Long Short-Term Memory-Networks for Machine Reading

Jianpeng Cheng **Li Dong** **Mirella Lapata**
ILCC, 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

Machine reading, the automatic understanding of text, remains a challenging task of great value for NLP applications. We propose a machine reader which processes text incrementally from left to right, while linking the current word to previous words stored in memory and implicitly discovering lexical dependencies facilitating understanding. The reader is equipped with a Long Short-Term Memory architecture, which differs from previous work in that it has a memory tape (instead of a memory cell) for adaptively storing past information without severe information compression. We also integrate our reader with a new attention mechanism in 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

Machine reading can be defined as the automatic understanding of text. The term is used to broadly describe various interrelated tasks ranging from answering reading comprehension questions (Clark et al., 2013), to fact and relation extraction (Etzioni et al., 2011; Fader et al., 2011), knowledge base population and construction (Ji and Grishman, 2011; Niu et al., 2012; Carlson et al., 2010), ontology learning (Poon and Domingos, 2010b), textual entailment (Dagan et al., 2005), and the creation of proposition stores (Peñas and Hovy, 2010; Schubert and Tong, 2003).

In order to understand texts, a machine reading system must provide facilities for: (1) extracting and representing meaning from natural language text, (2) storing meanings internally, and (3) working with stored meanings, to answer questions or 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. Our work presents a novel approach to machine reading which analyzes text without requiring traditional NLP pipelines (e.g., tagging, parsing) or extensive manual engineering. Our model draws inspiration from human language processing and the fact language comprehension is highly *incremental* with readers and listeners continuously extracting the meaning of utterances on a word-by-word basis.

English speakers 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, it has proven challenging to model long input sequences for at least two reasons. The first one concerns model training problems associated 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 reason relates to memory compression problems. As the input sequence gets compressed and blended

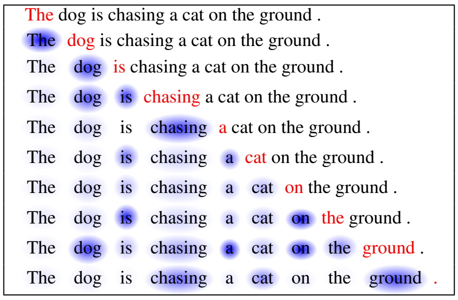


Figure 1: Illustration of our model while reading the sentence *The dog is chasing a cat on the ground.* Color *red* represents the current word being fixated, *blue* represents memories. Shading indicates the degree of memory activation.

into a single dense vector, sufficiently 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.

Recent work attempts to address the latter limitation using external memories (Weston et al., 2015; Sukhbaatar et al., 2015; Grefenstette et al., 2015). The idea is to use multiple memory slots to memorize a sequence; read and write operations for each slot are modeled as an attention mechanism depending on the current input token and the state of the neural controller. Inspired by this work, we equip our machine reader with a memory tape whose size grows with the input sequence.[1] As a point of departure from previous work, we embed the memory tape within an LSTM unit, enabling the model to recurrently read texts without any external state controller. The resulting model is a Long Short-Term Memory-Network (LSTMN) machine reader, which can be used for any sequence processing task.

Figure 1 gives an example of the model's reading behavior. The LSTMN processes text incrementally while learning which words and to what extent contribute to the meaning of the current word. It implicitly identifies lexical dependencies whilst modulating the memory required to extract appropriate meaning representations. We validate the performance of our model in three tasks, namely language modeling, sentiment analysis and natural language inference. In all cases, we achieve performance competitive or better to state-of-the-art models and superior to vanilla LSTM.

[1] It is also reasonable to restrict the size of the tape to simulate a limited memory span.

2 Related Work

Machine reading has been recently recognized as a significant milestone for artificial intelligence (Poon and Domingos, 2010a; Etzioni et al., 2006) since it leverages advances in the fields of natural language processing (NLP), machine learning, and probabilistic reasoning. Much previous work in this area analyzes text with traditional NLP pipelines such as tagging and parsing, mapping natural language to symbolic representations of meaning. More recently, a few approaches have used low dimensional embeddings of entities and relations to enhance symbolic representations such as first-order logic (Rocktäschel et al., 2015; Bordes et al., 2011). Our machine reader leverages neural networks to understand text from scratch without recourse to preprocessing tools or symbolic representations. It learns embeddings which can be integrated with downstream applications such as open information extraction.

Our model extends recent work on recurrent neural networks and their ability to solve sequence modeling and sequence-to-sequence transduction problems. 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). Efforts to handle well-known exploding or vanishing gradient problems associated with RNN training (Bengio et al., 1994) have led 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). Another bottleneck for downstream tasks (Bahdanau et al., 2014) is the memory compression effect. Since the model recursively combines all inputs into a single memory (which is typically too small) it has difficulty memorizing sequences (Zaremba and Sutskever, 2014). In the encoder-decoder architecture, this problem can be sidestepped with an attention mechanism which learns soft alignment *between* the encoding and decoding states (Bahdanau et al., 2014). To the best of our knowledge, no attempts have been made to model attention *within* a sequence encoder.

The idea of coupling neural networks with external memory resources dates back to Das et al. (1992) who connect a recurrent neural network state controller with an external memory stack for learning context free grammars. More recently, Weston et al. (2015) propose Memory Networks to explicitly segregate memory storage from the computation of the neural network. Their model is trained end-to-end with a memory addressing mechanism closely related to soft attention (Sukhbaatar et al., 2015). Meng et al. (2015) apply a variant of this model to machine translation. Their architecture explicitly stores the source sequence in memory, applies read-write transformations, and finally decodes the target sequence based on the transformed memory. Grefenstette et al. (2015) define a set of differentiable data structures (stacks, queues, and dequeues) as memories controlled by a recurrent neural network. Their model has shown promising results in simple sequence transduction tasks, such as copying. Tran et al. (2016) combine the LSTM with an external memory block component which interacts with its hidden state. A common theme across these models is the use of external memories that interact with a neural controller, whereas our work directly enhances the internal memory of an LSTM for reading and memorizing sequences.

3 The Machine Reader

In this section we propose a novel machine reader inspired by psycholinguistics. The core of the reader is a Long Short-Term Memory recurrent neural network with an extended memory tape that explicitly simulates the human memory span. The reader performs implicit dependency analysis with an attention-based memory addressing mechanism at every input time step. In the following we first review the standard Long Short-Term Memory unit and then present 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:

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

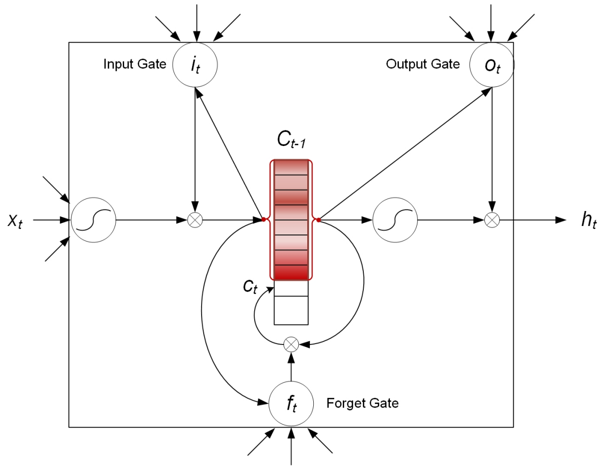


Figure 2: Long Short-Term Memory-Network.

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

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

where i, f, and o are gate activations. Compared to the standard RNN, the LSTM separates the memory c from the hidden state h, which interacts with the environment when computing the output. This network can in principle memorize long input histories, and future predictions can be made conditioned on these histories. In the case of language modeling, the model estimates the probability distribution of the next word as follows:

$$p(x_{t+1} = w | x_1, \cdots, x_t) = \exp(u(w, h_t))/Z \tag{4}$$

where w is a word in the vocabulary V, u is a scoring function that relates w to the current hidden state h_t, and $Z = \sum_{w' \in V} \exp(u(w', h_t))$ is the normalization constant.

3.2 Long Short-Term Memory-Network

We aim at developing a machine reader which explicitly stores memory segments so that it learns how to analyze and modulate past information in order to facilitate the understanding of present input. To this end, we modify the standard LSTM structure by replacing the memory cell with a memory network, whose size grows with the input sequence. The resulting Long Short-Term Memory-Network (LSTMN) stores the input at each time step with a unique memory slot, obviating the problem of information compression, while enabling adaptive modulation of the memory itself. Although it is feasible to apply both write and read operations to the memory network, we focus on the latter. We conceptualize the *read* operation as a means of 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. We use two sets of parameters for the memory network: a hidden state tape interacting with the environment (e.g., computing attention) and a memory tape representing what is actually stored in memory. At each time step, the model computes the memory activation based on the present input token, the previous attention vector, and the previous hidden tape. Then, it uses the adaptively weighted hidden contents to compute various gate activations like the LSTM. Finally, it mixes the memory contents with the current input token to obtain the new input memory. In this model, each input token corresponds to one memory slot, which stores the transformed input representation under the given context.

More formally, given the current input x_t, previous memory tape $C_{t-1} = (c_1, \cdots, c_{t-1})$, and previous hidden tape $H_{t-1} = (h_1, \cdots, h_{t-1})$, the model computes at time step t the values of c_t and h_t as:

$$a_i^t = v^T \tanh(W_h h_i + W_x x_t + W_{\tilde{h}} \tilde{h}_{t-1}) \tag{5}$$

$$s_i^t = \text{softmax}(a_i^t) \tag{6}$$

$$\begin{bmatrix} \tilde{h}_t \\ \tilde{c}_t \end{bmatrix} = \sum_{i=1}^{t-1} s_i^t \cdot \begin{bmatrix} h_i \\ c_i \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [\tilde{h}_t, x_t] \tag{8}$$

$$c_t = f_t \odot \tilde{c}_t + i_t \odot \hat{c}_t \tag{9}$$

$$h_t = o_t \odot \tanh(c_t) \tag{10}$$

where v, W_h, W_x and $W_{\tilde{h}}$ are the new weight terms of the network. Compared to the standard LSTM in Equations (1)–(3), we additionally introduce an attention layer to compute the adaptive memory representation $\tilde{c}_t$ and hidden representation $\tilde{h}_t$ (i.e., the attention vector), which are then used in computing the gated activations i_t, f_t, o_t and the memory to be remembered $\hat{c}_t$.

3.3 Deep LSTMNs

It is possible to construct deep LSTMNs by stacking multiple memory and hidden layers in an alternating fashion, resembling a stacked LSTM (Graves, 2013) or a multi-hop memory network (Sukhbaatar et al., 2015). This is achieved by feeding the output (attention vector) $\tilde{h}_t^k$ of layer k as input to the $(k+1)th$ layer. We apply skip-connections (Graves, 2013) across layers ($\text{input}_{k+1} = \text{input}_k + \text{output}_k$). This improves the information flow within the network and makes optimization easier.

4 LSTMNs for Dual Sequence Modeling

Natural language processing tasks such as machine translation are concerned with dual sequences rather than a single sequence. Central to these tasks is a dual sequence processor or encoder-decoder, where the second sequence (i.e., *target*) is being processed *conditioned* on the first one (i.e., *source*). In this section we draw connections between the internal attention mechanism of the LSTMN and the widely used attention mechanism between two sequences. We then explain how an LSTMN can be applied in the dual sequence processing task.

In general, intra-attention within a sequence and inter-attention between two sequences complement each other. While inter-attention derives the alignment between source and target tokens, intra-attention provides implicit dependency analysis within a sequence, resulting in enhanced memories that could benefit subsequent inter-alignment. In the following we propose two ways of using the LSTMN in a dual architecture, shown in Figure 3a and 3b, respectively.

Shallow Attention Fusion Shallow fusion simply treats the LSTMN as a separate module that can be readily used in a dual architecture (e.g., an encoder-decoder), in lieu of a standard RNN or LSTM. As shown in Figure 3a, both sequence processors are modeled as LSTMNs with intra-attention. Meanwhile, inter-attention is triggered when the target processor reads a target token. The architecture of this model is similar to *RNNSearch*, a neural machine translation model introduced in Bahdanau et al. (2014).

Deep Attention Fusion Deep fusion combines inter- and intra-attention initiated by the target processor when computing the new memory content. Let $A = [\alpha_1, \cdots, \alpha_m]$ and $Y = [\gamma_1, \cdots, \gamma_m]$ denote respectively the source memory tape and hidden tape, with m being the length of the source sequence conditioned upon. We compute inter-attention at time step t when processing the target sequence as follows:

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

$$p_j^t = \text{softmax}(b_j^t) \tag{12}$$

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

and then we can adaptively re-encode the source and transfer its representation to the target with another gating operation r_t:

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

$$c_t = f_t \odot \tilde{c}_{t-1} + i_t \odot \hat{c}_t + r_t \odot \tilde{\alpha}_t \tag{15}$$

$$h_t = o_t \odot \tanh(c_t) \tag{16}$$

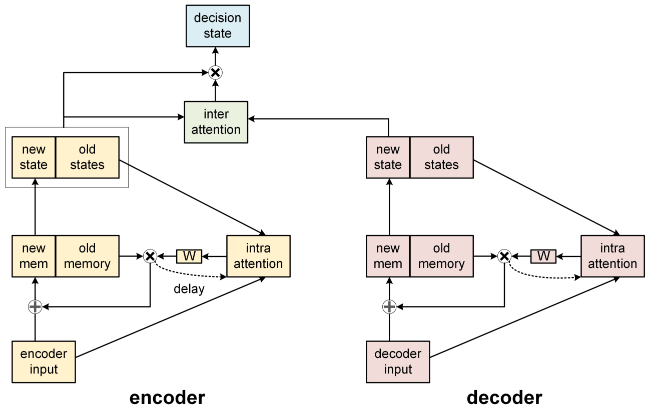
As shown in the above equations and Figure 3b, the major change of deep fusion lies in the recurrent storage of the inter-alignment vector in the target memory network. This tighter integration is potentially helpful for sequence to sequence transduction tasks which involve a significant amount of reordering, and is similar in spirit to the Grid LSTM *Reencoder* (Kalchbrenner et al., 2016).

5 Experiments

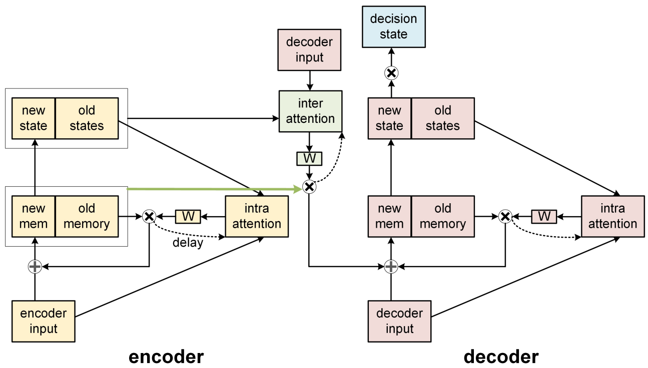
In this section we present our experiments for evaluating the performance of the LSTMN machine reader. We start with language modeling as it is a natural testbed for our model. We then assess the model's ability to extract meaning representations for generic sentence classification tasks such as sentiment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

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



(a) Decoder with shallow attention.



(b) Decoder with deep attention.

Figure 3: LSTMNs for sequence to sequence transduction.

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 the norm is greater than 5. The mini-batch size was set to 40. The dimensions of the LSTM/LSTMN and word embeddings were 300 and 150, respectively.

In this suite of experiments we compared a single-layer LSTMN and a stacked (multilayer) LSTMN (sLSTMN) against a variety of baselines. The first one is the 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. Finally, we implemented more sophisticated LSTM architectures, such as a gated-feedback LSTM (gLSTM; Chung et al. (2015)) and a depth-gated LSTM (dLSTM;

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: Model perplexities on the Penn Treebank.

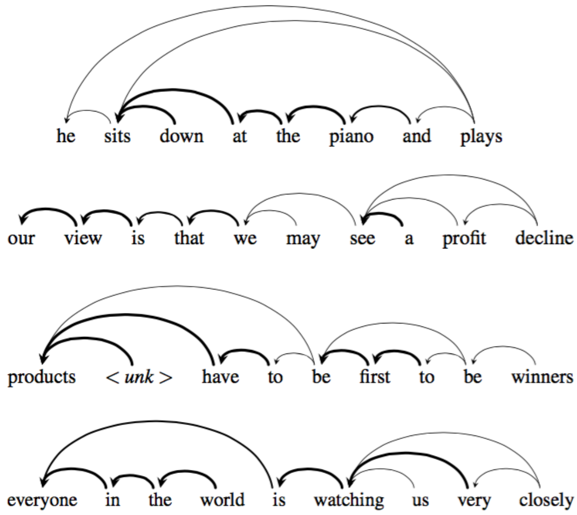


Figure 4: Examples of intra-attention (language modeling). Bold lines indicate stronger attention.

Yao et al. (2015)). The gated-feedback LSTM is a generalization of the clock-work RNN (Koutník et al., 2014), with feedback gates connecting the hidden states across multiple time steps as an adaptive control of the information flow. The depth-gated LSTM is a one dimensional special case of the Grid LSTM (Kalchbrenner et al., 2016), with a depth gate which connects memory cells of adjacent layers, thereby introducing dependencies between them. In general, both gLSTM and dLSTM are able to capture long-term dependencies to some degree, but they do not explicitly keep past memories. For all deep architectures, we set the number of layers to 3 in this experiment.

The results of the language modeling task are shown in Table 1. Perplexity results for KN-5 and RNN models are taken from Mikolov et al. (2015). As can be seen, the single-layer LSTMN outperforms these two baselines 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 six 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 valid bi-lexical relations (e.g., the dependency between *sits* and *at*, *sits* and *plays*, *everyone* and *is*, *is* and *watching*, and *products* and *have*).

5.2 Sentiment Analysis

Our second task concerns the prediction of sentiment labels of sentences. We used the Stanford Sentiment Treebank (Socher et al., 2013), 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.

We experimented with 1- and 2-layer LSTMNs. For the LSTMN models, 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. We used pre-trained `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 convolutional 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

Models	5-classes	2-classes
RAE (Socher et al., 2011)	43.2	82.4
RNTN (Socher et al., 2013)	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)	50.6	87.0
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	**49.3**	**87.3**
2-layer LSTMN	**49.8**	**87.9**

Table 2: Model accuracies (test set) on the Stanford Sentiment Treebank.

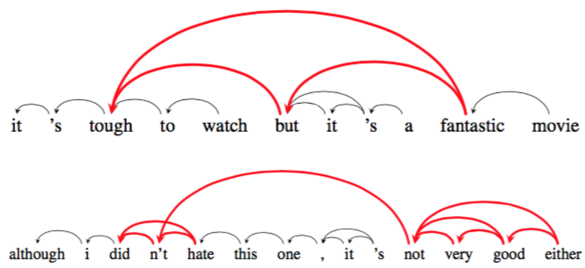


Figure 5: Examples of intra-attention (sentiment analysis). Bold lines (red) indicate attention between sentiment important words.

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.

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. On the fine-grained and binary classification tasks our 2-layer LSTMN performs close to the CT-LSTM (Tai et al., 2015) without recourse to any syntactic information. 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*.

5.3 Natural Language Inference

The ability to reason about the semantic relationship between two sentences is an integral part of machine reading. 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.

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 in a significant performance boost. 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 within each network $Y = [\gamma_1, \cdots, \gamma_m]$ and $H = [h_1, \cdots, h_n]$, and concatenate the two hidden vectors $[\gamma_{avg}, h_{avg}]$ 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 dimension of the LSTMN was 450. 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.

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 rep-

Models	Train	Test
BOW concatenation	60.9	59.8
Handcrafted features (Bowman et al., 2015)	99.7	78.2
LSTM (Bowman et al., 2015)	84.4	77.6
LSTM shared (Rocktäschel et al., 2016)	84.4	81.4
LSTM attention (Rocktäschel et al., 2016)	85.3	83.5
mLSTM (Wang and Jiang, 2016)	86.9	86.1
LSTMN	86.3	83.5
LSTMN shallow fusion	87.6	86.0
LSTMN deep fusion	88.5	**86.3**
LSTMN deep fusion (improved)	92.1	**89.0**

Table 3: Training and test accuracies (in %) for each model on the natural language inference task.

resentation of the premise (rather than basing its predictions on whole sentence embeddings). We also compared our models with a simple classifier (Bowman et al., 2015) which considers traditional features (such as the BLEU score between the premise and hypothesis, their word overlap, and so on) and a bag-of-words baseline which averages the pre-trained `Glove` embeddings for the words in each sentence and concatenates them to create features for a logistic regression classifier (first block in Table 3). As shown in the table, the LSTMNs achieve better performance than the LSTMs both with and without attention. We also observe that deep fusion improves over shallow fusion. One interpretation 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 a new state-of-the-art result in this task (86.3% accuracy).

A characteristic of the LSTMN is that at each time step it chooses the memories which are most relevant to the current input token. While this approach generally delivers improved memories, it does not have any advantage in sentence classification tasks like entailment where global representations of the premise and hypothesis are needed. To remedy this, we add an `EOS` token at the end of each sentence which encourages the network to summarize past memories (with intra-attention) and treat the last intra-attention vector as the global representation of the sentence. We use the concatenation and absolute difference of the global premise and hypothesis representations as two additional features for the classification task. To remove any effect of varying sentence length in a minibatch, we keep a mask variable recording the true length of each sentence, to accurately

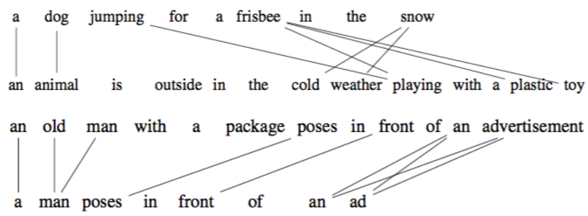


Figure 6: Inter-attention between a premise (upper sentence in each pair) and a hypothesis.

retrieve the last intra-attention vector and compute inter-attention. This modification further improves accuracy to 89.0%.

Table 6 shows two examples of attention between a premise and a hypothesis. As can be seen, the model associates *a dog* with *an animal* and *jumping* with *playing*. It also computes one-to-many and many-to-one alignments, associating *frisbee* with *playing plastic toy*, and *old man* with *man*, respectively.

6 Conclusions

We proposed a novel machine reader that processes sequences from left to right and implicitly discovers lexical dependencies on the fly while reading. The reader employs a Long Short-Term Memory architecture with an extended memory tape, explicitly storing all past input information without recursive memory compression. This architecture leads to implicit dependency analysis and adaptive memory modulation driven by an *internal* attention mechanism. Experimental results across three tasks show that our model yields performance superior to other LSTM variants without recourse to syntactic information or any form of additional annotation.

Although our experiments focused on LSTMs, the internal attention mechanism is general and can be applied to other types of networks such as the Gated Recurrent Unit (Cho et al., 2014). The way in which the current input interacts with past memories is also flexible and can vary depending on different tasks and network architectures. In the future, we would like to introduce a memory updating mechanism, which could be modeled as a *write* operation with another set of attention parameters. This mechanism would potentially allow the machine reader to correct misunderstandings which took place in the past.

References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *The International Conference on Learning Representations*.

[Bengio et al.1994] 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.

[Blunsom et al.2014] Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

[Bordes et al.2011] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI)*, pages 301–306, San Francisco, CA.

[Bowman et al.2015] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *Empirical Methods in Natural Language Processing*.

[Carlson et al.2010] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1306–1313, Atlanta, Georgia.

[Cho et al.2014] 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. *Empirical Methods in Natural Language Processing*.

[Chung et al.2015] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. *The International Conference on Machine Learning*.

[Clark et al.2013] 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, CA.

[Dagan et al.2005] 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*.

[Das et al.1992] 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.

[Etzioni et al.2006] Oren Etzioni, Michele Banko, and Michael J. Cafarella. 2006. Machine reading. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1517–1519, Boston, Massachusetts.

[Etzioni et al.2011] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *International Joint Conference on Artificial Intelligence*, pages 3–10.

[Fader et al.2011] Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

[Ferreira and Henderson1991] Fernanda Ferreira and John M. Henderson. 1991. Recovery from misanalyses of garden-path sentences. *Journal of Memory and Language*, 30:725–745.

[Frank and Bod2011] Stefan L. Frank and Rens Bod. 2011. Insensitivity of the human sentence-processing system to hierarchical structure. *Pyschological Science*, 22(6):829–834.

[Graves2013] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[Grefenstette et al.2015] 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.

[Hermann et al.2015] 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.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hochreiter1991] Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*.

[Irsoy and Cardie2014] Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

[Ji and Grishman2011] Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th*

Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 1148–1158, Portland, Oregon, USA.

[Kalchbrenner et al.2016] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2016. Grid long short-term memory. *The International Conference on Learning Representations*.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Empirical Methods in Natural Language Processing*.

[Kingma and Ba2015] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations*.

[Konieczny2000] Lars Konieczny. 2000. Locality and parsing complexity. *Journal of Psycholinguistics*, 29(6):627–645.

[Koutník et al.2014] Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. 2014. A clockwork rnn. *Proceedings of the 31st International Conference on Machine Learning*.

[Le and Mikolov2014] Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *International Conference on Machine Learning*.

[Lei et al.2015] Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1565–1575, Lisbon, Portugal.

[Meng et al.2015] Fandong Meng, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2015. A deep memory-based architecture for sequence-to-sequence learning. *arXiv preprint arXiv:1506.06442*.

[Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

[Mikolov et al.2015] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2015. Learning longer memory in recurrent neural networks. *Workshop at the The International Conference on Learning Representations*.

[Niu et al.2012] Feng Niu, Ce Zhang, Chirstopher Ré, and Jude Shavlik. 2012. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. In *Proceedings of the Second International Workshop on Searching and Integrating New Web Data Sources*, pages 25–28, Instabul, Turkey.

[Pascanu et al.2013] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning*.

[Peñas and Hovy2010] Anselmo Peñas and Eduard Hovy. 2010. Filling knowledge gaps in text for machine reading. In *Coling 2010: Posters*, pages 979–987, Beijing, China.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Poon and Domingos2010a] Hoifung Poon and Pedro Domingos. 2010a. Machine reading: A "killer app" for statistical relational ai. In *Proceedings of the AAAI-10 Workshop on Statistical Relational AI*, Atlanta, Georgia.

[Poon and Domingos2010b] Hoifung Poon and Pedro Domingos. 2010b. Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 296–305, Uppsala, Sweden, July. Association for Computational Linguistics.

[Rayner1998] Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422.

[Rocktäschel et al.2015] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129, Denver, Colorado.

[Rocktäschel et al.2016] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. *The International Conference on Learning Representations*.

[Rush et al.2015] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *Empirical Methods on Natural Language Processing*.

[Schubert and Tong2003] Lenhart Schubert and Matthew Tong. 2003. Extracting and evaluating general world knowledge from the brown corpus. In Graeme Hirst and Sergei Nirenburg, editors, *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, pages 7–13.

[Socher et al.2011] 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.

[Socher et al.2013] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. Citeseer.

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

[Tai et al.2015] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

[Tanenhaus et al.1995] 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.

[Tran et al.2016] Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory network for language modeling. *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

[Wang and Jiang2016] Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

[Weston et al.2015] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. *The International Conference on Learning Representations*.

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

[Zaremba and Sutskever2014] Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.