Incremental Non-Projective Dependency Parsing

Joakim Nivre
Växjö University, School of Mathematics and Systems Engineering
Uppsala University, Department of Linguistics and Philology
joakim.nivre@{msi.vxu.se,lingfil.uu.se}

Abstract

An open issue in data-driven dependency parsing is how to handle non-projective dependencies, which seem to be required by linguistically adequate representations, but which pose problems in parsing with respect to both accuracy and efficiency. Using data from five different languages, we evaluate an incremental deterministic parser that derives non-projective dependency structures in $O(n^2)$ time, supported by SVM classifiers for predicting the next parser action. The experiments show that unrestricted non-projective parsing gives a significant improvement in accuracy, compared to a strictly projective baseline, with up to 35% error reduction, leading to state-of-the-art results for the given data sets. Moreover, by restricting the class of permissible structures to limited degrees of non-projectivity, the parsing time can be reduced by up to 50% without a significant decrease in accuracy.

1 Introduction

Data-driven dependency parsing has been shown to give accurate and efficient parsing for a wide range of languages, such as Japanese (Kudo and Matsumoto, 2002), English (Yamada and Matsumoto, 2003), Swedish (Nivre et al., 2004), Chinese (Cheng et al., 2004), and Czech (McDonald et al., 2005). Whereas most of the early approaches were limited to strictly *projective* dependency structures, where the projection of a syntactic head must be continuous, attention has recently shifted to the analysis of *non-projective* structures, which are required for linguistically adequate representations, especially in languages with free or flexible word order.

The most popular strategy for capturing non-projective structures in data-driven dependency parsing is to apply some kind of post-processing to the output of a strictly projective dependency parser, as in *pseudo-projective parsing* (Nivre and Nilsson, 2005), *corrective modeling* (Hall and Novák, 2005), or *approximate non-projective parsing* (McDonald and Pereira, 2006). And it is rare to find parsers that derive non-projective structures directly, the notable exception being the non-projective spanning tree parser proposed by McDonald et al. (2005).

There are essentially two arguments that have been advanced against using parsing algorithms that derive non-projective dependency structures directly. The first is that the added expressivity compromises efficiency, since the parsing problem for a grammar that allows arbitrary non-projective dependency structures has been shown to be $\mathcal{NP}$ complete (Neuhaus and Bröker, 1997). On the other hand, most data-driven approaches do not rely on grammars, and with a suitable factorization of dependency structures, it is possible to achieve parsing of unrestricted non-projective structures in $O(n^2)$ time, as shown by McDonald et al. (2005).

The second argument against non-projective dependency parsing comes from the observation that, even in languages with free or flexible word order,

Proceedings of NAACL HLT 2007, pages 396–403,
Rochester, NY, April 2007.

most dependency structures are either projective or very nearly projective. This can be seen by considering data from treebanks, such as the Prague Dependency Treebank of Czech (Böhmová et al., 2003), the TIGER Treebank of German (Brants et al., 2002), or the Slovene Dependency Treebank (Džeroski et al., 2006), where the overall proportion of non-projective dependencies is only about 2% even though the proportion of sentences that contain some non-projective dependency is as high as 25%. This means that an approach that starts by deriving the best projective approximation of the correct dependency structure is likely to achieve high accuracy, while an approach that instead attempts to search the complete space of non-projective dependency structures runs the risk of finding structures that depart too much from the near-projective norm. Again, however, the results of McDonald et al. (2005) suggest that the latter risk is minimized if inductive learning is used to guide the search.

One way of improving efficiency, and potentially also accuracy, in non-projective dependency parsing is to restrict the search to a subclass of "mildly non-projective" structures. Nivre (2006) defines *degrees* of non-projectivity in terms of the maximum number of intervening constituents in the projection of a syntactic head and shows that limited degrees of non-projectivity give a much better fit with the linguistic data than strict projectivity, but also enables more efficient processing than unrestricted non-projectivity. However, the results presented by Nivre (2006) are all based on oracle parsing, which means that they only provide upper bounds on the accuracy that can be achieved.

In this paper, we investigate to what extent constraints on non-projective structures can improve accuracy and efficiency in practical parsing, using treebank-induced classifiers to predict the actions of a deterministic incremental parser. The parsing algorithm used belongs to the family of algorithms described by Covington (2001), and the classifiers are trained using support vector machines (SVM) (Vapnik, 1995). The system is evaluated using treebank data from five languages: Danish, Dutch, German, Portuguese, and Slovene.

The paper is structured as follows. Section 2 defines syntactic representations as labeled dependency graphs and introduces the notion of *degree* used to constrain the search. Section 3 describes the parsing algorithm, including modifications necessary to handle degrees of non-projectivity, and section 4 describes the data-driven prediction of parser actions, using history-based models and SVM classifiers. Section 5 presents the experimental setup, section 6 discusses the experimental results, and section 7 contains our conclusions.

2 Dependency Graphs

A dependency graph is a labeled directed graph, the nodes of which are indices corresponding to the tokens of a sentence. Formally:

Definition 1 Given a set R of dependency types (arc labels), a *dependency graph* for a sentence $x = (w_1, \ldots, w_n)$ is a labeled directed graph $G = (V, E, L)$, where:

1. $V = \{0, 1, 2, \ldots, n\}$
2. $E \subseteq V \times V$
3. $L : E \rightarrow R$

The set V of *nodes* (or *vertices*) is the set of non-negative integers up to and including n. This means that every token index i of the sentence is a node ($1 \leq i \leq n$) and that there is a special node 0, which will always be a root of the dependency graph. The set E of *arcs* (or *edges*) is a set of ordered pairs (i, j), where i and j are nodes. Since arcs are used to represent dependency relations, we will say that i is the *head* and j is the *dependent* of the arc (i, j). The function L assigns a dependency type (label) $r \in R$ to every arc $e \in E$. We use the notation $i \rightarrow j$ to mean that there is an arc connecting i and j (i.e., $(i, j) \in E$); we use the notation $i \stackrel{r}{\rightarrow} j$ if this arc is labeled r (i.e., $((i, j), r) \in L$); and we use the notation $i \rightarrow^* j$ and $i \leftrightarrow^* j$ for the reflexive and transitive closure of the arc relation E and the corresponding undirected relation, respectively.

Definition 2 A dependency graph G is *well-formed* if and only if:

1. The node 0 is a root, i.e., there is no node i such that $i \rightarrow 0$ (ROOT).
2. G is weakly connected, i.e., $i \leftrightarrow^* j$ for every pair of nodes i, j (CONNECTEDNESS).
3. Every node has at most one head, i.e., if $i \rightarrow j$ then there is no node k such that $k \neq i$ and $k \rightarrow j$ (SINGLE-HEAD).

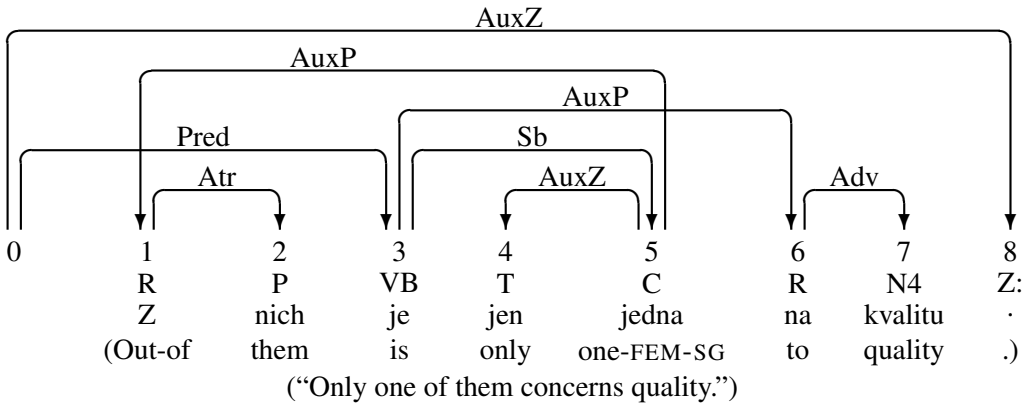

Figure 1: Dependency graph for Czech sentence from the Prague Dependency Treebank

The well-formedness conditions are independent in that none of them is entailed by any (combination) of the others, but they jointly entail that the graph is a tree rooted at the node 0. By way of example, figure 1 shows a Czech sentence from the Prague Dependency Treebank (Böhmová et al., 2003) with a well-formed dependency graph according to Definitions 1 and 2.

The constraints imposed on dependency graphs in Definition 2 are assumed in almost all versions of dependency grammar, especially in computational systems, and are sometimes complemented by a fourth constraint:

4. The graph G is projective, i.e., if $i \rightarrow j$ then $i \rightarrow^* k$, for every node k such that $i < k < j$ or $j < k < i$ (PROJECTIVITY).

Most theoretical formulations of dependency grammar regard projectivity as the norm but recognize the need for non-projective representations to capture non-local dependencies (Mel'čuk, 1988; Hudson, 1990). Finding a way of incorporating a suitably restricted notion of non-projectivity into practical parsing systems is therefore an important step towards a more adequate syntactic analysis, as discussed in the introduction of this paper.

In order to distinguish classes of dependency graphs that fall in between arbitrary non-projective and projective, Nivre (2006) introduces a notion of *degree* of non-projectivity, such that projective graphs have degree 0 while arbitrary non-projective graphs have unbounded degree.

Definition 3 Let $G = (V, E, L)$ be a well-formed dependency graph, let $G_{(i,j)}$ be the subgraph of G defined by $V_{(i,j)} = \{i, i+1, \ldots, j-1, j\}$, and let $\min(e)$ be the smallest and $\max(e)$ the largest element of an arc e in the linear order $<$:

1. The *degree* of an arc $e \in E$ is the number of connected components (i.e., weakly connected subgraphs) in $G_{(\min(e)+1,\max(e)-1)}$ that are not dominated by the head of e in $G_{(\min(e),\max(e))}$.
2. The *degree* of G is the maximum degree of any arc $e \in E$.

To exemplify the notion of degree, we note that the dependency graph in figure 1 has degree 1. The only non-projective arc in the graph is $(5, 1)$ and $G_{(2,4)}$ contains three connected components, each consisting of a single root node (2, 3, 4). Since exactly one of these, 3, is not dominated by 5 in $G_{(1,5)}$, the arc $(5, 1)$ has degree 1.

Nivre (2006) presents an empirical study, based on data from the Prague Dependency Treebank of Czech (Böhmová et al., 2003) and the Danish Dependency Treebank (Kromann, 2003), showing that more than 99.5% of all sentences occurring in the two treebanks have a dependency graph with a maximum degree of 2; about 98% have a maximum degree of 1; but only 77% in the Czech data and 85% in the Danish data have degree 0 (which is equivalent to assuming PROJECTIVITY). This suggests that limited degrees of non-projectivity may allow a parser to capture a larger class of naturally occurring syntactic structures, while still constraining the search to a proper subclass of all possible structures.[1]

[1] Alternative notions of mildly non-projective dependency structures are explored in Kuhlmann and Nivre (2006).

3 Parsing Algorithm

Covington (2001) describes a parsing strategy for dependency representations that has been known since the 1960s but not presented in the literature. The left-to-right (or incremental) version of this strategy can be formulated in the following way:

PARSE($x = (w_1, \ldots, w_n)$)
1 **for** $j = 1$ **up to** n
2 **for** $i = j - 1$ **down to** 0
3 LINK(i, j)

LINK(i, j) is a nondeterministic operation that adds the arc $i \rightarrow j$ (with some label), adds the arc $j \rightarrow i$ (with some label), or does nothing at all. In this way, the algorithm builds a graph by systematically trying to link every pair of nodes (i, j) ($i < j$). We assume that LINK(i,j) respects the ROOT and SINGLE-HEAD constraints and that it does not introduce cycles into the graph, i.e., it adds an arc $i \rightarrow j$ only if $j \neq 0$, there is no $k \neq i$ such that $k \rightarrow j$, and it is not the case that $j \rightarrow^* i$. Given these constraints, the graph G given at termination can always be turned into a well-formed dependency graph by adding arcs from the root 0 to any root node in $\{1, \ldots, n\}$.

Assuming that LINK(i, j) can be performed in some constant time c, the running time of the algorithm is $\sum_{i=1}^{n} c(i-1) = c(\frac{n^2}{2} - \frac{n}{2})$, which in terms of asymptotic complexity is $O(n^2)$. Checking ROOT and SINGLE-HEAD in constant time is easy, but in order to prevent cycles we need to be able to find, for any node k, the root of the connected component to which k belongs in the partially built graph. This problem can be solved efficiently using standard techniques for disjoint sets, including path compression and union by rank, which guarantee that the necessary checks can be performed in average constant time (Cormen et al., 1990).

In the experiments reported in this paper, we modify the basic algorithm by making the performance of LINK(i, j) conditional on the arcs (i, j) and (j, i) being permissible under different degree constraints:

PARSE($x = (w_1, \ldots, w_n)$, d)
1 **for** $j = 1$ **up to** n
2 **for** $i = j - 1$ **down to** 0
3 **if** PERMISSIBLE(i, j, d)
4 LINK(i, j)

The function PERMISSIBLE(i, j, d) returns **true** if and only if $i \rightarrow j$ and $j \rightarrow i$ have a degree less than or equal to d given the partially built graph G. Setting $d = 0$ gives strictly projective parsing, while $d = \infty$ corresponds to unrestricted non-projective parsing. With low values of d, we will reduce the number of calls to LINK(i, j), which will reduce the overall parsing time provided that the time required to compute PERMISSIBLE(i, j, d) is insignificant compared to the time needed for LINK(i, j). This is typically the case in data-driven systems, where LINK(i, j) requires a call to a trained classifier, while PERMISSIBLE(i, j, d) only needs access to the partially built graph G.[2]

4 History-Based Parsing

History-based parsing uses features of the parsing history to predict the next parser action (Black et al., 1992). In the current setup, this involves using features of the partially built dependency graph G and the input $x = (w_1, \ldots, w_n)$ to predict the outcome of the nondeterministic LINK(i, j) operation. Given that we use a deterministic parsing strategy, this reduces to a pure classification problem.

Let $\Phi(i, j, G) = (\phi_1, \ldots, \phi_m)$ be a feature vector representation of the parser history at the time of performing LINK(i, j). The task of the history-based classifier is then to map $\Phi(i, j, G)$ to one of the following actions:

1. Add the arc $i \xrightarrow{r} j$ (for some $r \in R$).
2. Add the arc $j \xrightarrow{r} i$ (for some $r \in R$).
3. Do nothing.

Training data for the classifier can be generated by running the parser on a sample of treebank data, using the gold standard dependency graph as an oracle to predict LINK(i, j) and constructing one training instance $(\Phi(i, j, G), a)$ for each performance of LINK(i, j) with outcome a.

The features in $\Phi(i, j, G) = (\phi_1, \ldots, \phi_m)$ can be arbitrary features of the input x and the partially built graph G but will in the experiments below be restricted to linguistic attributes of input tokens, including their dependency types according to G.

[2] Checking PERMISSIBLE(i, j, d), again requires finding the roots of connected components and can therefore be done in average constant time.

Language	Tok	Sen	T/S	Lem	CPoS	PoS	MSF	Dep	NPT	NPS
Danish	94	5.2	18.2	no	10	24	47	52	1.0	15.6
Dutch	195	13.3	14.6	yes	13	302	81	26	5.4	36.4
German	700	39.2	17.8	no	52	52	0	46	2.3	27.8
Portuguese	207	9.1	22.8	yes	15	21	146	55	1.3	18.9
Slovene	29	1.5	18.7	yes	11	28	51	25	1.9	22.2

Table 1: Data sets; Tok = number of tokens (*1000); Sen = number of sentences (*1000); T/S = tokens per sentence (mean); Lem = lemmatization present; CPoS = number of coarse-grained part-of-speech tags; PoS = number of (fine-grained) part-of-speech tags; MSF = number of morphosyntactic features (split into atoms); Dep = number of dependency types; NPT = proportion of non-projective dependencies/tokens (%); NPS = proportion of non-projective dependency graphs/sentences (%)

The history-based classifier can be trained with any of the available supervised methods for function approximation, but in the experiments below we will rely on SVM, which has previously shown good performance for this kind of task (Kudo and Matsumoto, 2002; Yamada and Matsumoto, 2003).

5 Experimental Setup

The purpose of the experiments is twofold. First, we want to investigate whether allowing non-projective structures to be derived incrementally can improve parsing accuracy compared to a strictly projective baseline. Secondly, we want to examine whether restricting the degree of non-projectivity can improve efficiency compared to an unrestricted non-projective baseline. In order to investigate both these issues, we have trained one non-projective parser for each language, allowing arbitrary non-projective structures as found in the treebanks during training, but applying different constraints during parsing:

1. Non-projective ($d = \infty$)
2. Max degree 2 ($d = 2$)
3. Max degree 1 ($d = 1$)

These three versions of the non-projective parser are compared to a strictly projective parser ($d = 0$), which uses the same parsing algorithm but only considers projective arcs in both training and testing.[3]

The experiments are based on treebank data from five languages: the Danish Dependency Treebank (Kromann, 2003), the Alpino Treebank of Dutch (van der Beek et al., 2002), the TIGER Treebank of German (Brants et al., 2002), the Floresta Sintáctica of Portuguese (Afonso et al., 2002), and the Slovene Dependency Treebank (Džeroski et al., 2006).[4] The data sets used are the training sets from the CoNLL-X Shared Task on multilingual dependency parsing (Buchholz and Marsi, 2006), with 20% of the data reserved for testing using a pseudo-random split. Table 1 gives an overview of the five data sets, showing the number of tokens and sentences, the presence of different kinds of linguistic annotation, and the amount of non-projectivity.

The features used in the history-based model for all languages include the following core set of 20 features, where i and j are the tokens about to be linked and the *context stack* is a stack of root nodes k in $G_{(i+1,j-1)}$, added from right to left (i.e., with the top node being closest to i):

1. Word form: i, j, $j+1$, $h(i)$.
2. Lemma (if available): i.
3. Part-of-speech: $i-1$, i, j, $j+1$, $j+2$, k, $k-1$.
4. Coarse part-of-speech (if available): i, j, k.
5. Morphosyntactic features (if available): i, j.
6. Dependency type: i, j, $l(i)$, $l(j)$, $r(i)$.

In the specification of features, we use k and $k-1$ to refer to the two topmost tokens on the context stack, and we use $h(\alpha)$, $l(\alpha)$ and $r(\alpha)$ to refer to the *head*,

[3] An alternative would have been to train all parsers on non-projective data, or restrict the training data for each parser according to its parsing restriction. Preliminary experiments showed that the setup used here gave the best performance for all parsers involved.

[4] This set does not include the Prague Dependency Treebank of Czech (Böhmová et al., 2003), one of the most widely used treebanks in studies of non-projective parsing. The reason is that the sheer size of this data set makes extensive experiments using SVM learning extremely time consuming. The work on Czech was therefore initially postponed but is now ongoing.

	Danish		Dutch		German		Portuguese		Slovene	
Constraint	**AS**	**ER**	**AS**	**ER**	**AS**	**ER**	**AS**	**ER**	**AS**	**ER**
Non-projective	88.13	8.34	86.79	36.18	89.78	21.51	90.59	11.39	76.52	6.83
Max degree 2	88.08	7.95	86.15	33.09	89.74	21.20	90.58	11.30	76.48	6.67
Max degree 1	88.00	7.33	85.12	28.12	89.49	19.28	90.48	10.36	76.40	6.35
Projective	87.05	–	79.30	–	86.98	–	89.38	–	74.80	–

Table 2: Parsing accuracy; AS = attachment score; ER = error reduction w.r.t. projective baseline (%)

the *leftmost dependent* and the *rightmost dependent* of a token α in the partially built dependency graph.[5] In addition to the core set of features, the model for each language has been augmented with a small number of additional features, which have proven useful in previous experiments with the same data set. The maximum number of features used is 28 (Danish); the minimum number is 23 (German).

The history-based classifiers have been trained using SVM learning, which combines a maximum margin strategy with the use of kernel functions to map the original feature space to a higher-dimensional space. More specifically, we use LIBSVM (Chang and Lin, 2001) with a quadratic kernel $K(x_i, x_j) = (\gamma x_i^T x_j + r)^2$. We use the built-in one-versus-one strategy for multi-class classification and convert symbolic features to numerical features using the standard technique of binarization.

Parsing accuracy is measured by the unlabeled attachment score (AS), i.e., the proportion of words that are assigned the correct head (not counting punctuation). Although the parsers do derive *labeled* dependency graphs, we concentrate on the graph structure here, since this is what is concerned in the distinction between projective and non-projective dependency graphs. Efficiency is evaluated by reporting the parsing time (PT), i.e., the time required to parse the respective test sets. Since both training sets and test sets vary considerably in size between languages, we are primarily interested in the relative differences for parsers applied to the same language. Experiments have been performed on a Sun-Blade 2000 with one 1.2GHz UltraSPARC-III processor and 2GB of memory.

[5]The lack of symmetry in the feature set reflects the asymmetry in the partially built graph G, where, e.g., only i can have dependents to the right at decision time. This explains why there are more features defined in terms of graph structure for i and more features defined in terms of string context for j.

6 Results and Discussion

Table 2 shows the parsing accuracy of the non-projective parser with different maximum degrees, both the raw attachment scores and the amount of error reduction with respect to the baseline parser. Our first observation is that the non-projective parser invariably achieves higher accuracy than the projective baseline, with differences that are statistically significant across the board (using McNemar's test). The amount of error reduction varies between languages and seems to depend primarily on the frequency of non-projective structures, which is not surprising. Thus, for Dutch and German, the two languages with the highest proportion of non-projective structures, the best error reduction is over 35% and over 20%, respectively. However, there seems to be a sparse data effect in that Slovene, which has the smallest training data set, has the smallest error reduction despite having more non-projective structures than Danish and Portuguese.

Our second observation is that the highest score is always obtained with an unbounded degree of non-projectivity during parsing. This seems to corroborate the results obtained by McDonald et al. (2005) with a different parsing method, showing that the use of inductive learning to guide the search during parsing eliminates the potentially harmful effect of increasing the size of the search space. Although the differences between different degrees of non-projectivity are not statistically significant for the current data sets,[6] the remarkable consistency across languages suggests that they are nevertheless genuine. In either case, however, they must be considered marginal, except possibly for Dutch, which leads to our third and final observation about accu-

[6]The only exception is the difference between a maximum degree of 1 and unrestricted non-projective for Dutch, which is significant according to McNemar's test with $\alpha = .05$.

	Danish		Dutch		German		Portuguese		Slovene	
Constraint	**PT**	**TR**	**PT**	**TR**	**PT**	**TR**	**PT**	**TR**	**PT**	**TR**
Non-projective	426	–	3791	–	24454	–	3708	–	204	–
Max degree 2	395	7.29	2068	45.46	17903	26.79	3004	18.99	130	36.39
Max degree 1	346	18.72	1695	55.28	13079	46.52	2446	34.04	108	47.05
Projective	211	50.53	784	79.32	7362	69.90	1389	62.55	429	79.00

Table 3: Parsing time; PT = parsing time (s); TR = time reduction w.r.t. non-projective baseline (%)

System	**Danish**	**Dutch**	**German**	**Portuguese**	**Slovene**
CoNLL-X McDonald et al.	84.79	79.19	87.34	86.82	73.44
CoNLL-X Nivre et al.	84.77	78.59	85.82	87.60	70.30
Incremental non-projective	84.85	77.91	85.90	87.12	70.86

Table 4: Related work (labeled attachment score)

racy, namely that restricting the maximum degree of non-projectivity to 2 or 1 has a very marginal effect on accuracy and is always significantly better than the projective baseline.

Turning next to efficiency, table 3 shows the parsing time for the different parsers across the five languages. Our first observation here is that the parsing time can be reduced by restricting the degree of non-projectivity during parsing, thus corroborating the claim that the running time of the history-based classifier dominates the overall parsing time. As expected, the largest reduction is obtained with the strictly projective parser, but here we must also take into account that the training data set is smaller (because of the restriction to projective potential links), which improves the average running time of the history-based classifier in itself. Our second observation is that the amount of reduction in parsing time seems to be roughly related to the amount of non-projectivity, with a reduction of about 50% at a max degree of 1 for the languages where more than 20% of all sentences are non-projective (Dutch, German, Slovene) but significantly smaller for Portuguese and especially for Danish. On the whole, however, the reduction in parsing time with limited degrees of non-projectivity is substantial, especially considering the very marginal drop in accuracy.

In order to compare the performance to the state of the art in dependency parsing, we have retrained the non-projective parser on the entire training data set for each language and evaluated it on the final test set from the CoNLL-X shared task (Buchholz and Marsi, 2006). Thus, table 4 shows *labeled* attachment scores, the main evaluation metric used in the shared task, in comparison to the two highest scoring systems from the original evaluation (McDonald et al., 2006; Nivre et al., 2006). The incremental non-projective parser has the best reported score for Danish and outperforms at least one of the other two systems for four languages out of five, although most of the differences are probably too small to be statistically significant. But whereas the spanning tree parser of McDonald et al. (2006) and the pseudo-projective parser of Nivre et al. (2006) achieve this performance only with special pre- or post-processing,[7] the approach presented here derives a labeled non-projective graph in a single incremental process and hence at least has the advantage of simplicity. Moreover, it has better time complexity than the approximate second-order spanning tree parsing of McDonald et al. (2006), which has exponential complexity in the worst case (although this does not appear to be a problem in practice).

7 Conclusion

In this paper, we have investigated a data-driven approach to dependency parsing that combines a deterministic incremental parsing algorithm with history-based SVM classifiers for predicting the next parser action. We have shown that, for languages with a

[7] McDonald et al. (2006) use post-processing for non-projective dependencies and for labeling. Nivre et al. (2006) use pre-processing of training data and post-processing of parser output to recover non-projective dependencies.

non-negligible proportion of non-projective structures, parsing accuracy can be improved significantly by allowing non-projective structures to be derived. We have also shown that the parsing time can be reduced substantially, with only a marginal loss in accuracy, by limiting the degree of non-projectivity allowed during parsing. A comparison with results from the CoNLL-X shared task shows that the parsing accuracy is comparable to that of the best available systems, which means that incremental non-projective dependency parsing is a viable alternative to approaches based on post-processing of projective approximations.

Acknowledgments

The research presented in this paper was partially supported by a grant from the Swedish Research Council. I want to thank Johan Hall and Jens Nilsson for their contributions to MaltParser, which was used to perform the experiments. I am also grateful to three anonymous reviewers for finding important errors in the preliminary version and for suggesting several other improvements for the final version.

References

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proc. of LREC*, 1698–1703.

E. Black, F. Jelinek, J. D. Lafferty, D. M. Magerman, R. L. Mercer, and S. Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proc. of the DARPA Speech and Natural Language Workshop*, 31–37.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In A. Abeillé, ed., *Treebanks: Building and Using Parsed Corpora*, chapter 7. Kluwer, Dordrecht.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proc. of TLT*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*, 149–164.

C.-C. Chang and C.-J. Lin, 2001. *LIBSVM: A Library for Support Vector Machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Y. Cheng, M. Asahara, and Y. Matsumoto. 2004. Deterministic dependency structure analyzer for Chinese. In *Proc. of IJCNLP*, 500–508.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. 1990. *Introduction to Algorithms*. MIT Press.

M. A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proc. of the Annual ACM Southeast Conference*, 95–102.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of LREC*.

Keith Hall and Vaclav Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proc. of IWPT*, 42–52

R. A. Hudson. 1990. *English Word Grammar*. Blackwell.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of TLT*.

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of CoNLL*, 63–69.

M. Kuhlmann and J. Nivre. 2006. Mildly non-projective dependency structures. In *Proc. of COLING-ACL, Posters*, 507–514.

R. McDonald and F-. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*, 81–88.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*, 523–530.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of CoNLL*, 216–220.

I. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

P. Neuhaus and N. Bröker. 1997. The complexity of recognition of linguistically adequate dependency grammars. In *Proc. of ACL-EACL*, 337–343.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL*, 99–106.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. of CoNLL*, 49–56.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proc. of CoNLL*, 221–225.

J. Nivre. 2006. Constraints on non-projective dependency graphs. In *Proc. of EACL*, 73–80.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

V. N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*, 195–206.