Predicting Embedded Syntactic Structures from Natural Language Sentences with Neural Network Approaches

Gregory Senay
Panasonic Silicon Valley Lab
Cupertino, CA 95014
gregory.senay@us.panasonic.com

Fabio Massimo Zanzotto
University of Rome Tor Vergata
Viale del Politecnico, 1, 00133 Rome, Italy
fabiomassimo.zanzotto@gmail.com

Lorenzo Ferrone
University of Rome Tor Vergata
Viale del Politecnico, 1, 00133 Rome, Italy
lorenzo.ferrone@gmail.com

Luca Rigazio
Panasonic Silicon Valley Lab
Cupertino, CA 95014
luca.rigazio@us.panasonic.com

Abstract

Syntactic parsing is a key component of natural language understanding and, traditionally, has a symbolic output. Recently, a new approach for predicting syntactic structures from sentences has emerged: directly producing small and expressive vectors that embed in syntactic structures. In this approach, parsing produces distributed representations. In this paper, we advance the frontier of these novel predictors by using the learning capabilities of neural networks. We propose two approaches for predicting the embedded syntactic structures. The first approach is based on a multi-layer perceptron to learn how to map vectors representing sentences into embedded syntactic structures. The second approach exploits recurrent neural networks with long short-term memory (LSTM-RNN-DRP) to directly map sentences to these embedded structures. We show that both approaches successfully exploit word information to learn syntactic predictors and achieve a significant performance advantage over previous methods. Results on the Penn Treebank corpus are promising. With the LSTM-RNN-DRP, we improve the previous state-of-the-art method by 8.68%.

1 Introduction

Syntactic structure is a key component for natural language understanding [8], with several studies showing that syntactic information helps in modeling meaning [21, 14, 25]. Consequently, a very active area in natural language processing is building predictors of *symbolic* syntactic structures from sentences; such predictors, called parsers, are commonly implemented as complex recursive or iterative functions. Even when learned from data, the recursive/iterative nature of parsers is generally not changed since learning is confined to a probability estimation of context-free rules [9, 6] or learning of local discriminative predictor ([23, 26]).

Despite the effort in building explicit syntactic structures, they are rarely used in that form for semantic tasks such as question answering [28], recognizing textual entailment [13], semantic textual similarity [1]. These tasks are generally solved by learning classifiers or regressors. Hence, syntactic structures are unfolded to obtain syntactic-rich feature vectors [14], used within convolution kernel functions [17], or guiding the application of recursive neural networks [25]. Syntactic structures are first discovered by parsers, then, unfolded by "semantic learners" in explicit or implicit syntactic feature vectors.

Distributed syntactic trees [30] have offered a singular opportunity to redraw the path between sentences and feature vectors used within learners of semantic tasks. These distributed syntactic trees embed syntactic trees in small vectors. Hence, a possibility is to learn functions to map sentences in distributed syntactic trees [29]. These functions have been called *distributed representation parsers* (DRPs) [29]. However, these distributed representation parsers suffer from major limitations because, due to data sparsity, these functions can only transform part-of-speech tag sequences in syntactic trees without the lexical information.

In this paper, we propose two novel approaches based on neural networks for building predictors of distributed syntactic structures. The first model is based on a multi-layer perceptron (MLP) which learns how to map sentences, transformed into vectors to distributed syntactic representations. The second model is based on a recurrent neural network (RNN) with long short-term memory (LSTM) which learns to directly map sentences to distributed trees. Both models show the ability to positively exploit words in learning these predictors and significantly outperform previous models [29].

The paper is organized as follows: Section 2 describes the background by reporting on the distributed syntactic trees and the idea of distributed representation parsers; Section 3 introduces our two novel approaches for distributed representation parsing: the model based on a multi-layer perceptron (MLP-DRP) and the model based on long short-term memory (LSTM-RNN-DRP); Section 4 reports on the experiments and the results. Finally, section 5 draws conclusions.

2 Background

2.1 Distributed Syntactic Trees: Embedding Syntactic Trees in Small Vectors

Embedding syntactic trees in small vectors [30] is a key idea which changes how syntactic information is used in learning. Stemming from the recently revitalized research field of Distributed Representations (DR) [18, 24, 4, 12, 25], distributed syntactic trees [30] have shown that it is possible to use small vectors for representing the syntactic information. In fact, feature spaces of subtrees underlying tree kernels [10] are fully embedded by these distributed syntactic trees.

We want to give an intuitive idea how this embedding works. To explain this idea, we need to start from the definition of tree kernels [10] used in kernel machines. In these kernels, trees T are seen as collections of subtrees $S(T)$ and a kernel $TK(T_1, T_2)$ between two trees performs a weighted count of common subtrees, that is:

$$TK(T_1, T_2) = \sum_{\tau_i \in S(T_1), \tau_j \in S(T_2)} \omega_{\tau_i} \omega_{\tau_j} \delta(\tau_i, \tau_j)$$

where ω_{τ_i} and ω_{τ_j} are the weights for subtrees τ_i and τ_j and $\delta(\tau_i, \tau_j)$ is the Kronecker's delta between subtrees. Hence, $\delta(\tau_i, \tau_j) = 1$ if $\tau_i = \tau_j$ else $\delta(\tau_i, \tau_j) = 0$. Distributed trees, in some sense, pack sets $S(T_{s_1})$ in small vectors. In the illustration of Figure 1, this idea is conveyed by packing images of subtrees in a small space, that is, the box under $DT(T_{s_1})$. By rotating and coloring subtrees, the picture in the box under $DT(T_{s_1})$ still allows us to recognize these subtrees. Consequently, it is possible to count how many subtrees are similar by comparing the picture in the box under $DT(T_{s_1})$ with the one under $DT(T_{s_2})$. We visually show that it is possible to pack subtrees in small boxes, hence, it should be possible to pack this information in small vectors.

The formal definition of these embeddings, called distributed syntactic trees $DT(T)$, is the following:

$$DT(T) = \sum_{\tau_i \in S(T)} \omega_i \vec{\tau}_i = \sum_{\tau_i \in S(T)} \omega_i \mathrm{dt}(\tau_i)$$

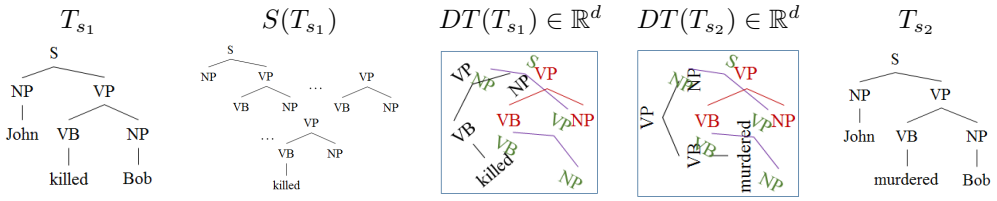

Figure 1: Distributed tree idea

where $S(T)$ is the set of the subtrees τ_i of T, $dt(\tau_i) = \vec{\tau}_i$ is a vector in $\mathbb{R}^d$ corresponding to the subtree τ_i, and ω_i is the weight assigned to that subtree. These vectors are obtained compositionally using vectors for node labels and shuffled circular convolution $\otimes$ as a basic composition function. For example, the last subtree of $S(T_{s_1})$ in Figure 1 has the following vector:

$$dt(T_1) = (\vec{S} \otimes (\vec{NP} \otimes \vec{John}) \otimes (\vec{VP} \otimes (\vec{VB} \otimes \vec{killed}) \otimes (\vec{VP} \otimes \vec{Bob})))$$

Vectors $dt(\tau_i)$ have the following property:

$$\delta(\tau_i, \tau_j) - \epsilon < |dt(\tau_i) \cdot dt(\tau_j)| < \delta(\tau_i, \tau_j) + \epsilon \tag{1}$$

with a high probability. Therefore, given two trees T_1 and T_2, the dot product between the two related, distributed trees approximates the tree kernel between trees $TK(T_1, T_2)$, that is:

$$DT(T_1) \cdot DT(T_2) = \sum_{\tau_i \in S(T_1), \tau_j \in S(T_2)} \omega_{\tau_i} \omega_{\tau_j} dt(\tau_i) \cdot dt(\tau_j) \approx TK(T_1, T_2)$$

with a given degree of approximation [30]. Hence, distributed syntactic trees allow us to encode syntactic trees in small vectors.

2.2 Distributed Representation Parsers

Building on the idea of encoding syntactic trees in small vectors [30], distributed representation parsers (DRPs) [29] have been introduced to predict these vectors directly from sentences. DRPs map sentence s to predicted distributed syntactic trees $DRP(s)$ (Figure 2), and represent the expected distributed syntactic trees $DT(T_s)$. In Figure 2, $DRP(s_1)$ is blurred to show that it is a predicted version of the correct distributed syntactic tree, $DT(T_{s_1})$. The DRP function is generally divided in two blocks: a sentence encoder SE and a transducer P, which is the actual "parser" as it reconstructs distributed syntactic subtrees. In contrast, the sentence encoder SE maps sentences into a distributed representation. For example, the vector $SE(s_1)$ represents s_1 in Figure 2 and contains subsequences of part-of-speech tags.

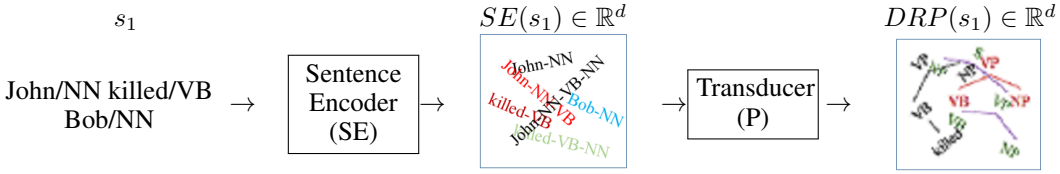

Figure 2: Visualization of the distributed representation parsing

Formally, a DRP is a function $DRP : \mathcal{X} \rightarrow R^d$ that maps sentences into $\mathcal{X}$ to distributed trees in R^d. The sentence encoder $SE : \mathcal{X} \rightarrow R^d$ maps sentences into $\mathcal{X}$ to distributed representation of sentence sequences defined as follows:

$$SE(s) = \sum_{seq_i \in SUB(s)} \vec{seq}_i$$

where $SUB(s)$ is a set of all relevant subsequences of s, and $\vec{seq}_i$ are nearly orthonormal vectors representing given sequences seq_i. Also, vectors seq_i are nearly orthonormal (c.f., Equation 1 applied to sequences instead of subtrees) and are obtained composing vectors for individual elements in sequences. For example, the vector for the subsequence $seq_1 =$ *John-NN-VB* is:

$$\vec{seq}_1 = \vec{John} \otimes \vec{NN} \otimes \vec{VB}$$

The transducer $P : R^d \rightarrow R^d$ is instead a function that maps distributed vectors representing sentences to distributed trees. In [29], P has been implemented as a square matrix trained with a partial least square estimate.

3 Predicting Distributed Syntactic Trees

Distributed representation parsing establishes a different setting for structured learning where a multi-layer perceptron (MLP) can help. In this novel setting, MLP are designed to learn functions that map sentences s or distributed sentences $SE(s)$ to low dimensional vectors embedding syntactic trees $DRP(s)$.

We thus explored two models: (1) a model based on a multi-layer perceptron to learn to transducers P that maps distributed sentences $SE(s)$ to distributed trees $DRP(s)$; (2) a model based on a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) which learns how to map sentences s as word sequences to distributed trees $DRP(s)$.

3.1 From Distributed Sentences to Distributed Trees with Multi-Layer Perceptrons

Our first model is based on a multi-layer perceptron (MLP) to realize the transducer $P_{MLP} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ (see Figure 2), which maps distributed sentences $SE(s)$ to distributed structures $DRP(s)$. The overall distributed representation parser based on the multi-layer perceptron is referred to as ***MLP-DRP***. To define our ***MLP-DRP*** model, we need to specify: (1) the input and the expected output of P_{MLP}; (2) the topology of the MLP.

We defined two classes of input and output for the transducer P_{MLP}: an *unlexicalized model* (UL) and a *lexicalized model* (L). In the *unlexicalized* model, input distributed sentences and output distributed trees do not contain words. Distributed sentences encode only sequences of part-of speech tags. We experimented with $SEQ_{UL}(s)$ containing sequences of part-of-speech tags up to 3. For example, $SEQ_{UL}(s_1) = \{$NN,NN-VB,NN-VB-NN,VB,VB-NN,NN$\}$ (see Figure 2). Similarly, distributed trees encode syntactic subtrees without words, for example, *(VP (VB NN))* . On the other hand, in the lexicalized model, input distributed sentences and output distributed trees are lexicalized. The lexicalized version of distributed sentences was obtained by concatenating previous part-of-speech sequences with their first words. For example, $Seq_{UL}(s_1) = \{$John-NN,John-NN-VB,John-NN-VB-NN,killed-VB,killed-VB-NN,Bob-NN$\}$. Distributed trees encode all the subtrees, including those with words.

Then, we setup a multi-layer perceptron that maps $x = SE(s)$ to $y' = DRP(s)$ and its expected output is $y = DT(T_s)$. The layer 0 of the network has the activation:

$$a^{(0)} = \sigma(W^{(0)}x + b^{(0)})$$

We selected a sigmoid function as the activation function σ:

$$\sigma(z) = \frac{1}{1 + exp(-z)}.$$

All intermediate $n - 2$ layers of the network have the following activation :

$$\forall n \in [1; N-2] : a^{(n)} = \sigma(W^{(n-1)}a^{(n-1)} + b^{(n-1)})$$

The final reconstructed layer, with output y', is done with a linear function:

$$y'(x) = W^{(N-1)}a^{(N-1)} + b^{(N-1)}$$

We learn the network weights by using the following cost function:

$$J(W, b; x, y', y) = 1 - \frac{y \cdot y'}{||y||\ ||y'||},$$

that evaluates the cosine similarity between y and y'.

Learning unlexicalized and lexicalized MLP-DRPs is feasible even if the two settings hide different challenges. The unlexicalized MLP-DRP exposes network learned with less information to encode. However, the model cannot exploit the important information on words. In contrast, the lexicalized MLP-DRP can exploit words but it has to encode more information. Experiments with the two settings are reported in Section 4.

3.2 From Word Sequences to Distributed Trees with Long Short Term Memory

Our second model is more ambitious: it is an end-to-end predictor of distributed syntactic trees $DRP(s)$ from sentences s. We based our approach on recurrent neural networks (RNN) since RNNs have already proven their efficiency to learn complex sequence-to-sequence mapping in speech recognition [16] and in handwriting [15]. Moreover, RNNs have been also successfully used to learn mapping functions between word sequences through sentence embedding vectors [7, 2].

Our end-to-end predictor of distributed syntactic structures is built on the recurrent neural network model with long-short term memory (LSTM-RNN) [20] to overcome the vanishing gradient problem. However, to increase computational efficiency, in this model the activation of the output gate of each cell does not depend on its memory state.

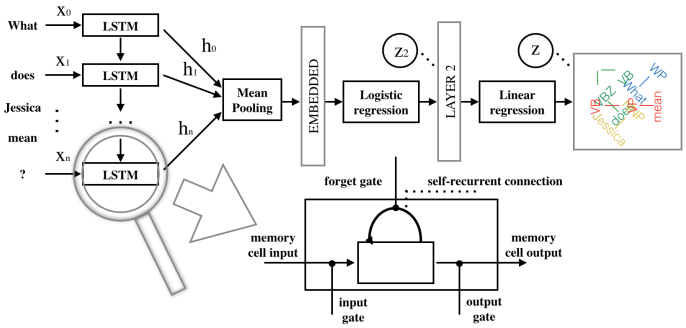

Figure 3: Structure of our LSTM-RNN-DRP encoder and a detail of the LSTM neuron

The resulting distributed representation parser LSTM-RNN-DRP is then defined as follows: Input sentences s are seen as word sequences. To each word in these sequences, we assigned a unit base vector $x_t \in \mathbb{R}^L$ where L is the size of the lexicon. x_t is 1 in the t-th component representing the word and 0 otherwise. Words are encoded with 4 matrices $W_i, W_c, W_f, W_o \in \mathbb{R}^{m \times L}$. Hence, m is the size of word encoding vectors. The LSTM cells are defined as follows: x_t is an input word to the memory cell layer at time t. i_t is the input gate define by:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{2}$$

where σ is a sigmoid. $\tilde{C}_t$ is the candidate values of the states of the memory cells:

$$\tilde{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c). \tag{3}$$

f_t is the activation of the memory cell's forget gates:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f). \tag{4}$$

Given i_t, f_t and $\tilde{C}_t$, C_t memory cells are computed with:

$$C_t = i_t \star \tilde{C}_t + f_t \star C_{t-1}, \tag{5}$$

where $\star$ is the element-wise product. Given the state of the memory cells, we compute the output gate with:

$$\begin{aligned} o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_1) \\ h_t &= o_t \star tanh(C_t) \end{aligned} \tag{6}$$

The non recurrent part of this model is achieved by an average pooling of the sequence representation $h_0, h_1, ..., h_n$, the 4 matrix W_* are concatenated into a single one: W, the U_* weight matrix into U

and the bias b_* into b (see Figure 3). Then, a pre-nonlinear function is computed with W, U and b, following by a linear function:

$$\begin{aligned} z_2 &= \sigma(Wx_t + U_{t-1} + b) \\ z &= W_2 z_2 + b_2 \end{aligned} \tag{7}$$

Finally, the cost function of this model is the cosine similarity between the reconstructed output z and $DT(T_s)$.

4 Experiments

This section explores whether our approaches can improve existing models for learning distributed representation parsers (DRPs). Similarly to [29], we experimented with the classical setting of learning parsers adapted to the novel task of learning DRPs.

In these experiments, all trainings are done with a maximum number of epochs of 5000. If a better result is not found on the validation set after a patience of 30 epochs, we stop the training. All deep learning experiments are done with the *Theano* toolkit [5, 3]. The dimension of the embedded vector after the mean pooling is fixed to 1024 and the second layer size is fixed to 2048. There dimensions are fixed empirically.

4.1 Experimental set-up

The experiment is based on the revised evaluation model for parsers adapted to the task of learning distributed representation parsers [29]. Here we use the **Penn Treebank** corpus for learning and predicting the embedded syntactic structures. The distributed version of the Penn Treebank contains distributed sentences $SE(s)$ along related oracle distributed syntactic trees $DT(T_s)$ for all the sections of the Penn Treebank. Distributed syntactic trees are provided for three different λ values: 0, 0.2 and 0.4. As in tree kernels, λ governs weights ω_{τ_i} of subtrees τ_i. For each λ, there are two versions of the data sets: an un-lexicalized version (UL), where sentences and syntactic trees are considered without words, and a lexicalized version (L), where words are considered. Because the LSTM-RNN-DRP approach is based on word sequence, only the lexicalized results are reported. As for parsing, the datasets from the Wall Street Journal (WSJ) section are divided in: sections 20-21 with 39,832 distributed syntactic trees for training, section 23 with 2,416 distributed syntactic trees for testing and section 24 with 1,346 distributed syntactic trees for parameter estimation.

The evaluation measure is the cosine similarity $cos(DRP(s), DT(T_s))$ between predicted distributed syntactic trees $DRP(s)$ and distributed syntactic trees $DT(T_s)$ of the distributed Penn Treebank, computed for each sentence in the testing and averaged on all the sentences.

We compared our novel models with respect to the model in [29], ZD-DRP (the baseline), and we respect the chain of building distributed syntactic representations that involve a symbolic parser SP, that is, $DSP(s) = DT(SP(s))$. In line with[29], as symbolic parser SP, we used the Bikel's parser.

4.2 Results and discussion

The question we want to answer with these experiments is whether MLP-DRP and LSTM-RNN-DRP can produce better predictors of distributed syntactic trees from sentences. To compare with previous results, we experimented with the distributed Penn Treebank set.

We experimented with d=4096 as the size of the space for representing distributed syntactic trees. We compared with a previous approach, that is ZD-DRP [29] and with the upper-bound of the distributed symbolic parser DSP.

Our novel predictors of distributed syntactic trees outperform previous models for all the values of the parameters (see Table 1). Moreover, our MLP-DRP captures better structural information than the previous model ZD-DRP. In fact, when λ is augmented, the difference in performance between our MLP-DRP and ZD-DRP increases. With higher λ, larger structures have higher weights. Hence, our model captures these larger structures better than the baseline system. In addition, our model is definitely closer to the distributed symbolic parser DSP in the case of unlexicalized trees. This is promising, as the DSP is using lexical information whereas our MLP-DRP does not.

Table 1: Predicting distributed trees on the Distributed Penn Treebank (section 23): average cosine similarity between predicted and oracle distributed syntactic trees. ZD-DRP is a previous baseline model, MLP-DRP is our model and DSP is a the Bikel's parser with a distributed tree function.

	unlexicalized trees			*lexicalized trees*		
Model	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$
ZD-DRP (baseline)	0.8276	0.7552	0.6506	0.7192	0.6406	0.0646
MLP-DRP	**0.8358**	**0.7863**	**0.7038**	**0.7280**	0.6740	0.4960
LSTM-RNN-DRP	-	-	-	0.7162	**0.7274**	**0.5207**
DSP	0.8157	0.7815	0.7123	0.9073	0.8564	0.6459

Our second approach LSTM-RNN-DRP, based on the word sequence, outperforms the other approaches for lexicalized setup. Results show a high improvement compared to the baseline (+8.68% absolute with $\lambda = 0.2$) and it shows this model can represent lexical information better than MLP-DRP under the same conditions.

Finally, our new models reduce the gap in performances with the DSP on the lexicalized trees by dramatically improving over previous models on $\lambda = 0.4$. The increase in performance of our approaches with respect to ZD-DRP is extremely important as it confirms that MLP-DRP and LSTM-RNN-DRP can encode words better.

5 Conclusion

This paper explores two novel methods to merge symbolic and distributed approaches. Predicting distributed syntactic structures is possible and our models show that neural networks can definitely play an important role in this novel emerging task. Our predictor based on a Multi-Layer Perceptron and Long-Short Term Memory Recurrent Neural Network outperformed previous models. This last method, RNN-LSTM-DRP is able, other than the word level, to predict the syntactic information from the sentence. This is a step forward to use these predictors that may change the way syntactic information is learned.

Future research should focus on exploring the promising capability of encoding words shown by recurrent neural networks with long-short term memory. But we think a combinaison of both our approaches can also increase the quality of our predictor due to the fact that each approach encode different information of the tree. This should lead a better predictor of distributed syntactic structures.

References

[1] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, and W. Guo. *sem 2013 shared task: Semantic textual similarity. In **SEM*, pages 32–43, USA, 2013. ACL.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014.

[3] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. Theano: new features and speed improvements. *arXiv:1211.5590*, 2012.

[4] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009.

[5] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of SciPy* , 2010.

[6] E. Charniak. A maximum-entropy-inspired parser. In *Proc. of the 1st NAACL*, 2000.

[7] K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.

[8] N. Chomsky. *Aspect of Syntax Theory*. MIT Press, Cambridge, Massachussetts, 1957.

[9] M. Collins. Head-driven statistical models for natural language parsing. *Comput. Linguist.*, 29(4), 2003.

[10] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *NIPS*, 2001.

[11] Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL02*. 2002.

[12] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12, 2011.

[13] I. Dagan, D. Roth, M. Sammons, and F.M. Zanzotto. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on HLT. Morgan&Claypool Publishers, 2013.

[14] Daniel Gildea and Daniel Jurafsky. Automatic Labeling of Semantic Roles. *Comp. Ling.*, 28(3), 2002.

[15] A. Graves. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

[16] A. Graves, A. R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, IEEE, 2013.

[17] D. Haussler. Convolution kernels on discrete structures. Tech.Rep., Univ. of California at S. Cruz, 1999.

[18] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press, Cambridge, MA., 1986.

[19] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8), 1997.

[21] B. MacCartney, T. Grenager, M. -C. de Marneffe, D. Cer, and C. D. Manning. Learning to recognize features of valid textual entailments. In *Proceedings of NAACL*, New York City, USA, 2006.

[22] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330, 1993.

[23] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Nat. Lang. Eng.*, 13(2), 2007.

[24] T. A. Plate. *Distributed Representations and Nested Compositional Structure*. PhD thesis, 1994.

[25] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*. 2011.

[26] Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of ICML*, 2011.

[27] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 2013.

[28] Ellen M. Voorhees. The trec question answering track. *Nat. Lang. Eng.*, 7(4):361–378, 2001.

[29] F.M. Zanzotto and L. Dell'Arciprete. Transducing sentences to syntactic feature vectors: an alternative way to "parse"? In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, 2013.

[30] F.M. Zanzotto and L. Dell'Arciprete. Distributed tree kernels. In *Proceedings of ICML*, 2012.