When Are Tree Structures Necessary for Deep Learning of Representations?

Jiwei Li¹, Dan Jurafsky¹ and Eduard Hovy²
¹Computer Science Department, Stanford University, Stanford, CA 94305
²Language Technology Institute, Carnegie Mellon University, Pittsburgh, PA 15213
jiweil,jurafsky@stanford.edu ehovy@andrew.cmu.edu

Abstract

Recursive neural models, which use syntactic parse trees to recursively generate representations bottom-up from parse children, are a popular new architecture, promising to capture structural properties like long-distance semantic dependencies. But understanding exactly which tasks this parse-based method is appropriate for remains an open question. In this paper we benchmark recursive neural models against sequential recurrent neural models, which are structured solely on word sequences. We investigate 4 tasks: (1) sentence-level sentiment classification; (2) matching questions to answer-phrases; (3) discourse parsing; (4) computing semantic relations (e.g., component-whole between nouns). We implement apply basic, general versions of recursive and recurrent models and apply to each task. Our analysis suggests that syntactic tree-based recursive models are very helpful for tasks that require representing long-distance relations between words (e.g., semantic relations between nominals), but may not be helpful in other situations, where sequence based recurrent models can produce equal performance. Our results offer insights on the design of neural architectures for representation learning.

1 Introduction

Deep learning based methods learn low-dimensional, real-valued vectors for word tokens, mostly from large-scale data corpus (e.g., Mikolov et al., 2013; Le and Mikolov, 2014; Collobert et al., 2011), successfully capturing syntactic and semantic aspects of text.

For tasks where the inputs are larger text units (e.g., phrases, sentences or documents), a convolutional model is first needed to aggregate tokens into a vector with fixed dimensionality that can be used as a feature for other NLP tasks. Models for achieving this usually fall into two categories: recurrent models and recursive models:

![Figure 1: Standard recursive neural network calculating a representation for the food is delicious based on the syntactic parse tree.](image)

Recurrent models deal successfully with time-series data (Pearlmutter, 1989; Dorffner, 1996) like speech recognition (Robinson et al., 1996; Lippmann, 1989; Graves et al., 2013) or handwriting recognition (Graves and Schmidhuber, 2009; Graves, 2012). They were also applied early on to NLP (Elman, 1990), modeling a sentence as tokens processed sequentially, at each step combining the current token with previously built embeddings. These models generally consider no linguistic structure aside from word order.

Recursive neural models (Williams and Zipser, 1989), by contrast, are structured by syntactic parse trees. Instead of considering tokens sequentially, recursive models combine neighbors based on the recursive structure of the parse trees, starting from the leaves and proceeding recursively in a bottom-up fashion until the root of tree is reached. Figure 1 shows a recursive neural networks building a distributed representation for the phrase the food is delicious, following the operation sequence ((the food) (is delicious)) rather than the sequential order (((the food) is) delicious). Many recursive models have been proposed (e.g., Paulus et al., 2014; Irsoy and Cardie, 2014), and applied to
various NLP tasks, among them entailment (Bowman, 2013; Bowman et al., 2014), sentiment analysis (Socher et al., 2013; Irsoy and Cardie, 2013; Dong et al., 2014), question-answering (Iyyer et al., 2014), relation classification (Socher et al., 2012; Hashimoto et al., 2013), and discourse (Li and Hovy, 2014; Li et al., 2014).

One possible advantage of recursive models is their potential for capturing long-distance dependencies: two tokens may be structurally close to each other, even though they are far away in word sequence. For example, a verb and its corresponding direct object can be far away in terms of tokens if many adjectives lies in between, but they are adjacent in the parse tree (Irsoy and Cardie, 2013). But we don’t know if this advantage is truly important, and if so for which tasks, or whether other issues are at play. Indeed, the reliance of recursive models on parsing is also a potential disadvantage, given that parsing is relatively slow, can be errorful, and parsers are also usually quite domain-dependent.

On the other hand, recent progress in multiple subfields of neural NLP has suggested that recurrent nets may be sufficient to deal with many of the tasks for which recursive models have been proposed. Recurrent models without parse structures have shown good results in sequence-to-sequence generation (Sutskever et al., 2014) for machine translation (e.g., Kalchbrenner and Blunsom, 2013 [Bahdanau et al., 2014; Luong et al., 2014]), parsing (Vinyals et al., 2014), and sentiment, where for example recurrent-based paragraph vectors (Le and Mikolov, 2014) outperform recursive models (Socher et al., 2013) on the Stanford sentiment-bank dataset.

Our goal in this paper is thus to investigate a number of tasks with the goal of understanding for which kinds of problems recurrent models may be sufficient, and for which kinds recursive models offer specific advantages. We investigate four tasks with different properties. Binary sentiment classification at the sentence level (Pang et al., 2002) and phrase level can help us understand the role of recursive models in dealing with semantic compositionally. Question Answer-Phrase Matching on the UMD-QA dataset (Iyyer et al., 2014) can help see whether parsing is useful for finding similarities between source question sentences and targets answer phrases. Semantic Relation Classification on the SemEval-2010 (Hendrickx et al., 2009) data tries to understand whether parsing is helpful in dealing with long-term dependencies, such as relations between two words that are far apart in the sequence. Discourse parsing (RST dataset) is useful for measuring the extent to which parsing improves discourse tasks that need to combine meanings of larger text units.

The principal motivation for this paper is to understand better when, and why, recursive models are needed to outperform simpler models by enforcing apples-to-apples comparison as much as possible. Therefore this paper might intrinsically suffer from the following weaknesses:

(1) This paper applies existing models to existing tasks, offering no novel algorithms or tasks or new state-of-the-art results. Our goal is rather an analytic one, to investigate different versions of recursive and recurrent models. We therefore pair each recurrent model with a matching recursive one (standard recurrent vs standard recursive, bidirectional recurrent vs bidirectional recursive, etc.).

(2) We only explore the most general or basic forms of recursive/recurrent models rather than various sophisticated algorithm variants. This is because fair comparison becomes more and more difficult as models get complex (e.g., the number of neural layers, number of hidden units within each layer, etc.). Thus all the neural models employed in this work are comprised of only one layer of neural convolution—despite the fact that deep neural models with multiple layers potentially come with more expressivity, flexibility and thus better results. Our conclusions might thus be limited to the algorithms employed in this paper, and it is unclear whether they can be extended to other variants or to the latest state-of-the-art. We return to this question in the final discussion.

The rest of this paper organized as follows: We detail versions of recursive/recurrent models in Section 2, present the tasks and results in Section 3, and conclude with discussions in Section 4.

2 Recursive and Recurrent Models

In this section, we introduce the recursive and recurrent models we consider.

2.1 Notations

We assume that the text unit $S$, which could be a phrase, a sentence or a document, is comprised of a sequence of tokens/words: $S =$
\{w_1, w_2, ..., w_{N_s}\}$, where $N_s$ denotes the number of tokens in $S$. Each word $w$ is associated with a $K$-dimensional vector embedding $e_w = \{e_w^1, e_w^2, ..., e_w^K\}$. The goal of recursive and recurrent models is to map the sequence to a $K$-dimensional $e_S$, based on its tokens and their corresponding embeddings.

2.2 Recurrent Models

**Standard Recurrent Models** A recurrent network successively takes word $w_i$ at step $i$, combines its vector representation $e_{w_i}$ with previously built embedding $e_{i-1}$ from time $i-1$, calculates the resulting current embedding $e_t$, and passes it to the next step. The embedding $e_t$ for the current time $t$ is thus:

$$e_t = f(W \cdot e_{t-1} + V \cdot e_{w_l})$$

(1)

where $W$ and $V$ are $K \times K$ dimensional convolutional matrices and $f(\cdot)$ denotes the activation function. Bias vectors are omitted. If $N_s$ denote the length of the sequence, $e_{N_s}$ represents the whole sequence $S$.

**Bidirectional Recurrent Models** (Schuster and Paliwal, 1997) add bidirectionality to the recurrent framework where embeddings for each time are calculated both forwardly and backwardly:

$$e_t^\rightarrow = f(W^\rightarrow \cdot e_{t-1}^\rightarrow + V^\rightarrow \cdot e_{w_l})$$

$$e_t^\leftarrow = f(W^\leftarrow \cdot e_{t-1}^\leftarrow + V^\leftarrow \cdot e_{w_l})$$

(2)

Normally, bidirectional models feed the concatenation vector calculated from both directions $[e_t^\rightarrow, e_t^\leftarrow]$ to the classifier.

**Long Short Term Memory (LSTM)** LSTM (Hochreiter and Schmidhuber, 1997; Schmidhuber, 1997) constitutes a sophisticated version of recurrent models LSTM holds promise to dampen the influence of unrelated information and learn long-term dependencies between tokens. LSTM has shown great advantages over standard recurrent models in multiple sequence-to-sequence prediction tasks like machine translation (e.g., (Bahdanau et al., 2014; Cho et al., 2014; Graves, 2013; Sutskever et al., 2014)), or handwriting recognition (e.g., (Graves et al., 2013)).

We adopt the version of LSTM described in (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2014) by associating each timestep with an input, memory and output gate, denoted by $i_t$, $f_t$ and $o_t$, the values of which are given by:

$$i_t = \sigma(W_i \cdot e_{x_t} + V_i \cdot e_{t-1})$$

$$f_t = \sigma(W_f \cdot e_{x_t} + V_f \cdot e_{t-1})$$

$$o_t = \sigma(W_o \cdot e_{x_t} + V_o \cdot e_{t-1})$$

(3)

where $\sigma$ denotes the sigmoid function, and $i_t$, $f_t$ and $o_t$ are scalars in $[0,1]$. The current timestep embedding $e_t$ is then given by:

$$l_t = \tanh(W_i \cdot e_{x_t} + V_i \cdot e_{t-1})$$

$$m_t = f_t \cdot m_{t-1} + i_t \cdot l_t$$

$$e_t = o_t \cdot m_t$$

(4)

As mentioned in Section 1, LSTM models we employ (along with gate model for recursive models) only contain one layer of convolution.

**Bidirectional LSTM** As in bidirectional recurrent models, Bi-LSTM obtains embeddings forwardly and backwardly and concatenates vectors calculated from both directions for classification.

**Recurrent Tensor Network** We extended the RNTN (Socher et al., 2013) recursive model to the recurrent version which enables richer convolution between early time output $e_{t-1}$ and current word embedding $e_{w_l}$. Adjusted from the recursive RNTN, let $e = [e_{t-1}, e_{w_l}]$ denote the concatenation of $e_{t-1}$ and $e_{w_l}$, the output embedding at $e_t$ is given by:

$$e_t = f(e^T \cdot W \cdot e + V \cdot e)$$

(5)

2.3 Recursive Models

**Standard Recursive Models** Recursive models rely on the structure of parse trees, where each leaf node corresponds to a word from the original sentence, computing a representation for each parent node based on its immediate children recursively in a bottom-up fashion until reaching the root of the tree. Concretely, for a given node $\eta$ in the tree and its left child $\eta_{left}$ (associated with vector representation $e_{left}$) and right child $\eta_{right}$ (associated with vector representation $e_{right}$), the standard recursive network calculates $e_\eta$ as follows:

$$e_\eta = f(W \cdot e_{left} + V \cdot e_{right})$$

(6)

where $W$ and $V$ are $K \times K$ dimensional convolutional matrices and $f(\cdot)$ denotes the activation function.
Bidirectional Recursive Models extend standard recursive models and consider both downward and upward information propagations along the tree [Irsoy and Cardie, 2013]. Each node $p$ is associated with two vectors, an upward vector $e^\uparrow_p$ and a downward vector $e^\downarrow_p$. $e^\downarrow_p$ is computed similarly as in standard recursive models based on two immediate children:

$$e^\downarrow_p = f(W^\downarrow \cdot e^\downarrow_{\text{left}} + V^\downarrow \cdot e^\downarrow_{\text{right}})$$  \hspace{1cm} (7)

The downward vector $e^\downarrow_p$ is computed based on its parent $h(\eta)$ and upward vector $e^\uparrow_p$:

$$e^\uparrow_p = f(W^\uparrow e^\uparrow_{h(\eta)} + V^\uparrow e^\uparrow_{\text{right}})$$ \hspace{1cm} (8)

where $W^\downarrow$, $W^\uparrow$, $V^\downarrow$ and $V^\uparrow$ denote convolutional matrixes. Intuitively, $e^\downarrow_p$ includes evidence from the descendants of $\eta$ while $e^\uparrow_p$ includes information from ancestors.

Note that bi-directional recursive models are suited for classification at the node level by considering upward and downward information, but not sentence level classification. No downward vector is computed for roots—downward vectors are computed based on parents—concatenating root embedding (upwardly computed) with leaf embeddings (downwardly computed) leads to vectors with different dimensionality. We therefore omit the bi-directional recursive model for sentence-level classification tasks.

Recursive Neural Tensor Network (RNTN) RNTN (Socher et al., 2013) enables richer composition between children embeddings. Let $e = [e^\text{left}, e^\text{right}]$ denote the concatenation of the two children’s vectors. RNTN computes the parent representation $e_\eta$ as follows:

$$e_\eta = f(e^T \cdot W \cdot e + V \cdot e)$$ \hspace{1cm} (9)

Gate Based Recursive Models (Gate-Recursive) We adjust the idea of LSTM to recursive model where the representation of a node preserves part of the previous information and adds in the convolutional part. Similarly, let $r_\eta$ and $z_\eta$ denote the control gates associated with node $\eta$, given by:

$$z_\eta = \sigma(W_z \cdot e^\text{left} + V_z \cdot e^\text{right})$$

$$r_\eta = \sigma(W_r \cdot e^\text{left} + V_r \cdot e^\text{right})$$ \hspace{1cm} (10)

$e_\eta$ is then given by:

$$e_\eta = z_\eta \cdot (e^\text{left} + e^\text{right}) + (1 - z_\eta)e^\prime_\eta$$

$$e^\prime_\eta = f(W \cdot r_\eta \cdot e^\text{left} + V \cdot r_\eta \cdot e^\text{right})$$ \hspace{1cm} (11)

3 Experiments

In this section, we detail our experimental settings and results. We consider 4 tasks, each representative of a different class of NLP tasks.

1. Binary sentiment classification on the Pang et al. (2002) dataset. This addresses the issues where supervisions only exist globally at top of long sequences.

2. Sentence-Target Matching on the UMD-QA dataset (Lyer et al., 2014): Learns matches between target and components in the source sentences, which are parse tree nodes for recursive models and different time-steps for recurrent models.

3. Semantic Relation Classification on the SemEval-2010 task (Hendrickx et al., 2009). Learns long-distance relationships between two words that may be far apart sequentially.

4. Discourse Parsing (Li et al., 2014; Hernault et al., 2010): Learns sentence-to-sentence relations based on calculated representations.

In each case we followed the protocols described in the original papers. We employed standard training frameworks for neural models: for each task, we used stochastic gradient decent using AdaGrad (Duchi et al., 2011) with minibatches (Cotter et al., 2011). Parameters are tuned using development dataset if already found from original datasets or from cross-validation if not. Derivatives are calculated from standard backpropagation (Goller and Kuchler, 1996). Parameters to tune include size of mini batches, learning rate, parameters for L2 penalizations. The number of running iterations is treated as a parameter to tune and the model achieving best performance on the development set is used as the final model to be evaluated.

3.1 Binary Sentiment Classification (Pang)

The sentiment dataset of Pang et al. (2002) consists of sentences with a sentiment label for each sentence. We divide the original dataset into training(8101)/dev(500)/testing(2000). No pre-training procedure as described in (Socher et al., 2011b) is employed. Word embeddings are initialized using skip-grams and kept fixed in the learning procedure. We trained skip-gram embeddings on the Wikipedia+Gigaword dataset using the word2vec package[1]. Sentence level embeddings are fed into a sigmoid classifier: For
bi-directional recurrent models, concatenations of embeddings calculated leftwardly and rightwardly are used as sentence-level vectors. As the concatenation of leftward and rightward vectors would double the dimensionality of embeddings, a further convolutional operation is used to preserve vector dimensionality:

\[ \vec{e}_t = f(W_L \cdot [\vec{e}^{-t}_t, \vec{e}^{+t}_t]) \] (12)

where \( W_L \) denotes a \( 2K \times K \) dimensional matrix.

Performances for 50- and 250-dimensional vectors are given in Figure 2; no large difference appears between standard recursive and recurrent models or RNTN-Recursive and RNTN-recurrent models.

Why don’t parse trees help on this task? One possible explanation is the distance of the supervision signal from the local compositional structure. The Pang et al. dataset has an average sentence length of 22.5 words, which means it takes multiple steps before sentiment related evidence comes up to the surface. It is therefore unclear whether local compositional operators (such as negation) can be learned; there is only a small amount of training data and the sentiment supervision only at the level of the sentence may not be easy to propagate down to deeply buried local phrases.

The question-answering dataset QANTA\(^2\) is comprised of 1,713 history questions (8,149 questions) and 2,246 literature questions (10,552 sentences), each question paired with an answer (Iyyer et al., 2014). Each answer is a token or short phrase. The task is defined as a multi-class classification task that matches a source question with a candidates phrase from a predefined pool of candidate phrases rather than sequence generation as in standard QA. We give an illustrative example here:

**Question:** He left unfinished a novel whose title character forges his father’s signature to get out of school and avoids the draft by feigning desire to join. Name this German author of The Magic Mountain and Death in Venice.

**Answer:** Thomas Mann [from the pool of phrases]

The model of Iyyer et al. (2014) minimizes the distances between answer embeddings and node embeddings along the parse tree of the question. Concretely, let \( c \) denote the correct answer to question \( S \), with embedding \( \vec{c} \), and \( z \) denoting any random wrong answer. The objective function sums over the dot product between representation for every node \( \eta \) along the question parse trees and the answer representations:

\[ L = \sum_{\eta \in \text{[parse tree]}} \sum_{z} \max(0, 1 - \vec{c} \cdot \vec{s}_{\eta} + \vec{z} \cdot \vec{s}_{\eta}) \] (13)

where \( \vec{s} \) denotes the embedding for parse tree node \( s \) calculated from the recursive neural model. Here the parse trees are dependency parses following (Iyyer et al., 2014).

Recurrent version models minimize the distance between the answer embedding and embeddings calculated from each timestep of the sequence:

\[ L = \sum_{t \in [1,N_s]} \sum_{z} \max(0, 1 - \vec{c} \cdot \vec{e}_t + \vec{z} \cdot \vec{e}_t) \] (14)

The concatenation of \( \vec{e}^{-t}_t \) and \( \vec{e}^{+t}_t \) (or \( \vec{e}^{-t}_t \) and \( \vec{e}^{+t}_t \)) in bi-directional models will double the dimensionality of embedding, a further convolutional operation is used to preserve vector dimensionality:

\[ \vec{e}_t = f(W_L \cdot [\vec{e}^{-t}_t, \vec{e}^{+t}_t]) \] (15)
where $W_L$ denotes a $2K \times K$ dimensional matrix. As demonstrated in Figure 4, we see no great difference between recurrent and recursive models.

The UMD-QA task represents a group of situations where because we have insufficient supervision about matching (it’s hard to know which node in the parse tree or which timestep provides the most direct evidence for the answer), decisions have to be made by looking at and iterating over all subunits (all nodes in parse trees or timesteps). Similar ideas can be found in pooling structures (e.g. Socher et al. (2011a)). We can draw a similar conclusion: in this task, recursive models look at the relevance to the target of linguistically meaningful units from parse trees, but this does not necessarily lead to better performance than simple sequential units.

3.2 Semantic Relationship Classification

SemEval-2010 Task 8 (Hendrickx et al., 2009) is to find semantic relationships between pairs of nominals, e.g., in “My [apartment]$_1$ has a pretty large [kitchen]$_2$” classifying the relation between [apartment] and [kitchen] as component-whole. The dataset contains 9 ordered relationships, so the task is formalized as a 19-class classification problem, with directed relations treated as separate labels; see Hendrickx et al. (2009); Socher et al. (2012) for details.

For the recursive implementations, we follow the neural framework defined in Socher et al. (2012). The path in the parse tree between the two nominals is retrieved, and the embedding is calculated based on recursive models and fed to a softmax classifier. Retrieved paths are transformed

Results are shown in Figure 5. Unlike for earlier tasks, here recursive models yield much better performance than the corresponding recurrent versions. Even standard recursive models perform better than LSTM.

These results suggest that it is the need to integrate structures far apart in the sentence that characterizes the tasks where recursive models surpass recurrent models. In parse-based models, the two target words are drawn together much earlier in the decision process than in recurrent models, which remembers one target until the other one appears.

3.3 Discourse Parsing

Our final task, discourse parsing based on the RST-DT corpus (Carlson et al., 2003), is to build a discourse tree for a document, based on assigning Rhetorical Structure Theory (RST) relations between elementary discourse units (EDUs). Because discourse relations express the coherence structure of discourse, they presumably express different aspects of compositional meaning than sentiment or nominal relations. A discourse relation classifier takes as input a vector embedding representation of each of the two EDUs, and computes the most likely relation (Li et al., 2014). We compare recursive versus recurrent methods for computing EDU embeddings. See Ji and Eisenstein () and Hernault et al. (2010) for more details on discourse parsing and the RST-DT corpus.

human-feature engineering. Again, because MV-RNN is difficult to adapt to a recurrent version, we do not employ this state-of-the-art model, adhering only to the general versions of recursive models described in Section 2, since our main goal is to compare equivalent recursive and recurrent models rather than implement the state of the art.
Representations for adjacent EDUs are fed into binary classification (whether two EDUs are related) and multi-class relation classification models, as defined in (Li et al., 2014). Following Li et al. (2014), a CKY bottom-up algorithm is used to compute the most likely discourse parse tree using dynamic programming. Models are evaluated in terms of three matrices: Span (on blank tree structures), nuclearity (on tree structures with nuclearity indication), and relation (on tree structures with rhetorical relation indication but no nuclearity indication\footnote{Nuclearity indicates which one of the two adjacent EDUs are more salient.}. To serve the purposes for plain comparison, no additional human-developed features are added.

Results are reported in Table\textsuperscript{1} We see no large differences between equivalent recurrent and recursive models. Again Bi-LSTM achieves the best performance.

### 4 Discussions

We compared recursive and recurrent neural models for representation learning on four separate NLP tasks for which recursive neural models are known to achieve good performance (Socher et al., 2012; Socher et al., 2013; Li et al., 2014; Iyyer et al., 2014).

Our results suggest that recurrent models are sufficient to capture the compositional semantics required for many NLP tasks. But in tasks like semantic relation extraction, in which single headwords need to be associated across a long distance, recursive models seem to shine. This suggests that for the many classes of tasks where long-distances play a role (such as Chinese-English translation) syntactic structures from recursive models might offer useful power.

Exploring these directions constitute our future work; for example our results illustrate that improving sequential models to handle such phenomena will require methods for modeling such long-distance dependencies.

Of course, it is hard to define what is exactly “fair” in this comparison-oriented paper: we force every model to be trained exactly the same way: AdaGrad with minibatches, same set of initializations, etc. However, it is not necessarily true that this is the optimal way to train every model. It is possible that different training strategies are tailored for and should be adopted for different models. In that sense, trying to be “fair” in the way we selected in this paper is in fact not being fair.

Additionally, we require that all sequence and tree models be of the same structure. Such a strategy may be problematic, since one model might be more flexible to other algorithm variants while the

| Model             | Span  | Nuclearity | Relation |
|-------------------|-------|------------|----------|
| Recursive         | 0.824 | 0.692      | 0.568    |
| Recurrent         | 0.815 | 0.684      | 0.574    |
| RNTN-Recursive    | 0.828 | 0.695      | 0.566    |
| RNTN-Reccurrent   | 0.830 | 0.687      | 0.560    |
| Gate-Recursive    | 0.828 | 0.702      | 0.562    |
| LSTM              | 0.838 | 0.696      | 0.578    |
| BiLSTM            | 0.840 | 0.702      | 0.580    |

Table 1: Discourse parsing performances from different EDUs models according to 3 metrics.
other might not. For example, we only employ a one-layer neural model structure for each model, to make fair comparison easier. However, it could be possible that one of the models is intrinsically better suited for deep (multi-layer) structure than the other. For instance, [Tai et al. (2015)] found that deep LSTM tree models outperform deep LSTM sequence models on the Stanford SentimentBank task, suggesting that depth may offer different or greater advantages in recursive models. We hope to explore these questions in our future work.

5 Acknowledgments

We would especially like to thank Richard Socher for insightful comments, advice, and suggestions. We would also like to thank Sam Bowman and other members of the Stanford NLP group, as well as the ACL reviewers for their helpful advice on various aspects of this work.

References

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

[Bowman et al.2014] Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2014. Recursive neural networks for learning logical semantics. arXiv preprint arXiv:1406.1827.

[Bowman2013] Samuel R Bowman. 2013. Can recursive neural tensor networks learn logical reasoning? arXiv preprint arXiv:1312.6192.

[Carlson et al.2003] Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. Springer.

[Cho et al.2014] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. The Journal of Machine Learning Research, 12:2493–2537.

[Graves and Schmidhuber2009] Alex Graves and Jürgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In Advances in Neural Information Processing Systems, pages 545–552.

[Graves et al.2013] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 6645–6649. IEEE.

[Graves2012] Alex Graves. 2012. Supervised sequence labelling with recurrent neural networks, volume 385. Springer.

[Hashimoto et al.2013] Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In EMNLP, pages 1372–1376.

[Graves et al.2011] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. 2011. Better minibatch algorithms via accelerated gradient methods. In Advances in Neural Information Processing Systems, pages 1647–1655.

[Dong et al.2014] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 49–54.

[Dorffner1996] Georg Dorffner. 1996. Neural networks for time series processing. In Neural Network World. Citeseer.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159.

[Elman1990] Jeffrey L Elman. 1990. Finding structure in time. Cognitive science, 14(2):179–211.

[Goller and Kuchler1996] Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In Neural Networks, 1996. IEEE International Conference on, volume 1, pages 347–352. IEEE.

[Gupta et al.2011] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. 2011. Better minibatch algorithms via accelerated gradient methods. In Advances in Neural Information Processing Systems, pages 1647–1655.

[Dong et al.2014] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 49–54.

[Dorffner1996] Georg Dorffner. 1996. Neural networks for time series processing. In Neural Network World. Citeseer.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159.

[Elman1990] Jeffrey L Elman. 1990. Finding structure in time. Cognitive science, 14(2):179–211.
Recent Achievements and Future Directions, pages 94–99. Association for Computational Linguistics.

[Hernault et al.2010] Hugo Hernault, Helmut Prendinger, Mitsuru Ishizuka, et al. 2010. Hilda: a discourse parser using support vector machine classification. Dialogue & Discourse, 1(3).

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

[Irsoy and Cardie2013] Ozan Irsoy and Claire Cardie. 2013. Bidirectional recursive neural networks for token-level labeling with structure. arXiv preprint arXiv:1312.0493.

[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.

[Iyyer et al.2014] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 633–644.

[Ji and Eisenstein] Yangfeng Ji and Jacob Eisenstein. Representation learning for text-level discourse parsing.

[Kalchbrenner and Blunsom2013] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In EMNLP, pages 1700–1709.

[Le and Mikolov2014] Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053.

[Li and Hovy2014] Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation.

[Li et al.2014] Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2061–2069.

[Lippmann1989] Richard P Lippmann. 1989. Review of neural networks for speech recognition. Neural computation, 1(1):1–38.

[Luong et al.2014] Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. arXiv preprint arXiv:1410.8206.

[Mikolov et al.2013] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In HLT-NAACL, pages 746–751. Citeseer.

[Pang et al.2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, pages 79–86. Association for Computational Linguistics.

[Paulus et al.2014] Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In Advances in Neural Information Processing Systems, pages 2888–2896.

[Pearlmutter1989] Barak A Pearlmutter. 1989. Learning state space trajectories in recurrent neural networks. Neural Computation, 1(2):263–269.

[Robinson et al.1996] Tony Robinson, Mike Hochberg, and Steve Renals. 1996. The use of recurrent neural networks in continuous speech recognition. In Automatic speech and speaker recognition, pages 233–258. Springer.

[Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Lstm can solve hard long time lag problems. In Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference, volume 9, page 473. MIT Press.

[Schuster and Paliwal1997] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. Signal Processing, IEEE Transactions on, 45(11):2673–2681.

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

[Socher et al.2011b] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 151–161. Association for Computational Linguistics.

[Socher et al.2012] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211. Association for Computational Linguistics.

[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. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1631–1642. Citeseer.
[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, pages 3104–3112.

[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. arXiv preprint arXiv:1503.00075.

[Vinyals et al.2014] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2014. Grammar as a foreign language. arXiv preprint arXiv:1412.7449.

[Williams and Zipser1989] Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. Neural computation, 1(2):270–280.