Investigations in Question Answering Architectures

Patrick Cho
Department of Computer Science
Stanford University

Sudarshan Seshadri
Department of Computer Science
Stanford University

Abstract

This paper implements a deep neural model for the Stanford Question Answering Dataset (SQuAD). We experiment with various types of attention, from basic to more complex models of attention. Our best model reaches an F1 score of 75.882 and an EM score of 65.698 on the test set.

1 Introduction

In a Question Answering (QA) task, a model is given a context and is asked to predict the answer to a corresponding question. Such problems have gained popularity due to their diverse applications as well as their contributions to natural language processing research. In tackling such tasks, models of attention are used to focus on relevant portions of the context. Creating these models of attention requires modeling the interaction between context and question. Deep neural models are employed in such tasks to learn meaningful relationships between contexts and their corresponding questions. To facilitate research in these topics, the Stanford NLP group created The Stanford Question Answering Dataset (SQuAD) [3]. This dataset consists of 100,000+ samples over 500+ articles of contexts, questions, and the corresponding answers. Each answer is a subsequence of words in the context, so the QA task can be simplified to predicting start and end indices of the answer within the context.

We compare various types of attention, including Basic Attention, Bidirectional Attention [4], and Coattention [5]. These attention models are embedded in deep neural network architectures used to perform answer prediction. We primarily model our architecture off of a simplified version of the Bi-Directional Attention Flow (BiDAF) architecture [4].

2 Related Work

2.1 RNN Encoders

RNN encoders can be used to represent a variable length sequence by a fixed-length vector [1]. The encoder reads each symbol of the input sequence x sequentially, changing its hidden state after each symbol. The output of an RNN encoder given x is the concatenation of the output of each symbol in x. These outputs encode information about the temporal interactions between symbols in x. Bidirectional RNN encoders perform this operation in two directions. The output of these two directions are usually then concatenated to form a single output.

2.2 Basic Attention

One problem faced by traditional encoder frameworks is that the encoder can be forced to encode information that is not necessarily relevant to the task at hand. For example, this problem can arise if the input is long or otherwise information-rich. In the case of machine translation, there is a certain alignment between the input and output text. This intuition inspires attention mechanisms that address the above issue by allowing the encoder to refer to specific parts of the input sequence [6] through attention. One basic form of attention for SQuAD models is as follows.

For a context hidden state c_i (where q_j represents question hidden states), the attention distribution is

$$\alpha^i = softmax([c_i^T q_1, \ldots, c_i^T q_M]) \in \mathbb{R}^M$$

We then use this attention distribution to produce the attention output a_i, a weighted sum of the question hidden states.

$$a_i = \sum_{j=1}^{M} \alpha_j^i q_j \in \mathbb{R}^{2h}$$

This is a simple but effective model for attention. Recent work has produced many improvements to this attention model.

2.3 Bidirectional Attention

An improvement over basic attention, Bidirectional Attention (BiDAF) [4] considers attention from context to question, as in Basic Attention, but also question to context. Given the same context and question hidden states c_i and q_j, respectively, we first compute a similarity matrix $S \in \mathbb{R}^{N \times M}$:

$$S_{ij} = w_{sim}^T [c_i; q_j; c_i \circ q_j] \in \mathbb{R}$$

where $w_{sim} \in \mathbb{R}^{6h}$ is a trainable weight vector. For the Context-to-Question (C2Q) attention, we first find the attention distribution α^i:

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

and use this to calculate the C2Q attention outputs a_i.

$$a_i = \sum_{j=1}^{M} \alpha_j^i q_j$$

Next, we perform Question-to-Context (Q2C) Attention. We first calculate

$$m_i = \max_j S_{ij} \in \mathbb{R}$$

and use m to calculate the attention distribution $\beta \in \mathbb{R}^N$ over context locations.

$$\beta = softmax(m) \in \mathbb{R}^N$$

We then calculate the Q2C attention outputs c' as a weighted sum of the context hidden states.

$$c' = \sum_{i=1} N\beta_i c_i \in \mathbb{R}^{2h}$$

Finally, the bidirectional attention output is, in general, some function of the context hidden states, the C2Q attention output, and the Q2C attention output. Empirically, a good output b_i is:

$$b_i = [a_i; c_i \circ a_i; c_i \circ c'] \in \mathbb{R}^{6h} \quad \forall i \in \{1, \ldots, N\}$$

2.4 Coattention

Similarly to BiDAF, Coattention [5] expands on Basic Attention by producing both C2Q and Q2C attention outputs. Given the same context and question hidden states c_i and q_j, we first compute projected question hidden states $q'_1, \ldots, q'_M$ using a non-linear transformation:

$$q'_j = \tanh(Wq_j + b) \in \mathbb{R}^d \quad \forall j \in \{1, \ldots, M\}$$

where W is a trainable weight matrix and b is a trainable bias vector. It is desirable to allow contexts and questions to attend to none of the corresponding question or context hidden states, respectively. Therefore, we add sentinel vectors $c_\emptyset \in \mathbb{R}^{2h}$ and $q'_\emptyset \in \mathbb{R}^{2h}$, which are trainable vectors. Our new context and question hidden states, respectively, are $\{c_1, \ldots, c_N, c_\emptyset\}$ and $\{q'_1, \ldots, q'_M, q'_\emptyset\}$. Now we compute the affinity matrix $L \in \mathbb{R}^{(N+1)\times(M+1)}$ which contains pairwise affinities for context and question hidden states.

$$L_{ij} = c_i^T q'_j \in \mathbb{R}$$

We first use L to compute C2Q attention outputs a_i.

$$\alpha^i = softmax(L_{i,:}) \in \mathbb{R}^{M+1}$$

$$a_i = \sum_{j=1}^{M+1} \alpha_j^i q'_j \in \mathbb{R}^{2h}$$

Next, we use L to compute Q2C attention outputs b_j.

$$\beta^j = softmax(L_{:,j}) \in \mathbb{R}^{N+1}$$

$$b_j = \sum_{i=1}^{N+1} \beta_i^j c_i \in \mathbb{R}^{2h}$$

We then use the C2Q attention distributions to take weighted sums of the Q2C attention outputs to yield second-level attention outputs s_i.

$$s_i = \sum_{j=1}^{M+1} \alpha_j^i b_j \in \mathbb{R}^l \quad \forall i \in \{1, \ldots, N\}$$

Finally, we use the second-level attention outputs and the first-level C2Q attention outputs and feed them through a bidirectional RNN encoder. This gives us hidden states u_i, the coattention encoding, which is used as the overall output of the Coattention layer.

$$\{u_1, \ldots, u_N\} = biRNN(\{[s_1; a_1], \ldots, [s_N; a_N]\})$$

3 Approach

To tackle the SQuAD problem, we use a simple version of the architecture used in the full BiDAF architecture [4]. Our model consists of five layers, as pictured below.

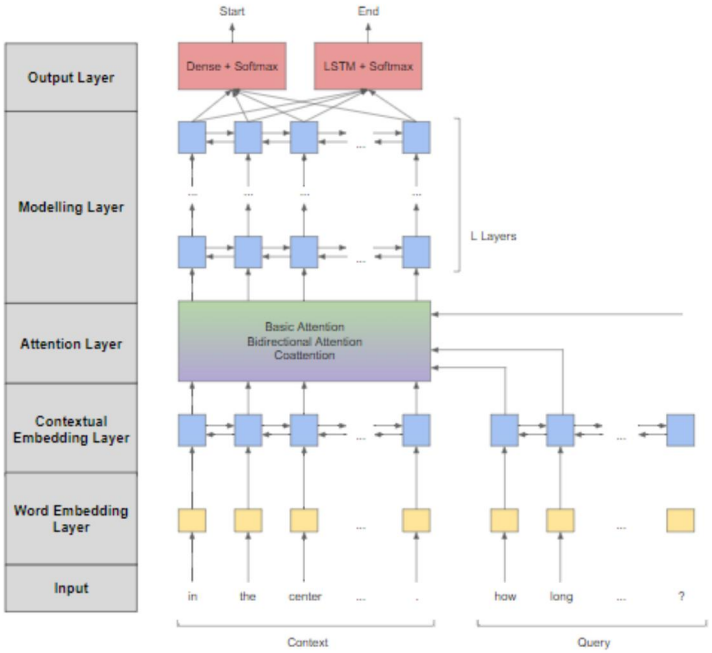


Figure 1: Overall Model architecture

1. **Word Embedding Layer** maps each word to a word vector using pretrained GloVe vectors.

2. **Contextual Embedding Layer** encodes the context and question into embedded hidden states.
3. **Attention Layer** produces a set of question-aware feature vectors for each word in the context. We experiment with Basic Attention, Bidirectional Attention, and Coattention.
4. **Modeling Layer** scans the context with RNN encoders given the attention output.
5. **Output Layer** provides the start and end index distributions to answer the given question.

3.1 Word Embedding Layer.

We have a pretrained word embedding matrix from GloVe, namely, $E \in \mathbb{R}^{|V| \times d}$ where $\mid V \mid$ is the size of the vocabulary and d is the embed vector size. We fix $d = 100$ for our experiments. Given a one-hot vector encoding x of some word w, we produce the embedded vector $e_w = Ex$.

3.2 Contextual Embedding Layer.

Our RNN Encoders are bidirectional encoders built on top of a Long Short-Term Memory Network (LSTM) [2]. We represent the context with a sequence of word embeddings $x_1, \ldots, x_N \in \mathbb{R}^d$, and the question with a sequence of word embeddings $y_1, \ldots, y_M \in \mathbb{R}^d$, obtained from the word embedding layer. We define $N = 600$ and $M = 30$ for all experiments. The RNN encoder is is shared between the contexts and the questions. This is done to enhance the expressive power of the encoder by leveraging shared representational power. Because the RNN Encoder is bidirectional, we have encoded outputs for the forward and the backward directions. The produced encodings are:

$$\{\overrightarrow{c_1}, \overleftarrow{c_1}, \ldots, \overrightarrow{c_N}, \overleftarrow{c_N}\}$$

$$\{\overrightarrow{q_1}, \overleftarrow{q_1}, \ldots, \overrightarrow{q_M}, \overleftarrow{q_M}\}$$

Concatenating the forward and backward directions gives us context hidden states:

$$c_i = [\overrightarrow{c_i}, \overleftarrow{c_i}] \in \mathbb{R}^{2h} \quad \forall i \in \{1 \ldots N\}$$

and question hidden states

$$q_j = [\overrightarrow{q_j}, \overleftarrow{q_j}] \in \mathbb{R}^{2h} \quad \forall j \in \{1 \ldots M\}$$

where we define $h = 200$ for all experiments. Downstream of this layer, only the encoded context and question hidden states are used; the context and question embeddings produced in the previous layer are not used. Therefore, from GloVe we extract features at the word level, and through RNN encoder, we capture higher level features, namely sentence level features.

3.3 Attention Layer.

We can use any of Basic Attention, Bidirectional Attention, and Coattention in this layer, whose equations are described in Related Work. Let the output of the attention layer be $g \in \mathbb{R}^{N \times l}$, where N is the number of context hidden states, and l is the length of the attention output, which varies with the type of attention used. We concatenate the context hidden states and the attention output g to produce $G \in \mathbb{R}^{N \times (l+2h)}$

3.4 Modeling Layer.

The blended representation G produced by the attention layer is passed to the modeling layer. The output of this layer encodes interactions within the context given the question. We use L layers of RNN encoders, implemented again as bi-directional LSTMs. For Basic and Bidirectional Attention, we use $L = 2$, but for Coattention, which contains an RNN encoder within the attention layer, we use $L = 1$. This maintains similar complexity between the different attention models.

The input of the first modeling layer is G, and the output is a matrix $M^1 \in \mathbb{R}^{N \times 2h}$. If there is a second modeling layer, then the input to the second layer is M^1, and the output is a matrix $M^2 \in \mathbb{R}^{N \times 2h}$. We refer the overall output of the Modeling Layer as M, regardless if it is actually M^1 or M^2.

3.5 Output Layer.

The SQuAD task requires finding the start and end index of the answer to a question withing a context. Therefore, there are two principle output distributions: the start index distribution p^1 and the end index distribution p^2. First, we obtain the start index distribution over the context by

$$p^1 = softmax(w_{(p^1)}^T[G; M])$$

where $w_{(p^1)}$ is a trainable weight vector. The end index distribution should be somewhat informed by the start index distribution, as predict an end index lower than a corresponding start index would result in a null answer. Therefore, to pass information from the start index to the end index, we pass $[p^1; M]$ to another RNN encoder layer, again implemented by a bidirectional LSTM. This produces output $M^{end} \in \mathbb{R}^{N \times 2h}$. We then calculate the end index distribution by:

$$p^2 = softmax(w_{(p^2)}^T[G; M^{end}])$$

where $w_{(p^2)}$ is a trainable weight vector.

Training: During training, the objective function is the sum of the cross-entropy loss for p^1 and p^2. Suppose θ is the set of trainable parameters of the overall model, there are T examples in a batch, and y_i^1 and y_i^2 are the true start and end indices, respectively, of the i-th answer. Then the loss function we minimize is:

$$L(\theta) = -\frac{1}{T}\sum_{i=1}^{T} \log(p_{y_i^1}^1) + \log(p_{y_i^2}^2)$$

Evaluation: For evaluation, we produce the start and end index (k, l) that maximizes $p_k^1 p_l^2$ subject to $k \leq l$. This is computed in linear time with the following algorithm.

Algorithm 1 Maximize Span

procedure SPANMAXIMIZATION(p^1, p^2)
    BestEnds ← [1,2,...,N]
    for *start* ∈ $\{N-1, N-2, \ldots, 1\}$ **do**
        if p^2[*BestEnds*[*start* + 1]] > p^2[*start*] **then**
            BestEnds[*start*] ← *BestEnds*[*start* + 1]
    JointProb ← $p^1 \circ p^2$[*BestEnds*]
    $k \leftarrow argmax$(*JointProb*)
    l ← *BestEnds*[k]
    return k, l

4 Experiments

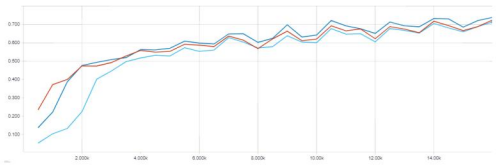
4.1 Dataset

For training our experiments, we use 86k examples for our training set, and 10k examples for our dev set, all taken from SQuAD. The data consists of contexts, questions, and the corresponding answers.

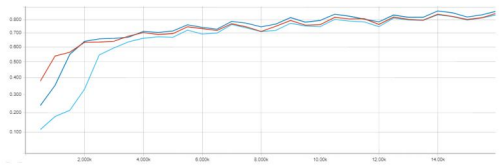
4.2 Model configurations for experiments

For all models, we train with an Adam optimizer with a learning rate of 0.001. All RNN Encoders have a dropout rate of 0.15. As stated in our approach, we use a GloVe embedding size of 100, a maximum context length of 600, a maximum question length of 30, and a hidden size of 200 for all RNN encoders.

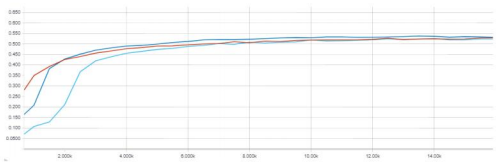
We ran three principle experiments. In the first we use Basic Attention, in the second we use Bidirectional Attention, and in the third we use Coattention. We also conducted hyperparameter tuning on the hyperparameter that affected performance most: number of modeling layers.



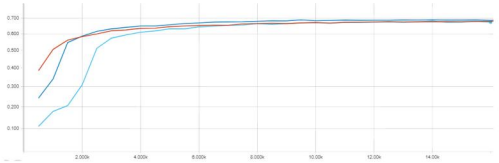
(a) train EM



(b) train F1

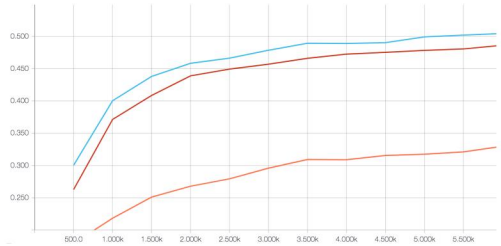


(c) dev EM

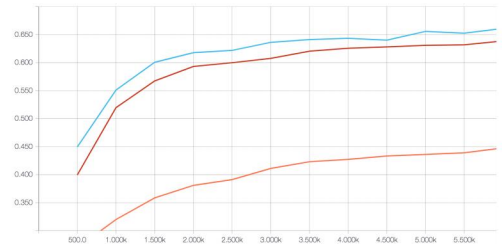


(d) dev F1

Figure 2: EM and F1 scores of three tested models on train and dev sets. All use two modeling layers. Red: Basic; Blue: BiDir; Teal: Coattn.



(a) dev EM



(b) dev F1

Figure 3: EM and F1 scores of three tested models on train and dev sets. All use Bidirectional Attention. Orange: Zero modeling layers; Red: One modeling layer; Teal: Two modeling layers.

4.3 Quantitative Results

We note that all three models of attention attain comparable EM and F1 scores on the dev set. In terms of computation time, basic attention trains the fastest while coattention takes significantly longer to train. The computational time differences are expected as they coincide with the increasing complexity of the models.

While the type of attention had little effect on performance, the number of LSTM layers affected performance by a large margin. Increasing from zero to one modeling layer gave a performance boost of about 0.2 in both EM and F1 scores. Increasing from one to two modeling layers gave a further performance boost of about 0.01. Further additions of modeling layers gave little performance boost and slowed down training significantly. Hence, we settled on two modeling layers in our attention experiments.

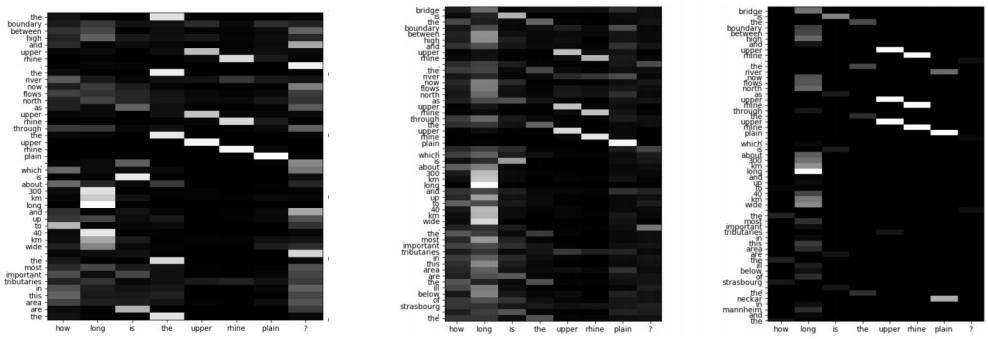
We note that while the types of attention had a marginal effect on performance when there were two modeling layers, the type of attention played a more important role when there were no modeling layers. In particular, with no modeling layers, bidirectional attention achieved a dev set F1 score of 0.47, while basic attention only achieved a dev set F1 score of 0.44. These results suggest that the addition of modeling layers diminishes the effect of more complex attention models.

4.4 Qualitative Results

We visualize some of the attention distributions from each of the three attention models. For each example, the left hand side represents the C2Q distribution, and if present, the right hand side represents the Q2C distribution.

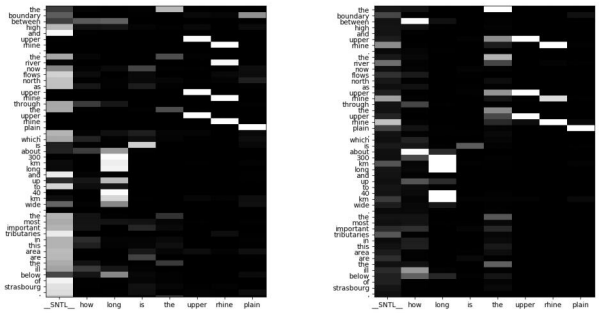
Table 1: EM and F1 scores for dev set official evaluation

Model Name	EM	F1
Basic	63.283	73.762
Bidirectional	**63.661**	**74.108**
Coattn	62.649	73.547



(a) Basic

(b) Bidirectional



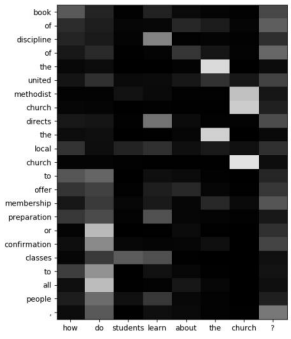
(c) Coattention

Figure 4: Attention distribution for "how long is the upper rhine plain?"

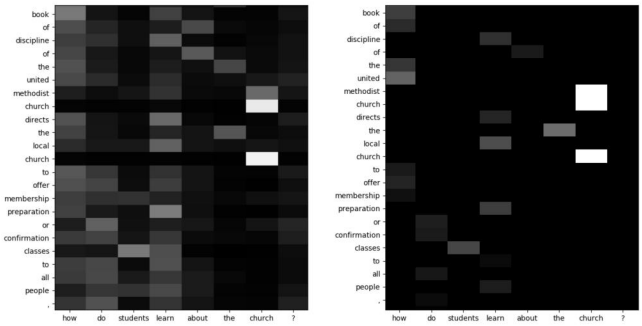
From this example for the question "how long is the upper rhine plain?", we view a portion of the attention distribution that we found insightful. As expected, words in the context strongly attend to the same word in the question and vice-versa. It is also interesting to note that in Coattention, the sentinel question word is the most frequently attended to. Interestingly, notice that in all three attention models, all words in the phrases "300 km long" and "40 km wide" attend to the word "long" in the question. Similarly, "long" in the question attends to the same phrases "300 km long" and "40 km wide" in the context in bidirectional attention and coattention. These phrases are intuitively useful to answer a question involving finding out how "long" something is.

From this example for the question "how do students learn about the church?", we again visualize a portion of the attention distributions. In this example, we note that there are interesting relations between words whose meanings are associated with each other. For example, "classes" in the context attends to and is attended to be "students" in the question. This is natural, as students and classes usually go hand in hand. We also notice some deficiencies in our attention models. None of them draw a strong relationship between "book" in the context and any of "students" or "learn", which would appear to be very natural connections to make.

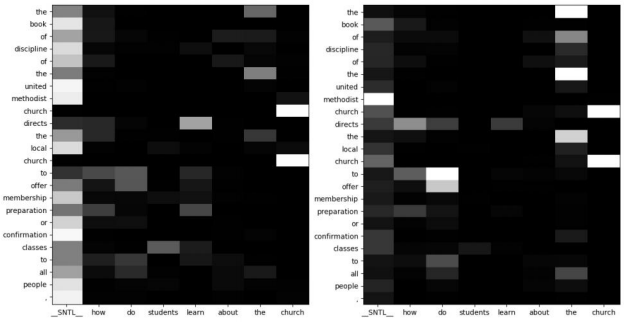
In both examples, we note that our Q2C distributions in bidirectional attention and coattention appear to strongly mimic the C2Q distributions. In particular, they both have the largest values between words that are the same, and share the trends discussed in the above examples. These visualizations may explain why we did not see a dramatic performance change when we swapped out different



(a) Basic



(b) Bidirectional



(c) Coattention

Figure 5: Attention distribution for ”how do students learn about the church?”

types of attention. We suspect the change in the attention distributions may be more stark with a more shallow model. In deep models, perhaps the affects of attention become washed out, as attention is mainly important in informing the model where to focus between context and question. On the other hand, if the model is sufficiently expressive, it may learn such relations even without complex attention.

5 Conclusion

We compared three attention models of varying complexity within a deep neural model for a question answering task. We found that although more complex attention does seem to marginally help our model, the impact of complex attention models such as bidirectional attention and coattention over basic attention is diminished as we deepen our architecture. Attention is important in helping reduce the amount of relevant information that models have to process, thus making more efficient use of downstream layers. However, increasing the number of downstream layers appears to somewhat diminish the importance of complex attention. We do see modest improvements in bidirectional attention over basic attention in terms of F1 and EM scores, but the improvement is not quite as dramatic as we expected. For further work, we would like to shrink the size of each RNN layer as well as the number in order to see if there is a more dramatic disparity between basic attention and the more complex attention models.

6 Acknowledgements

Thank you to the CS224N staff for putting together a great project and for all the help during office hours.

References

[1] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

[5] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[6] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *arXiv preprint arXiv:1708.02709*, 2017.