Question Answering on SQuAD Dataset

Zihuan Diao
Department of Computer Science
Stanford University
diaozh@stanford.edu

Junjie Dong, Jiaxing Geng
Department of Electrical Engineering
Stanford University
{junjied, jg755}@stanford.edu

Abstract

Machine comprehension (MC), answering a question based on a given context, has gained significant popularity in recent years. In this paper, we present an end-to-end neural network model for the SQuAD dataset. On the hidden test set, our single model achieved F1 score 79.9 and EM 70.7, and an ensemble of 7 models achieved F1 score 81.9 and EM 73.8.

1 Introduction

Machine comprehension has gained significant popularity over recent years due to its wide applications in real life as well as theoretical values for the field of natural language processing. One particular dataset that led to huge advancement in the field is the Stanford Question Answering Dataset (SQuAD) [1], which consists of more than 100,000 questions posed by crowdworkers on Wikipedia articles.

In the SQuAD machine comprehension task, given a context and a question, the machine needs to read and synthesize the context, and then answer the question in natural language based on information presented in the context. The context is a sequence of word tokens $\mathbf{W}^C = [w_1^C, \cdots, w_T^C]$, and the question is $\mathbf{W}^Q = [w_1^Q, \cdots, w_J^Q]$, where T is the length of the context, and J is the length of the question. The SQuAD dataset constrains the answer $\mathbf{A}$ to be a continuous sub-span of the context $\mathbf{C}$; therefore, we can represent the answer using a pair of start and end indices $\{a_s, a_e\}$, where $1 \leq a_s \leq a_e \leq T$.

The original organizers of the SQuAD challenge established a logistic regression baseline that achieved an F1 score of 51.0%, which was about 40% lower than human performance [1]. Since then, researchers have made significant progress applying deep learning approaches, especially variations of recurrent neural networks and attention mechanisms, to close the gap between machine performance and human performance on this dataset. Key innovations in recent models include using bi-directional attention flow to combine information in the context and the question [2], using self-attention mechanism to refine the context representation by matching the context against itself [3], using reinforcement learning and multi-step reasoning techniques to make predictions through multiple passes [4], and adding additional features such as part-of-speech and named entity tags to deep learning models [5].

In this project, we built an end-to-end deep learning model to perform machine comprehension on the SQuAD dataset. Our model combines techniques such as bi-directional attention flow and self-attention, and it is a variation of the model presented in [6]. The rest of this paper is organized as follows: Section 2 introduces our model architecture and decoding algorithm, Section 3 describes the SQuAD dataset and implementation details of the model, Section 4 gives both quantitative and qualitative analysis of the model. Finally, we shed light on future work and conclude the paper in Section 5.

2 Model

In this section, we introduce our model architecture that consists of an embedding layer, encoding layer, bi-directional attention layer, self-matching attention layer, and output layer. Figure 1 gives an overview of the multi-stage model architecture.

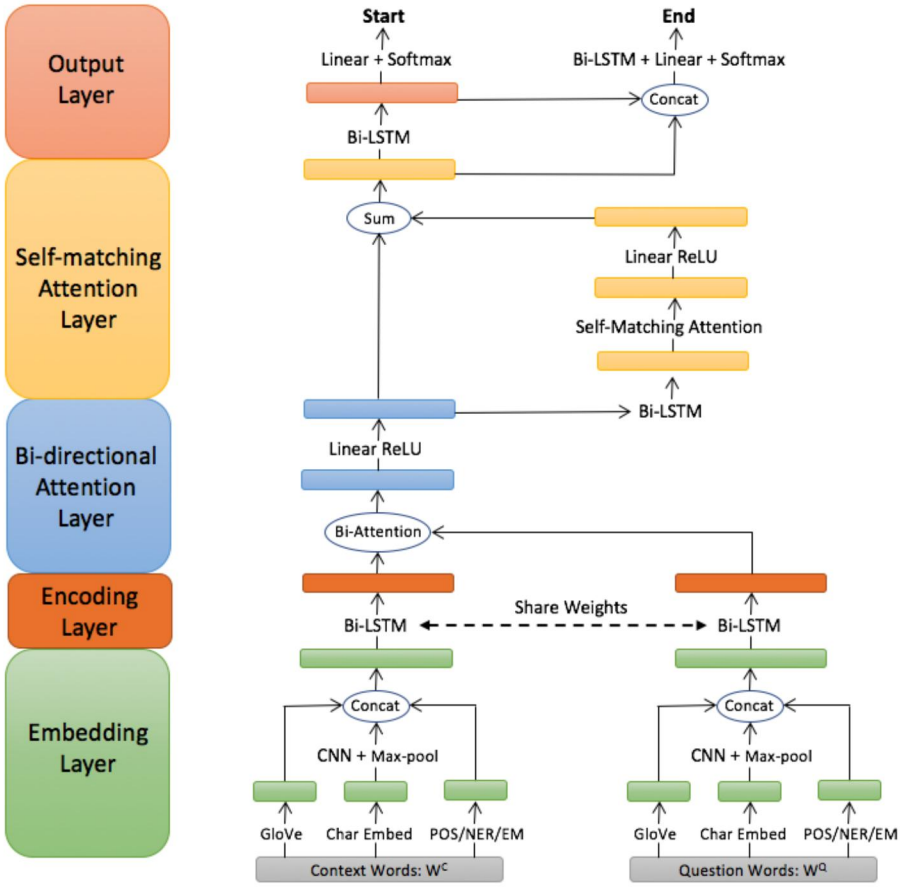

Figure 1: High level network architecture.

2.1 Embedding layer

The embedding layer is responsible for mapping each word token into a dense representation using word-level, character-level, and context-level features. At the word level, we use pretrained 300-dimensional GloVe word vectors to obtain a fixed-size word embedding for each word [7]. At the character level, we obtain embeddings using a 1D convolutional neural network. We use 124 filters of various sizes, and max-pool the outputs over the width of each word to obtain a fixed-size dense vector. The 20-dimensional embeddings of the characters are randomly initialized and learned during training. At the context level, we use pretrained models to extract part-of-speech and named entity features of each word [9]. We encode each feature using a 10-dimensional dense vector that is learned during training. Following [5], we also include a binary exact-match feature for each word to indicate whether a context word is present in the question, and the corresponding position in the question embedding is padded with zero. After the embedding layer, the word tokens $\mathbf{W}^C$ and $\mathbf{W}^Q$ are transformed into fixed-size dense representations $C = [c_1, \cdots, c_T]$ and $Q = [q_1, \cdots, q_J]$, where $c_t, q_j \in \mathbb{R}^{445}$.

2.2 Encoding layer

We use two bi-directional Long Short-Term Memory (LSTM) networks [10] to encode the context and question into new representations $\mathbf{H} = [h_1, \cdots, h_T] \in \mathbb{R}^{T \times 2d}$ and $\mathbf{U} = [u_1, \cdots, u_J] \in \mathbb{R}^{J \times 2d}$.

$$h_t = \text{BiLSTM}_C(h_{t-1}, c_t) \tag{1}$$

$$u_t = \text{BiLSTM}_Q(u_{t-1}, q_t) \tag{2}$$

This layer models the temporal interactions between the words in each piece of text. To enrich the encoder representation and make later similarity-based attention layers more effective, the weights of the two LSTM networks in this layer are shared.

2.3 Bi-directional attention layer

After the two previous layers, all the tokens in the context and question are transformed into high-level dense representations **H** and **U**. In this attention layer, we reference [2] and use a bi-directional attention flow mechanism to fuse the question representation into the context representation. Intuitively, this layer is responsible for understanding the question and generating a question-aware context representation.

Specifically, we compute attentions in two directions: context to question and question to context, both of which are derived from a shared similarity matrix $\mathbf{S} \in \mathbb{R}^{T \times J}$. Each entry S_{tj} represents the similarity between the t-th context word and the j-th question word, and it is computed by:

$$S_{tj} = w_s^T \cdot [h_t; u_j; h_t \circ u_j] \tag{3}$$

where $\circ$ is element-wise multiplication, $[;]$ is vector concatenation, and w_s is a trainable vector.

Context-to-question (C2Q) attention: For each word in the context, the context-to-question attention takes a weighted average of the question word representations based on how relevant to the context word each question word is. The attention weight $a_t \in \mathbb{R}^J$ for the t-th context word is computed by:

$$a_t = \text{softmax}(S_{t:}) \tag{4}$$

And subsequently we can compute the attended question vector by:

$$\tilde{u}_t = \sum_{j=1}^{J} a_{tj} \cdot u_j \tag{5}$$

where $\tilde{U} = [\tilde{u}_1, \cdots \tilde{u}_T] \in \mathbb{R}^{T \times 2d}$ contains the attended question vectors for the entire context.

Question-to-context (Q2C) attention: The question-to-context attention indicates which context words are most relevant to the question. Different from the context-to-question attention, here we only compute one attention vector $\tilde{h} \in \mathbb{R}^{2d}$ for the entire question. We compute the attention weight $b \in \mathbb{R}^T$ by:

$$b = \text{softmax}(\max{}_{col}(S)) \tag{6}$$

where the function $\max_{col}$ computes the maximum for each row across the columns. Then we can compute the attention vector as $\tilde{h} = \sum_t b_t \cdot h_t$. Intuitively, by using the $\max_{col}$ function, we pay attention to a context word as long as it is relevant to some part of the question. And this vector aggregates the most important information in the context for answering the question.

Finally, we combine both attention vectors and pass it through a linear ReLU layer to generate the question-aware context representations $\mathbf{G} = [g_1, \cdots, g_T] \in \mathbb{R}^{T \times 2d}$:

$$g_t = \text{ReLU}\left(W_g \cdot [h_t; \tilde{u_t}; h_t \circ \tilde{u}_t; h_t \circ \tilde{h}] + b_g\right) \tag{7}$$

where W_g is a trainable weight matrix, and b_g is a trainable bias.

2.4 Self-matching attention layer

The question-aware context representation effectively combines the information in the context and the question. However, each word in the context that is represented by g_t has limited knowledge about its surrounding context. Although in theory recurrent LSTM networks can model complex long-term dependencies, its effect is sometimes limited in practice. Inspired by R-NET [3], we match the question-aware context representation $\mathbf{G}$ against itself in this layer. This allows us to directly fuse relevant information from the entire context into each context word.

First we blend $\mathbf{G}$ using a bi-directional LSTM network to generate $\tilde{\mathbf{G}} = [\tilde{g}_1, \cdots, \tilde{g}_T] \in \mathbb{R}^{T \times 2d}$. Then in order to compute the self-matching attention vectors, we first construct a similarity matrix $S' \in \mathbb{R}^{T \times T}$ by:

$$S'_{ij} = w_m^T \cdot [\tilde{g}_i; \tilde{g}_j; \tilde{g}_i \circ \tilde{g}_j] - \infty \cdot \{i = j\} \tag{8}$$

where $w_m \in \mathbb{R}^{6d}$ is a trainable weight vector, and the second term is to force a word to match with words other than itself. We use a different similarity function from the original R-NET implementation because we find it to perform better for our model. Then the attention vectors $\mathbf{M} = [m_1, \cdots m_T] \in \mathbb{R}^{T \times 2d}$ can be computed as:

$$a_t = \text{softmax}(S'_{t:}) \tag{9}$$

$$m_t = \sum_{i=1}^{T} a_{ti} \cdot \tilde{g}_i \tag{10}$$

Finally, we produce the self-attended context representation $\tilde{\mathbf{M}} = [\tilde{m}_1, \cdots \tilde{m}_T] \in \mathbb{R}^{T \times 2d}$ by:

$$\tilde{m}_t = g_t + \text{ReLU}\left(W_m \cdot [m_t; \tilde{g}_t; m_t \circ \tilde{g}_t] + b_m\right) \tag{11}$$

Compared to R-NET which uses a gate structure to control information flow at the end of the self-matching layer [3], we pass not only m_t but also $\tilde{g}_t$ and $m_t \circ \tilde{g}_t$ to the linear ReLU layer. And we also add a residual connection so that the gradients can directly flow to the original question-aware context representation $\mathbf{G}$.

2.5 Output layer

Because the answers in the SQuAD dataset are all continuous sub-spans of the context, we can generate the answer by predicting the start and end positions of the answer in the context. In the output layer, we first blend the self-attended context representation $\tilde{\boldsymbol{M}}$ using a bi-directional LSTM network to produce $\boldsymbol{M_{start}}$. Then the probability distribution for the start position over the entire context is computed using a linear layer and the softmax function:

$$\boldsymbol{p_{start}} = \text{softmax}(\boldsymbol{M_{start}} \cdot w_{start}) \in \mathbb{R}^T \tag{12}$$

For the end position of the answer, we concatenate $\tilde{\boldsymbol{M}}$ and $\boldsymbol{M_{start}}$, and pass it to another bi-directional LSTM to produce $\boldsymbol{M_{end}}$. Then the probability distribution for the end position over the entire context is computed similarly by:

$$\boldsymbol{p_{end}} = \text{softmax}(\boldsymbol{M_{end}} \cdot w_{end}) \in \mathbb{R}^T \tag{13}$$

2.6 Decoding algorithm

Because we make predictions for the start and end positions a_s, a_e of the answer using two separate softmax distributions, simply picking the positions with the highest probabilities can result in invalid answers with $a_s > a_e$. Furthermore, after performing explanatory analysis on the dataset, we found that only 2.8% of the answers have length greater than 14, and the most frequent answer length is 1. Therefore, at test time, we determine a_s and a_e by first filtering out spans with $a_s > a_e$ or $a_e > a_s + 13$, and then taking the positions with the maximum probabilities. This improves the F1 score by more than 1% at test time.

3 Experiments

3.1 Dataset

SQuAD dataset is a machine comprehension dataset on Wikipedia articles with more than 100,000 questions [1]. The dataset is randomly partitioned into a training set (80%), a development set (10%), and a test set (10%). And the test set is not released to the public. In this project, we use two metrics to evaluate our model: F1 score and Exact Match (EM). The human performance on this dataset is F1: 91.221, EM: 82.304.

3.2 Implementation details

We use nltk to tokenize the sentences and extract the part-of-speech and named entity features [9]. All the LSTM networks have a hidden size of 100, and all the linear ReLU layers have an output size of 200. We use the 300-dimensional case-insensitive Common Crawl GloVe word embeddings [7], and do not retrain the embeddings during training. Switching from the Wikipedia+Gigaword GloVe embeddings used in the baseline to the Common Crawl version improved our F1 score on the dev set by more than 0.5%. We believe this is mainly because the Common Crawl version has a much larger vocabulary size, and is pretrained on a larger corpus.

For character-level features, we use random initialized 20-dimensional dense embeddings. According to a recently published paper [11], using ELMo, a deep contextualized character-level word representation greatly improves models' performance on the SQuAD dataset. We tried to use the pretrained ELMo model for this project, but had to later abandon it due to the ELMo model's huge size.

During training, We use Adam optimizer [12] with batch size 100, and an initial learning rate 0.001. We also apply dropout to the LSTM networks and fully connected layers with a dropout rate of 0.2 [13]. We also maintain moving averages of all the trainable parameters with an exponential decay rate of 0.999, and use the moving averages as model parameters at test time. The moving average technique alone gives us about 0.45% improvement on the F1 score on the dev set.

With the configurations described above, training a single model takes about 20 hours on the Azure NV6 machine, and about 10 hours using an Nvidia 1080-Ti GPU.

To further improve the model performance, we built an ensemble model using seven models with the same architecture but different random initialization. At test time, we sum the confidence scores of all candidate answers, and choose the answer with the highest score. This boosts our F1 score by 2.0, and EM by 3.1 on the test set. We also tried performing majority vote and breaking ties based on sum of confidence scores, but the performance on the dev set is not as good as the first method.

4 Results and analysis

In this section, we compare the performance of our model to state-of-the-art models, analyze the errors our model makes, and visualize the attention layer to give more intuition into how the model works.

4.1 Model performance

The performances of our model and several state-of-the-art models are summarized in Table 1. Our single model achieved 79.3 F1, 69.6 EM on the dev set, and 79.9 F1, 70.7 EM on the test set. An ensemble of seven models with different initialization achieved 80.8 F1, 72.4 EM on the dev set, and 81.9 F1, 73.8 EM on the test set. As of February 13, the best single model performance on the official SQuAD leader board is 87.8 F1, 80.9 EM.

4.2 Quantitative error analysis

To further analyze our model beyond the F1 and EM metrics, we break down the model performance by answers of different lengths and questions of different types in Figure 2. As shown in the left figure, the average F1 score decreases dramatically as the length of the correct answer increases.

Table 1: Results on the SQuAD Dataset

	Dev Set	Test Set
Single Model	**F1 / EM**	**F1 / EM**
Logistic Regression Baseline [1]	51.0 / 40.0	51.0 / 40.4
Dynamic Chunk Reader [14]	71.2 / 62.5	71.0 / 62.5
BiDAF [2]	77.3 / 67.7	77.3 / 68.0
R-NET [3]	79.5 / 71.1	79.7 / 71.3
Stochastic Answer Networks [15]	84.1 / 76.2	84.4 / 76.8
QANet (Google Brain & CMU)	- / -	87.8 / 80.9
Our Model	**79.3 / 69.6**	**79.9 / 70.7**
Ensemble Model	**F1 / EM**	**F1 / EM**
BiDAF [2]	80.7 / 72.6	81.1 / 73.3
R-NET [3]	82.8 / 75.6	82.9 / 75.9
Stochastic Answer Networks [15]	85.9 / 78.6	86.5 / 79.6
QANet (Google Brain & CMU)	- / -	89.0 / 82.7
Hybrid AoA Reader (HIT & iFLYTEK Research)	- / -	89.3 / 82.5
Our Model	**80.8 / 72.4**	**81.9 / 73.8**
Human Performance [1]	- / -	91.2 / 82.3

This is hidden by the simple F1/EM metrics because most of the answers are very short, and the overall model performance is not affected much by the questions with very long answers. In the right figure, we divide the questions into different types simply based on the first word of the question. The model performs very well on questions about times and named entities, but gives bad results for questions starting with "why".

Combining our observations on the two plots, we believe the model is good at finding relevant information in the contexts that is useful for answering the questions, but struggles on questions that require deeper logical reasoning and modeling longer-term dependencies, despite having both bi-directional and self-matching attention layers. Another reason is that the model lacks an iterative prediction mechanism that allows for multi-step reasoning [4], and our predictions on the start and end positions for the answer are very much uncorrelated.

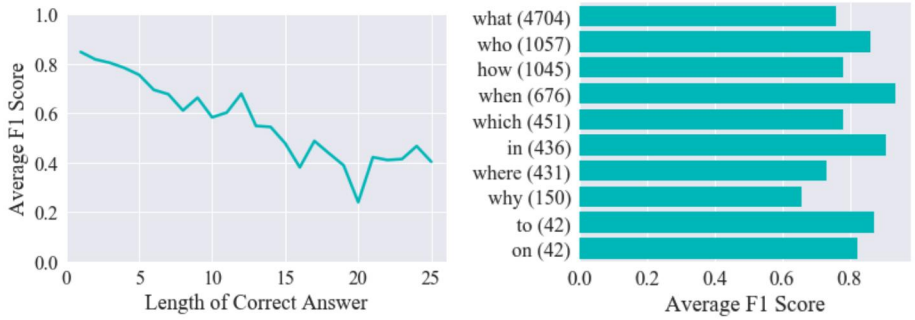

Figure 2: Average F1 score for answers of different lengths and questions of different types.

4.3 Qualitative error analysis

Here we provide two representative examples to show how the model fails to answer some questions.

4.3.1 Irrelevant words in the answer

As in the example shown in Figure 3, our model sometimes produces answers that includes several extra words which are irrelevant to the question. Based on observations on the attention similarity matrix, we found that the model is good at paying attention to relevant context information in these

examples, but our single-pass prediction mechanism that does not condition the end prediction on the start prediction makes it hard to accurately pinpoint an exact answer.

Context: As northwest Europe slowly began to warm up from 22,000 years ago onward, frozen subsoil and expanded alpine glaciers began to thaw and fall-winter snow covers melted in spring. Much of the discharge was routed to the Rhine and its downstream extension.
Question: What did frozen subsoil and expanded alpine glaciers begin to do?
Answer: thaw
Prediction: thaw and fall-winter snow covers melted in spring

Figure 3: Wrong prediction due to imprecise end boundary

4.3.2 Lack of logical reasoning

In this example shown in Figure 4, the keyword "Super Bowl" is not directly mentioned in the context. The model correctly pays attention to both "Seattle Seahawks" and "Arizona Cardinals", but is not able to use logical reasoning to identify which one is the correct answer. For humans with no prior knowledge about football, they can still answer the question correctly by reasoning that the Panthers beat the Seahawks to advance to the "NFC Championship Game", so the answer can only be the Arizona Cardinals. Answering this type of questions requires the model to reason beyond the basic semantic meanings of the sentences.

Context: The Panthers beat the Seattle Seahawks in the divisional round, running up a 31-0 halftime lead and then holding off a furious second half comeback attempt to win 31-24, avenging their elimination from a year earlier. The Panthers then blew out the Arizona Cardinals in the NFC Championship Game, 49-15, racking up 487 yards and forcing seven turnovers.
Question: Who did the Panthers play to advance to the Super Bowl?
Answer: Arizona Cardinals
Prediction: Seattle Seahawks

Figure 4: Wrong prediction due to lack of logical reasoning

4.4 Attention analysis

In order to gain more intuition into how the model works, we extract and visualize the attention similarity matrix of the bi-directional attention flow layer (denoted as S in Section 2.3).

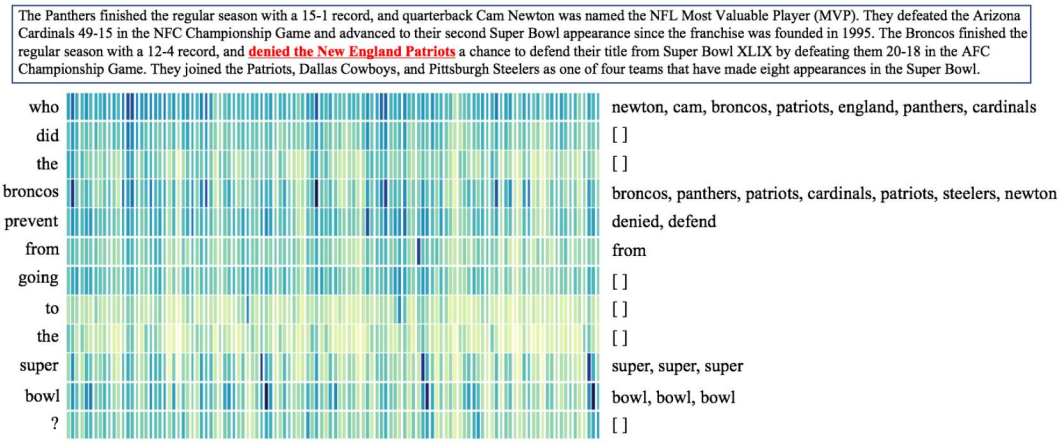

Figure 5: Attention similarity matrix of the bi-directional attention layer.

A larger value in the similarity matrix signifies a stronger mutual attention between a context word and a question word. We further apply a threshold to the matrix, and list out word pairs with attention scores above the given threshold. An example is shown in Figure 5. In this example, "who" matches with several player names and football teams mentioned in the context, and "prevent" matches with words with similar meanings such as "denied" and "defend". Furthermore, wo notice that noun phrases almost always match with themselves, this means a word in the context can effectively localize itself in the question and synthesize its surrounding information, and vice versa. In contrast, stop words such as "the" and "to" are not paid much attention to in most cases.

5 Conclusion

In this project, we implemented an end-to-end neural network model to perform reading comprehension on the SQuAD dataset [1]. Our approach combined several state-of-the-art techniques such as character CNN [8], bi-directional attention flow [2], and self attention [3, 6]. The ensemble model achieved F1 81.9 and EM 73.8 on the hidden test set, and ranked number one on the class leaderboard as of Feb 20. From detailed error analysis and attention analysis, we found that the model can effectively synthesize information in the sentences and pay attention to the most relevant information in the context, but is not good at predicting longer answers and struggles on questions that require deeper logical reasoning.

As of the submission of this report, our ensemble model F1 score is about 7.4% lower than the best model submitted to the public SQuAD leaderboard, and 9.3% lower than the human performance. There are several approaches that we plan to try in order to bridge this performance gap, including using multiple hops memory network to better exploit and reason about the attention layer outputs [4], using methods other than residual connections to control information flow at different stages of the model [16]-[18], using pretrained deep contextualized character-level embeddings such as ELMo if more computational resources are given [11], and running more experiments to tune the hyperparameters.

Acknowledgments

We would like to thank Richard Socher and the CS224N teaching group for providing help and guidance. We would also like to thank Microsoft for sponsoring us with Azure credits to perform experiments on GPU-enabled virtual machines.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. *CoRR*, abs/1606.05250, 2016.

[2] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional Attention Flow for Machine Comprehension. *CoRR*, abs/1611.01603, 2016.

[3] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated Self-Matching Networks for Reading Comprehension and Question Answering. *ACL*, 2017.

[4] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. ReasoNet: Learning to Stop Reading in Machine Comprehension. *CoRR*, abs/1609.05284, 2016.

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to Answer Open-Domain Questions. *CoRR*, abs/1704.00051, 2017.

[6] Christopher Clark and Matt Gardner. Simple and Effective Multi-Paragraph Reading Comprehension. *CoRR*, abs/1710.10723, 2017.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. 2014.

[8] Yoon Kim. Convolutional Neural Networks for Sentence Classification. *CoRR*, abs/1408.5882, 2014.

[9] Bird, Steven, Edward Loper, and Ewan Klein. Natural Language Processing with Python. O'Reilly Media Inc. 2009.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735-1780, 1997.

[11] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv*:1802.05365, 2018.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014) 1929-1958.

[14] Yang Yu, Wei Zhang, Kazi Saidul Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-End Reading Comprehension with Dynamic Answer Chunk Ranking. *CoRR*, abs/1610.09996, 2016

[15] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. Stochastic Answer Networks for Machine Reading Comprehension. *arXiv*:1712.03556, 2017.

[16] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. FusionNet: Fusing via Fully-Aware Attention with Application to Machine Comprehension. *CoRR*, abs/1711.07341, 2017.

[17] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway Networks. *CoRR*, abs/1505.00387, 2015.

[18] Rui Liu, Wei Wei, Weiguang Mao, and Maria Chikina. Phase Conductor on Multi-layered Attentions for Machine Comprehension. *CoRR*, abs/1710.10504, 2017.