Long Short-Term Memory-Networks for Machine Reading

Long Short-Term Memory-Networks for Machine Reading

Jianpeng Cheng, Li Dong and **Mirella Lapata**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
{jianpeng.cheng,li.dong}@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we address the question of how to render sequence-level networks better at handling structured input. We propose a machine reading simulator which processes text incrementally from left to right and performs shallow reasoning with memory and attention. The reader extends the Long Short-Term Memory architecture with a memory network in place of a single memory cell. This enables adaptive memory usage during recurrence with neural attention, offering a way to weakly induce relations among tokens. The system is initially designed to process a single sequence but we also demonstrate how to integrate it with an encoder-decoder architecture. Experiments on language modeling, sentiment analysis, and natural language inference show that our model matches or outperforms the state of the art.

1 Introduction

How can a sequence-level network induce relations which are presumed latent during text processing? How can a recurrent network attentively memorize longer sequences in a way that humans do? In this paper we design a machine reader that automatically learns to understand text. The term machine reading is related to a wide range of tasks from answering reading comprehension questions (Clark et al., 2013), to fact and relation extraction (Etzioni et al., 2011; Fader et al., 2011), ontology learning (Poon and Domingos, 2010), and textual entailment (Dagan et al., 2005). Rather than focusing on a specific task, we develop a general-purpose reading simulator, drawing inspiration from human language processing and the fact language comprehension is incremental with readers continuously extracting the meaning of utterances on a word-by-word basis.

In order to understand texts, our machine reader should provide facilities for extracting and representing meaning from natural language text, storing meanings internally, and working with stored meanings to derive further consequences. Ideally, such a system should be robust, open-domain, and degrade gracefully in the presence of semantic representations which may be incomplete, inaccurate, or incomprehensible. It would also be desirable to simulate the behavior of English speakers who process text sequentially, from left to right, fixating nearly every word while they read (Rayner, 1998) and creating partial representations for sentence prefixes (Konieczny, 2000; Tanenhaus et al., 1995).

Language modeling tools such as recurrent neural networks (RNN) bode well with human reading behavior (Frank and Bod, 2011). RNNs treat each sentence as a sequence of words and recursively compose each word with its previous *memory*, until the meaning of the whole sentence has been derived. In practice, however, sequence-level networks are met with at least three challenges. The first one concerns model training problems associated with vanishing and exploding gradients (Hochreiter, 1991; Bengio et al., 1994), which can be partially ameliorated with gated activation functions, such as the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), and gradient clipping (Pascanu et al., 2013). The second issue relates to memory compression problems. As the input sequence gets compressed and blended into a single dense vector, suf-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 551–561,
Austin, Texas, November 1-5, 2016.

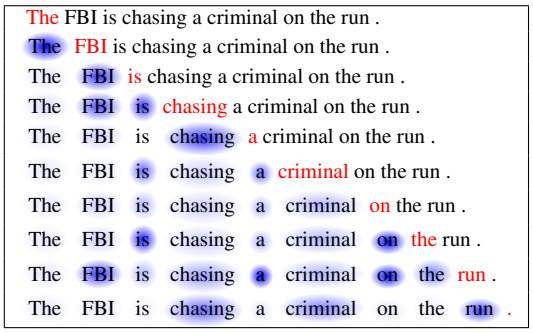


Figure 1: Illustration of our model while reading the sentence *The FBI is chasing a criminal on the run.* Color *red* represents the current word being fixated, *blue* represents memories. Shading indicates the degree of memory activation.

ficiently large memory capacity is required to store past information. As a result, the network generalizes poorly to long sequences while wasting memory on shorter ones. Finally, it should be acknowledged that sequence-level networks lack a mechanism for handling the structure of the input. This imposes an inductive bias which is at odds with the fact that language has inherent structure. In this paper, we develop a text processing system which addresses these limitations while maintaining the incremental, generative property of a recurrent language model.

Recent attempts to render neural networks more structure aware have seen the incorporation of external memories in the context of recurrent neural networks (Weston et al., 2015; Sukhbaatar et al., 2015; Grefenstette et al., 2015). The idea is to use multiple memory slots outside the recurrence to piece-wise store representations of the input; read and write operations for each slot can be modeled as an attention mechanism with a recurrent controller. We also leverage memory and attention to empower a recurrent network with stronger memorization capability and more importantly the ability to discover relations among tokens. This is realized by inserting a memory network module in the update of a recurrent network together with attention for memory addressing. The attention acts as a weak inductive module discovering relations between input tokens, and is trained without direct supervision. As a point of departure from previous work, the memory network we employ is internal to the recurrence, thus strengthening the interaction of the two and leading to a representation learner which is able to reason over shallow structures. The resulting model, which we term Long Short-Term Memory-Network (LSTMN), is a reading simulator that can be used for sequence processing tasks.

Figure 1 illustrates the reading behavior of the LSTMN. The model processes text incrementally while learning which past tokens in the memory and to what extent they relate to the current token being processed. As a result, the model induces undirected relations among tokens as an intermediate step of learning representations. We validate the performance of the LSTMN in language modeling, sentiment analysis, and natural language inference. In all cases, we train LSTMN models end-to-end with task-specific supervision signals, achieving performance comparable or better to state-of-the-art models and superior to vanilla LSTMs.

2 Related Work

Our machine reader is a recurrent neural network exhibiting two important properties: it is incremental, simulating human behavior, and performs shallow structure reasoning over input streams.

Recurrent neural network (RNNs) have been successfully applied to various sequence modeling and sequence-to-sequence transduction tasks. The latter have assumed several guises in the literature such as machine translation (Bahdanau et al., 2014), sentence compression (Rush et al., 2015), and reading comprehension (Hermann et al., 2015). A key contributing factor to their success has been the ability to handle well-known problems with exploding or vanishing gradients (Bengio et al., 1994), leading to models with gated activation functions (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), and more advanced architectures that enhance the information flow within the network (Koutník et al., 2014; Chung et al., 2015; Yao et al., 2015).

A remaining practical bottleneck for RNNs is memory compression (Bahdanau et al., 2014): since the inputs are recursively combined into a single memory representation which is typically too small in terms of parameters, it becomes difficult to accurately memorize sequences (Zaremba and Sutskever, 2014). In the encoder-decoder architecture, this problem can be sidestepped with an attention mechanism which learns soft alignments *between* the decoding states and the encoded memories (Bahdanau

et al., 2014). In our model, memory and attention are added *within* a sequence encoder allowing the network to uncover lexical relations between tokens.

The idea of introducing a structural bias to neural models is by no means new. For example, it is reflected in the work of Socher et al. (2013a) who apply recursive neural networks for learning natural language representations. In the context of recurrent neural networks, efforts to build modular, structured neural models date back to Das et al. (1992) who connect a recurrent neural network with an external memory stack for learning context free grammars. Recently, Weston et al. (2015) propose Memory Networks to explicitly segregate memory storage from the computation of neural networks in general. Their model is trained end-to-end with a memory addressing mechanism closely related to soft attention (Sukhbaatar et al., 2015) and has been applied to machine translation (Meng et al., 2015). Grefenstette et al. (2015) define a set of differentiable data structures (stacks, queues, and dequeues) as memories controlled by a recurrent neural network. Tran et al. (2016) combine the LSTM with an external memory block component which interacts with its hidden state. Kumar et al. (2016) employ a structured neural network with episodic memory modules for natural language and also visual question answering (Xiong et al., 2016).

Similar to the above work, we leverage memory and attention in a recurrent neural network for inducing relations between tokens as a module in a larger network responsible for representation learning. As a property of soft attention, all intermediate relations we aim to capture are soft and differentiable. This is in contrast to shift-reduce type neural models (Dyer et al., 2015; Bowman et al., 2016) where the intermediate decisions are hard and induction is more difficult. Finally, note that our model captures undirected lexical relations and is thus distinct from work on dependency grammar induction (Klein and Manning, 2004) where the learned head-modifier relations are directed.

3 The Machine Reader

In this section we present our machine reader which is designed to process structured input while retaining the incrementality of a recurrent neural network. The core of our model is a Long Short-Term Memory (LSTM) unit with an extended memory tape that explicitly simulates the human memory span. The model performs implicit relation analysis between tokens with an attention-based memory addressing mechanism at every time step. In the following, we first review the standard Long Short-Term Memory and then describe our model.

3.1 Long Short-Term Memory

A Long Short-Term Memory (LSTM) recurrent neural network processes a variable-length sequence $x = (x_1, x_2, \cdots, x_n)$ by incrementally adding new content into a single memory slot, with gates controlling the extent to which new content should be memorized, old content should be erased, and current content should be exposed. At time step t, the memory c_t and the hidden state h_t are updated with the following equations:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t] \tag{1}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{2}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3}$$

where i, f, and o are gate activations. Compared to the standard RNN, the LSTM uses additive memory updates and it separates the memory c from the hidden state h, which interacts with the environment when making predictions.

3.2 Long Short-Term Memory-Network

The first question that arises with LSTMs is the extent to which they are able to memorize sequences under recursive compression. LSTMs can produce a list of state representations during composition, however, the next state is always computed from the current state. That is to say, given the current state h_t, the next state h_{t+1} is conditionally independent of states $h_1 \cdots h_{t-1}$ and tokens $x_1 \cdots x_t$. While the recursive state update is performed in a Markov manner, it is assumed that LSTMs maintain unbounded memory (i.e., the current state alone summarizes well the tokens it has seen so far). This assumption may fail in practice, for example when the sequence is long

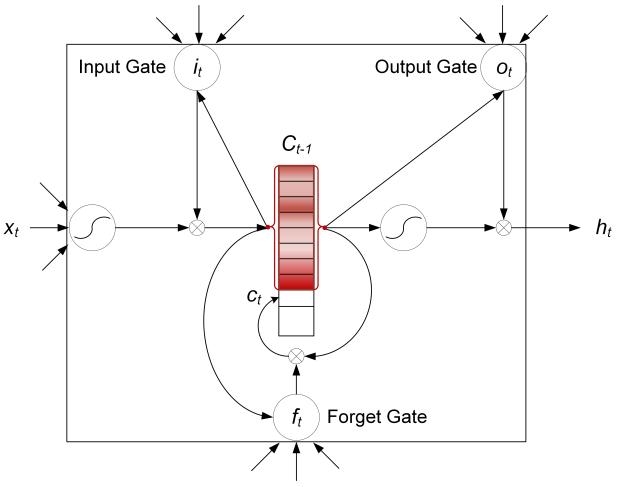


Figure 2: Long Short-Term Memory-Network. Color indicates degree of memory activation.

or when the memory size is not large enough. Another undesired property of LSTMs concerns modeling structured input. An LSTM aggregates information on a token-by-token basis in sequential order, but there is no explicit mechanism for reasoning over structure and modeling relations between tokens.

Our model aims to address both limitations. Our solution is to modify the standard LSTM structure by replacing the memory cell with a memory network (Weston et al., 2015). The resulting Long Short-Term Memory-Network (LSTMN) stores the contextual representation of each input token with a unique memory slot and the size of the memory grows with time until an upper bound of the memory span is reached. This design enables the LSTM to reason about relations between tokens with a neural attention layer and then perform non-Markov state updates. Although it is feasible to apply both write and read operations to the memories with attention, we concentrate on the latter. We conceptualize the *read* operation as attentively linking the current token to previous memories and selecting useful content when processing it. Although not the focus of this work, the significance of the *write* operation can be analogously justified as a way of incrementally updating previous memories, e.g., to correct wrong interpretations when processing garden path sentences (Ferreira and Henderson, 1991).

The architecture of the LSTMN is shown in Figure 2 and the formal definition is provided as follows. The model maintains two sets of vectors stored in a hidden state tape used to interact with the environment (e.g., computing attention), and a memory tape used to represent what is actually stored in memory.[1] Therefore, each token is associated with a hidden vector and a memory vector. Let x_t denote the current input; $C_{t-1} = (c_1, \cdots, c_{t-1})$ denotes the current memory tape, and $H_{t-1} = (h_1, \cdots, h_{t-1})$ the previous hidden tape. At time step t, the model computes the relation between x_t and $x_1 \cdots x_{t-1}$ through $h_1 \cdots h_{t-1}$ with an attention layer:

$$a_i^t = v^{\mathrm{T}} \tanh(W_h h_i + W_x x_t + W_{\tilde{h}} \tilde{h}_{t-1}) \quad (4)$$

$$s_i^t = \mathrm{softmax}(a_i^t) \quad (5)$$

This yields a probability distribution over the hidden state vectors of previous tokens. We can then compute an adaptive summary vector for the previous hidden tape and memory tape denoted by $\tilde{c}_t$ and $\tilde{h}_t$, respectively:

$$\begin{bmatrix} \tilde{h}_t \\ \tilde{c}_t \end{bmatrix} = \sum_{i=1}^{t-1} s_i^t \cdot \begin{bmatrix} h_i \\ c_i \end{bmatrix} \quad (6)$$

and use them for computing the values of c_t and h_t in the recurrent update as:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [\tilde{h}_t, x_t] \quad (7)$$

$$c_t = f_t \odot \tilde{c}_t + i_t \odot \hat{c}_t \quad (8)$$

$$h_t = o_t \odot \tanh(c_t) \quad (9)$$

where v, W_h, W_x and $W_{\tilde{h}}$ are the new weight terms of the network.

A key idea behind the LSTMN is to use attention for inducing relations between tokens. These relations are soft and differentiable, and components of a larger representation learning network. Although it is appealing to provide direct supervision for the attention layer, e.g., with evidence collected from a dependency treebank, we treat it as a submodule being optimized within the larger network in a downstream task. It is also possible to have a more structured relational reasoning module by stacking multiple memory and hidden layers in an alternating fashion, resembling a stacked LSTM (Graves,

[1] For comparison, LSTMs maintain a hidden vector and a memory vector; memory networks (Weston et al., 2015) have a set of key vectors and a set of value vectors.

2013) or a multi-hop memory network (Sukhbaatar et al., 2015). This can be achieved by feeding the output h_t^k of the lower layer k as input to the upper layer $(k+1)$. The attention at the $(k+1)$th layer is computed as:

$$a_{i,k+1}^t = v^{\mathrm{T}} \tanh(W_h h_i^{k+1} + W_l h_t^k + W_{\tilde{h}} \tilde{h}_{t-1}^{k+1}) \quad (10)$$

Skip-connections (Graves, 2013) can be applied to feed x_t to upper layers as well.

4 Modeling Two Sequences with LSTMN

Natural language processing tasks such as machine translation and textual entailment are concerned with modeling two sequences rather than a single one. A standard tool for modeling two sequences with recurrent networks is the encoder-decoder architecture where the second sequence (also known as the *target*) is being processed conditioned on the first one (also known as the *source*). In this section we explain how to combine the LSTMN which applies attention for intra-relation reasoning, with the encoder-decoder network whose attention module learns the inter-alignment between two sequences. Figures 3a and 3b illustrate two types of combination. We describe the models more formally below.

Shallow Attention Fusion Shallow fusion simply treats the LSTMN as a separate module that can be readily used in an encoder-decoder architecture, in lieu of a standard RNN or LSTM. As shown in Figure 3a, both encoder and decoder are modeled as LSTMNs with intra-attention. Meanwhile, inter-attention is triggered when the decoder reads a target token, similar to the inter-attention introduced in Bahdanau et al. (2014).

Deep Attention Fusion Deep fusion combines inter- and intra-attention (initiated by the decoder) when computing state updates. We use different notation to represent the two sets of attention. Following Section 3.2, C and H denote the target memory tape and hidden tape, which store representations of the target symbols that have been processed so far. The computation of intra-attention follows Equations (4)–(9). Additionally, we use $A = [\alpha_1, \cdots, \alpha_m]$ and $Y = [\gamma_1, \cdots, \gamma_m]$ to represent the source memory tape and hidden tape, with m being the length of the source sequence conditioned upon. We compute inter-attention between the input at time step t and tokens in the entire source sequence as follows:

$$b_j^t = u^{\mathrm{T}} \tanh(W_\gamma \gamma_j + W_x x_t + W_{\tilde{\gamma}} \tilde{\gamma}_{t-1}) \quad (11)$$

$$p_j^t = \mathrm{softmax}(b_j^t) \quad (12)$$

After that we compute the adaptive representation of the source memory tape $\tilde{\alpha}_t$ and hidden tape $\tilde{\gamma}_t$ as:

$$\begin{bmatrix} \tilde{\gamma}_t \\ \tilde{\alpha}_t \end{bmatrix} = \sum_{j=1}^{m} p_j^t \cdot \begin{bmatrix} \gamma_j \\ \alpha_j \end{bmatrix} \quad (13)$$

We can then transfer the adaptive source representation $\tilde{\alpha}_t$ to the target memory with another gating operation r_t, analogous to the gates in Equation (7).

$$r_t = \sigma(W_r \cdot [\tilde{\gamma}_t, x_t]) \quad (14)$$

The new target memory includes inter-alignment $r_t \odot \tilde{\alpha}_t$, intra-relation $f_t \odot \tilde{c}_t$, and the new input information $i_t \odot \hat{c}_t$:

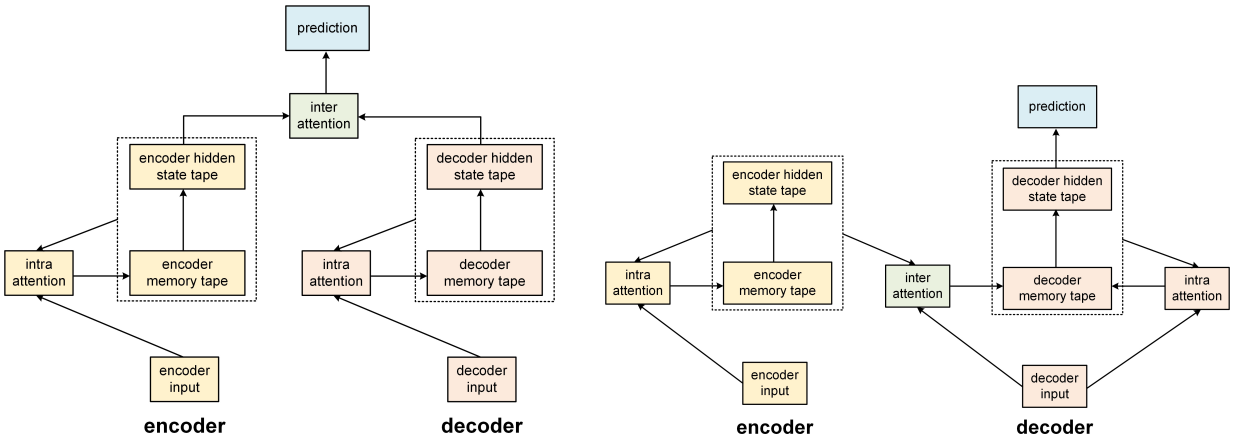
$$c_t = r_t \odot \tilde{\alpha}_t + f_t \odot \tilde{c}_t + i_t \odot \hat{c}_t \quad (15)$$

$$h_t = o_t \odot \tanh(c_t) \quad (16)$$

As shown in the equations above and Figure 3b, the major change of deep fusion lies in the recurrent storage of the inter-alignment vector in the target memory network, as a way to help the target network review source information.

5 Experiments

In this section we present our experiments for evaluating the performance of the LSTMN machine reader. We start with language modeling as it is a natural testbed for our model. We then assess the model's ability to extract meaning representations for generic sentence classification tasks such as sentiment analysis. Finally, we examine whether the LSTMN can recognize the semantic relationship between two sentences by applying it to a natural language inference task. Our code is available at `https://github.com/cheng6076/SNLI-attention`.



(a) Decoder with shallow attention fusion.

(b) Decoder with deep attention fusion.

Figure 3: LSTMNs for sequence-to-sequence modeling. The encoder uses intra-attention, while the decoder incorporates both intra- and inter-attention. The two figures present two ways to combine the intra- and inter-attention in the decoder.

Models	Layers	Perplexity
KN5	—	141
RNN	1	129
LSTM	1	115
LSTMN	1	**108**
sLSTM	3	115
gLSTM	3	107
dLSTM	3	109
LSTMN	3	**102**

Table 1: Language model perplexity on the Penn Treebank. The size of memory is 300 for all models.

5.1 Language Modeling

Our language modeling experiments were conducted on the English Penn Treebank dataset. Following common practice (Mikolov et al., 2010), we trained on sections 0–20 (1M words), used sections 21–22 for validation (80K words), and sections 23–24 (90K words for testing). The dataset contains approximately 1 million tokens and a vocabulary size of 10K. The average sentence length is 21. We use perplexity as our evaluation metric: $PPL = \exp(NLL/T)$, where NLL denotes the negative log likelihood of the entire test set and T the corresponding number of tokens. We used stochastic gradient descent for optimization with an initial learning rate of 0.65, which decays by a factor of 0.85 per epoch if no significant improvement has been observed on the validation set. We renormalize the gradient if its norm is greater than 5. The mini-batch size was set to 40. The dimensions of the word embeddings were set to 150 for all models.

In this suite of experiments we compared the LSTMN against a variety of baselines. The first one is a Kneser-Ney 5-gram language model (KN5) which generally serves as a non-neural baseline for the language modeling task. We also present perplexity results for the standard RNN and LSTM models. We also implemented more sophisticated LSTM architectures, such as a stacked LSTM (sLSTM), a gated-feedback LSTM (gLSTM; Chung et al. (2015)) and a depth-gated LSTM (dLSTM; Yao et al. (2015)). The gated-feedback LSTM has feedback gates connecting the hidden states across multiple time steps as an adaptive control of the information flow. The depth-gated LSTM uses a depth gate to connect memory cells of vertically adjacent layers. In general, both gLSTM and dLSTM are able to capture long-term dependencies to some degree, but they do not explicitly keep past memories. We set the number of layers to 3 in this experiment, mainly to agree with the language modeling experiments of Chung et al. (2015). Also note that that there are no single-layer variants for gLSTM and dLSTM; they have to be implemented as multi-layer systems. The hidden unit size of the LSTMN and all comparison models (except KN5) was set to 300.

The results of the language modeling task are shown in Table 1. Perplexity results for KN5 and RNN are taken from Mikolov et al. (2015). As can be seen, the single-layer LSTMN outperforms these

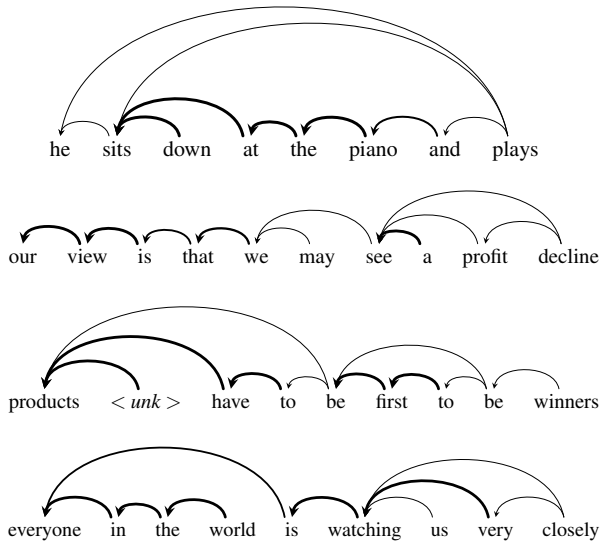


Figure 4: Examples of intra-attention (language modeling). Bold lines indicate higher attention scores. Arrows denote which word is being focused when attention is computed, but not the direction of the relation.

two baselines and the LSTM by a significant margin. Amongst all deep architectures, the three-layer LSTMN also performs best. We can study the memory activation mechanism of the machine reader by visualizing the attention scores. Figure 4 shows four sentences sampled from the Penn Treebank validation set. Although we explicitly encourage the reader to attend to any memory slot, much attention focuses on recent memories. This agrees with the linguistic intuition that long-term dependencies are relatively rare. As illustrated in Figure 4 the model captures some valid lexical relations (e.g., the dependency between *sits* and *at*, *sits* and *plays*, *everyone* and *is*, *is* and *watching*). Note that arcs here are undirected and are different from the directed arcs denoting head-modifier relations in dependency graphs.

5.2 Sentiment Analysis

Our second task concerns the prediction of sentiment labels of sentences. We used the Stanford Sentiment Treebank (Socher et al., 2013a), which contains fine-grained sentiment labels (very positive, positive, neutral, negative, very negative) for 11,855 sentences. Following previous work on this dataset, we used 8,544 sentences for training, 1,101 for validation, and 2,210 for testing. The average sentence length is 19.1. In addition, we also performed a binary classification task (positive, negative) after removing the neutral label. This resulted in 6,920 sentences for training, 872 for validation and 1,821 for testing. Table 2 reports results on both fine-grained and binary classification tasks.

Models	Fine-grained	Binary
RAE (Socher et al., 2011)	43.2	82.4
RNTN (Socher et al., 2013b)	45.7	85.4
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
DCNN (Blunsom et al., 2014)	48.5	86.8
CNN-MC (Kim, 2014)	48.0	88.1
T-CNN (Lei et al., 2015)	**51.2**	**88.6**
PV (Le and Mikolov, 2014)	48.7	87.8
CT-LSTM (Tai et al., 2015)	51.0	88.0
LSTM (Tai et al., 2015)	46.4	84.9
2-layer LSTM (Tai et al., 2015)	46.0	86.3
LSTMN	**47.6**	**86.3**
2-layer LSTMN	**47.9**	**87.0**

Table 2: Model accuracy (%) on the Sentiment Treebank (test set). The memory size of LSTMN models is set to 168 to be compatible with previously published LSTM variants (Tai et al., 2015).

We experimented with 1- and 2-layer LSTMNs. For the latter model, we predict the sentiment label of the sentence based on the averaged hidden vector passed to a 2-layer neural network classifier with ReLU as the activation function. The memory size for both LSTMN models was set to 168 to be compatible with previous LSTM models (Tai et al., 2015) applied to the same task. We used pretrained `300-D Glove 840B` vectors (Pennington et al., 2014) to initialize the word embeddings. The gradient for words with `Glove` embeddings, was scaled by 0.35 in the first epoch after which all word embeddings were updated normally.

We used Adam (Kingma and Ba, 2015) for optimization with the two momentum parameters set to 0.9 and 0.999 respectively. The initial learning rate was set to 2E-3. The regularization constant was 1E-4 and the mini-batch size was 5. A dropout rate of 0.5 was applied to the neural network classifier.

We compared our model with a wide range of top-performing systems. Most of these models (including ours) are LSTM variants (third block in Table 2), recursive neural networks (first block), or convolu-

tional neural networks (CNNs; second block). Recursive models assume the input sentences are represented as parse trees and can take advantage of annotations at the phrase level. LSTM-type models and CNNs are trained on sequential input, with the exception of CT-LSTM (Tai et al., 2015) which operates over tree-structured network topologies such as constituent trees. For comparison, we also report the performance of the paragraph vector model (PV; Le and Mikolov (2014); see Table 2, second block) which neither operates on trees nor sequences but learns distributed document representations parameterized directly.

The results in Table 2 show that both 1- and 2-layer LSTMNs outperform the LSTM baselines while achieving numbers comparable to state of the art. The number of layers for our models was set to be comparable to previously published results. On the fine-grained and binary classification tasks our 2-layer LSTMN performs close to the best system T-CNN (Lei et al., 2015). Figure 5 shows examples of intra-attention for sentiment words. Interestingly, the network learns to associate sentiment important words such as *though* and *fantastic* or *not* and *good*.

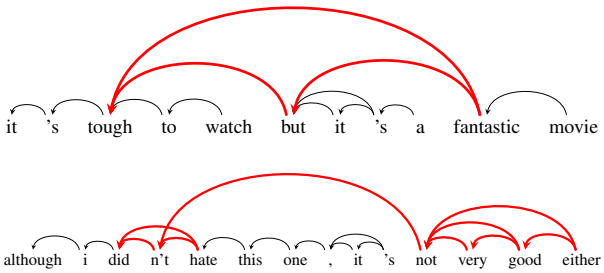


Figure 5: Examples of intra-attention (sentiment analysis). Bold lines (red) indicate attention between sentiment important words.

5.3 Natural Language Inference

The ability to reason about the semantic relationship between two sentences is an integral part of text understanding. We therefore evaluate our model on recognizing textual entailment, i.e., whether two premise-hypothesis pairs are entailing, contradictory, or neutral. For this task we used the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015), which contains premise-hypothesis pairs and target labels indicating their relation. After removing sentences with unknown labels, we end up with 549,367 pairs for training, 9,842 for development and 9,824 for testing. The vocabulary size is 36,809 and the average sentence length is 22. We performed lower-casing and tokenization for the entire dataset.

Recent approaches use two sequential LSTMs to encode the premise and the hypothesis respectively, and apply neural attention to reason about their logical relationship (Rocktäschel et al., 2016; Wang and Jiang, 2016). Furthermore, Rocktäschel et al. (2016) show that a non-standard encoder-decoder architecture which processes the hypothesis conditioned on the premise results significantly boosts performance. We use a similar approach to tackle this task with LSTMNs. Specifically, we use two LSTMNs to read the premise and hypothesis, and then match them by comparing their hidden state tapes. We perform average pooling for the hidden state tape of each LSTMN, and concatenate the two averages to form the input to a 2-layer neural network classifier with ReLU as the activation function.

We used pre-trained `300-D Glove 840B` vectors (Pennington et al., 2014) to initialize the word embeddings. Out-of-vocabulary (OOV) words were initialized randomly with Gaussian samples (μ=0, σ=1). We only updated OOV vectors in the first epoch, after which all word embeddings were updated normally. The dropout rate was selected from [0.1, 0.2, 0.3, 0.4]. We used Adam (Kingma and Ba, 2015) for optimization with the two momentum parameters set to 0.9 and 0.999 respectively, and the initial learning rate set to 1E-3. The mini-batch size was set to 16 or 32. For a fair comparison against previous work, we report results with different hidden/memory dimensions (i.e., 100, 300, and 450).

We compared variants of our model against different types of LSTMs (see the second block in Table 3). Specifically, these include a model which encodes the premise and hypothesis independently with two LSTMs (Bowman et al., 2015), a shared LSTM (Rocktäschel et al., 2016), a word-by-word attention model (Rocktäschel et al., 2016), and a matching LSTM (mLSTM; Wang and Jiang (2016)). This model sequentially processes the hypothesis, and at each position tries to match the current word with an attention-weighted representation of the premise (rather than basing its predictions on whole sentence embeddings). We also compared our mod-

Models	h	$\|\theta\|_M$	Test
BOW concatenation	—	—	59.8
LSTM (Bowman et al., 2015)	100	221k	77.6
LSTM-att (Rocktäschel et al., 2016)	100	252k	83.5
mLSTM (Wang and Jiang, 2016)	300	1.9M	86.1
LSTMN	100	260k	81.5
LSTMN shallow fusion	100	280k	84.3
LSTMN deep fusion	100	330k	84.5
LSTMN shallow fusion	300	1.4M	85.2
LSTMN deep fusion	300	1.7M	85.7
LSTMN shallow fusion	450	2.8M	86.0
LSTMN deep fusion	450	3.4M	**86.3**

Table 3: Parameter counts $|\theta|_M$, size of hidden unit h, and model accuracy (%) on the natural language inference task.

els with a bag-of-words baseline which averages the pre-trained embeddings for the words in each sentence and concatenates them to create features for a logistic regression classifier (first block in Table 3).

LSTMNs achieve better performance compared to LSTMs (with and without attention; 2nd block in Table 3). We also observe that fusion is generally beneficial, and that deep fusion slightly improves over shallow fusion. One explanation is that with deep fusion the inter-attention vectors are recurrently memorized by the decoder with a gating operation, which also improves the information flow of the network. With standard training, our deep fusion yields the state-of-the-art performance in this task. Although encouraging, this result should be interpreted with caution since our model has substantially more parameters compared to related systems. We could compare different models using the same number of total parameters. However, this would inevitably introduce other biases, e.g., the number of hyper-parameters would become different.

6 Conclusions

In this paper we proposed a machine reading simulator to address the limitations of recurrent neural networks when processing inherently structured input. Our model is based on a Long Short-Term Memory architecture embedded with a memory network, explicitly storing contextual representations of input tokens without recursively compressing them. More importantly, an intra-attention mechanism is employed for memory addressing, as a way to induce undirected relations among tokens. The attention layer is not optimized with a direct supervision signal but with the entire network in downstream tasks. Experimental results across three tasks show that our model yields performance comparable or superior to state of the art.

Although our experiments focused on LSTMs, the idea of building more structure aware neural models is general and can be applied to other types of networks. When direct supervision is provided, similar architectures can be adapted to tasks such as dependency parsing and relation extraction. In the future, we hope to develop more linguistically plausible neural architectures able to reason over nested structures and neural models that learn to discover compositionality with weak or indirect supervision.

Acknowledgments

We thank members of the ILCC at the School of Informatics and the anonymous reviewers for helpful comments. The support of the European Research Council under award number 681760 "Translating Multiple Modalities into Text" is gratefully acknowledged.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of the 2016 NAACL: HLT*, pages 1545–1554, San Diego, California.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2014 ICLR*, Banff, Alberta.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd ACL*, pages 655–665, Baltimore, Maryland.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 EMNLP*, pages 22–32, Lisbon, Portugal.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th ACL*, pages 1466–1477, Berlin, Germany.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 EMNLP*, pages 1724–1734, Doha, Qatar.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the 32nd ICML*, pages 2067–2075, Lille, France.

Peter Clark, Phil Harrison, and Niranjan Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *Proceedings of the 3rd Workshop on Automated KB Construction*, San Francisco, California.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Sreerupa Das, C. Lee Giles, and Guo zheng Sun. 1992. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*, pages 791–795. Morgan Kaufmann Publishers.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd ACL*, pages 334–343, Beijing, China.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd IJCAI*, pages 3–10, Barcelona, Spain.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 EMNLP*, pages 1535–1545, Edinburgh, Scotland, UK.

Fernanda Ferreira and John M. Henderson. 1991. Recovery from misanalyses of garden-path sentences. *Journal of Memory and Language*, 30:725–745.

Stefan L. Frank and Rens Bod. 2011. Insensitivity of the human sentence-processing system to hierarchical structure. *Pyschological Science*, 22(6):829–834.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1819–1827.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 EMNLP*, pages 1746–1751, Doha, Qatar.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 2015 ICLR*, San Diego, California.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd ACL*, pages 478–485, Barcelona, Spain.

Lars Konieczny. 2000. Locality and parsing complexity. *Journal of Psycholinguistics*, 29(6):627–645.

Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. 2014. A clockwork RNN. In *Proceedings of the 31st ICML*, pages 1863–1871, Beijing, China.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33rd ICML*, New York, NY.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st ICML*, pages 1188–1196, Beijing, China.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proceedings of the 2015 EMNLP*, pages 1565–1575, Lisbon, Portugal.

Fandong Meng, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2015. A deep memory-based architecture for sequence-to-sequence learning. In *Proceedings of ICLR-Workshop 2016*, San Juan, Puerto Rico.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of 11th Interspeech*, pages 1045–1048, Makuhari, Japan.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2015. Learning longer memory in recurrent neural networks. In *Proceedings of ICLR Workshop*, San Diego, California.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th ICML*, pages 1310–1318, Atlanta, Georgia.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 EMNLP*, pages 1532–1543, Doha, Qatar.

Hoifung Poon and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 296–305, Uppsala.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the 2016 ICLR*, San Juan, Puerto Rico.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 EMNLP*, pages 379–389, Lisbon, Portugal.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013a. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 EMNLP*, pages 1631–1642, Seattle, Washington.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 EMNLP*, pages 1631–1642, Seattle, Washingtton.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd ACL*, pages 1556–1566, Beijing, China.

Michael K. Tanenhaus, Michael J. Spivey-Knowlton, Kathleen M. Eberhard, and Julue C. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory network for language modeling. In *Proceedings of the 15th NAACL*, San Diego, CA.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 NAACL: HLT*, pages 1442–1451, San Diego, California.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the 2015 ICLR*, San Diego, USA.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33rd ICML*, New York, NY.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790*.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.