 7, there are several places appeared in the contexts like Fort Caroline and South Carolina which are confusingly close to the ground truth answer Parris Island. One potential solution is to collect more data of this type from either real or artificial text so that the model could distinguish similar words better.

- **Question:** What present-day area was this settlement near?
- **Context:** French Huguenots made two attempts to establish a haven in North America. In 1562, naval officer Jean Ribault led an expedition that explored Florida and the present-day Southeastern U.S., and founded the outpost of Charlesfort on Parris Island, South Carolina. The Wars of Religion precluded a return voyage, and the outpost was abandoned. In 1564, Ribault's former lieutenant René Goulaine de Laudonnière launched a second voyage to build a colony; he established Fort Caroline in what is now Jacksonville, Florida. War at home again precluded a resupply mission, and the colony struggled. In 1565 the Spanish decided to enforce their claim to La Florida, and sent Pedro Menéndez de Avilés, who established the settlement of St. Augustine near Fort Caroline. Menéndez' forces routed the French and executed most of the Protestant captives.
- **Answer:** Parris Island
- **Prediction:** Fort Caroline

Figure 7: A specific question that is very challenging for the model

7 Conclusion and Future work

As a summary, we observe that adding Answer Pointer to BiDAF and QANet improves both the F1 and EM performance. The best performance we achieve is 68.29 F1 score and 64.98 EM score on the development dataset, and 64.87 F1 score and 61.32 EM score on the test dataset. We also observe that larger QANet performs better than smaller QANet. For future work, we would like to explore Transformer-XL [14] for longer-term dependencies since our model is poor at learning words that are very far part. We could also Explore models with low memory use like Reformer [15] since our current model consumes lots of memory.

References

[1] 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.

[2] Shuohang Wang and JingJia Jiang. Machine comprehension using match-lstm and answer pointer. In *International Conference on Learning Representations (ICLR)*, 2017.

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

[4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2018.

[5] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996*, 2016.

[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] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

[8] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

[9] Dian Yu, Kai Sun, Dong Yu, and Claire Cardie. Self-teaching machines to read and comprehend with large-scale multi-subject question-answering data. *arXiv preprint arXiv:2102.01226*, 2021.

[10] Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. Attention-based convolutional neural network for machine comprehension. *arXiv preprint arXiv:1602.04341*, 2016.

[11] Anna Rogers, Matt Gardner, and Isabelle Augenstein. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *arXiv preprint arXiv:2107.12708*, 2021.

[12] 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.

[13] 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.

[14] 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.

[15] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.