Attend and Hop

Tin-Yun Ho
Computer Science
Stanford University
tinyunho@stanford.edu

Jade Huang
Computer Science
Stanford University
jayebird@stanford.edu

Abstract

We propose a novel architecture for natural language inference. On top of a traditional recurrent neural net with attention architecture, we add memory-based modules, residual connections, and richer word embeddings. With these, we are able to achieve 76.6% accuracy.

1 Introduction

Recognizing whether one statement entails another statement is a critical and unsolved challenge in natural language processing central for many industry use cases. Until recently, relevant corpuses were of insufficient size for training neural networks to this task. This changed with the release of the Stanford Natural Language Inference (SNLI) Corpus, which is orders of magnitude larger than other relevant corpuses available. We propose a bidirectional recurrent neural network (RNN) with attention trained using premise and hypothesis pairs to discriminate entailment from contradiction and neutral categorizations. Our novel additions to existing approaches include memory-based modules, residual connections, and richer word embeddings. With these, we are able to achieve 76.6% accuracy.

2 Review of existing work

The use of a (potentially multi-layer) bi-directional RNN layer for natural language inference problems, in particular Gated-Recurrent Unit (GRU) and Long short-term memory (LSTM) architectures, is common in literature, serving as a simple baseline and/or first-layer for most neural network-based architectures that have been published on the SNLI dataset (for example see the first neural network used on SNLI at [1]). In light of this, we also use this as our first layer.

Both above and as a subcomponent of this RNN layer, a large variety of attention mechanisms have been proposed, and the more complex attention mechanisms have also been the key ingredients to the state of the art results recently achieved on SNLI thus far. Broadly speaking (with some exceptions), these models tend to generate separate pooled sentence representations of the hypothesis and premise sentences using attention, and then combine the two sentences in a set of feed-forward layers to get to a final prediction.

Attention was first introduced to SNLI by [11]. Here, the first mechanism used was vanilla attention, which used the last output vector of the LSTM reading the hypothesis sentence, piped through an affine layer and softmax, to create a weighted-average single fixed-vector representation of the premise sentence. The second version of attention used was word-by-word attention, where instead of using a single distribution over all the premise LSTM outputs to generate a pooled average, uses separate distributions for each individual word in the premise to create a pooled representation of each word in the premise, with the representation of the final premise word being selected as the representation for the premise sentence. This second attention model achieves 83.5% on the test set.

The current leading result [2] (with 86.3% accuracy on the test set) has two even more complex attention mechanisms. First, it uses an extension of the LSTM which they call the "Long Short-Term Memory-Network (LSTMN)" layer, which enables any given hidden and cell state to be calculated with respect to not only the immediately prior time step hidden/cell states, but also based on an attention-based pooling of all prior time step hidden/cell states. On top of the resulting RNN output states, they also use a "Deep Fusion Layer", which enables a different distribution over every hidden and cell state of a first 'source' sentence to inform the value taken for the hidden/cell states of a

second 'target' sentence. Besides the attention mechanisms cited here, others have also been used, including attention within sentences or 'inner attention' [8], attention focused on matching alignment between words in the premise and hypothesis, combined with LSTM to create an 'mLSTM' [12], and more.

What unites these approaches, importantly, is that each time a round of attention is applied, it is only done once. Hence, what all of these approaches lack, is an approach to attention in which multiple rounds of attention (and potentially a varying number based on need) can be done over the representation of each sentence, thus allowing each round of attention to focus more specifically on a different aspect of the target sentence outputs and piece together the results sequentially, just as a human might piece facts together one by one (as opposed to all at once). One could imagine as an extension to existing work the specific attention used within the RNN and above the RNN layer could be treated as a hyperparameter in which you slotted in the attention mechanism of your choice, and then after the RNN layer, multiple rounds of attention could be used, creating successfully more accurate/useful representations of the premise and hypothesis sentences.

This alternative approach, which we end up implementing below, is inspired by the 'Dynamic Memory Network' work by [7], in which questions trigger an iterative attention process which allows the model to condition a current round of attention on both the inputs and the result of previous iterations of attention. They found that with such an iterative approach, this caused the attention of each iteration to be significantly more focused on individual facts as opposed to spread out evenly over the entire set of facts (and thus not able to conclude as much). An upgraded version of this model currently holds the state of the art on the facebook bAbi dataset [13], so we decided to try adapting this iterative attention approach to our problem.

One thing we noted in reviewing this work is that in [7], having more attention iterations than necessary tended to degrade performance on simpler tasks (such as for 5 iterations on sentiment analysis). This reminded us of work from computer vision done in a recent ImageNet winning paper [4], in which it was shown that for extremely deep neural networks (in this case on the order of 100+), it was possible for the neural net not only to overfit the training data, but even worse start to perform more and more poorly on training error (not just validation error). This theoretically shouldn't be happening because extraneous layers in a deeper network could just learn to pass previous layers forward by becoming like identity functions, but in practice, this seems very hard to do. So instead, the authors made residual connections between layers, as they found it is easier for higher layers to learn to zero themselves out (so that no residual is added to the existing input) than it is to learn to become like an identity function. To ward off the potential risk of this issue harming us as well, we also took the existing iterative attention model cited above but made the links between each iteration of attention residual.

Finally, in looking at existing SNLI results, it was clear that the winning architectures tended to use as large as possible of vocabularies they could find with the largest word embedding (most use glove 840B 300 dimensional word embeddings), and the current state of the art result not only uses this massive word embedding but trains on top of it, indicating that for them overfit was not a problem even with such a large embedding (some models performing less than current state of the art do not train the embedding). Even more astounding, rather than have an unknown token represent the out of vocabulary words, they create new vectors for each out of vocabulary word and train them from scratch.

Based on this, we hypothesize that if we could create an even richer word embedding than this state of the art, we could improve our results further. Inspired by the idea of creating different word vectors for different senses of the same word from [5], we use an external NLP toolkit called sPaCy to create a different word vector for any given vocabulary word's different part of speech uses. Furthermore, inspired by the idea of not just using a single monolithic unknown token to represent each out of vocabulary word, but to actually use external resources to identify/categorize them more finely from [10], we also use sPaCy to create different a variety of different unknown tokens based on part of speech, entity type, dependency parse labels.

3 Approach

3.1 Data

We utilize the Stanford Natural Language Inference (SNLI) Corpus [1], a collection of 570k, the largest of its kind, human-written English pairs of single sentences manually labeled with the labels *entailment, contradiction,* and *neutral*. As the sentences are based on descriptions of image captions, and thus very life-style domain focused, they do not require as much real-world knowledge of politics/science/etc. to judge the correct label. There are a total of 13,981,666 words with 39,865 unique words.

An example of one sentence entailing another would be *A soccer game with multiple males playing* entails *Some men are playing a sport*. An example of one sentence not entailing another would be *An older and younger man smiling* and *Two men are smiling and laughing at the cats playing on the floor*.

3.2 Architecture

We implement a deep bidirectional RNN with GRU cells in order to capture context from both the left and the right in our sequences of text with multiple layers as well as to capture long-distance dependencies in our text which could be lost in a conventional RNN.

3.2.1 Sentence Encoding Module

First we feed the premise and hypothesis respectively through two independent deep bidirectional RNNs with GRU cells. The bidirectionality of the network allows us to capture context from the left and right of each word in the sequence. Each sentence is fed word-by-word through the RNN, resulting in an encoding of the entire sentence in a single vector. We experimented with pre-initializing the words with pretrained Glove embeddings as well as randomly initializing the words from a normal distribution. In both cases, we updated the embeddings as part of training.

3.2.2 Memory Module

Having fed our premise word-by-word into our deep bidirectional GRU, we then compute multiple rounds of attention in the style of Dynamic Memory Networks. During each iteration, the attention mechanism attends over the word-by-word outputs from the top GRU layers while taking into consideration the sentence encoding of the hypothesis. We utilize the 'general global' form of attention proposed by [9]. This global attentional model considers all hidden states of the premise when deriving the context vector. First, we derive an alignment vector a_p, whose size equals the number of time steps on the premise side, by comparing the resulting hypothesis state $\mathbf{h}_h$ with each premise hidden state $\bar{\mathbf{h}}_p$.

$$\mathbf{a}_p = align(\mathbf{h}_h, \bar{\mathbf{h}}_p) = \frac{exp(score(\mathbf{h}_h^T \mathbf{W}_a \bar{\mathbf{h}}_p))}{\sum_{p'} exp(score(\mathbf{h}_h, \bar{\mathbf{h}}_{p'}))} \tag{1}$$

While Luong et al. explored various options for the $score$ equation, we utilize the "general" global attention formulation where $\mathbf{W}_a$ is a learnable parameter:

$$score(\mathbf{h}_h, \bar{\mathbf{h}}_p) = \mathbf{h}_h^T \mathbf{W}_a \bar{\mathbf{h}}_p \tag{2}$$

We then compute the context vector as a weighted average over all premise hidden states using our alignment vector by computing the dot product between all hidden premise states and their corresponding alignment vector:

$$\mathbf{c}_p = \mathbf{h}_p \mathbf{a}^T \tag{3}$$

When computing multiple rounds of attention, or memory "hops", for hop k (and where we originally started with the hypothesis vector as $\mathbf{m}_0$), we first take the context vector created for that given hop's round of attention (which attends over the premise given the latest round of memory) $\mathbf{c}_k$, concatenate it with an existing memory vector from the prior hop $\mathbf{m}_{k-1}$, and feed it through an affine layer (with a different set of weights for each hop) with a ReLU nonlinearity to produce a new memory based on the combined representation of the premise and earlier memory, $\mathbf{m}_k$. The equations describing this are below

$$\mathbf{m}_0 = \mathbf{h}_h \tag{4}$$

$$\mathbf{a}_k = align(\mathbf{m}_{k-1}, \bar{\mathbf{h}}_p) \tag{5}$$

$$\mathbf{c}_k = \mathbf{h}_p \mathbf{a}_k^T \tag{6}$$

$$\mathbf{m}_k = ReLU([\mathbf{c}_k; \mathbf{m}_{k-1}]\mathbf{W}_k + b_k) \tag{7}$$

When utilizing residual connections (which we use in some architectures and not others), rather than replacing the memory vector at each hop, we add the latest memory vector calculation as a residual $\mathbf{r}_k$ to the prior hop step's memory vector calculation, thus creating a skip connection similar to that found in [4].

$$\mathbf{r}_k = ReLU([\mathbf{c}_k; \mathbf{m}_{k-1}]\mathbf{W}_k + b_k) \tag{8}$$

$$\mathbf{m}_k = \mathbf{m}_{k-1} + \mathbf{r}_k \tag{9}$$

Note that in the above equations, we could essentially have slotted in whatever attention we chose to create our $\mathbf{c}_k$ vector for each hop, but due to limited resources, we stuck with the exact attention formulation above. We also experimented with creating a skip connection by updating our hypothesis sentence vector representation with our attended premise with each hop, but this did not perform as well as the above residual formulation.

$$\mathbf{h}_h^{t+1} = \mathbf{c} + \mathbf{h}_h^t \tag{10}$$

3.2.3 Prediction Module

Having produced a final representation of the premise with attention, memory, and residual connections, we then concatenate this with our hypothesis sentence encoding, and feed this combined representation of our premise and hypothesis through affine layers leading to a softmax cross-entropy loss over our gold standard labels.

3.3 Enriched Vocabulary

In addition to utilizing pretrained Glove word embeddings, we seek to create even richer word embeddings leveraging the spaCy toolkit to differentiate tokens found in our Glove vocabulary by part of speech and further differentiate unknown tokens not found in our Glove vocabulary by part of speech, entity type, and its position in its dependency tree. In doing so, we distinguish our known words which may have many senses with different parts of speech, such as in the case of river *banks*, noun, versus a plane that *bank*s, verb, as it lands. We distinguish our unknown tokens, previously lumped together into one umbrella category of unknowns, by their unique attributes, resulting in a set of rich unknown tokens.

4 Experiments and Results

We evaluate the effectiveness of our model using standard accuracy, precision/recall/F1 measures on the SNLI data. The SNLI corpus is split into a training set of 550,152 sentence pairs, a development set of 10,000 sentence pairs, and a test set of 10,000 sentence pairs. We first removed all sentence pairs with inconclusive labels of "-", i.e. neither entailment nor contradiction nor neutral due to a non-majority vote from the annotators, resulting in a training set of 549,367 sentence pairs, a development set of 9,842 sentence pairs, and a test set of 9,824 sentence pairs. We utilize the development set to select our hyperparameters and only ran our models once on the test set after training all of our models at least 10 epochs and setting our hyperparameter choices in stone.

4.1 Training Details

We train using the SNLI training set and shuffle mini-batches as we proceed where our mini-batches size is 128. Since our data includes variable-length sentences, we utilize a padding token "<PAD>" to pad each of our sentences to the maximum sentence length to expedite the matrix multiplications in our neural network. Our deep bidirectional GRU has two or three layers with 300-dimensional embeddings, resulting in a 600-dimensional embeddings as we concatenate the forward and backward states.

We initialize our parameters using Xavier weight initialization [3]. We experiment with initializing with 300-dimensional Glove 840B word vectors as well as randomly initializing 300-dimensional word vectors with a random normal distribution.

We train for at least 10 epochs using the Adam optimizer [6] with default parameters. Additionally, we utilize L2 loss on the parameters of our affine layers as well as dropout with probability $0.2 - 0.3$ to the inputs and outputs of our bi-GRU layers as well as the outputs of our affine layers. We performed a hyperparameter search using the following learning rates $[1e-2, 1e-3, 3e-4, 1e-4]$ and following l2 regularization $[0.0, 1e-4, 3e-4, 1e-3]$. Our final affine layers leading up to the softmax have a reduced dimensionality of 100.

Our code is implemented in Tensorflow. When running on a g2.2x GPU device on AWS, it takes about 3 hours to iterate over the entire training data once.

4.2 Two-way Classification

We first experimented with converting the original three-way classification task of predicting [entailment, contradiction, neutral] to a two-way classification task of predicting [entailment, not entailment] by merging together the contradiction and neutral classes under one umbrella of *not entailment*.

4.2.1 Baselines

We first implemented the baselines described in Bowman et al. [1] including logistic regression based on a set of six unlexicalized and lexicalized features. The features included BLEU score, length difference between hypothesis and premise, word overlap, unigram and bigram indicators, cross-unigrams, and cross-bigrams. Similarly as described in the paper, we achieved an increasing accuracy as we built up to using all six features. The resulting F1 scores on the dev set and test set are shown in Table 1.

System	**Dev(F1)**	**Test(F1)**
Unlexicalized	0.666	0.667
Unigrams only	0.690	0.696
Lexicalized	0.695	0.695

Table 1: F1 Scores of Lexicalized Classifier on Dev and Test sets

4.2.2 Sentence Encoding Models

Next we moved onto sentence encoding models also described in Bowman et al. with two adjustments: we implemented global general attention from Luong et al. as well as used a GRU instead of an LSTM. Otherwise, our architecture was similar to that used in Bowman et al.: our deep biGRU encodes both the premise and hypothesis separately, then we concatenate both representations and feed forward the combined representation through 3 affine layers leading up to a softmax layer. For both of these models, we trained word embeddings from scratch.

System	**Train (% acc)**	**Dev (% acc)**
Attention + 2-layer biGRU	82.785	80.3886
Attention + 2-layer biGRU + 2 hops	85.612	84.034

Table 2: Experimental results on two-way classification task.

As expected, this deeper model achieved higher accuracy than our baseline classifiers with hand-crafted features which achieved around 70% accuracy on average. We saw that when we added two rounds of attention, our model's accuracy increased on both the train and development set, showing that even adding one additional round of attention over the premise with respect to the hypothesis allowed our model a gain in accuracy.

4.3 Three-way Classification

Encouraged by the positive results of adding multiple rounds of attention, we then moved onto the original three-way classification problem of distinguishing between [entailment, contradiction, neutral].

For our three-way experiments, we utilized Glove 840B 300d to pre-initialize our word embeddings and trained these embeddings throughout training. Our bidirectional GRU has two layers, and we used a consistent learning rate of $1e-4$ and l2 regularization on affine layers of $1e-4$. For each of our models, we utilized general global attention. We created our enriched vocabulary using Glove 6B 300d pretrained vectors.

For most of our models, the gaps between the train, development, and test accuracies are quite small, suggesting that our model is able to learn from the train set and generalize well over the unseen development set without overfitting the train set. It is troubling that for the two models using hypothesis residual connections of always adding the hypothesis representation to our latest attended premise vector, the test accuracy is significantly lower than that of the train and development set, suggesting that those particular models were unable to generalize to the unseen test set.

Based on a typical confusion matrix (where the values are percentage of total samples) like the one below for the final model with 3 hops, we see that the neutral class is relatively similarly likely to be confused with both the entailment and contradiction classees. And as expected, the model is more likely to incorrectly predict that a sample is labeled

System	Train (% acc)	Dev (% acc)	Test (% acc)
300d biGRU RNN	77.4757	75.87	75.49
300d biGRU RNN + 3 hops	78.006	76.367	76.038
300d biGRU RNN + 3 hops + hypothesis residual	77.318	75.9868	58.85
300d biGRU RNN + 5 hops + hypothesis residual + enriched	75.56	75.678	60.65995
300d biGRU RNN + 3 hops + memory residual + enriched	77.34	76.7	**76.63**

Table 3: Experimental results on three-way classification task.

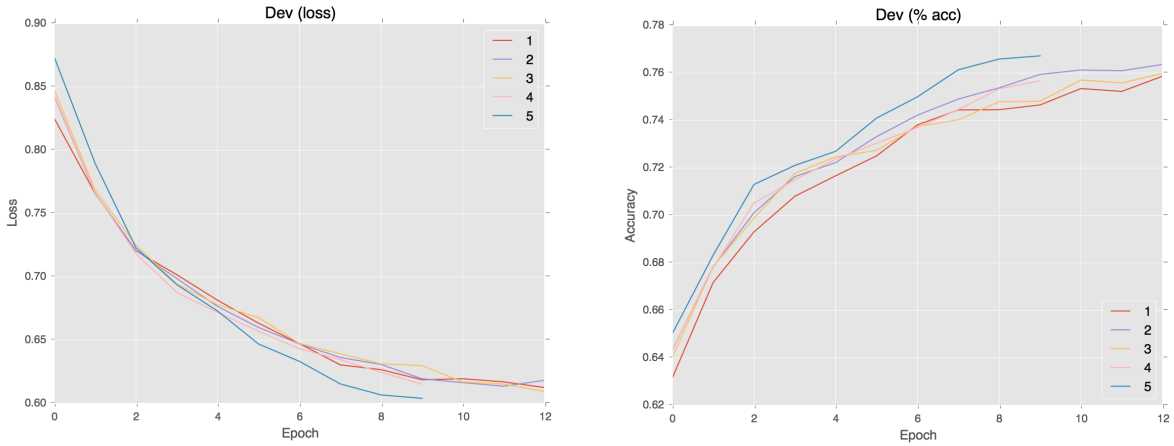

Figure 1: Loss and accuracy curves on the development set. The numbers in the legend correspond to the experiments listed in order in Table 3.

entailment when it is actually neutral than to incorrectly predict that the sample is labeled entailment when it is actually contradicting, with a similar relationship holding when looking at incorrect predictions for contradiction.

Confusion matrix (% of total)	Predict Entail	Predict Neutral	Predict Contradict
Labeled Entail	30%	3%	2%
Labeled Neutral	6%	23%	4%
Labeled Contradict	4%	4%	24%

Table 4: Confusion matrix for our best-performing model on the test set.

From the F1 metrics table for the final model with 3 hops below, we see that there is not a significant divergence between total precision/recall/F1 scores nor average F1 scores for the 3 classes. However, we do see that the model is much more likely to be precise about its contradictions, and much better at recall on its entailment labels, with the final outcome being that its F1 score for entailment is slightly higher.

Classification metrics	Precision	Recall	F1
Entail	.74	.86	.79
Neutral	.75	.69	.72
Contradict	.82	.74	.78
Total	.77	.77	.77

Table 5: Precision, Recall, and F1 scores for our best-performing model on the test set.

Finally, when looking at some examples of what our best model gets wrong, in some cases, it just hasn't learned what certain words mean or that certain pairs of words are analogous/contradicting. However, in a surprising number of cases, the problem itself requires a broader intuitive sense of the world not derivable from the sentences themselves which our model also does not have (as in the example of the girls with dresses).

- **Premise:** Group of young adults posing for picture near spanish - language sign .
  Hypothesis: The people are tourists .
  Prediction: contradiction
  Gold Label: entailment
- **Premise:** Two dirt bike riders , one wearing green and the wearing blue and white , are jumping a hill .
  Hypothesis: Two guys are driving buses
  Prediction: entailment
  Gold Label: neutral
- **Premise:** A lady is helping another woman work in a silver compartment , which is most likely related to nurse - work .
  Hypothesis: Two woman are trying to finish orders from a doctor
  Prediction: neutral
  Gold Label: contradiction
- **Premise:** People are on an escalator waiting to get to their destination while looking outside of the glass that makes up the wall .
  Hypothesis: People are riding on the escalator .
  Prediction: contradiction
  Gold Label: neutral
- **Premise:** Girls dressed in red stand in a line .
  Hypothesis: It is a special day .
  Prediction: contradiction
  Gold Label: entailment

4.4 Hops

Looking closer at our attend and hop mechanism with our model initialized with Glove vectors with 3 rounds of global attention, we can see that the way our model performs attention, it is nearly akin to a human reading left-to-right. Furthermore, while during the first round of attention, our "<PAD>" characters receive some amount of attention weight, during the second and third round, our model learns to zero these out, seemingly recognizing that the "<PAD>" characters are irrelevant to the verdict of [entailment, neutral, contradiction].

Looking at a specific example, we have the following premise and hypothesis:

Premise: Woman with four children with <unk> faces outside .
Hypothesis: Kids have their faces <unk> like animals .

First as we look at the alignment vector of weights, the vector of scores corresponding to the premise vector which we compute in comparison with the hypothesis vector. As we proceed from hop to hop, the alignment weights shift in magnitude from left to right, as if reading left to right. In the first hop, we mainly focus on *Woman*, the second hop *children* and *outside*, and the third hop *faces* and *outside*.

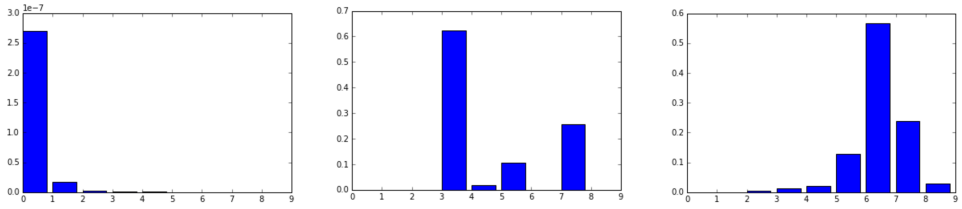

Figure 2: Alignment weights (y-axis) over the premise (x-axis) of *Woman with four children with <unk> faces outside*, hops increasing from left to right from one to three.

Upon examination of the complete alignment vector including the padded characters, we see that in the first hop, the padded characters, placed before the the actual premise text, receive some attention from the alignment scores. However, as we move to more hops, we begin to ignore the padded characters and give zero alignment weights to them, hopefully indicating that our model learns that padded characters are not essential to our final objective.

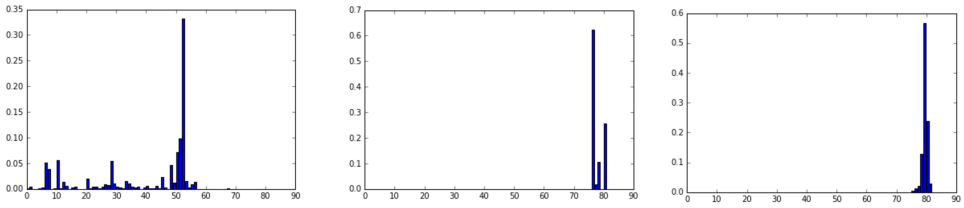

Figure 3: Alignment weights (y-axis) over the premise (x-axis) of *Woman with four children with <unk> faces outside* including padding characters placed in the beginning of the sentence, hops increasing from left to right from one to three.

5 Conclusion

With our limited time and resources, we never got to test out the full barrage of things we wanted to test. We even had to end our existing experiments early because we ran out of time/money (our best model was actually continuing to improve on dev loss/accuracy when we stopped it). We were hoping to begin with just general attention and hops, but move on quickly to testing out richer types of attention in the hops, and potentially even different types of attention in each hop. The initial bi-GRU layer as well was only a baseline, which in the future we'd like to replace with richer RNN structures. In the future, we would hope to be able to try out (with more financial support) these other experiments we never got around to.

References

[1] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[2] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *CoRR*, abs/1601.06733, 2016.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[5] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *ACL*, 2012.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[7] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015.

[8] Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090, 2016.

[9] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[10] Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *CoRR*, abs/1410.8206, 2014.

[11] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664, 2015.

[12] Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. *CoRR*, abs/1512.08849, 2015.

[13] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. *CoRR*, abs/1603.01417, 2016.