Reading Comprehension with Two Layer Attention

Urjit Patel, Vincent Chabot
Center for Data Science - New York University
{up276, vec241}@nyu.edu

Summary

Making machines able to answer the question through reading comprehension is yet the big area of research in Natural Language Processing. Many research has already been done on reading comprehension task. Specially attention based neural network models have shown significant improvement in reading comprehension task. Herman et al. (2015) [1] defined a methodology which allows to develop a class of attention based deep neural network that learns to read documents and answer complex question with minimal prior knowledge of language structure. Later Chen et al. (2016) [2] proposed a simplified version of the same model which was able to achieve 7-10% accuracy improvement on CNN and Daily Mail data sets, we call it as Stanford reader. In this paper, we conduct a research on both models and present our two layers attention model (with question feeding in second layer) which is inspired from attentive reader (Herman st al., 2015) and Standford reader(Chen st al., 2016). We show that our approach achieves 74.15% accuracy which is almost 1.5% higher than accuracy achieved by Chen et al. with Stanford reader.

1 Introduction

Reading comprehension is defined as the level understanding of a text. This understanding comes from the interaction between the words that are written and how they trigger knowledge outside the text [1]. Many attempts have been done in recent years to teach machine to read and comprehend. Except last two years, supervised approaches have largely been missing because of scarcity of enough labeled training data.

Although creating supervised data set for reading comprehension is difficult, few researchers have explored labeled data set creation specifically for reading comprehension task. DeepMind researchers data sets[2] (Herman et al., 2015) created a labeled data set in form of context-question-answer from large corpus of CNN and Daily Mail articles. The main body of the article creates a context and question is created by replacing named entity from summary sentence. In addition, named entities in the whole data set were replaced by anonymous tokens to make sure that model does not learn or build any word knowledge over the entity names. In contrast with CNN and Daily Mail data set, Researchers at Toyota Technological Institute at Chicago (Onishi et al., 2016 [3]) created the Who-did-what data set by using different two articles of the same event to form the question and answer. Other reading comprehension data sets include MCtest (Richardson et al., 2013 [4]), Children's Book Test (Hill et al., 2016 [5]) etc.

Abundance of recent work on these new data sets has shown that Neural methods give efficient performance in reading comprehension task. Herman et al.(2015)[1] proposed a first attention based neural network model, attentive reader, to address the reading comprehension task on CNN and Daily Mail dataset. Chen et al.(2016) [2] proposed a simplified version of attentive reader which outperforms the original attentive reader model. Hill et al.(2015) applied the memory networks

[1] https://en.wikipedia.org/wiki/Reading_comprehension
[2] https://github.com/deepmind/rc-data

(Weston et al., 2014 [6]) to the task of text comprehension. Kobayashi et al.(2016) proposed the Dynamic Entity Representation model which uses the weighted attention mechanism and max-pooling over contextual embeddings of vectors for each named entity. Pointer Networks (Vinyals et al., 2015 [7]) uses attention mechanism to select the answer in the context rather than taking blend version of words from the context. Inspired from Pointer Networks, Rudolph et al. (2016)[8] proposed a attentive sum reader (without entity anonymization) which uses attention directly to compute the answer probability.

Attention based neural networks are the heart of almost all the proposed architecture for reading comprehension task. We believe that attention models proposed by Herman et al. and Chen st al. have promising argument of anonymizing entities as we are more interested in finding the answer using context rather than word knowledge. We did our analysis on both the proposed models, Attentive reader and Stanford reader, and also implemented a new architecture where we use a two layer attention approach to find the answer entity. Our belief is that adding an extra layer of attention should give the network a chance to refine the attention over contextual hidden states. In this paper, we first discuss about the attention mechanism and proposed models Attentive Reader(Herman et al.) and Stanford Reader(Chen wt al.). Second, we discuss our experimental model inspired from the stanford and attentive reader. At the end, we present the comparison analysis of our model with proposed models.

2 Background: Reading Comprehension

Attention Mechanisms in Neural Networks are inspired from the visual attention mechanism found in humans. Attention has a long history in Image Recognition. Example includes Larochelle et al.(2010) [9], denil et al. (2012) [10], and Mnih et al. (2014) [11] etc. But only recently have attention mechanisms made their way into recurrent neural networks architectures that are typically used in Natural Language Processing. Bahdanau et al. (2015) [12] introduced Attention model for the Machine Translation task which was built upon RNN Encoder-Decoder architecture proposed by Cho et al.(2014)[13]. They proposed a novel architecture which consists of a bidirectional RNN as an encoder and a decoder searches through a source sentence during decoding. At each time stamp based on the current decoding word, network assigns the special attention to each of the hidden states coming from the encoder.

For a reading comprehension task, the same kind of structure can be applied, where instead of using target word translation, we can use question representation to decide where to give special attention in the context. Generally, we pass the data as (document, question, answer) triple (d,q,a), d = (p1,p2,...,pn) and q = (x1,x2,...,xm) are the tokens of document and question sentence. We also have E as the set of all entities from all the documents. q contains exactly one unknown token, we call it as "@placeholder". The goal is to infer correct entity a $\in$ d $\cap$ E that placeholder corresponds to.

Attention model architecture overview

The basic architecture of any attention model includes the following three components:

1. Encoder module:

All the words from document and question are mapped to a d-dimentional vectors using an embedding matrix. We can also use pre-trained vectors. Hence we would have $p1, p2, ..pn \in R^d$ for the document words and $x1, x2, .., xm \in R^d$ for the question. Encoder can be any Recurrent Neural Network, LSTM (Hochreiter et al. 1997 [14]) or GRU (Cho at al. 2014 [13]), which takes the word vectors as an input and maps it to contextual embedding which would have the context information of the neighbour words. As we have two input sources, one for document text and second is the question on this text, it would require two RNN encoders for each of these.

for document,

$h_i = RNN(h_{i-1}, p_i)$, i = 1,2,,n

same for question,

$q_i = RNN(q_{i-1}, x_i)$, i = 1,2,,m

where h_i,$q_i \in R^h$, h is the dimension of these contextual embeddings. Generally we are interested in the last hidden state of the question encoder as it would have the information of all question words.

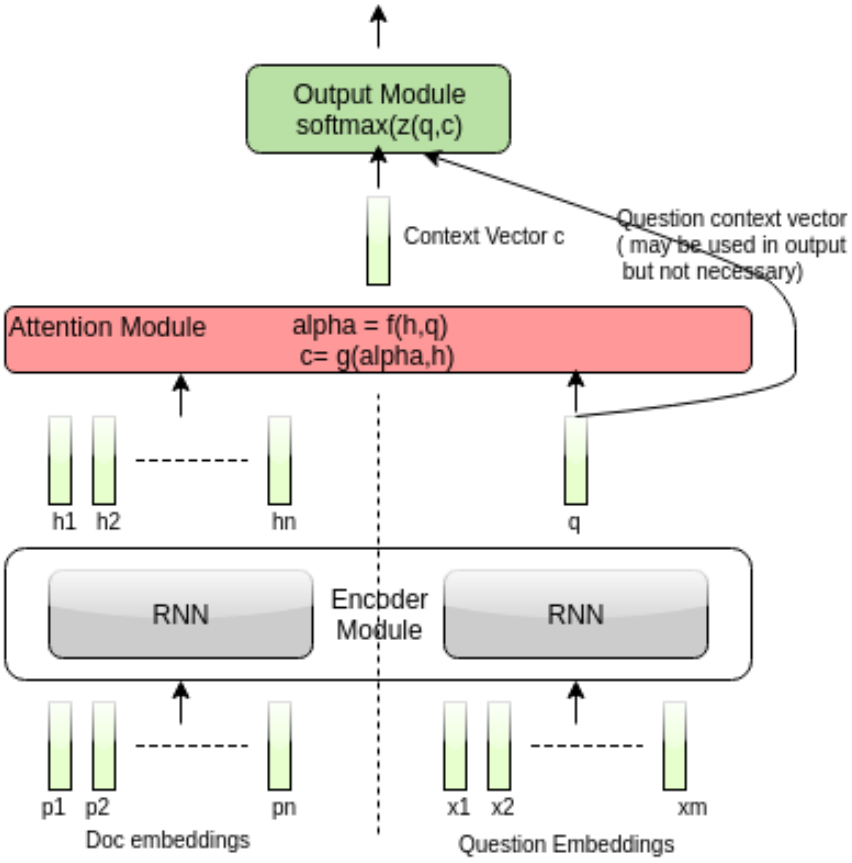

Figure 1: General Attention Architecture for Reading Comprehension

2. Attention module

Attention module takes the source text contextual embeddings h_i vector representation and question vector representation q_i as an input and assigns the attention scores α_i to each of the document embeddings vectors h_i. Using the attention score, it then generates the context vector c as the function of h_i to find the required answer from the source text.

$\alpha = f(h, q)$

$c = g(\alpha, h)$

f is the function of the attention layer, which can be either a dot product, an MLP, or a Bilinear term. g is just the linear operation between the attention scores and the document contextual embedding.

3. Output module

Output module can be either any type of neural networks or a Linear layer followed by a softmax, which takes the context vector of the document embedding information h and may take question vector q as an input and assigns probability score to each of the target answers. We may then go with top scored entity as an answer for our question.

$a = \text{softmax}(z(c, q))$

Where z is the function of outpout layer, which can be an MLP layer or just a simple direct prediction from context vector c.

3 Related Work: Reading Comprehension

Attentive Reader

Attentive reader was the leading attention architecture proposed by Herman et al. (2015). They used bidirectional single layer RNN (LSTM) (Schuster et al. (1997) [15]) to encode the document and the question, such that at each time stamp we can have information coming from both sides of the word in the document. For document encoding, at each time stamp they concatenate hidden vectors coming from both the directions. For question encoding they just use last vectors generated at the end in both directions and concatenate them. For attention they used a tanh layer with a weight matrix to learn the relationship between document vector and question vector. After creating attentional context vector, they use again tanh with weight matrix which combines context vector and question vector to find the score for all the words present in the vocab.

Stanford Reader

Chan et al. in 2016 claimed that performance can be improved by simplifying original attentive reader. They proved their claim with their experiments by outperforming the original model by 7-10%. Instead of LSTM (Hochreiter et al. 1997 [14]), they used GRU (Cho at al. 2014 [13]) for their bi-directional RNN. Apart from that, they used bilinear term instead of tanh to calculate the relevance between question and document contextual embedding. Once they obtained weighted contextual embedding (attention vector), they directly predict the scores for the entities using simple linear layer. In contrast, the original model (attentive reader) uses non-linear tanh layer before making the final predictions. Their believe that it is sufficient for the model to learn to return the entity to which it gives more attention. They just predict scores for possible entities rather than all the words of the vocab.

4 Attentional Model without Non-linearity tanh

We took the Stanford reader implemented by Chen et al. (2016) as our baseline. We use the same model for our early experiments, and we played with few components. One of our experiments was to make the network even simpler by just doing the dot product between the document contextual embedding and the question context for attention. We were able to achieve 74.20% accuracy with this framework.

5 Our Model: Two Layer Attentional Model

The current models are just using the one attention layer before going for the final prediction. We believe that, adding a second attention layer on the top of the first attention layer can provide network a chance to refine the attention before making the final prediction with output layer. We implemented two models with the two layers of attention.

Encoding

For the Encoding part we used the same logic which is used by Chen et al.

We used bi-directional GRU to get encode the document words and question.

for document vecors $p1, p2, ..., pn \in R^d$,

$\vec{h_i} = RNN(\vec{h_{i-1}}, p_i)$, i = 1,2,,n

$h_i = RNN(h_{i-1}, p_i)$, i = 1,2,,n

where $\vec{h_i}$ and h_i are hidden vectors from both direction at i^{th} time stamp. We concat both vectors to get the one vector at each time stamp.

$d_i = concat(\vec{h_i}, h_i)$, $d_i \in R^h$, where h is two times of the dimension of h_i.

For the question, we just take the last vectors generated in both the direction by GRU and concatenate. We used this encoding scheme for both versions of our implementation.

Model Architecture

- **Layer 1:**
  Layer 1 for both models are same. we used bilinear term with softmax to find the attention vetor c_1. For output layer, we apply tanh non-linearity on the linear layer. As we have another following layer, adding non linearity is essential here.

  attention:
  $\alpha 1_i = softmax_i(q^T W_{a1} d_i)$
  $c_1 = \sum_{i=1} \alpha 1_i d_i$
  output:
  $o_1 = softmax(tanh(W_{o1} c_1))$

- **Layer 2:**

  Model 1 (without question feeding):
  Our intention here is to use the information coming from output layer o_1 of the first layer and refine the context vector c_1 somehow. Hence, we add a simple dot product attention layer which takes both, document contextual embedding d and output vector o_1 to find the refined attention scores $\alpha 2_i$, and then new context vector c_2. As we observed good result with direct prediction for final output, we use the same to get our final output o_2.

  attention:
  $\alpha 2_i = softmax_i(o_1 d_i)$
  $c_2 = \sum_{i=1} \alpha 2_i d_i$
  output:
  $o_2 = softmax(W_{o2} c_2)$

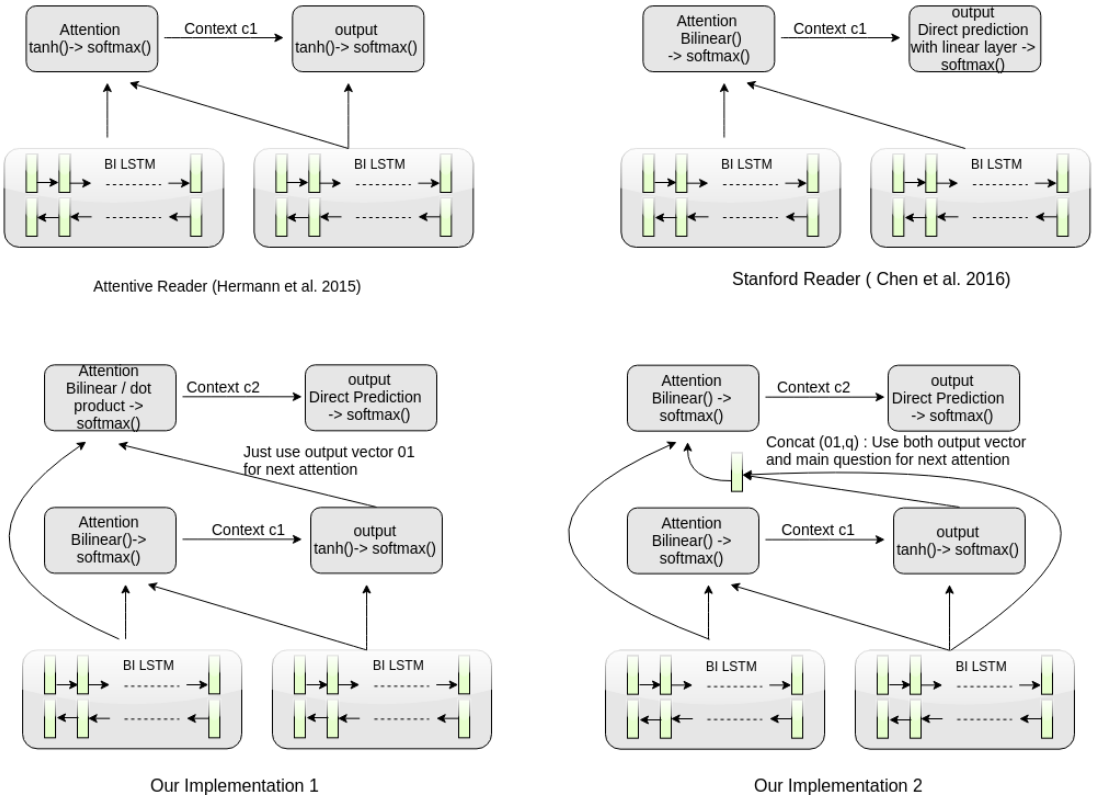

Figure 2: Comparison of our architecture with others

Model 2 (with question feeding):

With the above implementation, we were able to beat the Stnaford reader (Chen et al.), but we were not able to get more than 74.20%. We decided to pass the question embedding in second layer in our second implementation.

$o'_1 = concat(o_1, q)$

We concatenate the first layer output o_1 with the question embedding q. Now instead of passing just o_1 to second layer, we pass this concatenated vector o'_1 having the question contextual information. Because of dimension miss match, we can not apply simple dot product between document contextual embedding d and o'_1. hence we used bilinear term for second layer attention too.

attention:

$\alpha 2_i = softmax_i(o'_1 W_{a2} d_i)$

$c_2 = \sum_{i=1} \alpha 2_i d_i$

output:

$o_2 = softmax(W_{o2} c_2)$

Final output is just the linear layer with the softmax. With this architecture, we were able to get almost 74.20% accuracy.

6 Experiment settings

Data set and Embeddings

Previous work on Reading Comprehension has proved that, CNN and Daily Mail data set [3] with entity anonimization (Herman et al. 2015) could be used to build and evaluate attention based framework. We referred the same data set and use only CNN data set to build and evaluate our system so that we can compare our result with the previous work done by Herman et al. 2015 and Chen et al. 2016.

From 93k CNN articles, they created document-query-answer triples by turning the summary bullet points into cloze [16] style questions by replacing one entity with a placeholder.To prevent the system from learning from word knowledge, all entities were replaced by abstract entity markers and shuffled each time.

	CNN
Train samples	380298
Dev samples	3924
Test samples	3198
avg. sentences/ document	around 32
avg. tokens/ document	around 760
avg. tokens/ question	around 12
avg. entities/ document	around 26

Training

To train our attention based neural networks, we used Tesla k80 GPU servers available at NYU. We considered only top 50k most frequent words to create our vocabulary and took all other words as <unk>. We initialize our embedding matrix using 100 dimensional GloVe word vectors [17]. Encoder GRU weights were initialized with Gaussian distribution with 0 mean and 0.1 standard deviation. Other parameter details are given in the table,

[3] https://github.com/deepmind/rc-data

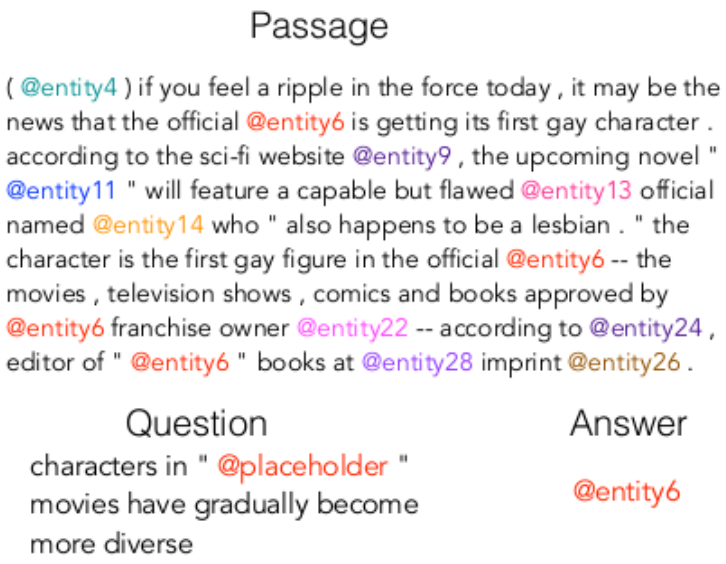

Figure 3: Example of the CNN data set after annonimization (Image from Chen et al.)

Parameter	**Value**
Encoder	Bidirectional GRU
Embedding dimension	100 (tried with 50 and 200 too)
vocab size	50000
No. of epochs	50 to 100
Batch size	64
Learning rate	0.1
Dropout rate	0.4
Stopping criteria	max epoch 100
Optimizer	Gradient Descent
Gradient Clipping limit	10

Evaluation

For evaluation, we just used accuracy on a development and a test set as our matrix so that we can compare our results with Herman et al.(2015) and Chen et al.(2016).

7 Results

Experiment 1

We experimented with both the pre-claimed models (attentive reader and Stanford reader) on the CNN Data set. We started our experiments with Stanford reader as our baseline. As a first milestone, we were able to achieve the same accuracy as Chen et al. with the Stanford reader.

Experiment 2

As Chen et al. were able to increase the accuracy by 7-10 % from attentive reader by making it simpler. We thought to make it more simpler by just using dot product between document contextual embedding and question embedding to get the attention vector. Surprisingly, we were able to increase the accuracy to 74.20%. We also tried with MLP attention layer instead of dot product, but it was not able to beat the 74% accuracy limit.

Experiment 3

For our first implementation, where we are not feeding the question embedding in second attention layer, we tried with different combinations of attention and output layers. Highest accuracy 73.80% was achieved with dot product attention and MLP tanh output layer for first layer, and Bilinear attention and linear layer output for second layer.

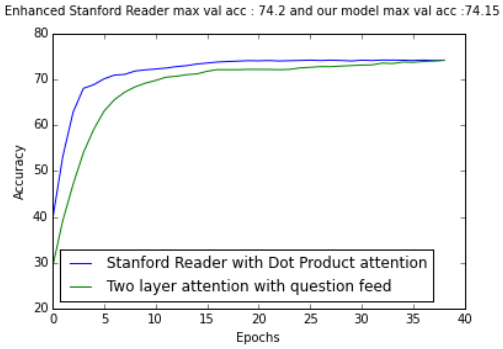

Figure 4: Best Models : Validation Accuracy Results

Experiment 4

For our second implementation, we concat the question contextual embedding with the output of first layer and pass it to second layer attention. Here also we tried with different combinations of attention and output layers. Highest accuracy 74.15% was achieved with Bilinear attention and MLP tanh output layer for first layer, and Bilinear attention and linear output layer for second layer.

Model	**Attn. L1**	**o/p L1**	**Attn. L2**	**o/p L2**	**Dev**
Attentive Reader (Herman et al.)	MLP Tanh	MLP Tanh	-	-	61.6
Stanford Reader (Chen et al.)	Bilinear	Linear	-	-	72.5
Stanford Reader (We tried)	Dot Prod	Linear	-	-	**74.20**
Stanford Reader (We tried)	MLP tanh	Linear	-	-	**73.98**
Our model w/o question feeding	Dot Prod	MLP Tanh	Dot Prod	Linear	73.65
Our model with question feeding	Dot Prod	MLP Tanh	Bilinear	Linear	**73.80**
Our model w/o question feeding	Bilinear	MLP Tanh	Dot Prod	Linear	74.01
Our model with question feeding	Bilinear	MLP Tanh	Bilinear	Linear	**74.15**

8 Conclusion

In conclusion, simple attention frameworks works good for Reading Comprehension Task on CNN data set. Our experiments proves the same thing, where we were able to beat the Chen et al. Stanford reader's accuracy using just dot product in first attention layer. Our proposed two layer attention architecture (74.15% accuracy) with question feeding was also able to beat the accuracy of Chen et al. Although, as we can see in accuracy plot (figure 4), simple framework (stanford reader with our modification) learns fast than our architecture (two layer attention), We believe it as a potential network for difficult reading comprehension task then CNN where network needs to get one chance to adjust the attention to find the answer from the passage.

Statement of Collaboration

Urjit : : Stanford Reader Implementation, Two Layer architecture brainstorming, Two Layer architecture Implementation and experiments, Report Writing.

Vincent : : Data set selection-cleaning-modifications, Stanford Reader enhancement and experiments, Two Layer architecture brainstorming, Report Writing.

References

[1] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[2] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016.

[3] Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Who did what: A large-scale person-centered cloze dataset. *arXiv preprint arXiv:1608.05457*, 2016.

[4] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 3, page 4, 2013.

[5] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

[6] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[7] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[8] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*, 2016.

[9] Hugo Larochelle and Geoffrey E Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251, 2010.

[10] Misha Denil, Loris Bazzani, Hugo Larochelle, and Nando de Freitas. Learning where to attend with deep architectures for image tracking. *Neural computation*, 24(8):2151–2184, 2012.

[11] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[13] 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.

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

[15] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[16] Wilson L Taylor. Cloze procedure: a new tool for measuring readability. *Journalism and Mass Communication Quarterly*, 30(4):415, 1953.

[17] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.