Machine Comprehension Using Bidirectional Attention

Behrooz Ghorbani
Department of Electrical Engineering
Stanford University
ghorbani@stanford.edu

Amirhossein Kiani
Department of Biomedical Informatics
Stanford University
akiani@stanford.edu

Abstract

In this paper, we built a machine comprehension model by leveraging the Stanford Question Answering Dataset (SQuAD) [3]. Our model is inspired by the Bidirectional Attention Flow (BiDAF) model as described in [4]. Overall our model was able to achieve EM score of 66.78 and F1 score of 76.52 on the test dataset. We use CNN character embedding, Bidirectional Attention Flow, a stacked bidirectional LSTM layer and dynamic programming to achieve near state of the art accuracy metrics. We additionally built an interactive visualization service which allows for thorough error analysis and real-time interaction with the model.

1 Introduction

Machine Comprehension (MC) is the task in which given a paragraph of context text, a machine is asked to predict an answer to a question related to the context. MC has become significantly popular in recent years due to its applications in building AI assistants as well as its theoretical values for language and neural network research.

Released by the Stanford NLP group, Stanford Question Answering Dataset (SQuAD) [3] provides and invaluable dataset for training and testing of machine comprehension tasks. SQuAD consists of 100K+ questions posed on 500+ Wikipedia articles which are crowd-sourced via Amazon Turk. Each question is answered three times by the crowd workers via selecting the span of text from the context that provides an answer to the posed question. SQuAD additionally provides a public competition leaderboard on stanford-qa.com which at the time of this writing was topped by a team achieving EM score of 82.48 and F1 score of 89.28.

1.1 Problem Definition

Let $c = \{c_1, c_2, ..., c_N\}$ be the sequence of context words of size N and $q = \{q_1, q_2, ..., q_M\}$ the sequence of question words with size M. Our model is tasked to learn a function $f : (q, c) \rightarrow (start, end)$, where $1 \leq start \leq end \leq N$ define the indices for the span of the words in the context that corresponds to the answer of the question.

2 Related Work

Developing effective question answering systems on the SQuAD has attracted considerable attention. In particular, [4, 7, 6] have studied various attention mechanisms to extract and combine the information in the question and the context in an effective manner. Namely, [4] introduces a bi-directional attention flow from the context to the question and from the question to the context. [7] has incorporated "Dynamic Coattention Network" which involves attending over representations that are themselves attention outputs. [1] introduced CNN character embeddings for NLP which can be a great substitute for representing out-of-vocabulary words. [5, 8] introduce new frameworks on how to combine the tasks of predicting the start and end of the answer.

2.1 Our Approach

Inspired by [4], we employ Bidirectional Attention Flow in our model. Compared to [4], we use a less complicated model, with minimal loss of accuracy. We experiment with adding further extensions to the model as described in Section 3 and provide the accuracy implications of such extensions. [1]

2.2 Architecture

The context, c, and the question, q are inputs to the model. After studying the size distribution of the context and questions, we chose $N = 300$ and $M = 30$. Samples that have smaller context or question than the provided parameters N and M, are padded accordingly. Samples with larger question or context size are discarded in training time and truncated in test time.

Embedding Layer: The raw words are converted to vectors using GloVe [2] word embeddings of size 100. Similar to [1, 7] we augmented GloVe word vectors with CNN character embeddings. We used a kernel size of 5 and 100 different filters to learn the character embeddings. This extra enhancement improved our F1 performance by about 1%.

Contextual Layer: The output of the embedding layer is fed to a 1-layer Bidirectional LSTM with hidden size $h = 200$. To encourage weight sharing among the model components, the same LSTM processes both the context and the question vectors. This layer is tasked with modeling the temporal interaction between the words. We used a dropout value of 0.2 on the inputs and outputs of the LSTM to avoid over-fitting. This layer outputs a forward and a backward representation for each word in the input. We concatenated the forward and the backward representations and ended up with a representation of size $2h$ for each word. The final output of this layer is therefore, $R_c \in \mathbb{R}^{N \times 2h}$ and $R_q \in \mathbb{R}^{M \times 2h}$.

Attention Layer: We employ bidirectional attention to combine the information from the question and the context. We form a similarity matrix $S \in \mathbb{R}^{N \times M}$ such that $\forall i \leq N, j \leq M$:

$$S_{i,j} = w_{sim}^T[(R_c)_i; (R_q)_j; (R_c)_i \circ (R_q)_j]$$

where $w_{sim} \in \mathbb{R}^{6h}$ is a similarity word vector and $\circ$ corresponds to element-wise product. For each word i in the context we have the context-to-question distribution of the form

$$\alpha^i = \text{softmax}(S_{i,\cdot}) \in \mathbb{R}^M$$

and the context-to-question attention output of the form

$$a_i = \sum_{j=1}^{M} \alpha_j^i (R_q)_j \in \mathbb{R}^{2h}.$$

We also use question-to-context attention distribution and output of the form

$$m_i = \max_j S_{i,j}, \qquad \beta = \text{softmax}(m) \in \mathbb{R}^N, \qquad c' = \sum_{i=1}^{N} \beta_i c_i \in \mathbb{R}^{2h}.$$

It should be noted that the similarity matrix and the attention distributions are appropriately masked in the padded locations. For each position i in the context, we output $(h_a)_i = [(R_c)_i; a_i; (R_c)_i \circ a_i; (R_c)_i \circ c']$ as the output of the attention layer.

Modeling Layer: The output of the attention layer, h_a, is fed to a two-layer bidirectional LSTM. This layer is tasked with using the attention output to model the probability distribution of the context conditioned on the question. We empirically observe that it is beneficial to increase the dropout value for this layer. In our experiments, we use dropout value of 0.5 on the inputs and outputs of the LSTMs. For each position i in the context, the bidirectional LSTM outputs $\overrightarrow{b}_i$ corresponding to the forward pass and $\overleftarrow{b}_i$ corresponding to the backward pass of the data. $\overrightarrow{b}_i$ and $\overleftarrow{b}_i$ are concatenated and passed as the output of the modeling layer at position i.

Output Layer: We use two separate weight vectors $w_1, w_2 \in \mathbb{R}^{2h}$ for predicting start and end positions. For each $1 \leq i \leq N$, we compute

$$L_i^s = w_1^T[\overrightarrow{b}_i, \overleftarrow{b}_i], \qquad L_i^e = w_2^T[\overrightarrow{b}_i, \overleftarrow{b}_i]$$
$$p^s = \text{softmax}(L^s), \qquad p^e = \text{softmax}(L^e).$$

During prediction time, we use dynamic programing to find the pair $1 \leq i \leq j \leq N$ that maximize $p_i^s p_j^e$ and we declare them as the chosen answer.

[1] After the competition period, our code will be available at https://github.com/akiani/cs224.

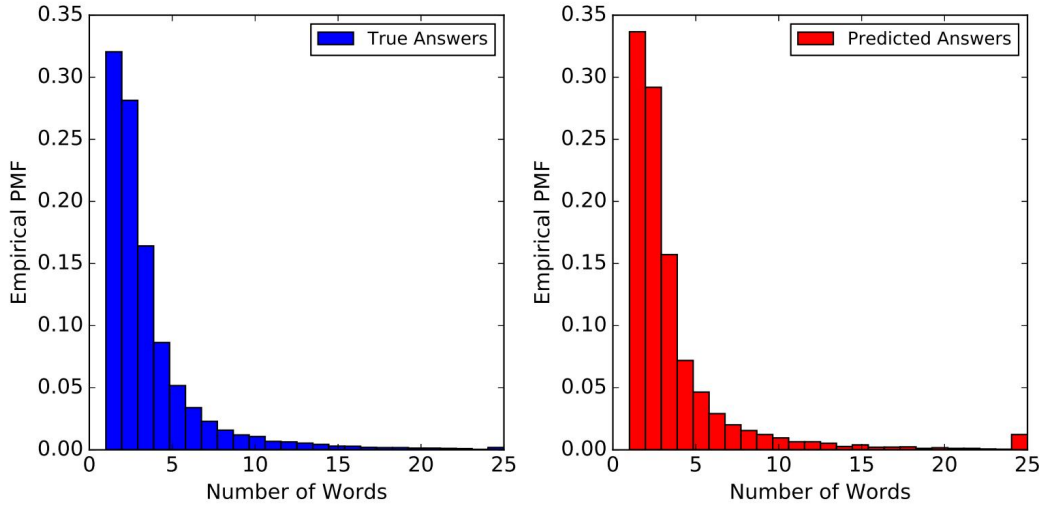

Figure 1: Comparison of the distribution of the length of the predicted answers with the length of the true answers. Note that the model has a tendency to give extremely long answers (more than 25 words).

Our model yields F1 score of 76.52 and an EM score of 66.78 on the test set and F1 score of 76.23 and EM of 66.24 on the development set. Figure 4.1 compares the distribution of the answer lengths of our model with the distribution of the true answer lengths of the development set. The two distributions match closely, however, it is evident that our prediction model has a tendency to generate spurious long answers of length more than 25.

3 Experimentation Process

In this section, we provide an overview of the experiments we ran for the task and their subsequent effect on the accuracy of the model. In particular, we list some of the extensions that were unsuccessful and subsequently removed from the final model.

Inspired by [8], we combined the prediction of the start and the end position together. In this extension, we directly assign probabilities to each admissible answer $(c_i, \cdots, c_j)$, $i \leq j \leq K + i$, where K is the maximum allowed answer length. To do this, in the output layer of the model each valid answer $(c_i, \cdots, c_j)$ is represented by $[\overrightarrow{b}_i; \overleftarrow{b}_j] \in R^{2h}$. These representations are given to logistic regression layer to determine the most probable chunk.

We can see that this approach increases the number of predicted classes by a factor of $O(K)$. In other words, previously we had to predict the start and end pairs from N possible options but now the number of possible choices are $N \times K$. This increase in the number of parameters of the last layer induces both computational and statistical costs. However, since this framework does not assume independence of the start and end positions, we expected it to have more representation power compared to the model predicting the start and the end separately. In our experiments we observed that the cost of representing each chunk separately overweights its benefits. In particular, adding this extension reduces F1 score by nearly 1%. Due to this, we excluded this extension from our final model.
As Figure 3 suggests, our error analysis shows that in the majority of the examples where the model is making a mistake the overlap with the true answer is essentially zero. This suggests that the bottle-neck in our model is fundamental. Changing the output layer as suggested above seems unlikely to be able to solve this issue.

Experiment	F1	EM
Baseline Model	40.08	29.13
Substituting LSTM in the Contextual Layer	41.7	31.02
Bidirectional Attention Addition	46.27	34.07
Modeling Layer Addition	68	53.44
Increased Dropout + Dynamic Programming + CNN Addition	70.39	55.44

Table 1: An overview of the attempted experiments. The listed accuracies are evaluated from the local devset.

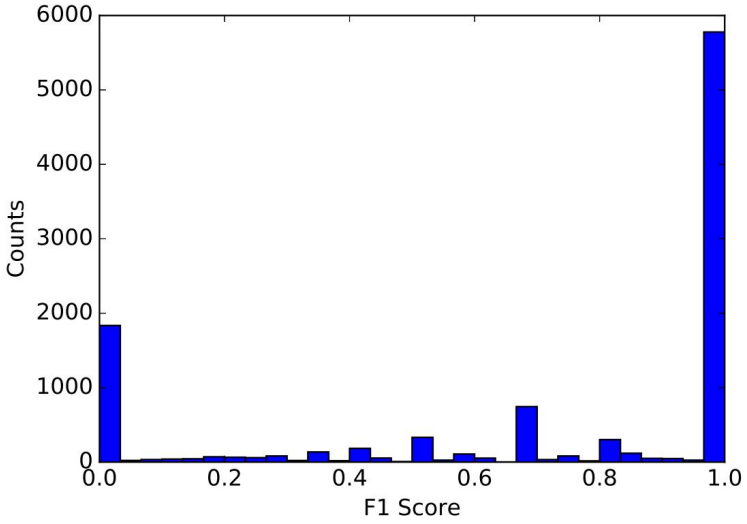

Figure 2: The majority of the performance loss in our model is due to examples that have **zero** F1 score. In other words, the model has not understood the context / question pair. The empirical probability mass function

Following the example of [4], we tried adding a third LSTM layer for determining the end position. We did not observe any significant improvement in doing so. In fact, our current two-layer modeling layer is already prone to over-fitting and we have to use dropout of rate 50% on its inputs and outputs to prevent this.

Furthermore, We did not observe any significant improvements from increasing the number of hidden units (from 200 to 300) or increasing the GloVe embedding dimension (from 100 to 200). We experimented with concatenating the output of the attention layer to the modeling layer output and sending that to the output layer, as suggested in [4]. This extension also showed any significant improvement in our accuracy and therefore was excluded from our final model.

4 Error Analysis

4.1 Statistical Error Analysis

Statistical error analysis of our predictions provides interesting insights into our model. These insights can be used to improve the performance of the model in the future projects.

As shown in Figure 4.1, we observe that our model has lower prediction power for answers that appear further away from the beginning of the context. This phenomenon suggests that the long sequence of words appearing before the answer tends to confuse the model. An interesting future research direction can be to prune the paragraph from the irrelevant information before feeding it to the main model.

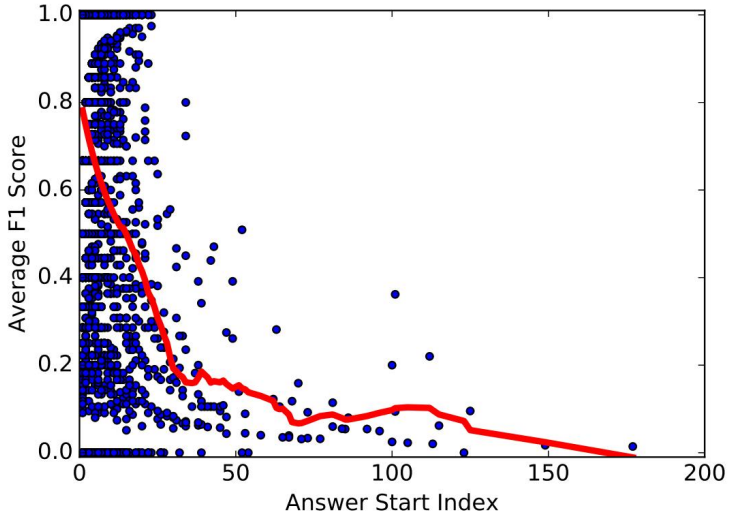

Figure 3: Our model has less predictive power on the answers that appear later in the context. Here each blue dot corresponds to the F1 score on an example. The red curve is smoothed average F1 score.

A second insight we gain from our statistical analysis is that the model tends to do worse when the true answer is longer.

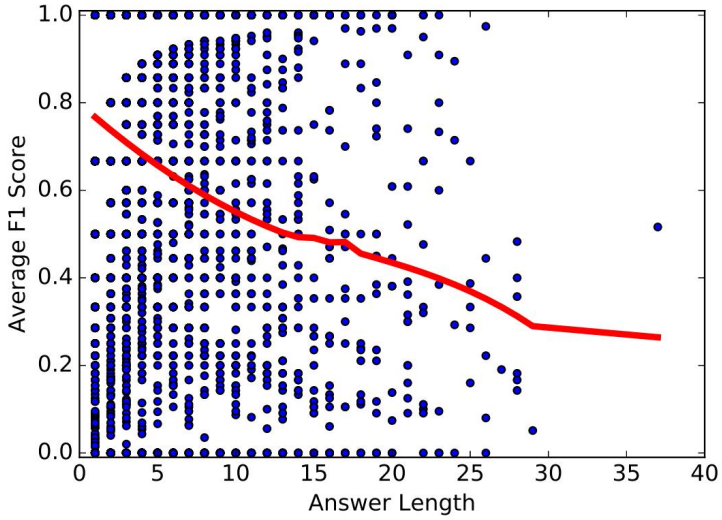

Figure 4: Our model has lower $F1$ score on examples with longer answers. Here each blue dot corresponds to the F1 score on an example. The red curve is smoothed average F1 score.

4.2 Error Examples

We used our interactive visualization engine to validate the findings of the statistical error analysis. Here are two examples in which aforementioned issues surface.

4.2.1 Answer is far away from the beginning and end of the context

As demonstrated in the statistical error analysis section, the model seems to perform poorly when the answer is many words away from the beginning and end of the sentence. This could be a result of vanishing gradients or overall complexity of state propagation in the LSTM layers used in the model. The probability distribution for an example of such case is shown in 4.2.1.

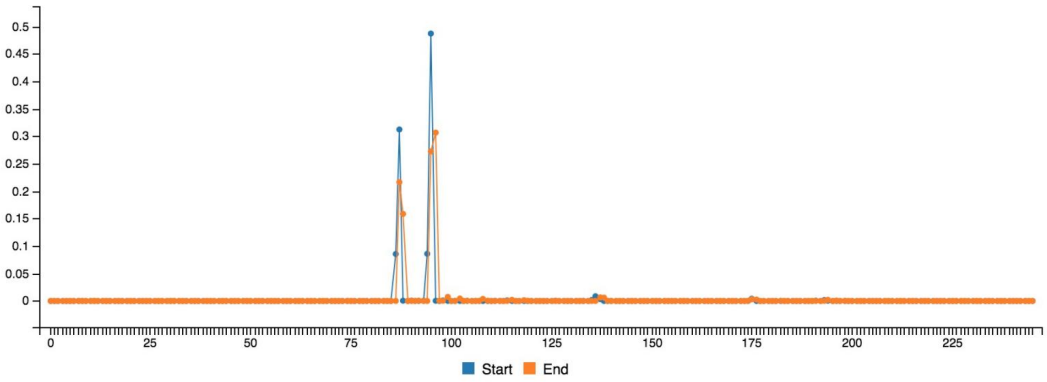

Figure 5: Model performs poorly when the correct answer appears in the middle of a long paragraph. In this case the true answer starts on position 174 but the model has calculated a very low probability distribution for that region.

4.2.2 True answer is many words long

The following is an example in which the correct answer is many words long and the model chooses a shorter answer. This could also be due to the human subjectively selecting a more thorough answer.

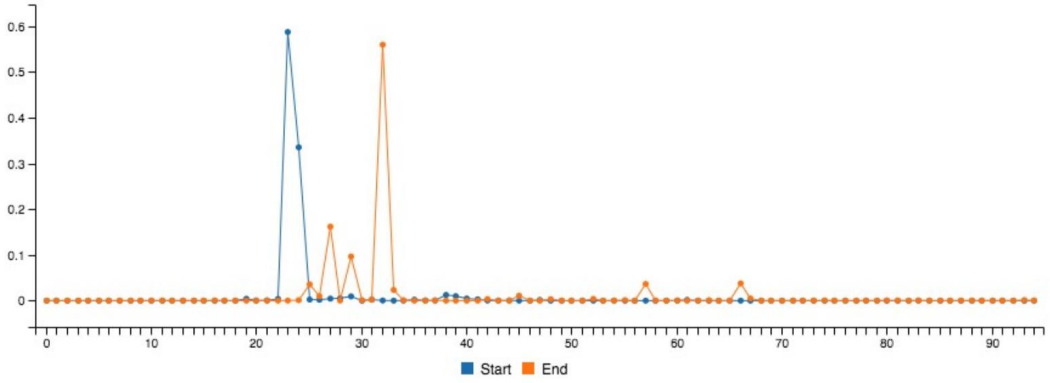

Figure 6: Model prefers small answers whereas the humans prefer more thorough and long answer. The predicted answer is the span of 23 to 32 whereas the human selected answer is the span 23 to 63.

To see an interactive visualization of the error cases please see the Visualization Engine section and our demo video in supplementary materials.

4.3 Visualization Engine

In order to do a through error analysis and provide a way for users to interact with our model, we built an interactive visualization engine on top of the model. The front-end for the engine is written in HTML/JavaScript (D3, jQuery, ...) and the back-end is written using the Flask Microframework.

When the visualization service is launched the model is loaded and is run on the development dataset to generate 20 examples from three categories: (1) Perfect match [F1=1], (2) Almost match [.7=<F1<1] and (3) Poor match [F1<.7]. These examples are then pre-loaded into the user interface for the user to select and analyze. The user interface additionally allows the user to input a custom question and context to be used as model input and predict the answer in real-time.

The visualization engine provides the following *real-time* visualizations for each example:

- Question's attention vector to each word in the context.
- A word cloud of words in the context in which each word is semi-scaled to reflect the attention to each word (with predicted answer highlighted)
- Probability distribution of predicted start and end positions.

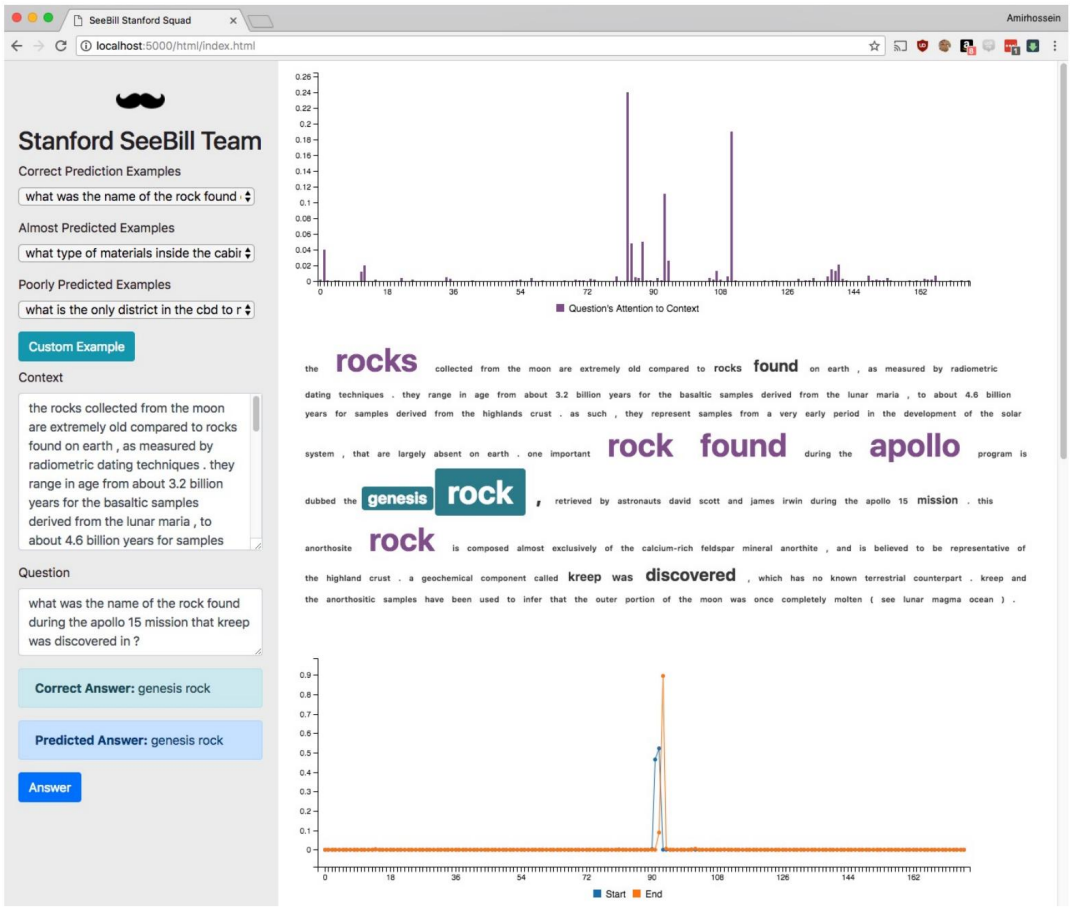

Figure 7: Screen shot of our visualization engine.

- A heat-map of values for each question word's attention towards each word in the context.

5 Conclusion

We used Bidirectional attention flow, CNN character embedding, and stacked bidirectional LSTMs to achieve near state of the art result in SQuAD dataset. We performed detailed error analysis on our model and pointed out a few scenarios in which our model's predictive performance is low. In particular, we identified that in cases where the answer appears in the middle of the context or the answer is lengthy our model under-performs. These observations can be used to come up with new approaches to increase the predictive performance of the model in future. For example, based on our observations, pruning the context from the material unrelated to the question can be a promising direction.

References

[1] KIM, Y., JERNITE, Y., SONTAG, D., AND RUSH, A. M. Character-aware neural language models. *CoRR abs/1508.06615* (2015).

[2] PENNINGTON, J., SOCHER, R., AND MANNING, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

[3] RAJPURKAR, P., ZHANG, J., LOPYREV, K., AND LIANG, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[4] SEO, M., KEMBHAVI, A., FARHADI, A., AND HAJISHIRZI, H. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* (2016).

[5] WANG, S., AND JIANG, J. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* (2016).

[6] WANG, W., YANG, N., WEI, F., CHANG, B., AND ZHOU, M. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2017), vol. 1, pp. 189–198.

[7] XIONG, C., ZHONG, V., AND SOCHER, R. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* (2016).

[8] YU, Y., ZHANG, W., HASAN, K., YU, M., XIANG, B., AND ZHOU, B. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996* (2016).