Experiments with a Higher-Order Projective Dependency Parser

Xavier Carreras
Massachusetts Institute of Technology (MIT)
Computer Science and Artificial Intelligence Laboratory (CSAIL)
32 Vassar St., Cambridge, MA 02139
carreras@csail.mit.edu

Abstract

We present experiments with a dependency parsing model defined on rich factors. Our model represents dependency trees with factors that include three types of relations between the tokens of a dependency and their children. We extend the projective parsing algorithm of Eisner (1996) for our case, and train models using the averaged perceptron. Our experiments show that considering higher-order information yields significant improvements in parsing accuracy, but comes at a high cost in terms of both time and memory consumption. In the multilingual exercise of the CoNLL-2007 shared task (Nivre et al., 2007), our system obtains the best accuracy for English, and the second best accuracies for Basque and Czech.

1 Introduction

Structured prediction problems usually involve models that work with factored representations of structures. The information included in the factors determines the type of features that the model can exploit. However, richer representations translate into higher complexity of the inference algorithms associated with the model.

In dependency parsing, the basic *first-order* model is defined by a decomposition of a tree into head-modifier dependencies. Previous work extended this basic model to include *second-order* relations—i.e. dependencies that are adjacent to the main dependency of the factor. Specifically, these approaches considered sibling relations of the modifier token (Eisner, 1996; McDonald and Pereira, 2006). In this paper we extend the parsing model with other types of second-order relations. In particular, we incorporate relations between the head and modifier tokens and the children of the modifier.

One paradigmatic case where the relations we consider are relevant is PP-attachment. For example, in "They sold 1,210 cars in the U.S.", the ambiguity problem is to determine whether the preposition "in" (which governs "the U.S.") is modifying "sold" or "cars", the former being correct in this case. It is generally accepted that to solve the attachment decision it is necessary to look at the head noun within the prepositional phrase (i.e., "U.S." in the example), which has a grand-parental relation with the two candidate tokens that the phrase may attach—see e.g. (Ratnaparkhi et al., 1994). Other ambiguities in language may also require consideration of grand-parental relations in the dependency structure.

We present experiments with higher-order models trained with averaged perceptron. The second-order relations that we incorporate in the model yield significant improvements in accuracy. However, the inference algorithms for our factorization are very expensive in terms of time and memory consumption, and become impractical when dealing with many labels or long sentences.

2 Higher-Order Projective Models

A dependency parser receives a sentence $\mathbf{x}$ of n tokens, and outputs a labeled dependency tree y. In the tree, a labeled dependency is a triple $\langle h, m, l \rangle$, where $h \in [0 \ldots n]$ is the index of the head token,

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 957–961,
Prague, June 2007.

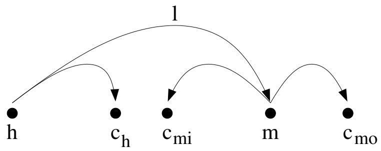

Figure 1: A factor in the higher-order parsing model.

$m \in [1 \ldots n]$ is the index of the modifier token, and $l \in [1 \ldots L]$ is the label of the dependency. The value $h = 0$ is used for dependencies where the head is a special *root-symbol* of the sentence. We denote by $\mathcal{T}(\mathbf{x})$ the set of all possible dependency structures for a sentence $\mathbf{x}$. In this paper, we restrict to projective dependency trees. The dependency tree computed by the parser for a given sentence is:

$$y^*(\mathbf{x}) = \arg\max_{y \in \mathcal{T}(\mathbf{x})} \sum_{f \in y} \text{score}(\mathbf{w}, \mathbf{x}, f)$$

The parsing model represents a structure y as a set of factors, $f \in y$, and scores each factor using parameters $\mathbf{w}$. In a first-order model a factor corresponds to a single labeled dependency, i.e. $f = \langle h, m, l \rangle$. The features of the model are defined through a feature function $\phi_1(\mathbf{x}, h, m)$ which maps a sentence together with an unlabeled dependency to a feature vector in $\mathbb{R}^{d_1}$. The parameters of the model are a collection of vectors $\mathbf{w}_1^l \in \mathbb{R}^{d_1}$, one for each possible label. The first-order model scores a factor as $\text{score}_1(\mathbf{w}, \mathbf{x}, \langle h, m, l \rangle) = \phi_1(\mathbf{x}, h, m) \cdot \mathbf{w}_1^l$.

The higher-order model defined in this paper decomposes a dependency structure into factors that include children of the head and the modifier. In particular, a factor in our model is represented by the signature $f = \langle h, m, l, c_h, c_{mi}, c_{mo} \rangle$ where, as in the first-order model, h, m and l are respectively the head, modifier and label of the main dependency of the factor; c_h is the child of h in $[h \ldots m]$ that is closest to m; c_{mi} is child of m inside $[h \ldots m]$ that is furthest from m; c_{mo} is the child of m outside $[h \ldots m]$ that is furthest from m. Figure 1 depicts a factor of the higher-order model, and Table 1 lists the factors of an example sentence. Note that a factor involves a main labeled dependency and three adjacent unlabeled dependencies that attach to children of h and m. Special values are used when either of these children are null.

The higher-order model defines additional

	m	h	c_h	c_{mi}	c_{mo}
They	1	2	-	-	-
sold	2	0	-	1	5
1,200	3	4	-	-	-
cars	4	2	-	3	-
in	5	2	4	-	7
the	6	7	-	-	-
U.S.	7	5	-	6	-

Table 1: Higher-order factors for an example sentence. For simplicity, labels of the factors have been omitted. A first-order model considers only $\langle h, m \rangle$. The second-order model of McDonald and Pereira (2006) considers $\langle h, m, c_h \rangle$. For the PP-attachment decision (factor in row 5), the higher-order model allows us to define features that relate the verb ("sold") with the content word of the prepositional phrase ("U.S.").

second-order features through a function $\phi_2(\mathbf{x}, h, m, c)$ which maps a head, a modifier and a child in a feature vector in $\mathbb{R}^{d_2}$. The parameters of the model are a collection of four vectors for each dependency label: $\mathbf{w}_1^l \in \mathbb{R}^{d_1}$ as in the first-order model; and $\mathbf{w}_{\text{h}}^l$, $\mathbf{w}_{\text{mi}}^l$ and $\mathbf{w}_{\text{mo}}^l$, all three in $\mathbb{R}^{d_2}$ and each associated to one of the adjacent dependencies in the factor. The score of a factor is:

$$\begin{aligned}\text{score}_2(\mathbf{w}, \mathbf{x}, \langle h, m, l, c_h, c_{mi}, c_{mo} \rangle) = \\ \phi_1(\mathbf{x}, h, m) \cdot \mathbf{w}_1^l + \phi_2(\mathbf{x}, h, m, c_h) \cdot \mathbf{w}_{\text{h}}^l + \\ \phi_2(\mathbf{x}, h, m, c_{mi}) \cdot \mathbf{w}_{\text{mi}}^l + \phi_2(\mathbf{x}, h, m, c_{mo}) \cdot \mathbf{w}_{\text{mo}}^l\end{aligned}$$

Note that the model uses a common feature function for second-order relations, but features could be defined specifically for each type of relation. Note also that while the higher-order factors include four dependencies, our modelling choice only exploits relations between the main dependency and secondary dependencies. Considering relations between secondary dependencies would greatly increase the cost of the associated algorithms.

2.1 Parsing Algorithm

In this section we sketch an extension of the projective dynamic programming algorithm of Eisner (1996; 2000) for the higher-order model defined above. The time complexity of the algorithm is $O(n^4 L)$, and the memory requirements are $O(n^2 L + n^3)$. As in the Eisner approach, our algorithm visits sentence spans in a bottom up fashion, and constructs a chart with two types of dynamic programming structures, namely open and closed structures—see Figure 2 for a diagram. The dynamic programming structures are:

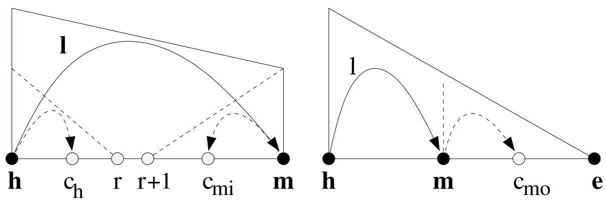

Figure 2: Dynamic programming structures used in the parsing algorithm. The variables in boldface constitute the index of the chart entry for a structure; the other variables constitute the back-pointer stored in the chart entry. Left: an open structure for the chart entry $[h, m, l]_O$; the algorithm looks for the r, c_h and c_{mi} that yield the optimal score for this structure. Right: a closed structure for the chart entry $[h, e, m]_C$; the algorithm looks for the l and c_{mo} that yield the optimal score.

- Open structures: For each span from s to e and each label l, the algorithm maintains a chart entry $[s, e, l]_O$ associated to the dependency $\langle s, e, l \rangle$. For each entry, the algorithm looks for the optimal splitting point r, sibling c_h and grand-child c_{mi} using parameters $\mathbf{w}^l_1$, $\mathbf{w}^l_{\mathrm{h}}$ and $\mathbf{w}^l_{\mathrm{mi}}$. This can be done in $O(n^2)$ because our features do not consider interactions between c_h and c_{mi}. Similar entries $[e, s, l]_O$ are maintained for dependencies headed at e.
- Closed structures: For each span from s to e and each token $m \in [s \ldots e]$, the algorithm maintains an entry $[s, e, m]_C$ associated to a partial dependency tree rooted at s in which m is the last modifier of s. The algorithm chooses the optimal dependency label l and grand-child c_{mo} in $O(nL)$, using parameters $\mathbf{w}^l_{\mathrm{mo}}$. Similar entries $[e, s, m]_C$ are maintained for dependencies headed at e.

We implemented two variants of the algorithm. The first forces the root token to participate in exactly one dependency. The second allows many dependencies involving the root token. For the single-root case, it is necessary to treat the root token differently than other tokens. In the experiments, we used the single-root variant if sentences in the training set satisfy this property. Otherwise we used the multi-root variant.

2.2 Features

The first-order features $\phi_1(\mathbf{x}, h, m)$ are the exact same implementation as in previous CoNLL system (Carreras et al., 2006). In turn, those features were inspired by successful previous work in first-order dependency parsing (McDonald et al., 2005). The most basic feature patterns consider the surface form, part-of-speech, lemma and other morphosyntactic attributes of the head or the modifier of a dependency. The representation also considers complex features that exploit a variety of conjunctions of the forms and part-of-speech tags of the following items: the head and modifier; the head, modifier, and any token in between them; the head, modifier, and the two tokens following or preceding them.

As for the second-order features, we again base our features with those of McDonald and Pereira (2006), who reported successful experiments with second-order models. We add some patterns to their features. Let dir be "right" if $h < m$, and "left" otherwise; let $\mathrm{form}(x_i)$ and $\mathrm{cpos}(x_i)$ return the surface form and coarse part-of-speech of token x_i, respectively. The definition of $\phi_2(\mathbf{x}, h, m, c)$ is:

- $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{cpos}(x_m) \cdot \mathrm{cpos}(x_c)$
- $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{cpos}(x_c)$
- $dir \cdot \mathrm{cpos}(x_m) \cdot \mathrm{cpos}(x_c)$
- $dir \cdot \mathrm{form}(x_h) \cdot \mathrm{form}(x_c)$
- $dir \cdot \mathrm{form}(x_m) \cdot \mathrm{form}(x_c)$
- $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{form}(x_c)$
- $dir \cdot \mathrm{cpos}(x_m) \cdot \mathrm{form}(x_c)$
- $dir \cdot \mathrm{form}(x_h) \cdot \mathrm{cpos}(x_c)$
- $dir \cdot \mathrm{form}(x_m) \cdot \mathrm{cpos}(x_c)$

3 Experiments and Results

We report experiments with higher-order models for the ten languages in the multilingual track of the CoNLL-2007 shared task (Nivre et al., 2007).[1]

In all experiments, we trained our models using the averaged perceptron (Freund and Schapire, 1999), following the extension of Collins (2002) for structured prediction problems. To train models, we used "projectivized" versions of the training dependency trees.[2]

[1] We are grateful to the providers of the treebanks that constituted the data for the shared task (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003).

[2] We obtained projective trees for training sentences by running the projective parser with an oracle model (that assigns a score of +1 to correct dependencies and -1 otherwise).

	Catalan	Czech	English
First-Order, no averaging	82.07	68.98	83.75
First-Order	86.15	75.96	87.54
Higher-Order, c_h	87.50	77.15	88.70
Higher-Order, c_h c_{mo}	87.68	77.62	89.28
Higher-Order, c_h c_{mi} c_{mo}	88.04	78.09	89.59

Table 2: Labeled attachment scores on validation data (~10,000 tokens per language), for different models that exploit increasing orders of factorizations.

3.1 Impact of Higher-Order Factorization

Our first set of experiments looks at the performance of different factorizations. We selected three languages with a large number of training sentences, namely Catalan, Czech and English. To evaluate models, we held out the training sentences that cover the first 10,000 tokens; the rest was used for training.

We compared four models at increasing orders of factorizations. The first is a first-order model. The second model is similar to that of McDonald and Pereira (2006): a factor consists of a main labeled dependency and the head child closest to the modifier (c_h). The third model incorporates the modifier child outside the main dependency in the factorization (c_{mo}). Finally, the last model incorporates the modifier child inside the dependency span (c_{mi}), thus corresponding to the complete higher-order model presented in the previous section.

Table 2 shows the accuracies of the models on validation data. Each model was trained for up to 10 epochs, and evaluated at the end of each epoch; we report the best accuracy of these evaluations. Clearly, the accuracy increases as the factors include richer information in terms of second-order relations. The richest model obtains the best accuracy in the three languages, being much better than that of the first-order model. The table also reports the accuracy of an unaveraged first-order model, illustrating the benefits of parameter averaging.

3.2 Results on the Multilingual Track

We trained a higher-order model for each language, using the averaged perceptron. In the experiments presented above we observed that the algorithm does not over-fit, and that after two or three training epochs only small variations in accuracy occur. Based on this fact, we designed a criterion to train models: we ran the training algorithm for up to three days of computation, or a maximum of 15 epochs. For Basque, Chinese and Turkish we could complete the 15 epochs. For Arabic and Catalan, we could only complete 2 epochs. Table 3 reports the performance of the higher-order projective models on the ten languages of the multilingual track.

	training		test	
	sent./min.	mem.	UAS	LAS
Arabic	1.21	1.8GB	81.48	70.20
Basque	33.15	1.2GB	81.08	75.73
Catalan	5.50	1.7GB	92.46	87.60
Chinese	1461.66	60MB	86.20	80.86
Czech	18.19	1.8GB	85.16	78.60
English	15.57	1.0GB	90.63	89.61
Greek	8.10	250MB	81.37	73.56
Hungarian	5.65	1.6GB	79.92	75.42
Italian	12.44	900MB	87.19	83.46
Turkish	116.55	600MB	82.41	75.85
Average	-	-	84.79	79.09

Table 3: Performance of the higher-order projective models on the multilingual track of the CoNLL-2007 task. The first two columns report the speed (in sentences per minute) and memory requirements of the training algorithm—these evaluations were made on the first 1,000 training sentences with a Dual-Core AMD Opteron™ Processor 256 at 1.8GHz with 4GB of memory. The last two columns report unlabelled (UAS) and labelled (LAS) attachment scores on test data.

4 Conclusion

We have presented dependency parsing models that exploit higher-order factorizations of trees. Such factorizations allow the definition of second-order features associated with sibling and grand-parental relations. For some languages, our models obtain state-of-the-art results.

One drawback of our approach is that the inference algorithms for higher-order models are very expensive. For languages with many dependency labels or long sentences, training and parsing becomes impractical for current machines. Thus, a promising line of research is the investigation of methods to efficiently incorporate higher-order relations in discriminative parsing.

Acknowledgments

I am grateful to Terry Koo, Amir Globerson and Michael Collins for their helpful comments relating this work, and to the anonymous reviewers for their suggestions. A significant part of the system and the code was based on my previous system in the CoNLL-X task, developed with Mihai Surdeanu and Lluís Màrquez at the UPC. The author was supported by the Catalan Ministry of Innovation, Universities and Enterprise.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

X. Carreras, M. Surdeanu, and L. Màrquez. 2006. Projective dependency parsing with perceptron. In *Proc. CoNLL-X*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP-2002*.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. COLING*.

J. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In H. C. Bunt and A. Nijholt, editors, *New Developments in Natural Language Parsing*, pages 29–62. Kluwer Academic Publishers.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL-2006*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. ACL*.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of EMNLP-CoNLL*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

A. Ratnaparkhi, J. Reinar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proc. of the ARPA Workshop on Human Language Technology*.