Analyzing Modeling Layers for the SQuAD Challenge

Timothy Anderson
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
timothy.anderson@stanford.edu

Abstract

This project analyzes the effect of adding language modeling layers in neural networks used for the SQuAD challenge. We test several different architectures with zero, one, and two modeling layers in conjunction with three different attention mechanisms: dot-product attention, quadratic attention (a simple variant of dot-product attention), and bidirectional attention flow (BiDAF). The results show that a single additional language modeling layer has the biggest effect on the accuracy of the model. While more complex attention mechanisms do have an effect, the effect for adding modeling layers is much more dramatic. Results also show that quadratic attention exhibits worse performance than dot-product across all model types. These results suggest that future work should focus on replicating the gains from a single modeling layer, such as by adding successive attention layers or residual connections.

1 Introduction

Question answering is one of the fundamental tasks in machine comprehension—and natural language processing in general—because of its linguistic and computational challenges, as well as practical interest. In its most basic form, the question answering problem is: for a question sequence $\{q_j\}_{j=1}^{M}$ and context sequence $\{c_i\}_{i=1}^{N}$, we want to identify the start index i_{start} and end index i_{end} of the context sequence that answers the question.

Such tasks pose two main challenges: learning long-term dependencies, and building correlations between representations of the question and context passages. For the former, LSTM [1] and similar recurrent neural network architectures have largely overcome this issue, and the recent popularity of bidirectional recurrent architectures has largely reduced long-term dependencies as a source of trouble. The latter task, however, still poses significant challenges and has been addressed mainly through attention mechanisms.

At their core, attention layers take as input key-value pairs, and generate outputs based on the similarity between the keys and values. Basic attention mechanisms will output for each key a weighted linear combination of the keys and/or values, where the weights are generated based on some type of similarity metric (e.g. dot-product). Attention mechanisms have shown stunning success in a variety of language and non-language tasks, and in particular have been critical to building robust question answering systems.

A large number of robust architectures have been presented for question answering tasks, specifically for the Stanford Question Answer Dataset (SQuAD) [2]. A common thread between these is the use of *language modeling layers*, (bidirectional) recurrent layers following the attention layer that is used to build a more accurate latent representation of the context sequence. Here, we analyze the effect of language modeling layers to gain insight on how these impact model performance. In addition to varying the number of modeling layers, we also test using several different attention mechanisms to separate the effects of attention from the that of modeling layers.

In what follows, we present a description of the architectures and attention mechanisms tested, followed by results comparing these models with existing SQuAD models.

2 Setup

The model architecture used here most closely resembles that in [3]. In [3], the model used LSTM cells for all RNN layers, and included a character-level CNN as input to the initial recurrent embedding layer for the context and question. To provide contrast with these choices, we 1) use gated recurrent unit (GRU) cells (from [4]) instead of LSTM cells, and 2) forgo the character-level CNN input. [3] showed that the character-level CNN improves the accuracy slightly, specifically when there are out of vocabulary words, but the gains are minimal compared with other modifications in their ablation study.

A description of our network architecture is given below, followed by an outline of the different attention mechanisms.

2.1 Network Architectures

The neural network models used breaks down into seven blocks:

1. **Embeddings:** pretrained 100-dimensional GloVe vectors. We represent the context vectors as $\{\boldsymbol{x}_i\}_{i=1}^N$ and question vectors as $\{\boldsymbol{y}_j\}_{j=1}^M$ with $\boldsymbol{x}_i, \boldsymbol{y}_j \in \mathbb{R}^d$.
2. **Encoder:** we pass the word vectors through a bidirectional GRU (biGRU) architecture to find hidden state viectors:
$$\{\boldsymbol{c}_i\}_{i=1}^N = \text{biGRU}(\{\boldsymbol{x}_i\}_{i=1}^N)$$
$$\{\boldsymbol{q}_j\}_{j=1}^M = \text{biGRU}(\{\boldsymbol{y}_j\}_{j=1}^M)$$
with $\boldsymbol{c}_i, \boldsymbol{q}_j \in \mathbb{R}^h$.
3. **Attention:** we test three different attention mechanisms, which are described below. We will notate the output of the attention layer as:
$$\{\boldsymbol{b}_i\}_{i=1}^N = \text{attention}(\{\boldsymbol{c}_i\}_{i=1}^N, \{\boldsymbol{q}_j\}_{j=1}^M)$$
with $\boldsymbol{b}_i \in \mathbb{R}^{4h}$ or $\boldsymbol{b}_i \in \mathbb{R}^{6h}$.
4. **Modeling layer**: in the one- and two-layer configurations, we pass the representations through a language modeling layer. The modeling layer passes the output of the attention layer through successive bidirectional RNN layers to build a neural language model. Here, we use GRU cells, although other studies (e.g. [3]) use LSTM cells. The purpose of the modeling layer is to build a more accurate latent representation of each step in the context sequence.
5. **Output:** the outputs from the attention layers are passed through a fully-connected layer with ReLU nonlinearity:
$$\boldsymbol{b}'_i = \text{ReLU}(\boldsymbol{W}_{FC}\boldsymbol{b}_i + \boldsymbol{v}_{FC}) \quad \forall\, i = 1, ..., N$$
6. **Predictions:** the nonlinearity layer outputs are passed to two separate affine layers to compute logits for the start and end words, respectively:
$$\text{logits}_i^{\text{start}} = \boldsymbol{w}_{start}^T \boldsymbol{b}'_i + u_{start}, \quad \text{logits}_i \in \mathbb{R} \quad \forall\, i = 1, ..., N$$
$$\text{logits}_i^{\text{end}} = \boldsymbol{w}_{end}^T \boldsymbol{b}'_i + u_{end}, \quad \text{logits}_i \in \mathbb{R} \quad \forall\, i = 1, ..., N$$
then use these to compute start and end probabilities:
$$p^{\text{start}} = \text{softmax}(\text{logits}_i^{\text{start}}), \quad p^{\text{start}} \in \mathbb{R}^N$$
$$p^{\text{end}} = \text{softmax}(\text{logits}_i^{\text{end}}), \quad p^{\text{end}} \in \mathbb{R}^N$$
7. **Loss:** the loss is calculated as the sum of the cross entropy of the starting position and ending position i.e.:
$$L = L^{\text{start}} + L^{\text{end}} = -\log(p^{\text{start}}) - \log(p^{\text{end}})$$

Fig. 1 shows the architecture used in our experiments. The architecture is primarily inspired by that in [3] (and our architecture diagram is based off fig. 1 of that paper).

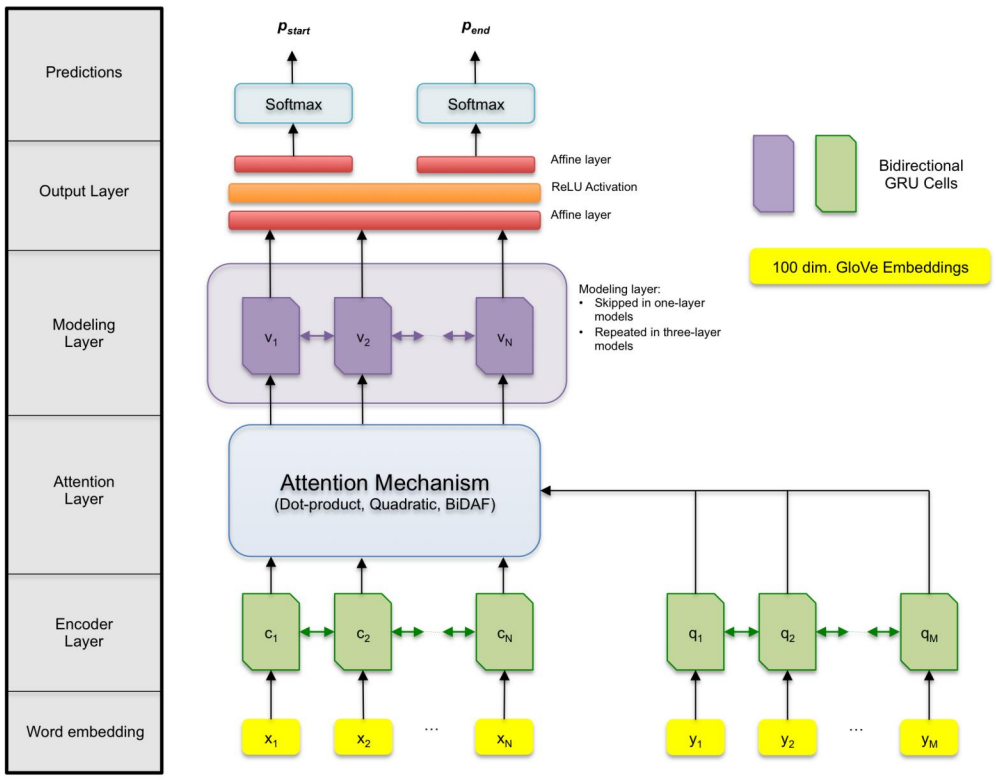


Figure 1: Network structure used for the below experiments. Based off network structure figure from [3]. The two- and three-layer models are created by adding additional modeling layers.

2.2 Attention Mechanisms

In order to normalize for the effect of attention on accuracy compared with the effect of language modeling layers, we use three types of attention layers in our model. The attention layers are described below.

2.2.1 Dot-Product Attention

At their core, attention mechanisms use similarities between the context and question to computed a weighted sum of the question vector. The simplest type of attention is dot-product attention. The equations for dot-product attention are:

$$\boldsymbol{e}^i = [\boldsymbol{c}_i^T \boldsymbol{q}_1, \boldsymbol{c}_i^T \boldsymbol{q}_2, ..., \boldsymbol{c}_i^T \boldsymbol{q}_M], \quad \boldsymbol{e}_i \in \mathbb{R}^M$$

$$\alpha^i = \text{softmax}(\boldsymbol{e}_i)$$

$$\boldsymbol{a}_i = \sum_{j=1}^{M} \alpha_j^i \boldsymbol{q}_j$$

$$\boldsymbol{b}_i = [\boldsymbol{c}_i; \boldsymbol{a}_i], \quad \boldsymbol{b}_i \in \mathbb{R}^{4h}$$

The intuition is that the words in the question will be similar to (or the same as) the words in the question, and therefore dot-product similarity between the question and context words will be a good indicator for which words are most relevant.

2.2.2 Quadratic Attention

As a simple extension of dot-product attention, we propose a novel modified version we will call *quadratic* attention. Here, instead of computing the simple inner-product between the question and

context vectors, we compute the quadratic inner product with respect to a matrix $\boldsymbol{W}_A \in \mathbb{R}^{h \times h}$:

$$\boldsymbol{e}^i = [\boldsymbol{c}_i^T \boldsymbol{W}_A \boldsymbol{q}_1, \boldsymbol{c}_i^T \boldsymbol{W}_A \boldsymbol{q}_2, ..., \boldsymbol{c}_i^T \boldsymbol{W}_A \boldsymbol{q}_M], \quad \boldsymbol{e}_i \in \mathbb{R}^M$$

$$\alpha^i = \text{softmax}(\boldsymbol{e}_i)$$

$$\boldsymbol{a}_i = \sum_{j=1}^{M} \alpha_j^i \boldsymbol{q}_j$$

$$\boldsymbol{b}_i = [\boldsymbol{c}_i; \boldsymbol{a}_i], \quad \boldsymbol{b}_i \in \mathbb{R}^{4h}$$

where $\boldsymbol{W}_A$ is a trainable parameter matrix. The intuition behind this modification is that taking the inner-product between two vectors may be too simple a model for computing their similarity, but projecting the vectors onto a different basis then taking the inner-product allows for more geometric expressivity in the model.

2.2.3 Bidirectional Attention Flow

Bidirectional Attention Flow (BiDAF) from [3] follows a similar approach, except here the similarity score $\boldsymbol{S}_{ij}$ is computed as a weighted sum of context, question, and element-wise product of context and hidden states:

$$\boldsymbol{S}_{ij} = \boldsymbol{w}_{sim}^T [\boldsymbol{c}_i; \boldsymbol{q}_j; \boldsymbol{c}_i \circ \boldsymbol{q}_j]$$

where $\boldsymbol{w}_{sim} \in \mathbb{R}^{6h}$ is a weight vector. We then compute the weights and weighted sum of the question vectors:

$$\alpha^i = \text{softmax}(\boldsymbol{S}_{i:})$$

$$\boldsymbol{a}_i = \sum_{j=1}^{M} \alpha_j^i \boldsymbol{q}_j \quad \forall\, i = 1, ..., N$$

From here, BiDAF also computes a weighted sum of the context vectors:

$$m_i = \max_j \boldsymbol{S}_{ij} \quad \forall\, i = 1, ..., N$$

$$\beta = \text{softmax}(\boldsymbol{m})$$

$$\boldsymbol{c}' = \sum_{i=1}^{N} \beta_i \boldsymbol{c}_i$$

This step (query-to-context attention or Q2C) is the primary difference between BiDAF and dot-product attention. The output from dot-product attention is only a linear combination of the question vectors, whereas BiDAF also outputs a linear combination of the context vectors. This means that the attention outputs both depend on the similarity of the context with respect to the question, as well as the question's similarity to the context.

We then calculate the attention output as:

$$\boldsymbol{b}_i = [\boldsymbol{c}_i; \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{c}']$$

3 Results

We present results for a total of nine model configurations: zero, one, or two modeling layers with each attention type. All models are trained with Adam optimizer [7] and models with the same number of modeling layers are trained with the same learning rate. The results for the various network architectures are summarized in table 1, and F1 and loss curves are shown in figs. 2 and 3, respectively.

The most immediately apparent results is that the addition of a single language modeling layer significantly increases the exact match (EM) and F1 scores. Between all the variations on the model architecture, we observe the biggest jump when a language modeling layer is added. We can think of the language modeling layer as creating more complex hierarchical embeddings of each step of the context sequence, so when we add one layer after the attention layer, we are essentially repeating the

	Exact Match		F1	
	Dev	Test	Dev	Test
Zero modeling layers				
0-layer Bi-GRU + Dot Product (baseline)	29.29%	–	40.02%	–
0-layer Bi-GRU + Quadratic	27.54%	–	38.00%	–
0-layer Bi-GRU + BiDAF	**31.21%**	–	**42.12%**	–
One modeling layer				
1-layer Bi-GRU + Dot Product	49.10%	–	64.33%	–
1-layer Bi-GRU + Quadratic	48.56%	–	63.17%	–
1-layer Bi-GRU + BiDAF	**50.59%**	–	**65.05%**	–
Two modeling layers				
2-layer Bi-GRU + Dot Product	**50.66%**	62.646	**65.55%**	72.483
2-layer Bi-GRU + Quadratic	48.56%	–	63.17%	–
2-layer Bi-GRU + BiDAF (trained for 10 epochs)	47.84%	–	63.26%	–
Human [2]	80.3%	77.0%	90.5%	86.8%
Logistic Regression [2]	40.0%	40.4%	51.0%	51.0%
Dynamic Co-Attention Network [5]	65.4%	75.6%	66.2%	75.9%
BiDAF Network [3]	65.4%	75.6%	66.2%	75.9%
R-NET [6]	72.3%	72.3%	80.6%	80.7%

Table 1: Summary of results and comparison to other models.

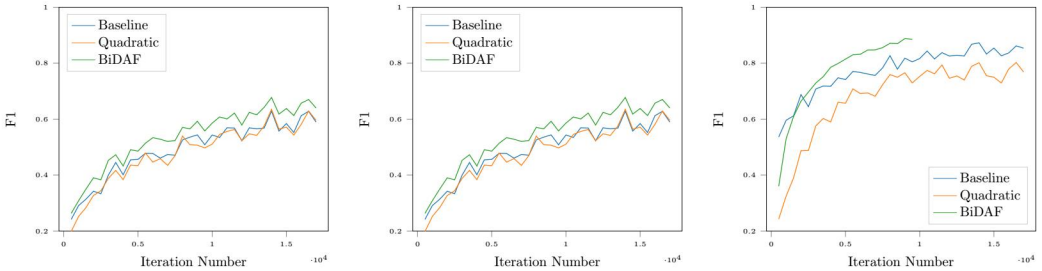
network's operation on its context, except over the attended context representations. This shows that attention in conjunction with a recurrent modeling layer can be very powerful, and corresponds to the propositions in [6].

Along this same line, we see that adding a second language modeling layer causes only a small increase in accuracy. While the accuracy does increase for all models (except two-layer BiDAF), the increase is relatively small. Though this may be due to training difficulties—larger models tend to require much more hyperparameter tuning—even with further hyperparameter tuning, it is highly unlikely that the increase from the second language modeling layer would increase the accuracy as much as does the first layer. Between small gains in accuracy and the much greater difficulty of tuning and overfitting, successive modeling layers alone may not be the best choice for building better question answering systems.

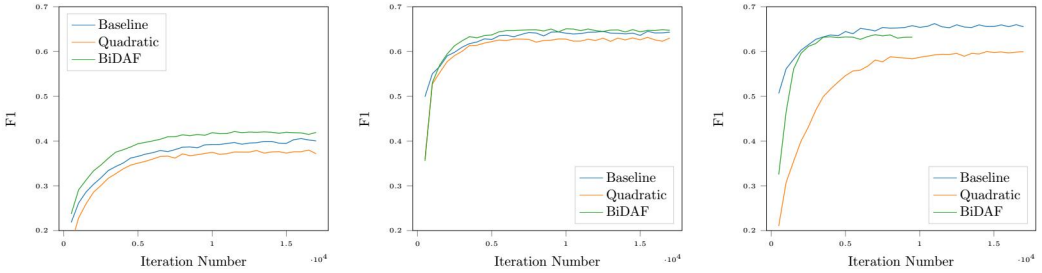
4 Discussion

Table 2 shows the output for an example question/context pair. This particular question/context pair demonstrates several notable attributes of the results. Most apparently, we see that BiDAF and dot-product attention perform more strongly than quadratic attention. This is inline with the training results (tab. 1): for the zero- and one-layer models, quadratic attention is the weakest model. Quadratic attention is nearly identical to dot-product attention, except that the inner product is taken with respect to a matrix $\boldsymbol{W}_A$. The performance of this type of model is worse than dot-product attention for all model types, showing that the inner-product with respect to a matrix provides a poor comparison of vector similarity. In the example from table 2, one-layer quadratic attention locates the approximate part of the context where the answer lies ("third"), it does not actually locate the answer ("third étude"). This mistake is telling because quadratic attention chooses a word that is the wrong part of speech (an adjective), while dot-product attention and BiDAF both identify that the answer should contain a noun. This indicates that the quadratic similarity metric is obscuring the relationships between question and context. It is possible that constraining $\boldsymbol{W}_A$ to be in the positive semidefinite cone (or even positive definite cone) could improve performance, but this would create a constrained optimization problem that becomes enormously more difficult to solve than the already difficult-to-optimize network.

Secondly—and more interestingly—table 2 shows that the one-layer quadratic attention model still locates the right part of the sentence for the answer, while the zero-layer quadratic model completely misses it. A similar effect is observed for BiDAF: the zero-layer model incorrectly identifies the

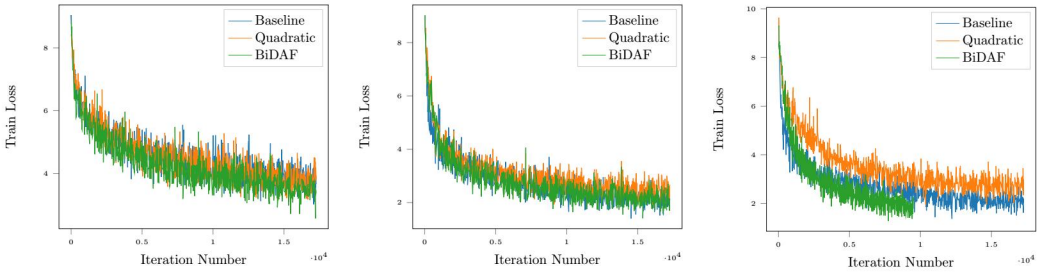


(a) Training Set - 1 layer (b) Training Set - 2 layer (c) Training Set - 3 layer

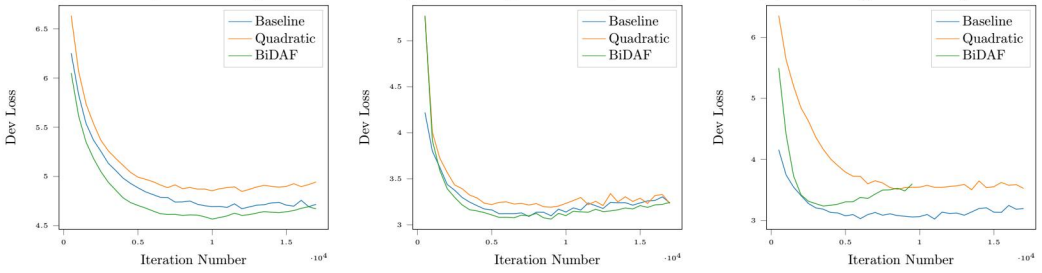


(d) Dev Set - 1 layer (e) Dev Set - 2 layer (f) Dev Set - 3 layer

Figure 2: F1 scores for training and development sets. We see that the BiDAF model tends to fit the training data most closely, but it is only in the 3-layer model that it significantly overfits the training data. We also observe that while the extra modeling layer does not significantly increase the development set F1, it does increase the training set F1 value, showing that simply adding the extra layer can make the model more prone to overfitting.



(a) Training Set - 1 layer (b) Training Set - 2 layer (c) Training Set - 3 layer



(d) Dev Set - 1 layer (e) Dev Set - 2 layer (f) Dev Set - 3 layer

Figure 3: Loss curves for training and development sets. As with the F1 curves, we see that the BiDAF models tend to overfit the most quickly. In particular, the 3-layer BiDAF model overfits after only a few epochs, indicating both that the learning rate was perhaps a bit too high in this trial, and that this model is more sensitive to the optimization parameters than the quadratic and dot-product attention models.

starting word, but the one-layer model identifies this correctly. These results indicate that adding the modeling layers can actually improve the effectiveness of the attention mechanism. Indeed, the models with different attention mechanisms show similar changes in performance (tab. 1) as successive modeling layers are added, which supports our hypothesis that the modeling layers have a larger effect on performance than the attention layer by itself. This is likely because the modeling layer forms a type of synergy with the attention mechanism, in that it provides a "buffer" latent representation between the attention and output layers that can correctly redirect the outputs from the attention layer, and during training helps to more efficiently backpropagate error into the trainable parameters of the attention layer.

Finally, we see that more modeling layers makes the model more sensitive to hyperparameter tuning and optimization, and is more prone to overfitting. This effect is especially pronounced with the two-layer models: even though the same learning rate is used for all three, we see that BiDAF overfits very quickly while the other two models still train well. Adding the second modeling layer increased the network's sensitivity to hyperparameters and made the network more prone to overfitting. Indeed, the development loss (fig. 3) plot for the two-layer model shows that BiDAF begins to overfit after about 3 epochs, whereas dot-product attention does not overfit until around 10 epochs.

5 Conclusion and Future Work

This work analyzed the effect of language modeling layers on question-answer performance. The results presented here show that language modeling layers have the largest effect on performance, but the complexity of the attention mechanism will consistently increase or decrease model performance. Furthermore, the first language modeling layer has an outsized effect on the performance compared to other changes to the network.

Future work should focus on replicating the gains from adding a single modeling layer in successive modeling layers. Two possible routes to doing this would be adding residual connections across the attention layers or additional attention layers (involving Q2C and C2Q attention) between the modeling layers. Residual connections would preserve the information from the input throughout the depth of the network, and attention layers between modeling layers (possibly in conjunction with residual connections) would allow the network to attend over successive hierarchical representations of the context. However, larger architectures such as this would come with their own host of optimization challenges similar to what we have shown here for two-layer models, so training difficulties may be of great concern or even prohibitive when using these networks.

6 Acknowledgements

Thank you to the CS 224N teaching staff for their extremely helpful writeup for the default final project, and in particular Abi See for providing the starter code for this project.

References

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

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. (ii), 2016.

[3] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional Attention Flow for Machine Comprehension. pages 1–13, 2016.

[4] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 2014.

[5] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic Coattention Networks For Question Answering. pages 1–14, 2016.

[6] Microsoft Research Asia Natural Language Computing Group. R-Net: Machine Reading Comprehension With Self-Matching Networks *. *arXiv*, pages 1–11, 2017.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *Iclr*, pages 1–15, 2015.

Question: "in which étude of neumes rythmiques do the primes 41, 43, 47 and 53 appear in ?" **Correct answer:** the third étude
Dot-product 1: "prime numbers have influenced many artists and writers. the french composer olivier messiaen used prime numbers to create _ametrical_ music through 'natural phenomena'. in works such as la _nativité_ du seigneur (1953) and quatre études de rythme (_1949-50_), he simulatenously employs motifs with lengths given by different prime numbers to create unpredictable rhythms : the primes 41, 43, 47 and 53 appear in the third étude, 'neumes _rythmiques_ ' . according to messiaen this way of composing was 'inspired by the movements of nature, movements of free and unequal durations'." **Dot-product 1 answer:** third étude
Quadratic 0: "prime numbers have influenced many artists and writers. the french composer olivier messiaen used prime numbers to create _ametrical_ music through 'natural phenomena'. in works such as la _nativité_ du seigneur (1953) and quatre études de rythme (_1949-50_), he simulatenously employs motifs with lengths given by different prime numbers to create unpredictable rhythms : the primes 41, 43 , 47 and 53 appear in the third étude, 'neumes _rythmiques_ ' . according to messiaen this way of composing was 'inspired by the movements of nature, movements of free and unequal durations'." **Quadratic 0 answer:** 43, 47
Quadratic 1: "prime numbers have influenced many artists and writers. the french composer olivier messiaen used prime numbers to create _ametrical_ music through 'natural phenomena'. in works such as la _nativité_ du seigneur (1953) and quatre études de rythme (_1949-50_), he simulatenously employs motifs with lengths given by different prime numbers to create unpredictable rhythms : the primes 41, 43, 47 and 53 appear in the third étude, 'neumes _rythmiques_ ' . according to messiaen this way of composing was 'inspired by the movements of nature, movements of free and unequal durations'." **Quadratic 1 answer:** third
BiDAF 0: "prime numbers have influenced many artists and writers. the french composer olivier messiaen used prime numbers to create _ametrical_ music through 'natural phenomena'. in works such as la _nativité_ du seigneur (1953) and quatre études de rythme (_1949-50_), he simulatenously employs motifs with lengths given by different prime numbers to create unpredictable rhythms : the primes 41, 43, 47 and 53 appear in the third étude, 'neumes _rythmiques_ ' . according to messiaen this way of composing was 'inspired by the movements of nature, movements of free and unequal durations'." **BiDAF 0 answer:** quatre études de rythme (_1949-50_), he simulatenously employs motifs with lengths given by different prime numbers to create unpredictable rhythms : the primes 41, 43, 47 and 53 appear in the third étude
BiDAF 1: "prime numbers have influenced many artists and writers. the french composer olivier messiaen used prime numbers to create _ametrical_ music through 'natural phenomena'. in works such as la _nativité_ du seigneur (1953) and quatre études de rythme (_1949-50_), he simulatenously employs motifs with lengths given by different prime numbers to create unpredictable rhythms : the primes 41, 43, 47 and 53 appear in the third étude, 'neumes _rythmiques_ ' . according to messiaen this way of composing was 'inspired by the movements of nature, movements of free and unequal durations'." **BiDAF 1 answer:** third étude

Table 2: Example output for different attention types with zero- and one-layer models. (We omit the dot-product attention zero-layer model since the output was the same as for the one-layer model.) We see that adding modeling layers improves the effectiveness of a given attention mechanism. We also observe that dot-product and BiDAF attention find the correct answer, while quadratic attention partially misses the answer. (*Key:* correct answer, predicted start , predicted end)