Transition-based Dependency Parsing Using Recursive Neural Networks

Pontus Stenetorp[*]
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8439 Japan
pontus@stenetorp.se

Abstract

In this work, we present a general compositional vector framework for transition-based dependency parsing. The ability to use transition-based algorithms allows for the application of vector composition to a large set of languages where only dependency treebanks are available, as well as handling linguistic phenomena such as non-projectivities which pose problems for previously proposed methods. We introduce the concept of a Transition Directed Acyclic Graph that allows us to apply Recursive Neural Networks for parsing with existing transition-based algorithms. Our framework captures semantic relatedness between phrases similarly to a constituency-based counterpart from the literature, for example predicting that "a financial crisis", "a cash crunch" and "a bear market" are semantically similar. Currently, a parser based on our framework is capable of achieving 86.25% in Unlabelled Attachment Score for a well-established dependency dataset using only word representations as input, falling less than 2% points short of a previously proposed comparable feature-based model.

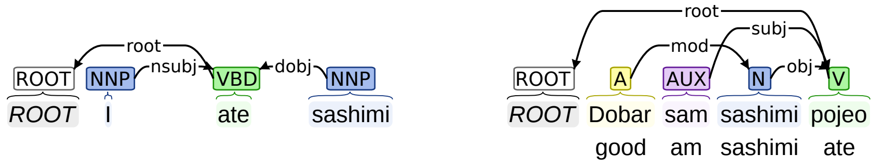
1 Introduction

Word representations induced using a variety of methods such as unsupervised slot-filling-style tasks (Collobert and Weston, 2008) or co-occurrence statistics (Turney and Pantel, 2010) are becoming a standard resource within the natural language processing community and it has been demonstrated that these representations can boost performance for tasks such as dependency parsing (Koo et al., 2008), named entity recognition and syntactic chunking (Turian et al., 2010).

Recently, a number of studies have used word representations to compose representations of phrases. Approaches for this task have ranged from manually defining composing operators (Mitchell and Lapata, 2008), learning linear composition functions (Baroni and Zamparelli, 2010) to learning non-linear composition functions (Socher et al., 2010), the last of which is the focus of this work. These composed representations have proven to be useful for tasks such as sentiment analysis (Socher et al., 2011) and compound similarity (Hermann and Blunsom, 2013).

While previous work has demonstrated how to learn non-linear composition functions for compounds and phrases in a constituency tree setting, there is a plethora of languages for which only dependency treebanks are available (Hajič et al., 2009; Haverinen et al., 2010). In this work we introduce a compositional vector framework for transition-based dependency parsing algorithms that can capture semantic relations between phrases. Quantitatively, at the current stage a parser based on our framework can achieve an Unlabelled Attachment Score (UAS) of 86.25% for dependency parsing on the CoNLL 2008 Shared Task Data Set using without using any manually crafted features.

[*]Currently at the University of Tokyo.



(a) English. (b) Croatian (“I ate good sashimi”).

Figure 1: Example sentences with labelled syntacto-semantic dependencies.

2 Dependency Parsing

The foundations for dependency grammars were laid down by Tesnière (1959), but it is only relatively recently that dependency parsing has become a major field of study. This increase in attention is partially due to community efforts such as the 2006 and 2007 CoNLL Shared Tasks (Buchholz and Marsi, 2006; Nilsson et al., 2007), but also since the dependency grammar framework easily lends itself to languages with free word order. As a result, annotated resources are readily available for a diverse, large set of languages. A key difference between constituency parsing and dependency parsing is in the unit that is considered when parsing. For constituency parsing words are composed into constituents (phrases) that are recursively composed into more complex constituents until they form a sentence, while dependency parsing establishes the relation between words.

Formally, a dependency graph $G = \{V, E\}$ for a sentence $S = \{w_1, \ldots, w_{|S|}\}$ is a labelled directed graph consisting of a set of vertices V, corresponding to words, and a set of labelled edges (arcs) E, corresponding to the dependency relations. Each edge can be expressed as $i \xrightarrow{l} j$ where i denotes the dependent, l the dependency type and j the head. Figure 1 shows two sentences annotated with dependency graphs. For the remainder of this work we will mainly consider unlabelled dependency graphs since it will simplify the explanation of our framework. We will, however, briefly explain how the framework can be extend to the labelled case.

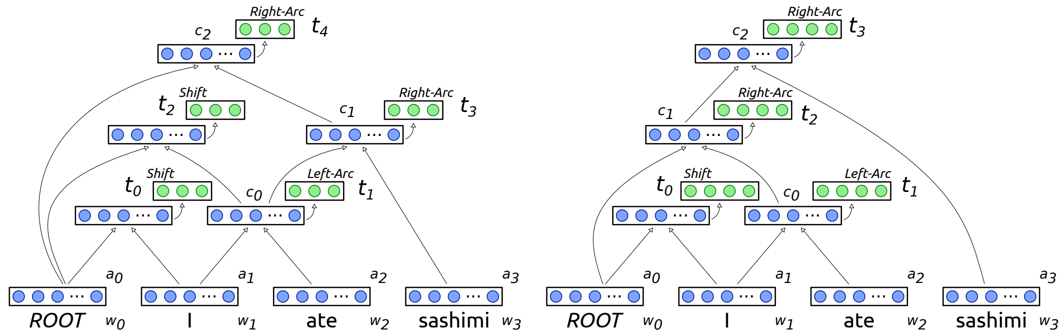
Dependency parsing algorithms generally fall within two major classes; graph-based and transition-based. Graph-based algorithms consider the complete set of possible dependency graphs and transition-based algorithms iteratively construct the dependency graph by processing the sentence sequentially.[1] While they both have strengths and weaknesses, in this work we will only consider transition-based algorithms, leaving parsing using compositional vectors for graph-based algorithms as future work.

For our example sentences (Figure 1), the key difference between the English (Figure 1a) and Croatian (Figure 1b) sentence is that the latter exhibits a crossing or *non-projective* edge.[2] This phenomenon can be handled for transition-based dependency parsing using approaches such as making algorithmic changes (Covington, 2001) or applying pre/post-processing heuristics to attempt to recover non-projective edges (Nivre and Nilsson, 2005). As the latter can trivially be done for any projective algorithm we will only demonstrate how our framework applies to a non-projective algorithmic variant.

3 Parsing Using Vector Composition

In this section we review a previously proposed model that uses vector composition for constituency parsing, introduce our framework for applying Recursive Neural Network (RNN) models to transition-based dependency parsing and then demonstrate how it can be applied to three previously proposed transition-based algorithms.

[1] For a thorough formal introduction to transition-based algorithms we refer the reader to Nivre (2008).
[2] Our use of Croatian is primarily motivated by the need for a plausible minimal non-projective example.



(a) *Arc-Standard*. (b) *Arc-Eager*.

Figure 2: Transition DAGs for two transition-based parsing algorithms for our English example sentence (Figure 1a). The variables correspond to those in Tables 1 and 2 respectively.

	Transition	Stack	Buffer	Arcs	Compositions
		[$ROOT_{w_0}$]	[I_{w_1}, $\mathrm{ate}_{w_2}, \dots$]		
t_0	*Shift* ⇒	[$ROOT_{w_0}$, I_{w_1}]	[ate_{w_2}, $\mathrm{sashimi}_{w_3}$]		
t_1	*Left-Arc* ⇒	[$ROOT_{w_0}$]	[ate_{w_2}, $\mathrm{sashimi}_{w_3}$]	I → ate	$c_0 = p([a_1; a_2])$
t_2	*Shift* ⇒	[$ROOT_{w_0}$, ate_{w_2}]	[$\mathrm{sashimi}_{w_3}$]		
t_3	*Right-Arc* ⇒	[$ROOT_{w_0}$]	[ate_{w_2}]	sashimi → ate	$c_1 = p([c_0; a_3])$
t_4	*Right-Arc* ⇒	[ ]	[$ROOT_{w_0}$]	ate → *ROOT*	$c_2 = p([a_0; c_1])$

Table 1: *Arc-Standard* transitions and compositions for the sentence in Figure 1a.

3.1 Constituency Trees

Socher et al. (2010) introduced an RNN model that operates on a Directed Acyclic Graph[3] (DAG) and demonstrated how it can be used to compose phrasal representations in a constituency parsing framework. Formally, the model uses a single composition matrix $W_C \in \mathbb{R}^{d \times Pd}$ where P is the number of the parents at each non-source vertex, d is the dimensionality and a representation $a_i \in \mathbb{R}^d$ is assigned to each word in a sentence. The algorithm then composes the representations by performing a topological traversal,[4] applying the composition matrix and a non-linearity f to the concatenated representation of the parents of each vertex $p = f(W_C[p_0; \dots; p_P]) \in \mathbb{R}^d$ resulting in a representation of the same dimensionality as its parents. The authors performed structured prediction of a binary constituency parse tree in a greedy fashion by scoring whether or not an adjacent pair of representations should be composed. Using their RNN model they demonstrated that it could achieve similar performance as a well-established feature-based constituency parser implementation and qualitatively showed that the resulting representations could be used to find semantically similar, yet syntactically different, phrases.

3.2 Dependency Trees

While the RNN model works well for constituency trees, it is conceptually difficult to apply it to dependency trees due to the core assumption of an RNN that all non-source vertices have an equal number of parents. For example, in Figure 1a the "ate" vertex has two parents while its child has one, making the use of a single composition matrix impossible without making major modifications to the model. Furthermore, in order to parse the sentence in Figure 1b we need an algorithm and grammar formalism capable of handling non-projectivities, which rules out the approach proposed by Socher et al. (2010).

Fortunately, these problems can be addressed if we are able to cast the vector composition as a transition-based dependency parsing problem. Transition-based algorithms employs a stack and a buffer that are jointly referred to as a *configuration*. A configuration is initialised by putting all the

[3] Since RNNs in their most general formulation operates on a DAG we use DAG terminology rather than the mixed DAG/tree terminology used by Socher et al. (2010). This leads to a few changes such as roots becoming sinks, leaves sources and the parent/child relationships being inverted.

[4] Parent vertices are visited before their children.

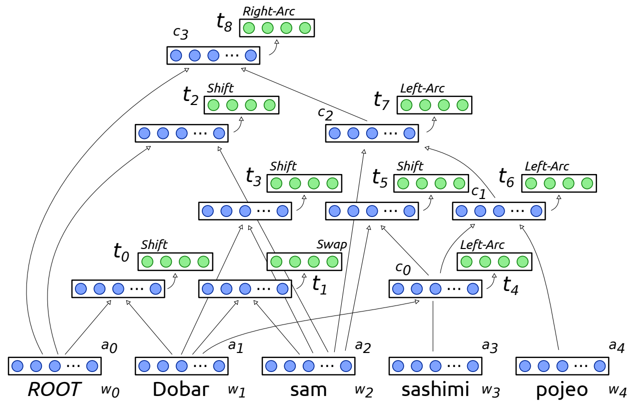


Figure 3: Transition DAG for the *Swp-Lazy* parsing algorithm for our Croatian example sentence (Figure 1b). The variables correspond to those in Table 3.

words of the sentence in the buffer and a dummy *ROOT* word w_0 on the stack. The algorithm then iteratively predicts a transition for the current configuration, manipulates the stack/buffer and create edges until it ends up in a terminating state.

Dependency parsing transitions can be categorised as either creating a dependency edge or simply manipulating the current configuration without creating an edge. The former are in a sense compositional since they attach a dependent to its head and establish their syntacto-semantic relationship. Using this distinction, we introduce the concept of a *Transition DAG* that is constructed when parsing. To predict the next transition, we apply the composition matrix W_C to the top-most elements of the stack and buffer to create a composed representation that we feed to a SoftMax layer that will predict the next transition. Intuitively, this determines whether two words are in a head/dependent relationship. By feeding the composed representation to an additional SoftMax layer we could also predict the dependency type. Repeating this iterative process produces phrasal representations while jointly parsing the sentence. We illustrate this for our example sentences (Figure 1) and the two well-established projective algorithms *Arc-Standard* and *Arc-Eager* as well as for the non-projective *Swp-Lazy* (Nivre et al., 2009) algorithm in Tables 1 to 3 and Figures 2 and 3.

Informally, the key difference between the *Arc-Standard* and *Arc-Eager* algorithms is that the former ensures that all dependents have been attached to a word before it is attached to its head, while the latter create edges *eagerly* by attaching a dependent to its head as early as possible in the parsing process. In practice, these two algorithms tend to achieve similar performance. *Swp-Lazy* is an extension to the *Arc-Standard* algorithm that allows for the re-ordering of words on the stack/buffer using a *Swap* transition to enable the processing of non-projective edges.

As *Arc-Standard* is arguably the most canonical of the three algorithms, we will walk through the steps to parse our English example sentence and create its transition DAG (Figure 2a) using oracle transitions (Table 1). From our initial configuration we determine that "I" is not to be attached to "*ROOT*" by predicting *Shift* t_0. As "I" is to be attached to the verb "ate" we predict *Left-Arc* t_1 and assign the composition of the head and the dependent as the new representation c_0 ("I ate"). Since the verb does not yet have all of its dependents attached we predict *Shift* t_2 instead of attaching the verb to "*ROOT*". "sashimi" is then assigned as the dependent of "ate" with a *Right-Arc* t_3 transition and we compose c_1 ("I ate sashimi"). Now that all dependents have been attached to the verb we attach it to "*ROOT*" using *Right-Arc* t_4 and arrive at a terminating state.

Since phrases and the transitions used to compose them are contextual in nature, we make one extension to our initial framework by allowing it to observe the top h representations on the stack/buffer and refer to this as the *horizon*.[5] Additionally, there is a need to represent the *ROOT* vertex and an empty stack/buffer to ensure that there are inputs for the composition matrix for all possible configurations and we simply introduce dummy vertices for each of the three cases and update their vector representations jointly with the word representations.

[5] This is similar to a context-aware RNN, but our context consists of representations from the stack/buffer rather than from neighbouring of words.

	Transition	Stack	Buffer	Arcs	Compositions
		[$ROOT_{w_0}$]	[I_{w_1}, $\mathrm{ate}_{w_2}, \ldots$]		
t_0	*Shift* ⇒	[$ROOT_{w_0}$, I_{w_1}]	[ate_{w_2}, $\mathrm{sashimi}_{w_3}$]		
t_1	*Left-Arc* ⇒	[$ROOT_{w_0}$]	[ate_{w_2}, $\mathrm{sashimi}_{w_3}$]	I → ate	$c_0 = p([a_1; a_2])$
t_2	*Right-Arc* ⇒	[$ROOT_{w_0}$, ate_{w_2}]	[$\mathrm{sashimi}_{w_3}$]	ate → *ROOT*	$c_1 = p([a_0; c_0])$
t_3	*Right-Arc* ⇒	[…, ate_{w_2}, $\mathrm{sashimi}_{w_3}$]	[ ]	sashimi → ate	$c_2 = p([c_1; a_3])$

Table 2: *Arc-Eager* transitions and compositions for the sentence in Figure 1a.

	Transition	Stack	Buffer	Arcs	Compositions
		[$ROOT_{w_0}$]	[Dobar_{w_1}, $\mathrm{sam}_{w_2}, \ldots$]		
t_0	*Shift* ⇒	[$ROOT_{w_0}$, Dobar_{w_1}]	[sam_{w_2}, $\mathrm{sashimi}_{w_3}, \ldots$]		
t_1	*Swap* ⇒	[$ROOT_{w_0}$]	[sam_{w_2}, $\mathrm{Dobar}_{w_1}, \ldots$]		
t_2	*Shift* ⇒	[$ROOT_{w_0}$, sam_{w_2}]	[Dobar_{w_1}, $\mathrm{sashimi}_{w_3}, \ldots$]		
t_3	*Shift* ⇒	[…, sam_{w_2}, Dobar_{w_1}]	[$\mathrm{sashimi}_{w_3}$, pajeo_{w_4}]		
t_4	*Left-Arc* ⇒	[$ROOT_{w_0}$, sam_{w_2}]	[$\mathrm{sashimi}_{w_3}$, pajeo_{w_4}]	Dobar → sashimi	$c_0 = p([a_1; a_3])$
t_5	*Shift* ⇒	[…, sam_{w_2}, $\mathrm{sashimi}_{w_3}$]	[pajeo_{w_4}]		
t_6	*Left-Arc* ⇒	[$ROOT_{w_0}$, sam_{w_2}]	[pajeo_{w_4}]	sashimi → pojeo	$c_1 = p([c_0; a_4])$
t_7	*Left-Arc* ⇒	[$ROOT_{w_0}$]	[pajeo_{w_4}]	sam → pojeo	$c_2 = p([a_2; c_1])$
t_8	*Right-Arc* ⇒	[ ]	[$ROOT_{w_0}$]	pojeo → *ROOT*	$c_3 = p([a_0; c_2])$

Table 3: *Swp-Lazy* transitions and compositions for the sentence in Figure 1b.

4 Experiments

In this section we describe our training strategies and preliminary experiments for dependency parsing and phrasal vector composition for a model created by applying our compositional vector framework to a transition-based algorithm.

4.1 Model and Training

As a first investigation into whether our framework is suitable for constructing models for dependency parsing and vector composition, we create a simple model using the *Arc-Standard* algorithm that we used for our example in the previous section. We limit ourselves to using global weight updates and apply a greedy search strategy at test time by selecting the most likely transition according to the SoftMax layer for each iteration.

We generate gold transition DAGs, such as the one in Figure 2a, for each sentence in the training data using oracle transitions. Given the gold transition DAGs we can then train our model using the same back-propagation through structure (Goller and Kuchler, 1996) strategy that was originally proposed for RNNs[6] by minimising the cross-entropy loss for the transition actions, propagating the errors from the sinks to the sources.

As initial word representation we use the non-scaled, unnormalised 200-dimensional vectors pre-trained vectors provided by Turian et al. (2010). When initialising the composition weight matrix W_C we use a strategy similar to Socher et al. (2013) and add a constant to the diagonal elements of each block sub-matrix of W_C. This factor depends on the distance to the top of the stack/buffer of the representation being affected by the block sub-matrix and is defined as $\frac{1}{i+2}$ where i is is the distance to the top of stack/buffer. This leads to gradually less initial influence for distant representations on the stack/buffer ($[\frac{1}{2}, \frac{1}{3}, \ldots]$). All other weights, including the representations for the empty-stack/buffer and *ROOT*, are initialised using the normalised initialisation heuristic proposed by Glorot and Bengio (2010).

To minimise our loss function we use mini-batches combined with the diagonal version of AdaGrad (Duchi et al., 2011). AdaGrad is particularly helpful for infrequent words since it adapts the learning rate for each parameter and provides larger updates for rare parameters. We tune our set of hyper-parameters on a development set and for the final evaluation we use a batch size of 64, a learning rate of 0.1 and apply a weight-decay regularisation of 10^{-4} for the composition matrix/SoftMax layer and 10^{-6} for the word representations. Lastly, through preliminary experiments on the development set we found that a horizon of three performed well.

[6] Further RNN training details can be found in Socher et al. (2010).

	Model	UAS
(1)	*This work*	86.25%
(2)	Surdeanu and Manning (2010)	88.06%
(3)	Johansson and Nugues (2008)	**92.45%**

(a) UAS scores for a selection of models.

(a)	a financial crisis
1st	a cash crunch
2nd	a bear market
(b)	hammer out their own plan
1st	work out their own compromise
2nd	enact the cut this year
(c)	to run their computerized trading strategies
1st	to determine buy and sell orders
2nd	to pick up more shares today
(d)	from $ 142.7 million , or 78 cents a share
1st	from $ 367.1 million , or $ 2.05 a share
2nd	from the sale of its First Chicago Investment Advisors unit

(b) Nearest neighbour phrases.

Table 4: Results from our preliminary experiments.

4.2 Evaluation and Parameter Tuning

We evaluate our method quantitatively for dependency parsing on the Wall Street Journal portion of the CoNLL 2008 Shared Task Data Set (Surdeanu et al., 2008) using the pre-defined training, development and test splits. Development and hyper-parameter optimisation uses only the development set so that the test set is only used to generate the final results. We use the official evaluation script provided by the shared task organisers and UAS as our performance measure. The choice of UAS as opposed to Labelled Attachment Score (LAS) is primarily motivated by the fact that we are still developing and improving our framework. Since the particular transition-based algorithm we use as the basis for our model is projective, we apply the MaltParser (Nivre et al., 2007) set of dependency parsing tools to projectivise the dependency trees used for training.

To qualitatively evaluate the ability of our model to compose phrasal vector representations we calculate the nearest neighbour phrases in the representation space. We do this by training our model only on the training set, and then parse the sentences of the development set. This yields representations for each phrase in the development set and we qualitatively analyse the sentences which were deemed to be most similar according to our model using the cosine similarity. Since our choice of evaluation corpus uses the same underlying texts as previous work, we should be able to expect similar patterns for phrasal similarity.

5 Results and Analysis

In Table 4a we can find our performance in relation to both an *Arc-Standard* model[7] (2) that uses the same transition-based algorithm as our model and the best performing model from the shared task (3). The performance of our model (1) is approaching that of the comparable feature-based model, which is highly encouraging, especially since our model employs a pure feature-learning strategy. However, as expected due to limiting the complexity of our model, it fails to achieve results competitive with the state-of-the-art. But we need to keep in mind that we are yet to apply more advanced training strategies and that other methods have access to manually crafted features for both words and tree structures. In regards to computational complexity, our model can parse slightly over 60 sentences per second on a single-core modern desktop computer.

In Table 4b we present four phrases and their two closest neighbours according to our model. For (a) our model successfully captures that the phrases are similar since they describe times of financial distress. (b) describe taking action as a group and (c) express trading with financial instruments. However, (d) presents a case where describing the intuition behind the model output becomes difficult. The two first phrases appears to describe the distribution of dividends while the latter describes the sale of a portion of a company. Potentially this could be due to some sense of profit being expressed in all three phrases, but as with all qualitatively analysis it is best to be cautious and not to draw far-reaching conclusions. Overall, we find that our model does indeed capture similarity comparable to what has qualitatively been observed for previous compositional vector parsing models.

[7] Results from Surdeanu and Manning (2010), the original model publication pre-dates the corpus.

An observation we made during development was that although Socher et al. (2010) used a frequency cut-off and replaced infrequent words with “unknown” place holders and normalised digits, we found no tangible performance benefits from this line of pre-processing. This finding is encouraging since models such as ours without explicitly defined word surface features are expected to have difficulties when encountering words that were not observed in the training data.

6 Future Work

While it is encouraging that our model can learn useful feature representations and perform as high as 86.25% in UAS, it is justifiable to investigate the possibility of training strategies more advanced than global updates with greedy search to strengthen our model (Huang et al., 2012). In fact, it may turn out that doing so may improve the phrasal representations which would ideally be evaluated for an extrinsic task. However, we leave such experimental analysis for future work.

Having established a base-line model that uses a unified representation of phrases, there are a number of interesting possibilities for extensions to the parsing model itself that would not be possible using previous approaches to transition-based parsing. One could, for example, try to eliminate the need for a horizon cut-off by compressing the stack/buffer in a fashion similar to recursive auto-encoders to allow for the model to learn how to observe contextual information rather than tuning a horizon hyper-parameter.

As we rely on the word representations induced by Turian et al. (2010) for pre-training, it is meaningful to ask to what extent this choice has an effect on parsing performance. While there have been investigations into the effects of different algorithms and representation dimensionalities (Turian et al., 2010) as well as corpus sizes (Stenetorp et al., 2012) for tasks such as named entity recognition, there has been no such analysis published for constituency or dependency parsing. Ideally, we should therefore investigate alternative models, such as the recently introduced Skip-gram model (Mikolov et al., 2013), to induce a range of alternative representations. Given the modularity of neural network-based models it would then be straightforward to re-train the model to evaluate the performance impact of the various representations.

Also, while our model is able to compose phrasal representations, we have observed that it has difficulties creating representations for full sentences. This problem seems to be inherent in the RNN model since it uses a single composition matrix. Thus, as we compose each representation in a sentence the model will tend to “wash out” representations that were composed early on in the parsing process. Ideally, the model would be informed about what information is the most salient, but how to do this remains an open research question.

Lastly, and most encouraging, we have the possibility of evaluating our model for the 19 languages from the CoNLL 2006 and 2007 Shared Tasks. One of the possible obstacles for this has already been cleared with the recently published multilingual Polyglot word embeddings (Al-Rfou et al., 2013), our work constitutes the second piece necessary for applying compositional vector parsing to a much wider set of languages than was previously possible.

7 Conclusions

In this work we have introduced the first framework for applying vector composition to transition-based parsing. By applying this framework to an existing transition-based algorithm we constructed a model that is capable of learning to compose phrasal representations while jointly parsing dependency tree structures, thus overcoming the weakness of previously proposed models that can only be trained using constituency tree annotations. The resulting phrase representations are jointly learned along with the weights to perform transition-based dependency parsing and are capable of capturing semantic relatedness such as syntactically different phrases expressing financial trading. We have also demonstrated that our model can learn to parse dependency structures without the need for hand-crafted features, albeit as-of-yet falling short of performing on-par with existing state-of-the-art feature-based models.

Acknowledgments

We would like to thank Kazuma Hashimoto, Hubert Soyer and Richard Socher for several helpful discussions regarding the ideas of this paper. Also, we would like to thank Goran Topić for providing us with a minimal non-projective Croatian sentence[8] and Sampo Pyysalo for his feedback on drafts of the final manuscript. Lastly, we would like to thank the anonymous reviewers for their comments and feedback. An earlier version of this work was rejected as a short-paper on semantics submitted to the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed Word Representations for Multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, 2013.

Marco Baroni and Roberto Zamparelli. Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010.

Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, 2006.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 2008.

Michael A Covington. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, 2001.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 2010 International Conference on Artificial Intelligence and Statistics*, 2010.

Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, 1996.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, 2009.

Katri Haverinen, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Filip Ginter, and Tapio Salakoski. Treebanking Finnish. In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories*, 2010.

Karl Moritz Hermann and Phil Blunsom. The Role of Syntax in Vector Space Models of Compositional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

Liang Huang, Suphan Fayong, and Yang Guo. Structured Perceptron with Inexact Search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2012.

Richard Johansson and Pierre Nugues. Dependency-based Syntactic–Semantic Analysis with PropBank and NomBank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, 2008.

Terry Koo, Xavier Carreras, and Michael Collins. Simple Semi-supervised Dependency Parsing. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, 2008.

[8] Interestingly, a non-projective sentence for any language must contain at least four words.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *the First International Conference on Learning Representations*, 2013.

Jeff Mitchell and Mirella Lapata. Vector-based Models of Semantic Composition. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, 2008.

Jens Nilsson, Sebastian Riedel, Deniz Yuret, et al. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the Eleventh Conference on Computational Natural Language Learning*, 2007.

Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.

Joakim Nivre and Jens Nilsson. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 2007.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th international conference on parsing technologies*, 2009.

Richard Socher, Christopher D. Manning, and Andrew Y Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, 2010.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 2011.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing With Compositional Vector Grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

Pontus Stenetorp, Hubert Soyer, Sampo Pyysalo, Sophia Ananiadou, and Takashi Chikayama. Size (and Domain) Matters: Evaluating Semantic Word Space Representations for Biomedical Text. In *Proceedings of the 5th International Symposium on Semantic Mining in Biomedicine*, 2012.

Mihai Surdeanu and Christopher D. Manning. Ensemble models for dependency parsing: Cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, 2008.

Lucien Tesnière. *Eléments de syntaxe structurale*. Klincksieck Paris, 1959.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010.

Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188, 2010.