# Ruminating QANet

**Rafael Rafailov,**
rafailov@stanford.edu

## Abstract

The QANet model has shown great results on machine comprehension tasks. Unlike more recent Transformer architectures (BERT) it maintains separate encodings for the context and the query. The interaction between those encodings is carried out by a single attention layer, which has been noted could not be a sufficiently powerful inference procedure. The aim of this work is to combine the QANet architecture with the multi-hop attention structure presented in the Ruminating Reader model. This project has two main contributions: 1) We combine both models and add further extension, via a multi-layer iterative attention mechanism and 2). We propose an entirely new fully-convolutional Ruminating Block structure with dual attention mechanisms, which improves performance by 1 point in both EM and F1 scores. This is a default project, participating in the Non-PCE division.

## 1 Introduction and Related Work

The problem of Machine Comprehension, involves a short paragraph (context) and a question (query), related to it, with the goal to output the answer location within the given text. The BiDAF model Seo et al. (2018) utilizes a bi-directional LSTM network to encode both the context and query input, which are then passed through an attention layer and final modelling layers. The QANet model of Yu et al. (2018) (Figure 1 A))uses ideas from the Transformer model (Vaswani et al. (2017)) to replace recurrent neural networks in the BiDAF framework with an "Encoder Block" (Shown in Figure 1 B))that consist of several layers of local convolutions and a self-attention module that models global dependencies in the text. This architecture has the advantage that when learning the encoding at each word in the paragraph the model has access to all other words, versus using only the the hidden state representations of an RNN network. In addition this model is fully convolutional, avoiding the sequential nature of RNNs and thus speeding-up training. Until the introduction of BERT (Devlin et al. (2018)) QANet was considered state of the art. On the other hand. the Ruminating Reader (Gong and Bowman (2017)) uses ideas from Dhingra et al. (2017), Sordoni et al. (2017) and Wang and Jiang (2016) to extend the basic BiDAF model with a two-hop attention mechanism, which the authors refer to as a "Ruminating Layer". The main idea of the layer is to use a two-hop attention layer to construct query-aware context encodings, thus creating a more complex inference scheme between the query and context representation. At the time of it's publishing the Ruminating Reader matched state of the art. The goal of this work is to extend the QANet model with a **Ruminating Chain**, which stacks ruminating blocks together to from a depth-recurrent mechanism, similar to ResNet. In addition we develop the idea further by introducing a new convolutional-based ruminating block architecture with two attention mechanism that recurrently re-computes query-aware context representations **and** context-aware query representations, which improves performance over the original ruminating chain by 1 point (both in EM and F1 scores). In addition we implement several other extensions of the base code that we have been provided with.

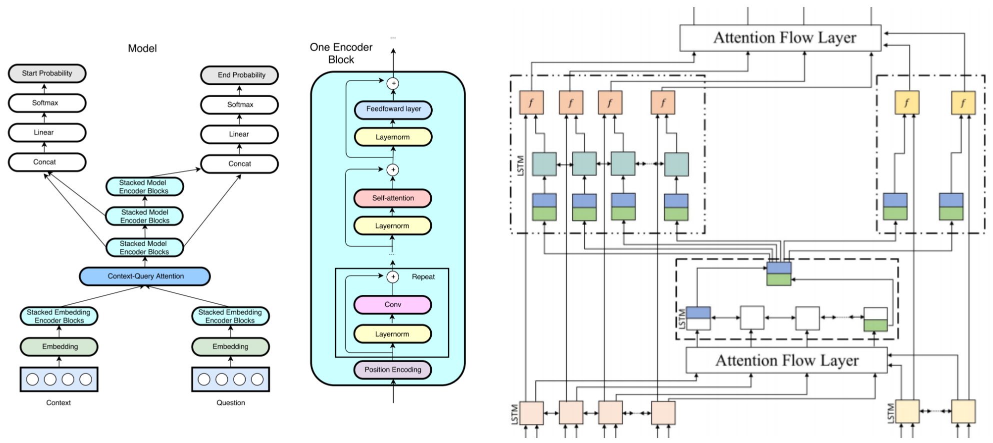

Figure 1: **A**: Original QANet architecture (Yu et al. (2018)), with Encoder Block, **B**: Original Ruminating Block architecture (Gong and Bowman (2017)).

## 2 Models

### 2.1 Joint Word Embedding

The first step of the model is a word-embedding block for both the context and the query. We obtain both pre-trained embeddings for words and characters from the files provided with the project. A character-based word embedding is produced from the character embedding via a standard convolutional model. At this point, both the pre-trained GloVe word embedding and the character based embedding are concatenated and passed through a projection layer, followed by a Highway Network block. This differs from the standard BiDAF approach, which concatenates both the word and character-based embedding, however, in order to keep the size of the model manageable and still maintain relatively high-dimensional word representations I opted for this aggregation approach. This layer outputs query embeddings $\mathbf{q}_{emb,i} \in \mathbb{R}^H$, $i = 1, \dots, M$ and context embeddings $\mathbf{c}_{emb,i} \in \mathbb{R}^H$, $i = 1, \dots, N$, where $H$ is the dimensionality of the hidden representation (in this model, similarly to QANet we use $H = 128$). These vectors are stacked together to obtain $\mathbf{c}_{emb} \in \mathbb{R}^{H \times N}$ and $\mathbf{q}_{emb} \in \mathbb{R}^{H \times M}$.

### 2.2 Encoder Block

The Encoder block architecture (shown in Figure 1. A) follows fairly closely the original paper (Yu et al. (2018)). We experimented with using a trainable positional encoding, i.e. $i \to e_i \in \mathbb{R}^H$, instead of the sinusoidal encoding used in Vaswani et al. (2017) (which was also applied in the QANet model). That modification did not seem to impact performance, but slowed training mildly (consistent with the experiments carried out in Vaswani et al. (2017)), so we opted to maintain the original positional encoding. We pass the output from the embedding layer through an encoder block (both query and context encoder blocks share the same weights) to obtain query encodings $\mathbf{q} \in \mathbb{R}^{H \times M} = \text{EncoderBlock}(\mathbf{q}_{emb})$ and context encodings $\mathbf{c} \in \mathbb{R}^{H \times N} = \text{EncoderBlock}(\mathbf{c}_{emb})$.

### 2.3 Attention and Summarization Architecture: Model 1

I trained two main models. Model 1 is based on QANet with an additional ruminating block, based on Gong and Bowman (2017), with a few small changes, described below.

#### 2.3.1 Attention Layer

The attention layer in our model follows the same general structure as the one provided with the starting code distribution. We experimented with two modifications:

- Computing the similarity score as a generalized quadratic function $\mathbf{S}_{ij} = \mathbf{c}_i^T W \mathbf{q}_j + w_c^T \mathbf{c}_i + w_q^T \mathbf{q}_j$, where $W, w_c$ and $w_q$ are trainable parameters.
- Adding several attention heads (in our implementation between 4 to 8) working in parallel. The final output of the attention layer is an averaging of all attention heads, via a trainable depth-wise convolution.

Using a general quadratic form for computing the similarity score fully nests the basic attention model, however this modification slowed down training significantly, so I reverted back to the original similarity computation. Surprisingly, adding multiple attention heads did not improve performance either (will comment on this in the Results section), so finally we decided to stick with the original formulation. The attention layer returns attention vectors $\mathbf{g}_i \in \mathbb{R}^{4H}$.

The ruminating Block consist of three separate layers: the Sumamrization Layer, the Context Ruminating Layer and the Query Ruminating Layer. Here I will present my implementation, which differs somewhat from the original model. In particular, we apply iterative procedure, similar to a stacking ResNet blocks. We stack $T$ consecutive Ruminating Blocks (which share weights), all variables withing block $t$ are specified with a $t$ superscript.

### 2.3.2 Summarization layer

The sumamrization layer is a bi-directional LSTM module applied to the output of the attention layer. In particular

$$\mathbf{s}_{i,fwd}^t = \text{LSTM}(\mathbf{m}_{i-1}^t, \mathbf{g}_i^t) \in \mathbb{R}^{2H}$$

$$\mathbf{s}_{i,rev}^t = \text{LSTM}(\mathbf{m}_{i+1}^t, \mathbf{g}_i^t) \in \mathbb{R}^{2H}$$

$$\mathbf{s}_i^t = [\mathbf{s}_{i,fwd}^t; \mathbf{s}_{i,rev}^t] \in \mathbb{R}^{4H}$$

Also, similarly to the encoder-decoder NMT structure, we get the LSTM state

$$\mathbf{s}^t = [\mathbf{s}_{len\_c,fwd}^t; \mathbf{s}_{1,rev}^t] \in \mathbb{R}^{4H}$$

by concatenating the first and last hidden states of the module.

### 2.3.3 Query Ruminating Layer

The basic Query Ruminating Layer has the same structure as the original model, with the exception that our model is iterative with a skip connection to the query encoder output $\mathbf{q}$.

$$\begin{aligned}\mathbf{z}_i^t &= \tanh(W_{Q_z}^1 \mathbf{s}^t + W_{Q_z}^2 \mathbf{q}_i + b_{Q_z})\\ \mathbf{f}_i^t &= \sigma(W_{Q_f}^1 \mathbf{s}^t + W_{Q_f}^2 \mathbf{q}_i + b_{Q_f})\\ \mathbf{q}_i^t &= \mathbf{f}_i^t \circ \mathbf{q}_i + (1 - \mathbf{f}_i^t) \circ \mathbf{z}_i^t\end{aligned}$$

Notice that here the block always has access to the query encoding. The process is recurrent as we re-compute the **attention block** (and thus the summary) at each ruminating iteration. Information from the previous iteration flows through the Summarization Layer.

### 2.3.4 Context Ruminating Layer

In the original architecture, the attention summary state vector $\mathbf{s}^t$, is stacked $N$ times and then an LSTM module is applied in order to encode positional information. This seemed very inefficient and somewhat wasteful, so we dispense with that step and use directly the attention summary layer hidden states, which already encode positional information. In more detail, we have the below computation:

$$\mathbf{z}_i^t = \tanh(W_{C_z}^1 \mathbf{s}_i^t + W_{C_z}^2 \mathbf{c}_i + b_{C_z})$$

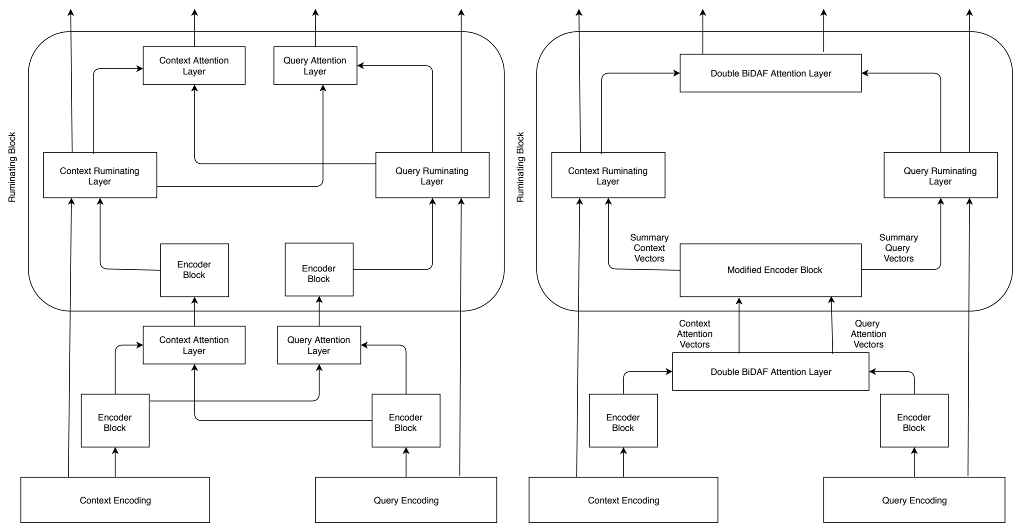

Figure 2: **A**: Chart for Model 2 Ruminating Blocks as implemented and presented here, **B**: Alternative for Model 2 Ruminating BLock. The Double BiDAF attention layers combined both the Query and Context Attention layers, which share weights for the combination of the joint similarity matrix. The modified encoder is a unit that computes positional embeddings and local convolutions on bith inputs seperately, but applies self-attention over both sets of vectors jointly..

$$\mathbf{f}_i^t = \sigma(W_{C_f}^1 \mathbf{s}_i^t + W_{C_f}^2 \mathbf{c}_i + b_{C_f})$$

$$\mathbf{c}_i^t = \mathbf{f}_i^t \circ \mathbf{c}_i + (1 - \mathbf{f}_i^t) \circ \mathbf{z}_i^t$$

This way we obtain iteration-$t$ encodings, $\mathbf{c}^t$, $\mathbf{q}^t$ which we pass again through the attention layer to obtain $\mathbf{g}^{t+1} = \text{Attention}(\mathbf{c}^t, \mathbf{q}^t)$.

### 2.4 Attention and Summarization Architecture: Model 2

Model 2's architecture is presented in Figure 2 A). In the QANet (and Model 1) all RNN units usually used for encoding and modelling have been replaced with convolutional based encoder blocks. Thus the summarization layer is the only recurrent-based unit in the Model 1. The way it's currently used has two major drawbacks:

1. As discussed in Vaswani et al. (2017), there is meaningful information loss when using RNNs for encoding, even in the bi-directional case, while the self-attention mechanism has proved much more robust, due to having access to the entire sequence at encoding time.
2. Sequential computation of RNN based architectures over the summarization layer is slower than a purely convolutional one, which can take advantage of GPUs and other accelerators.

We can address both of these points by replacing the bi-directional LSTM summarizaton layer with an encoding block. We can then process the context ruminating layer in the same way as in Model 1, however we would have to adopt a different procedure for the Query Ruminating block computation, since we no longer have a single attention state vector $\mathbf{s}$. One potential idea is to simply drop that unit in the ruminating block, however ablation studies performed in Gong and Bowman (2017) showed that both the context and query ruminating blocks have meaningful contribution to performance. Our approach is to add an additional attention block. In Model 2, we deploy two context-query attention layers. The first attention layer, which we refer to as the Context Attention is identical to the one used in Model 1 and outputs attention vectors $\mathbf{g}_c^{t+1} = \text{ContextAttention}(\mathbf{c}^t, \mathbf{q}^t)$. The query attention layer is described below.

### 2.4.1 Query Attention Layer

The Query Attention layer is similar to the Context Attention layer. We compute the matrices $\mathbf{S}_q$, $\bar{\mathbf{S}}_q$ and $\bar{\bar{\mathbf{S}}}_q$ in the same fashion as the regular Attention Layer (these matrices depend on the current iteration $t$, but we omit the superscript to keep notation cleaner). These matrices have a subscript $q$, as we use separate trainable similarity parameters from the Context Attention block. However, we exchange the procedures for the Context-to-Question and Question-to-Context attention computation within the attention layer itself. In more detail:

$$\mathbf{a}^t_{q,j} = \sum_{i=1}^{N} \bar{\bar{\mathbf{S}}}_{q,ij} \mathbf{c}^t_i \in \mathbb{R}^H \;\; \forall j \in \{1, \ldots, M\}$$

$$\mathbf{S}'_q = \bar{\bar{\mathbf{S}}}_q^T \bar{\mathbf{S}}_q \in \mathbb{R}^{M \times M}$$

$$\mathbf{b}^t_{q,j} = \sum_{i=1}^{M} \mathbf{S}'_{q,ij} \mathbf{q}^t_j \in \mathbb{R}^H \;\; \forall j \in \{1, \ldots, M\}$$

And finally $\mathbf{g}^{t+1}_{q,j} = [\mathbf{q}^t_j; \mathbf{a}^t_{q,j}; \mathbf{q}^t_j \circ \mathbf{a}^t_{q,j}; \mathbf{q}^t_j \circ \mathbf{b}^t_{q,j}] \in \mathbb{R}^{4H} \;\; \forall j \in \{1, \ldots, M\}$.

In the main implementation in this project the Query Attention and Context Attention are separate layers, however they could also share weights in the computation of the similarity matrix. With the benefit of hindsight, this would have three advantages:

1. We could speed up computation, since we do not have to re-estimate the similarity matrix (which could be expensive) and avoid re-computing the softmax matrices.
2. We could speed up learning as signal backpropagates to the weights in the similarity computation along both branches of the model.
3. Decrease overfitting, by reducing the number of model parameters.

We recommend that this setup is used in further development.

### 2.4.2 Summarization Layer

In Model 2 the summarization layer is another Encoder Block, that operates on the output of both attention layers described in the previous point. We have $\mathbf{s}^t_c = \text{EncoderBlock}(\mathbf{g}^t_c)$ and $\mathbf{s}^t_q = \text{EncoderBlock}(\mathbf{g}^t_q)$. Since the Encoder Block utilizes positional encoding, the resulting summarization vectors already have embedded positional information and we do not need to take further steps in that direction.

An alternative idea is to apply the encoder block to the concatenated tensor $[\mathbf{s}^t_c, \mathbf{s}^t_q] = \text{EncoderBlock}([\mathbf{g}^t_c, \mathbf{g}^t_q])$, which would allow the self-attention mechanism access to both the query and context attention vectors at the same time. We would only need a small modification to apply the positional encoding and convolutional mapping to each tensor separately and concatenate them before the self-attention module is computed. This is one modification that we could test as further development, a schematic for an alternative Model 2, including these modifications is shown in Figure 2 B).

### 2.4.3 Ruminating Layers

The context ruminating layer has the same structure as before, utilizing the summarization vectors $\mathbf{s}^t_c$. The query ruminating layer has the structure:

$$\mathbf{z}^t_i = \tanh(W^1_{Q_z} \mathbf{s}^t_{q,i} + W^2_{Q_z} \mathbf{q}_i + b_{Q_z})$$
$$\mathbf{f}^t_i = \sigma(W^1_{Q_f} \mathbf{s}^t_{q,i} + W^2_{Q_f} \mathbf{q}_i + b_{Q_f})$$
$$\mathbf{q}^t_i = \mathbf{f}^t_i \circ \mathbf{q}_i + (1 - \mathbf{f}^t_i) \circ \mathbf{z}^t_i$$

Using a separate summarization vector at each point in the query allows us to bring in a more information-rich representation than using a single summary vector, which was the case in Model 1.

### 2.5 Modelling block

For both Model 1 and Model 2, the final modelling block is identical to the one presented in the QANet architecture, consisting of 3 stacks with shared weights, each consisting of 7 shallow Encoder Blocks. It operates on the output of the context attention layer in the final ruminating block iteration $\mathbf{g}_c^T$.

## 3 Results and Discussion

We implemented the models presented above in several stages, which serve as a basic ablation study.

1. The first model we trained consisted of the base BiDAF model provided with a 2-step Ruminating Chain. The goal here was the gauge the value added of the ruminating chain. Within 30 epochs the model reached a top dev set EM score of 57.33, F1 score of 60.92 and AvNA of 70.54. This handily outperforms the baseline scores of 55, 58 and 65 respectively.

2. The second model we trained was Model 1, described above. This model achieved significantly better **dev set** results of 63.59 EM and 67.02 F1 scores in 30 epochs. Results on the **test set** were 60.034 EM and 63.695 F1 scores.

3. After implementing Model 1, we experimented with modifying the attention layer with a richer structure. In particular we added a quadratic similarity measure in the bi-directional attention layer, i.e $\mathbf{S}_{ij} = \mathbf{c}_i^T W \mathbf{q}_j + w_c^T \mathbf{c}_i + w_q^T \mathbf{q}_j$, as well as a parallel multi-head mechanism with 4 attention heads. Learning turned out to be quite slow in this case (although consistent) and after 30 epochs, we did not obtain a workable model. This improved slightly, when we initialized the matrix $W$ with a diagonal equal to the vector corresponding to the $\mathbf{c}_i \circ \mathbf{q}_j$ term in the similarity computation, obtained from a trained Model 1, however the model still did not reach good performance in meaningful time. As a next step in the study we reverted back to the original similarity matrix computation ( $\mathbf{S}_{ij} = \mathbf{w}_{sim}^T[\mathbf{c}_i, \mathbf{q}_j, \mathbf{c}_i \circ \mathbf{q}_j]$), maintaining the multi-head mechanism. Surprisingly this slowed down learning somewhat (as compared to using a single-head attention), but did not result in improved performance. We will discuss further below.

4. The final model trained was Model 2, described in Section 2.4. The model achieved top performance on the **dev set** of 63.89 EM and 67.52 F1 scores. The performance on the **test set** was 61.048 EM and 64.63 F1 scores (available on the NON-PCE Leaderboard under RMR), getting an improvement of 1 point on both the EM and F1 scores over Model 1.

The most surprising observation from the above results, was that adding a more expressive similarity measure and multiple attention heads, did not improve performance and slowed down learning. We believe that this may be due to the structure of the ruminating chain. This block is essentially a recurrent neural network, with hidden states $\mathbf{g}^t$ and $\{\mathbf{g}_c^t, \mathbf{g}_q^t\}$ for Model 1 and 2 respectively, and inputs $\mathbf{q}$ and $\mathbf{c}$ at each iteration. In fact the context and query ruminating layers serve similar function to the gating mechanisms of an LSTM. We could also interpret the attention output as a hidden state for the ruminating block and the attention summarization layer as a hidden state mapping. Recurrent Neural networks have been proven to be Turing Complete (for a survey and recent results Pérez et al. (2019)). It seems plausible that by including a deep enough ruminating chain, we could learn any algorithm for computing attention vectors, even with a relatively simple internal representation. This would explain why adding more general similarity mechanisms and multi-headed attention did not result in improved performance. The optimal attention mechanism within the given class, could already be reachable with the simpler formulation, via the ruminating chain recurrence. In Pérez et al. (2019) it is also proved that the Transformer model of Vaswani et al. (2017) is Turing complete (our Encoder Block is based on the same self-attention mechanism), which could also be a reason for the improved performance of Model 2 over Model 1, however any interaction between two such systems seems likely to be quite complex and at this point I have no definitive proof of this hypothesis.

## 4 Conclusion and Further Steps

In this project we extended the Ruminating Reader model into a full recurrent Ruminating Chain. We showed that this modification results in increased performance over the basic BiDAF model. We also combined the QANet model with a ruminating chain and showed additional improved performance. We proposed an entirely new dual attention, fully-convolutional Ruminating Block, based on self-attention. The proposed model resulted in better performance over our original model by 1 point in EM and F1 scores. There are multiple venues for analysis and further work here.

- One of our surprising discoveries was that adding more complex attention mechanisms did not improve performance and we stipulated that this was due to the recurrent structure of the ruminating chain. One avenue for further development would be to test this hypothesis more extensively. This would include varying the length of the ruminating chain with different attention and summarization mechanisms.
- We proposed an alternative form for Model 2 with shared weights in both attention layers and a summarization procedure, via a single pass through an encoder block, with self-attention over both sets of attention vectors. We plan to train this model as well and compare it to the original form of Model 2.
- If our hypothesis about the expressiveness of the ruminating chain is correct, it seems there would be little room for further improvement coming from more elaborate attention layers. Within the confines of the non-PCE framework, that means we could explore more expressive encodings, for example stacking multiple encoder blocks for the query and context, or perhaps applying the architecture proposed in the alternative Model 2 to the word encodings as well, i.e. $[\mathbf{c}, \mathbf{q}] = \text{EncoderBlock}([\mathbf{c}_{emb}, \mathbf{q}_{emb}])$, similarly to the proposed BERT architecture for question answering.
- Many of the models trained showed continued improvement well into the training procedure. At least one showed improved performance with additional training and learning-rate fine tuning. I believe that we could get additional improvement from both Model 1 and Model 2 with some fine tuning.
- I explored the output of both Model 1 and Model 2, however it was difficult to find any particular common characteristic of questions that were answered wrong. Perhaps with additional time and investigation we could discover potential issues and make amendments to the model.

## 5 Acknowledgements

I am grateful to Professor Manning and the entire course staff for their instruction, guidance, patient help and feedback. The assignments were really instructive and the default final project was very well done and presented a real opportunity to develop my skills further, for which I am grateful. I would also like to thank for the generous amount of Azure credits provided and making sure my models are never cut short in training. Thank you and all the best.

## References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv preprint*. `http://arxiv.org/abs/1810.04805`.

Dhingra, B., Liu, H., Cohen, W. W., and Salakhutdinov, R. (2017). Gated-attention readers for text comprehension. *ArXiv preprint*. `https://arxiv.org/abs/1606.01549`.

Gong, Y. and Bowman, S. R. (2017). Ruminating reader: Reasoning with gated multi-hop attention. *ArXiv preprint*. `http://arxiv.org/1704.07415`.

Pérez, J., Marinković, J., and Barceló, P. (2019). On the turing completeness of modern neural network architectures. *ICLR 2019, ArXiv preprint*. `https://arxiv.org/abs/1901.03429`.

Seo, M. J., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2018). Bidirectional attention flow for machine comprehension. *Arxiv preprint*. `https://arxiv.org/abs/1611.01603`.

Sordoni, A., Bachman, P., and Bengio, Y. (2017). Iterative alternating neural attention for machine reading. *ArXiv preprint*. `http://arXiv:1606.02245`.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Łukasz Kaiser, and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, page 5998–6008. `https://arxiv.org/abs/1706.03762`.

Wang, S. and Jiang, J. (2016). Machine comprehension using match-lstm and answer pointer. *Arxiv preprint*. `https://arxiv.org/abs/1804.09541`.

Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., and Le, Q. V. (2018). Qanet: Combining local convolution with global self-attention for reading comprehension. *Arxiv preprint*. `https://arxiv.org/abs/1608.07905`.

## 6 Appendix: Development and Contributions

1. I wrote the code for the joint word embedding, although it was based on the one we did for Assignment 5.
2. I adapted the Encoder Block from two public github repos and the original model tensorflow repo. I would estimate I modified about 20% of the code. For example, I implemented a trainable Positional Encoder and the modified encoder block described in the alternative Model 2.
3. I modified the bi-directional attention model we were provided with to include a general quadratic similarity function and multiple parallel heads, as well as the final combination layer.
4. Model 2 is entirely an original idea and I implemented the Query Attention Layer myself, using the provided Attention module as a template.
5. I wrote the whole Ruminating Block from scratch for both Model 1 and Model 2.