Variants of Attention for Neural Question Answering

Spenser Anderson
Department of Aeronautics and Astronautics
Stanford University
Stanford, CA 94305
aspenser@stanford.edu

Abstract

The Bidirectional Attention Flow (BiDAF) model (Seo et al., 2016) for neural question answering uses recurrent neural network contextual encoders, a bidirectional (passage-to-question and question-to-passage) attention mechanism, a recurrent modeling layer, and pointer networks to perform SQuAD-style question answering. We augment BiDAF with several modern attention-based neural architectural components (primarily drawn from Wang et al. (2017)), including nonlinear additive attention, self-attention, gated attention, and attention-pooling for conditioning a pointer network. This leads to substantial improvements in the model's performance, indicating the importance of more involved attention mechanisms to modern neural question answering.

1 Introduction

Neural attention mechanisms are becoming increasingly important to modern natural language processing tasks. Attention mechanisms allow a stage of neural computation to refer back to or "attend to" either the input sequence or a computed intermediate sequence, providing shortcut connections to aide with gradient descent, and an inductive bias that has proven powerful for NLP. Typically an attention mechanism computes a convex combination of the sequence being attended to, where the combination coefficients come from some learned similarity function. But there are many ways to implement an attention mechanism, and many ways to integrate attention mechanisms into a neural network model for an NLP task. Many of the first architectures to make use of attention (e.g. Bahdanau et al., 2014) added it on top of a Seq2Seq style architecture, giving the decoder the ability to attend to encoder representations. Since then, attention has been more elaborately integrated into NLP models, culminating in the very high-performing Transformer model (Vaswani et al., 2017) that is composed almost entirely of attention mechanisms. The most recent and high-performing models make heavier and more involved use of attention, leading one to suspect that a model cannot go wrong by adding more attention. This trend raises the question: How important are each of these attention mechanisms?

In this report we explore the impact of various attention mechanism augmentations on the performance of a modern neural question-answering system. Namely, we start with a BiDAF model, and add several additional attention mechanisms from the higher-performing R-Net model, examining the performance impact of each additional component.

BiDAF (Seo et al., 2016) contains many of the basic architectural elements that are common in deep learning NLP systems: word and character embeddings are passed through a bidirectional recurrent neural network (RNN) to create contextual embeddings for the passage and query, which are then passed through an attention mechanism to allow the passage and query embeddings to interact, and the output of this attention layer is passed to a pointer network that selects the span of the passage that should contain the answer. These features make BiDAF a sensible baseline model for many NLP tasks, as it contains most of the components that are considered critical to modern NLP, but not very many of the latest and greatest new components. R-net (Wang et al., 2017) makes more involved use of attention – it's attention mechanism is nonlinear and more heavily parameterized, and it supplements this attention mechanism with a further *self-attention* layer. It even finds a way to include attention in it's pointer network, conditioning the network on an attention-pooling of the query embeddings. Progressively adding R-net components to BiDAF allows us to study the impact of each of these additional components.

2 Related Work

This work directly builds on the BiDAF model proposed by Seo et al. (2016), and draws its major architectural improvements from the R-net model of Wang et al. (2017). One of the earliest papers to use attention was Bahdanau et al. (2014), which proposed an architecture much like the "seq2seq with attention" models that became a dominant paradigm for sequence transduction tasks for some time after its publication.

This work focuses on a study of particular recurrent attention mechanisms. Attention has been highly successful as an architectural element of NLP systems in recent years, even being found to be capable of completely replacing the recurrent networks that are usually central to the design of neural networks for NLP, as in the Transformer of Vaswani et al. (2017). These attention-heavy systems are currently state-of-the-art on a broad array of natural language processing tasks, especially when pre-trained as in BERT (Devlin et al., 2018). But here we instead focus on a careful study of the impact of several attention mechanisms on a more "traditional" RNN-based architecture.

3 Approach

We implement the major components of R-net (Wang et al., 2017) into a baseline model based on the Bidirectional Attention Flow (Seo et al., 2016) neural question-answering model. The baseline model is implemented as described in the cited paper, but without character-based word embeddings implemented. A diagram of this model is shown in Figure 1.

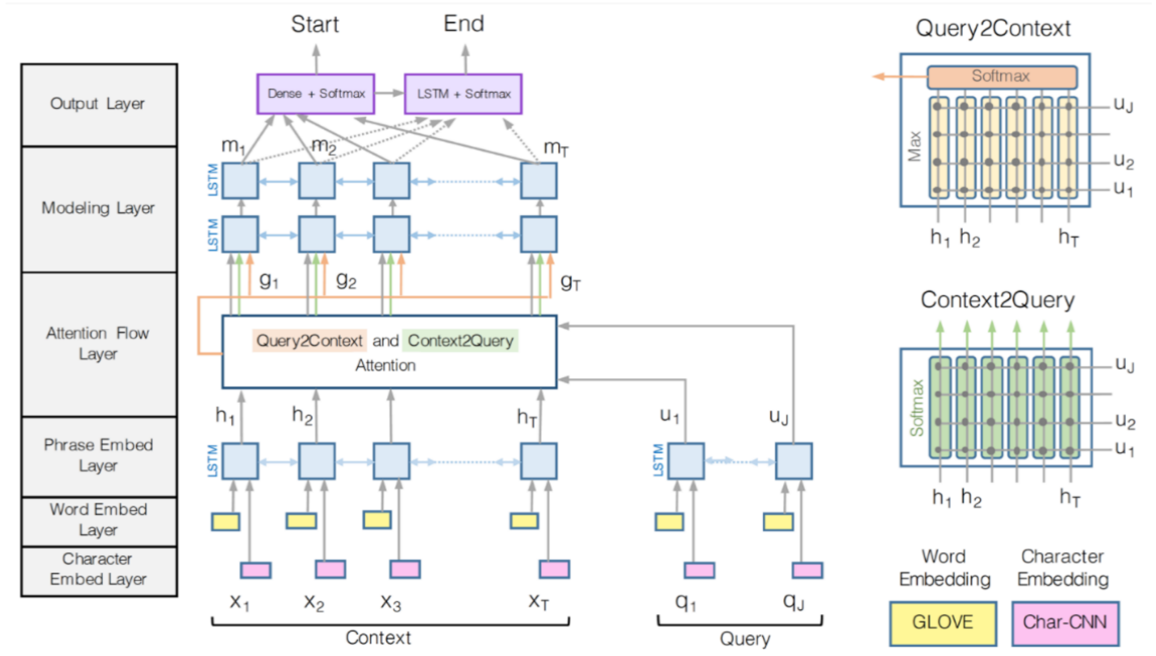

Figure 1: A schematic of the baseline BiDAF architecture.

This model is augmented with several additional architectural elements. A schematic of the final R-net architecture is shown in Figure 2.The model implemented here has the following components (both the baseline and added components are described):

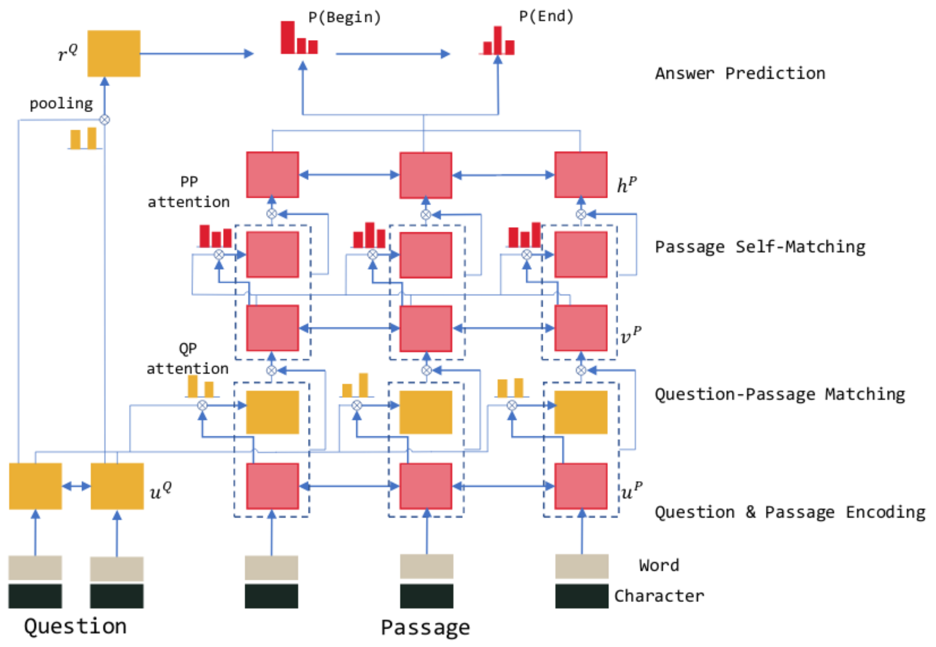

Figure 2: A schematic of the R-net architecture.

- **Word embedding layer:** Word-level embeddings of the passage and query are loaded as pre-trained GloVe embeddings (Pennington et al., 2014), $\{w_t^P\}_{t=1}^n$ and $\{w_t^Q\}_{t=1}^m$. These embeddings are held constant throughout training.

  Character embeddings are trained for each character, and are passed through a single-layer max-pool convolutional network followed by a highway network to produce character-level embeddings for each word, $\{c_t^P\}_{t=1}^n$ and $\{c_t^Q\}_{t=1}^m$. The final word embedding is the concatenation of the GloVe vector and the character-based embedding, $\{e_t^P\}_{t=1}^n = \{[c_t^P; w_t^P]\}_{t=1}^n$ and $\{e_t^Q\}_{t=1}^m = \{[c_t^Q; w_t^Q]\}_{t=1}^m$ This character-based embedding is not done in the baseline, but is added for this project. The character-level embeddings allow the model to learn to reason about word morphology, and also allows some limited meaning to be learned about out-of-vocabulary words.

- **Contextual embedding layer:** Once a vector representation for each word has been constructed, a representation of the meaning of these words in their context is computed. To do this, the sequences of word embedding vectors are passed through a bidirectional recurrent neural network, whose hidden states at each index in the sequence is taken as the contextual embedding vector,

$$h_t^Q = GRU(h_{t-1}^Q, e_t^Q) \tag{1}$$
$$h_t^P = GRU(h_{t-1}^P, e_t^P). \tag{2}$$

  Weights are shared between the two contextual-embedding RNNs, so that these layers can focus on learning contextual meanings in English generally – the later layers are responsible for question-passage interaction effects.

- **Attention layer:** The contextual embeddings are then passed through an attention layer, allowing the model to learn how to relate passage words to query words. Two types of attention are considered: the BiDAF attention model described in Seo et al. (2016), and the R-net attention mechanism used in Wang et al. (2017). The details of BiDAF can be found in the paper, but the primary difference between the two types of attention is that BiDAF-style attention computes a similarity matrix $\mathbf{S}_{ij} = w_{sim}^T[h_i^Q; h_j^P; h_i^Q \circ h_t^P]$, whereas R-Net computes a form of additive attention that is produced by a recurrent neural network layer as follows.

Define $c = att(\mathcal{S}, x)$ to be an attention vector computed by using the vector x to "attend to" the set of vectors $\mathcal{S}$. That is, c will be a convex combination of the elements of $\mathcal{S}$, where the coefficients of the convex combination are determined based on some similarity measure comparing the similarity of each element of $\mathcal{S}$ to x. BiDAF uses the softmax-normalized columns of the similarity matrix $\mathbf{S}$ defined above to compute these coefficients, and R-net does additive attention, computing

$$s_j = v^T \tanh(W_s \mathcal{S}_j + W_x x) \tag{3}$$

$$a_i = \exp(s_i) / \sum_{j=1}^{|\mathcal{S}|} \exp(s_j) \tag{4}$$

$$c = \sum_{i=1}^{|\mathcal{S}|} a_i \mathcal{S}_i. \tag{5}$$

In these expressions, first unnormalized similarity scores s_j are computed through a non-linear similarity function with parameters v, W_s, and W_x. These unnormalized logits are softmax-normalized to give the coefficients of a convex combination, and the attention vector c is the resulting combination.

The output of the attention layer is given by a recurrent layer whose t^{th} input is a vector that results from using the t^{th} passage contextual embedding h_t^P to attend to the entire question. This builds a question-aware repreentation of the passage. Specifically, the attention layer computes

$$v_t^P = GRU(v_{t-1}^P, \text{gate}([h_t^P, c_t^{att}])), \tag{6}$$

where $c_t^{att} = att(h^Q, [h_t^P; v_{t-1}])$, the result of using both the RNN hidden state and the current passage embedding to attend to the question embedding, allowing the RNN to influence the attention mechanism. Note that the attention vector c_t is passed through the RNN after concatenating the passage embedding h_t^P, allowing the passage embedding to pass through this layer directly as well.

The attention layer of R-net also utilizes a gate function on the RNN input. The gate function is computed as $\text{gate}(x) = \text{sigmoid}(W_g x) \circ x$, and allows this attention layer to ignore components of the passage if it so chooses, focusing on the most relevant words.

- **Self-Attention Layer:** R-net follows the passage-query attention layer with a self-attention mechanims. This allows the model to learn relationships between passage words, relating them to help compute the answer span. The self-attention is defined similarly to the passage-question attention,

$$y_t^P = GRU(y_{t-1}^P, \text{gate}([v_t^P; c_t^{self-att}])), \tag{7}$$

  where now $c_t^{self-att} = att(v^P, v_t^P)$, using the current attention vector v_t^P to attend to all other attention vectors v^P. Note that as before, the self-attention is concatenated with the question-passage attention to allow the previous layer to flow directly through this layer, and note that a gate allows this layer to selectively pass information through it.

  BiDAF has no self-attention layer, so this is a pure addition to the baseline model, rather than a change to an existing layer.

- **Modeling layer:** The modeling layer is additional processing on top of the previous layer's outputs, to model the question-aware passage representation and help produce an answer. BiDAF uses such a layer, and the modeling layer is simply another RNN stacked on top of the attention-layer outputs, $m_t^P = GRU(m_{t-1}^P, v_t^P)$.

  R-net does not use this layer with this terminology. However, note that the structure of the self-attention layer is very similar to the structure of BiDAF's modeling layer – it is an RNN that takes in the output of the attention layer – but the R-net self-attention RNN *also* takes as input the self-attention vector $c_t^{self-att}$. So the self-attention layer of R-net is a strictly more expressive version of the modeling layer in BiDAF. In fact, if the RNN in R-net's self-attention layer is a deep RNN, the second layer of the deep RNN could be considered a nearly identical analog of BiDAF's 'modeling layer.'

- **Output layer:** Both models compute outputs by generating a distribution over passage indices (a 'pointer network'). BiDAF takes the output of the modeling RNN (call this sequence m^P), and computes an output sequence o^P by passing these sequences through yet another RNN, $o_t^P = GRU(o_{t-1}^P, m_t^P)$. The distribution over start and end tokens is computed as

$$p_{start} = \text{softmax}(W_{start} \cdot [v_1^P, \cdots, v_n^P; m_1^P, \ldots, m_n^P]) \tag{8}$$

$$p_{end} = \text{softmax}(W_{end} \cdot [v_1^P, \cdots, v_n^P; o_1^P, \ldots, o_n^P]), \tag{9}$$

where recall that v_t^P was the output of the attention layer.

The R-net model uses pointer networks as originally described in Vinyals et al. (2015), which is a slightly different architecture, and it also conditions this pointer network on a pooled vector embedding of the query. Specifically, first a pooled question vector is computed as $r^Q = att(u^Q, V_r^Q)$, where V_r^Q is a learned parameter. Then outputs are computed using a recurrent network over outputs. We compute $c_1 = att(y^P, r^Q)$, and $p_{start}(i)$ is the i^{th} element of the softmax-normalized attention coefficients.

Then $r_2^Q = GRU(r^Q, c_1)$ is computed by a single step of an RNN. We compute $c_2 = att(y^P, r_2^Q)$, and $p_{end}(j)$ is the j^{th} element of the softmax-normalized attention coefficients. The RNN would be reused to produce more pointer distributions if more were necessary, but we only need two positions to define a span, so we stop here.

To predict 'no answer,' an out-of-vocabulary token is appended to the start of the passage sequence, so that the model can learn to point to this token to represent an unanswerable question.

- **Loss function:** The loss function for both models is the sum of the negative log-likelihood for the start and end tokens. If i is the true start token and j is the true end token, then we minimize

$$L = -\log p_{start}(i) - \log p_{end}(j). \tag{10}$$

- **Generating Predictions:** Predictions are generated by finding the span with the largest $p_{start}(i) \cdot p_{end}(j)$ where $i \leq j$ and $j - 1 + 1 \leq L_{max}$. Because 'no answer' is represented by $i = j = 0$, we can predict 'no answer' organically in this framework.

4 Experiments

We conduct experiments using the SQuAD 2.0 question-answering database by Rajpurkar et al. (2018). This database consists of sets of examples that each consist of a passage $\mathcal{P}$, a query $\mathcal{Q}$, and an answer $\mathcal{A}$ that is always a contiguous span located in the passage, if the question is answerable. The question can also be un-answerable, in which case the model is expected to return a special "no answer" token.

Models are evaluated using the F1 score as the primary evaluation metric, though exact-match (EM) scores are also reported for reference. Table 1 reports performance on the course dev and test sets, which are each composed of half of the official SQuAD dev set (the official SQuAD test set is not publicly available, so we do not evaluate against it). Because submissions for evaluation on the test set were limited, only the final model's performance on this dataset is recorded.

All models are trained with a learning rate of 0.5 using the Adadelta optimizer. All models are trained for 30 epochs, which takes 12-15 hours on a cloud NV-6 GPU machine for the baseline model, and as long as 30 hours for the more complicated R-net model. Hidden layer sizes of 100 are used throughout, with 300-element GloVe word vectors being held constant during training. All recurrent layers are bidirectional, and GRU cells are used for all recurrent networks in order to reduced training time (relative to LSTMs). When character embeddings are trained, 20-dimensional character embeddings are trained, and are passed through a character-level convolutional layer to produce 100-dimensional character-based word embeddings.

Performance results of models with various architectural elements are shown in Table 1. The best-performing model has been submitted to the non-PCE dev and test leaderboards under the name "`aspenser`".

Both character-based word embeddings and self-attention led to significant improvements in model performance. This is expected, as both are common elements of modern neural NLP architectures. The other added components either added small performance gains or did not improve performance.

	Dev		Test	
Model	**EM** (Δ)	**F1** (Δ)	**EM**	**F1**
Baseline	55.99 (-)	59.29 (-)	-	-
+ Character embedding	57.23 (+1.24)	61.34 (+2.05)	-	-
+ R-net attention	56.78 (-0.45)	60.81 (-0.53)	-	-
+ Gated Attention	57.12 (+0.34)	61.31 (+0.50)	-	-
+ Self-Attention	59.56 (+2.44)	63.03 (+1.72)	-	-
+ R-net output	**60.11** (+0.55)	**63.62** (+0.59)	59.053	62.638

Table 1: Exact-match and F1 scores of various SQuAD models under study. In parentheses next to each value is the absolute incremental improvement over the model in the previous row.

This is intuitive, as the other components represent smaller tweaks to the model, rather than the addition of an entirely new layer. It should also be noted that only the last row of the table was subjected to any focused hyper-parameter tuning due to the need to implement the final model relatively quickly. Even this final model was completed with time for only 3-4 tuning runs.

With all of the R-net components added into BiDAF, a little over 4 points of improvement are seen in F1 score, consistent with the difference between BiDAF and R-net on the SQuAD 1.1 leaderboard. However, I expected slightly better perfomance considering the competitiveness of the course non-PCE leaderboard. The relatively low performance may be due to the fact that architectures more recent and high-performing than R-net were popular for implementation for this project, and therefore may make up a large portion of the leaderboard.

5 Analysis

The quantative results in the previous section suggest that the character embeddings and the self-attention layer are the most important improvements that R-net has, relative to BiDAF.

The impact of character embeddings can likely be explained by the fact that words that were previously represented by only an "out of vocabulary" (OOV) token are now represented by a more expressive vector that can capture the morphology of the OOV word and match it with other words. In question answering, it is often the case that the question being asked is about a named person or event that is very frequently OOV. For instance, one passage about a Chinese scholar named Tugh has the question "What was the least notable publication of Tugh's academy?" where "Tugh" is an OOV word. The passage is full of OOV words such as "Zhu," "Xi," "neo-Confucianism," and others, so being able to reason about which OOV word is being asked about in the query is likely crucial.

To better understand how the self-attention layer works, consider the visualization of an example attention distribution in Figure 3. This figure shows the self-attention coefficients for a small region surrounding the position of the correct answer (France). Each column is a vector of attention coefficients corresponding to a single element of the self-attention sequence y_t^P. Note that the model is learning to put more attention on the correct answer. What's interesting is that even at words far removed from the correct answer, the model is choosing to heavily represent the vector for the correct answer in the attention combination. This means that many of the inputs to the self-attention RNN will be composed of largely the vector for the answer. It's particularly intriguing that for the timesteps y_t^P corresponding to the position of the correct answer, the answer if represented *less* than at the surrounding positions.

This suggests that the self-attention layer allows the model to emphasize the correct answer in its output representation by making it dominate the representation coming out of the self-attention layer.

6 Conclusion

We see a 4-point improvement in the F1 score of the BiDAF model after implementing all of the R-Net attention augmentations. This is consistent with the level of improvement seen when the R-Net paper was originally released, though the absolute score is substantially lower because this work used the more challenging SQuAD 2.0 database, rather than the original SQuAD database (Rajpurkar et al., 2016) used in the original R-Net paper.

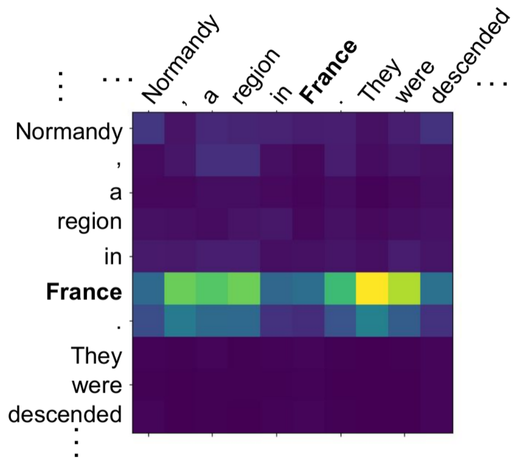

Figure 3: A visualization of the self-attention coefficients for a small region of the sequence surrounding the position of the answer. Note that the answer (France) is receiving much more attention in the self-attention layer.

Experiments show that character embeddings and a self-attention mechanism seem to be the two most important differences between the baseline and the final model configuration. The large impact of character embeddings is likely caused by the dramatic increase in the amount of information that is passed into the model about out-of-vocabulary words. Without a character-based embedding, the model knows nothing about out-of-vocabulary words except for the fact that they are out-of-vocabulary. With character embeddings, morphological similarities can be discerned, and different out-of-vocabulary words in the question and passage can be compared. This is likely especially important because it is often the case that the question is about a named person or event in the passage, which is frequently represented by at least one out-of-vocabulary word.

Self-attention likely causes substantial improvement because it is a more expressive version of the modeling layer that is present in BiDAF. BiDAF uses a bidirectional recurrent network over the passage-to-query and query-to-passage attention layer, whereas R-net similarly uses a bidirectional recurrent network *augmented* with attention. This allows the model to learn how to relate the words in the passage in order to reason toward the answer, and allows this layer to emphasize spans in the passage that are likely to contain the answer.

A major drawback of the attention mechanisms explored in this project is that they all rely heavily on recurrent neural networks. R-net replaces the non-recurrent passage-to-query and query-to-passage attention mechanism in BiDAF with a recurrent version, and also adds a recurrent self-attention layer. Each additional recurrent layer added to the model substantially increases the runtime of a forward pass through model. After adding all of the R-net attention mechanisms to the BiDAF model, it takes over twice as long to train. This makes the model more expensive to deploy, and also limits the amount of hyper-parameter optimization that can be realistically performed. This highlights the significance of recent advances in non-RNN-based attention-heavy architectures such as the Transformer.

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014), "Neural machine translation by jointly learning to align and translate." *CoRR*, abs/1409.0473, URL `http://arxiv.org/abs/1409.0473`.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018), "BERT: pre-training of deep bidirectional transformers for language understanding." *CoRR*, abs/1810.04805, URL `http://arxiv.org/abs/1810.04805`.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014), "Glove: Global vectors for word representation." In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–

1543, URL http://www.aclweb.org/anthology/D14-1162.

Rajpurkar, Pranav, Robin Jia, and Percy Liang (2018), “Know what you don’t know: Unanswerable questions for squad.” *CoRR*, abs/1806.03822, URL http://arxiv.org/abs/1806.03822.

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang (2016), “Squad: 100, 000+ questions for machine comprehension of text.” *CoRR*, abs/1606.05250, URL http://arxiv.org/abs/1606.05250.

Seo, Min Joon, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi (2016), “Bidirectional attention flow for machine comprehension.” *CoRR*, abs/1611.01603, URL http://arxiv.org/abs/1611.01603.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin (2017), “Attention is all you need.” In *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), 5998–6008, Curran Associates, Inc., URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly (2015), “Pointer networks.” In *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), 2692–2700, Curran Associates, Inc., URL http://papers.nips.cc/paper/5866-pointer-networks.pdf.

Wang, Wenhui, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou (2017), “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)*, volume 1, 189–198.