Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Networks

Richard Socher, Christopher D. Manning, Andrew Y. Ng
Department of Computer Science
Stanford University
richard@socher.org, manning@stanford.edu, ang@cs.stanford.edu

Abstract

Natural language parsing has typically been done with small sets of discrete categories such as NP and VP, but this representation does not capture the full syntactic nor semantic richness of linguistic phrases, and attempts to improve on this by lexicalizing phrases only partly address the problem at the cost of huge feature spaces and sparseness. To address this, we introduce a recursive neural network architecture for jointly parsing natural language and learning vector space representations for variable-sized inputs. At the core of our architecture are context-sensitive recursive neural networks (CRNN). These networks can induce distributed feature representations for unseen phrases and provide syntactic information to accurately predict phrase structure trees. Most excitingly, the representation of each phrase also captures semantic information: For instance, the phrases "decline to comment" and "would not disclose the terms" are close by in the induced embedding space. Our current system achieves an unlabeled bracketing F-measure of 92.1% on the Wall Street Journal dataset for sentences up to length 15.

1 Introduction

Syntactic parsing has been a central task in natural language processing because of its key importance in mediating between linguistic expression and meaning. For example, much work has shown the importance and usefulness of syntactic representations for subsequent tasks such as relation extraction and semantic role labeling [GP02].

Syntactic descriptions standardly use coarse discrete categories such as NP for noun phrases, or PP for prepositional phrases. However, the work of Klein and Manning [KM03] greatly improved parsing results through manual feature engineering and defining more fine-grained syntactic categories which better capture phrases with similar behavior. Recent successful work in discriminative parsing has also shown gains from careful engineering of features [TKC$^+$04, FKM08]. Another main direction for parsing gains has been through lexicalization [Col03, Cha00]. In lexicalized parsers each rule is associated with a lexical item. This is very effective for solving ambiguous attachment decisions, among other things, but requires models with vast numbers of features, and complex shrinkage estimation schemes to deal with the sparsity of observations of words. Instead of lexicalizing rules, Petrov et al. demonstrated how more fine-grained syntactic categories can be learned automatically [PBTK06]. However, subdividing a category like NP into 30 or 60 subcategories can only provide a very limited and poor representation of phrase meaning and semantic similarity.

Like Petrov, we tackle the problem of representing categories and phrases. However, unlike previous work on parsing, our architecture jointly learns how to parse and how to represent phrases in a continuous vector space of features, without manual engineering. The learned feature representations capture syntactic and semantic information and are the only input to the parser. In our experiments we show that these representations can help to inform parsing decisions, be used for learning category classifiers and capture interesting similarities between previously unseen phrases.

Our joint parsing and feature learning architecture is inspired by the recent reinvigoration of neural networks which jointly learn continuous feature representations for words [BDVJ03], and language models or classifiers. Collobert and Weston [CW08] have shown that neural networks can perform well on sequence labeling language processing tasks while also learning appropriate features. Their framework has interesting properties: (i) they use the same convolutional neural network architecture for several different tasks, (ii) their performance is competitive on many of these tasks and (iii) they introduce a method that can use large quantities of unlabeled text for learning syntactico-semantic distributed word representations. However, their model is lacking in that it cannot represent the recursive structure inherent in natural language. They partially circumvent this problem by using either independent window-based classifiers or a convolutional layer.

We introduce a neural network which learns recursive structure. Like Collobert and Weston we jointly learn the necessary features. However, our architecture allows us to not only embed single lexical units but also to embed unseen phrases and variable-sized inputs in a syntactically coherent order. It has already been shown [CW08, TRB10] that neural network induced word representations can be used successfully as features for tasks such as named entity recognition (NER). Our task for evaluating this architecture is unlabeled syntactic parsing of English sentences into their phrase structure.

At the core of our technique are recursive neural networks (RNN). RNNs have first been introduced by [GK96] to learn distributed representations of structured objects such as logical terms.

There has been previous work that used neural networks for parsing. Henderson [Hen03] introduced a left-corner parser to estimate probabilities of parsing decisions conditioned on the parsing history. The input to Henderson's model are pairs of frequent words and their part-of-speech (POS) tags. The latter come from a separate POS tagger. In contrast, our work incorporates POS tagging in a joint framework and lets the POS and phrase categories influence the feature representations. A parser which also uses recursive neural networks was presented by Menchetti et al. [MCFP05] who used recursive neural networks to rank possible phrase attachments in an incremental parser. They also ranked trees created by the Collins parser [Col03] and provide some analysis of the tree representations. In our approach we jointly learn phrase feature embeddings, parsing decisions as well as categories for each phrase and do not rely on another symbolic parsing algorithm.

There have been attempts at using neural networks to describe phrases. For instance, Elman [Elm91] used recurrent neural networks to create representations of sentences from a simple toy grammar and to analyze the linguistic expressiveness of the resulting representations. Words were represented as one-on vectors[1] which was feasible since the grammar only included a handful of words.

2 Parsing with Context-Sensitive Recursive Neural Networks

We propose a general learning architecture which aims to find structure in inputs of variable length. With the input being a sequence of words,[2] our goals are as follows: First, the model assigns to each word a continuous representation in an n-dimensional embedding space. These representations should capture syntactic and semantic information. They could be initialized randomly or pre-trained on unlabeled corpora to capture certain context statistics. Second, we want to use a neural network to induce a score for each pair of neighboring words which measures how likely these two words are to be combined into a phrase. The only input to this network will be the representations of the two words. As a side product of computing the syntactic score, the network will also *collapse* the two-words into an n-dimensional representation of the resulting phrase. This new phrase embedding then replaces the words in the sequence and will possibly become a child of another phrase spanning more words. This process continues until the whole input sentence is mapped to the embedding space.

[1] One-on vectors have a length equal to the vocabulary size. They contain all zeros except at the word's index where they are one.

[2] Note that, while we focus on syntactic parsing of natural language, recursive neural networks are applicable to structure prediction in other modalities.

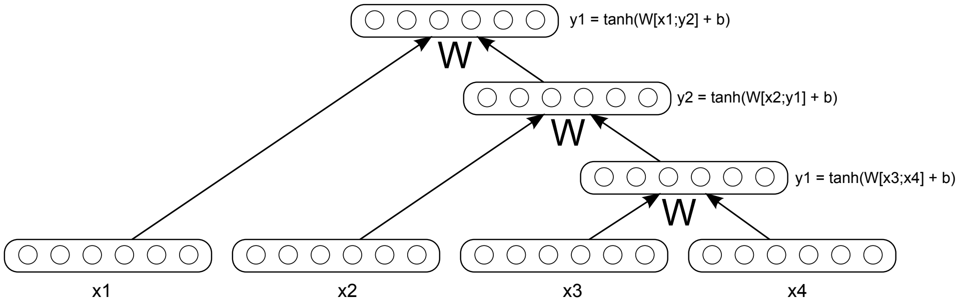

Figure 1: An example tree with a simple Recursive Neural Network: The same weight matrix is replicated and used to compute all non-leaf node representations. Leaf nodes are n-dimensional vector representations of words.

We first describe recursive neural networks and how they were used in previous approaches. We explain how they can be modified to jointly learn representations and predict structure. We then define the optimization objective and draw connections to structure prediction. Lastly, we show possible extensions for joint learning of the parser and category classifiers for phrases in the tree.

2.1 Recursive Neural Networks

Unlike standard neural networks, recursive neural networks (RNNs) are able to process structured inputs by repeatedly applying the same neural network at each node of a directed acyclic graph (DAG). In the past they have only been used in settings where another (often symbolic) component was first used to create directed acyclic graphs. These DAGs were subsequently given as input to the RNN. In such a setting, each non-leaf node of the DAG is associated with the same neural network. Fig.1 shows an example of a DAG which is a binary tree. In other words, all network replications share the same weights. The inputs to all these replicated feedforward networks are either given by using the children's labels to look up the associated representation or by their previously computed representation.

While in principle n-ary DAGS can be used as input to the RNN, we now focus on binary trees for ease of exposition.[3] Fig. 1 shows an instance of a RNN applied to a given binary tree. Assume we have an input of vectors $(x_1, \ldots, x_n)$ each of which has the same dimensionality $x_i \in \mathbb{R}^n$. In the original formulation these were one-on vectors [GK96]. Assume we are also given a binary tree in the form of branching triplets $(p \rightarrow c_1 c_2)$. Each such triplet denotes that a parent node p has two children and each c_k can be either an input x_i or a non-terminal node in the tree. For the example in Fig.1, we would get the triples $((y_1 \rightarrow x_3 x_4), (y_2 \rightarrow x_2 y_1), (y_1 \rightarrow x_1 y_2))$. Note that in order to replicate the neural network and compute node representations in a bottom up fashion, the parent must have the same dimensionality as each of its children. Given this tree structure, we can now compute activations for each node from the bottom up by

$$p \quad = \quad \tanh(W[c_1; c_2] + b), \tag{1}$$

where $c_1, c_2, p \in \mathbb{R}^{n \times 1}$, the term $[c_1; c_2]$ denotes the concatenation of the two child column vectors resulting in a $\mathbb{R}^{2n \times 1}$ vector and hence $W \in \mathbb{R}^{n \times 2n}$. At the top one can then stack some classification or regression layer.

2.2 Recursive Neural Networks for Structure Prediction

We now extend RNNs to construct the tree in a bottom-up fashion. We first compute the input to the parser in a similar way as [CW08]. Assume we are given a list of words. The input vectors $(x_1, \ldots, x_n)$ come from a look-up table of dimensionality $L \in \mathbb{R}^{n \times |V|}$, were $|V|$ is the size of the vocabulary. Each word in the input sequence has an associated index k into this table. Mathematically, the look-up can be seen as a simple projection layer where we use a binary vector b which is

[3] Focusing on binary trees will also allow us to develop efficient (cubic) dynamic programming algorithms for training and testing.

zero in all positions except at the kth index,

$$x_i = Lb_k \in \mathbb{R}^n, \tag{2}$$

All these word vectors are separate and in the order of their corresponding words.

2.2.1 Model 1: Greedy RNN

Given two such vectors, the goal of the network is twofold. First, it computes a new representation of the phrase which would combine the vectors of the two children into a new vector (such as p in Eq.1). Second, it scores how likely this is a correct phrase.

The algorithm takes the first pair of neighboring vectors, defines them as potential children of a phrase which spans both of these nodes: $(c_1, c_2) := (x_1, x_2)$, concatenates them and gives them as input to a neural network. This network can have multiple layers, as long as the top layer has the same dimensionality as each child/input vector. For simplicity of exposition, we restrict the network to one layer and use Eq.1 to compute the parent. Using the representation of this top layer, we can compute a score using a simple inner product with a row vector $W^{score} \in \mathbb{R}^{1\times n}$:

$$s_{1,2} = W^{score} p. \tag{3}$$

This score measures how well the two words are combined into a phrase. Using a set of training trees, we train the network so that correct parsing decisions (collapses into a phrase) will receive higher scores and incorrect decisions will receive lower scores (see Sec. 3 for our supervised learning algorithm).

After computing the score for the first pair, the network is shifted by one position and takes as input vectors $(c_1, c_2) = (x_2, x_3)$ and again computes a potential parent node and a score. This process repeats until it hits the last pair of words in the sentence: $(c_1, c_2) = (x_{n-1}, x_n)$. Next, it selects the pair which received the highest score. The scoring layer's output number is then removed and the top layer $p^{(n)}$ will represent this potential phrase and replace both children. For instance, consider the sequence (x_1, x_2, x_3, x_4) and assume the top score was obtained by the pair (x_3, x_4). After the first pass, the new sequence then consists of $(x_1, x_2, p_{(3,4)})$. The process repeats and treats the new vector $p_{(3,4)}$ like any other input vector. For instance, subsequent states could be either: $(x_1, p_{(2,(3,4))})$ or $(p_{(1,2)}, p_{(3,4)})$. Both states would then finish with a deterministic choice of collapsing the remaining two states into one parent to obtain $(p_{(1,(2,(3,4)))})$ (which coincides with the example tree in Fig.1) or $(p_{((1,2),(3,4))})$ respectively. Hence, the same network is *recursively* applied until all vectors are collapsed. The tree is then recovered by unfolding the collapsing decisions.

In our experiments, we will denote this greedy, bottom-up model as Model 1. Due to its greediness, this algorithm might not attain the optimal tree structure. In the following sections we will introduce extensions to this baseline.

2.2.2 Model 2: Greedy, Context-sensitive RNN

Parsing decisions are often context dependent. Hence, we can extend the basic RNN structure prediction model above by allowing the representations of context words to modify the parsing decision. The only difference to the above equations is in the first layer where we add as input the distributed representations of the t words to the left and t words to the right of the current constituent as context. The corresponding equations and an instance with $t = 1$ and two hidden layers is shown in Fig. 2.

2.2.3 Model 3: Greedy, Context-sensitive RNN and Category Classifier

One of the main advantages of our approach is that each phrase has associated with it a distributed feature representation. We can leverage on this representation by adding to each CRNN parent node (after removing the scoring layer) a simple softmax layer to predict class labels such as syntactic categories or named entity classes. Similarly to the contextualized collapsing decision, this extra layer may incorporate context. When minimizing the cross-entropy error of this softmax layer, the error will backpropagate and influence both the CRNN parameters as well as the word representations.

$$
\begin{aligned}
s &= W^{score} p \qquad (4)\\
p &= \tanh(W[x_{-1}; c_1; c_2; x_{+1};] + b^{(1)})
\end{aligned}
$$

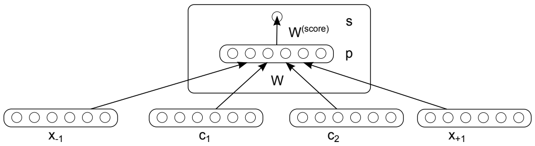

Figure 2: One context-sensitive recursive neural network which is replicated for each pair of possible input vectors. Dashed lines indicate context units of surrounding words. This network is different to the original RNN formulation in that it allows multiple layers at each node, predicts a score for being a correct phrasal constituent and uses context words.

2.2.4 Model 4: Global, Context-sensitive RNN and Category Classifier

Instead of greedily collapsing the best pairs, we can formulate a global, regularized risk objective in a max-margin framework [RBZ07, TKC$^+$04]. For each sentence, we can use dynamic programming parsing algorithms to efficiently find the globally optimal tree. Let the training data consist of (sentence, tree) pairs: (x_i, y_i). We denote the set of all possible trees that can be constructed from an input sentence as $A(x_i)$. We want to maximize the following objective:

$$J = \sum_i s(x_i, y_i) - \max_{y \in A(x_i)} \left(s(x_i, y) + \Delta(y, y_i) \right), \qquad (5)$$

where the structure loss Δ penalizes trees more when they deviate from the correct tree. We compute the total score of each tree as the sum of scores of each collapsing decision:

$$s(x_i, y_i) = \sum_{d \in T(y_i)} s_d(c_1, c_2). \qquad (6)$$

A span is a pair of indices which indicate the left and right most leaf nodes under a node in the tree. Let $T(y_i)$ denote the set of spans coming from all non-terminal nodes of the tree. Similar to [TKC$^+$04] we choose as our loss function a penalization of incorrect spans and add a penalization term λ to each incorrect decision:

$$\Delta(y, y_i) = \sum_{d \in T(y)} \lambda \mathbf{1}\{d \notin T(y_i)\}. \qquad (7)$$

3 Learning

We first describe the standard backprop algorithm and its modification for RNNs. We then describe the global optimization for our structure prediction problem.

3.1 Error Back-propagation Through Structure

In order to maximize the objective in Eq.5, we compute the derivative by using backpropagation through structure (BTS) [GK96]. Note that below we will use f as a sigmoid-like function instead of the specific case of $\tanh$. General backpropagation applies to RNNs such as that in Eq.4 through the normal use of δ's. The hidden layer backprop derivatives for tree i are:

$$\frac{\partial}{\partial W} J_i = \sum_{d \in T(y)} [c_1; c_2]^{(d)} [\delta^{(d+1)}]^T_{1:n} + CW, \qquad \delta^{(d)} = \left(W^T \delta^{(d+1)} \right) \circ f'([c_1; c_2]^{(d)}), \qquad (8)$$

where C is a weight regularization constant and we denote by $[\cdot]_{1:n}$, the first n elements of a vector. In the case of this δ, we consider the left child, for the right child we use indices $n + 1 : 2n$.

3.2 Subgradient Methods

Our objective J of Eq.5 is not differentiable due to the hinge loss. Therefore, we will generalize gradient ascent via the subgradient method [RBZ07] which computes a gradient-like direction called

Method	F1
Model 1 (Greedy RNN)	76.55
Model 2 (Greedy, context-sensitive RNN)	83.36
Model 3 (Greedy, context-sensitive RNN + category classifier)	87.05
Model 4 (Global, context-sensitive RNN + category classifier)	92.06
Left Corner PCFG, [MC97]	90.64
Current Implementation of the Stanford Parser, [KM03]	**93.98**

Table 1: Unlabeled Bracketing F-measure computed by evalb on section 22 of the WSJ dataset. Maximum sentence length is 15.

the subgradient. For any of our parameters such as W, the gradient becomes:

$$\frac{\partial J}{\partial W} = \sum_i \frac{\partial s(x_i, y_i)}{\partial W} - \frac{\partial s(x_i, y_{\max})}{\partial W} \tag{9}$$

Finding the highest scoring tree (including the penalization) can be achieved with the dynamic programming algorithm CKY. Even though our grammar is not context free and only consists of the rule $X \rightarrow XX$, we can use CKY-chart parsing to find the highest scoring tree. We attach the left and right context to each cell in the chart and whenever two charts are combined, the new cell inherits the contexts outside of their children cells and ignores the contexts between them. In other words, the context consists of the embedded representation of the words next to each cell's span. When two cells are combined the left cell will contribute its left context and the right cell its right context.

4 Experiments

In these preliminary experiments we show the type of semantic and syntactic information the phrase representations capture and how they can be used as input for making parsing decisions.

4.1 Parsing and Tagging

We pre-train the lookup table using the method of [CW08] on the English Gigaword corpus, section AFP which contains 611,506,174 words. Our vocabulary consists of all words that occurred more than twice in this corpus, keeping capitalization intact. This resulted in a vocabulary size of 15,942 words. All remaining words are mapped to an unknown word category. We also change all digits inside words to a generic number token '2'.

Our word and phrase representations are 100-dimensional. We train all models on the Wall Street Journal section of the Penn Tree Bank using the standard splits. Since we are still improving and developing the algorithm, we only present results on the development test section 22. To expedite development we restrict training and testing to sentences of maximum length 15.

Table 1 shows unlabeled bracketing F-measure for the different parsing models introduced above as well as two other systems. F-measure is the harmonic mean of precision and recall computed based on the spans of all nodes in the tree (see [MS99] for details). Unlike most previous systems, our parser does not provide a parent with information about the syntactic categories of its children. During training of the syntactic scoring network we only use their distributed representations. This shows that our learned, continuous representations capture enough syntactic information to make good parsing decisions.

While our parser does not yet perform as well as the current version of the Stanford parser, it performs respectably (1.92% difference in unlabeled F1 to the widely used Stanford parser) without any feature engineering or explicit usage of POS tags which are usually the main input for making parsing decisions.[4] It already outperforms previous PCFG based systems [MC97] where less dataset-specific features were implemented. Unlike many current parsers, our system is not tailored

[4] Note however, that the POS classifier uses the phrase representations and backpropagates its errors, modifying the parser's parameters.

Word	Initial Collobert&Weston embedding	After RNN training
the	a, its, an, this, his, their, UNK	an, The, no, some, A, these, another
and	,, or, but, as, –, ., that, for, in	or, but, And, But, &, least
in	at, on, from, for, over, after, and, as	In, from, into, since, for, For, like, with,
that	which, but, ,, and, –, as, for, or, about, if	what, who, if, this, some, which, If
said	added, says, –, while, but, reported, on	says, fell, added, did, rose, sold, reported
he	she, it, they, which, also, now, who, we	they, I, we, you, It, she, it, He, We, They
share	high, higher, business, market, current, stock, lower, increase, price, financial	increase, bank, income, industry, issue, state, sale, growth, unit, president
when	after, while, before, if, but, where, as	where, how, which, during, including

Table 2: Visualization of word embeddings after RNN training: Word embeddings are more likely to group according to their part-of-speech categories, even if sentence initial capitalization differs.

to the intricacies of the WSJ dataset. Furthermore, it provides more than the syntactic structure since each phrase representation also includes semantic features as we will see in the next section.

On a 2.6GHz laptop our current matlab implementation needs 72 seconds to parse 421 sentences of length less than 15.

Accuracy of part-of-speech (POS) tagging is 93.86% using the same train and test set as the parser. There are 27 different non-terminal categories such as noun phrase, verb phrase, prepositional phrase etc. The category classifier has an accuracy of 82%.

4.2 Modification to Word Embeddings

We use word embeddings that were pre-trained on unlabeled corpora [CW08]. In this section we qualitatively investigate how these word embeddings change after training our full recursive neural network model. Table 2 shows words from different parts-of-speech.

4.3 Phrase Feature Embeddings and Nearest Neighbor Phrases

The CRNN architecture is useful beyond its ability to parse sentences. Even incorrect parses may yield an embedding that captures information on the constituents. In this section we give a qualitative analysis of our learned phrase features. We first embed complete phrases from the WSJ dataset into the syntactico-semantic feature space. In table 3 we show some sentences and list their nearest neighbors in embedding space.

Our examples show that the learned embedding captures several interesting semantic or syntactic similarities between sentences or phrases. For instance (A) captures changes in financial quantities (all digits are mapped to 2). If one is interested in company statements, the group in (B) might be of interest.

Groups around (C) and (D) are especially interesting since they show that by considering the actual structure seemingly different lexical units end up with a similar meaning. For instance, *To decline to elaborate* is close in embedding space to *to not disclose*. Furthermore, for the phrase *a spokesman declined to elaborate*, the closest sentence is about another *spokesman [who] refused to identify....*

If one is interested in the task of named entity recognition and a rare city or village has never been seen during training, the embedding of the unknown word with a comma and possibly an abbreviation of a US state can be a very useful feature for identifying locations. This is captured in the embeddings as seen in group (E).

Examples in (F) capture positive news about companies and sentences in (G) suggest something bad has happened to a group of people.

5 Conclusion

We introduce a recursive neural network architecture which can be used to jointly parse sentences and map each phrase into a common feature space. While performance is still 1.9% behind that of the widely used Stanford parser, our parser can accurately recover unlabeled tree structures using only

Center Phrase and Nearest Neighbors	
(A) Sales grew almost 2 % to 222.2 million from 222.2 million.	
1. Sales surged 22 % to 222.22 billion yen from 222.22 billion.	
2. Revenue fell 2 % to 2.22 billion from 2.22 billion.	
3. Sales rose more than 2 % to 22.2 million from 22.2 million.	
4. Volume was 222.2 million shares, more than triple recent levels.	
(B) I had calls all night long from the States , he said.	
1. Our intent is to promote the best alternative, he says.	
2. We have sufficient cash flow to handle that, he said.	
3. Currently, average pay for machinists is 22.22 an hour, Boeing said.	
4. Profit from trading for its own account dropped, the securities firm said.	
(C) A DPC spokesman declined to elaborate on the group's new plan	
1. Those two offers were private and the spokesman refused to identify the bidding companies.	
2. Among the firms were Merrill Lynch & Co. and Dean Witter Reynolds Inc.	
3. The real key is to have the economy working and interest rates down.	
4. The market upheaval apparently has n't triggered any cash crunch yet.	
(D) Hess declined to comment.	**(E)** Columbia, S.C
1. PaineWebber declined to comment.	1. Greenville, Miss
2. Phoenix declined to comment.	2. UNK, Md
3. Campeau declined to comment.	3. UNK, Miss
4. Coastal would n't disclose the terms.	4. UNK, Calif
(F) Fujisawa gained 22 to 2,222	**(G)** We were lucky
1. Mochida advanced 22 to 2,222.	1. It was chaotic
2. Commerzbank gained 2 to 222.2.	2. We were wrong
3. Paris loved her at first sight.	3. People had died
4. Profits improved across Hess 's businesses.	4. They still are

Table 3: Visualization of phrase embeddings: Phrases and their nearest neighbors. Distance is often correlated with semantic and syntactic similarities of the phrases.

the distributed phrase representations. Furthermore, it provides semantic information even to previously unseen words and phrases. We provide examples of sentence embeddings that demonstrate possible applications of the learned features such as named entity recognition, semantic entailment, finding sentence polarity and information retrieval. Exploring these applications is part of the future work.

Acknowledgments

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government. This work was also supported in part by the DARPA Deep Learning program under contract number FA8650-10-C-7020. We thank Jiquan Ngiam, Quoc Le, Andrew Maas and David McClosky for their constructive feedback.

References

[BDVJ03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[Cha00] Eugene Charniak. A maximum-entropy-inspired parser. In *ACL*, pages 132–139, 2000.

[Col03] Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637, 2003.

[CW08] R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.

[Elm91] Jeffrey L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2-3):195–225, 1991.

[FKM08] Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. Efficient, feature-based, conditional random field parsing. In *ACL*, pages 959–967, 2008.

[GK96] C. Goller and A. Küchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the International Conference on Neural Networks (ICNN-96)*, 1996.

[GP02] Daniel Gildea and Martha Palmer. The necessity of parsing for predicate argument recognition. In *ACL*, pages 239–246, 2002.

[Hen03] James Henderson. Neural network probability estimation for broad coverage parsing. In *EACL*, pages 131–138, 2003.

[KM03] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *ACL*, pages 423–430, 2003.

[MC97] Christopher D. Manning and Bob Carpenter. Probabilistic parsing using left corner language models. In *In Proceedings of the 5th ACL/SIGPARSE International Workshop on Parsing Technologies. MIT*, pages 147–158. Kluwer Academic Publishers, 1997.

[MCFP05] Sauro Menchetti, Fabrizio Costa, Paolo Frasconi, and Massimiliano Pontil. Wide coverage natural language processing using kernel methods and neural networks for structured data. *Pattern Recognition Letters*, 26(12):1896–1906, 2005.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[PBTK06] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *ACL*, pages 433–440, 2006.

[RBZ07] Nathan Ratliff, J. Andrew (Drew) Bagnell, and Martin Zinkevich. (Online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*, March 2007.

[TKC$^+$04] Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. Max-margin parsing. In *In Proceedings of EMNLP*, 2004.

[TRB10] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pages 384–394, 2010.