Constraints on 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

Abstract

We investigate a series of graph-theoretic constraints on non-projective dependency parsing and their effect on *expressivity*, i.e. whether they allow naturally occurring syntactic constructions to be adequately represented, and *efficiency*, i.e. whether they reduce the search space for the parser. In particular, we define a new measure for the *degree* of non-projectivity in an acyclic dependency graph obeying the single-head constraint. The constraints are evaluated experimentally using data from the Prague Dependency Treebank and the Danish Dependency Treebank. The results indicate that, whereas complete linguistic coverage in principle requires unrestricted non-projective dependency graphs, limiting the degree of non-projectivity to at most 2 can reduce average running time from quadratic to linear, while excluding less than 0.5% of the dependency graphs found in the two treebanks. This is a substantial improvement over the commonly used projective approximation (degree 0), which excludes 15–25% of the graphs.

1 Introduction

Data-driven approaches to syntactic parsing has until quite recently been limited to representations that do not capture non-local dependencies. This is true regardless of whether representations are based on constituency, where such dependencies are traditionally represented by empty categories and coindexation to avoid explicitly discontinuous constituents, or on dependency, where it is more common to use a direct encoding of so-called non-projective dependencies.

While this "surface dependency approximation" (Levy and Manning, 2004) may be acceptable for certain applications of syntactic parsing, it is clearly not adequate as a basis for deep semantic interpretation, which explains the growing body of research devoted to different methods for correcting this approximation. Most of this work has so far focused either on post-processing to recover non-local dependencies from context-free parse trees (Johnson, 2002; Jijkoun and De Rijke, 2004; Levy and Manning, 2004; Campbell, 2004), or on incorporating nonlocal dependency information in nonterminal categories in constituency representations (Dienes and Dubey, 2003; Hockenmaier, 2003; Cahill et al., 2004) or in the categories used to label arcs in dependency representations (Nivre and Nilsson, 2005).

By contrast, there is very little work on parsing methods that allow discontinuous constructions to be represented directly in the syntactic structure, whether by discontinuous constituent structures or by non-projective dependency structures. Notable exceptions are Plaehn (2000), where discontinuous phrase structure grammar parsing is explored, and McDonald et al. (2005b), where non-projective dependency structures are derived using spanning tree algorithms from graph theory.

One question that arises if we want to pursue the structure-based approach is how to constrain the class of permissible structures. On the one hand, we want to capture all the constructions that are found in natural languages, or at least to provide a much better approximation than before. On the other hand, it must still be possible for the parser not only to search the space of permissible structures in an efficient way but also to learn to select the most appropriate structure for a given sentence with sufficient accuracy. This is the usual tradeoff

between expressivity and complexity, where a less restricted class of permissible structures can capture more complex constructions, but where the enlarged search space makes parsing harder with respect to both accuracy and efficiency.

Whereas extensions to context-free grammar have been studied quite extensively, there are very few corresponding results for dependency-based systems. Since Gaifman (1965) proved that his projective dependency grammar is weakly equivalent to context-free grammar, Neuhaus and Bröker (1997) have shown that the recognition problem for a dependency grammar that can define arbitrary non-projective structures is $\mathcal{NP}$ complete, but there are no results for systems of intermediate complexity. The pseudo-projective grammar proposed by Kahane et al. (1998) can be parsed in polynomial time and captures non-local dependencies through a form of gap-threading, but the structures generated by the grammar are strictly projective. Moreover, the study of formal grammars is only partially relevant for research on data-driven dependency parsing, where most systems are not grammar-based but rely on inductive inference from treebank data (Yamada and Matsumoto, 2003; Nivre et al., 2004; McDonald et al., 2005a). For example, despite the results of Neuhaus and Bröker (1997), McDonald et al. (2005b) perform parsing with arbitrary non-projective dependency structures in $O(n^2)$ time.

In this paper, we will therefore approach the problem from a slightly different angle. Instead of investigating formal dependency grammars and their complexity, we will impose a series of graph-theoretic constraints on dependency structures and see how these constraints affect expressivity and parsing efficiency. The approach is mainly experimental and we evaluate constraints using data from two dependency-based treebanks, the Prague Dependency Treebank (Hajič et al., 2001) and the Danish Dependency Treebank (Kromann, 2003).

Expressivity is investigated by examining how large a proportion of the structures found in the treebanks are parsable under different constraints, and efficiency is addressed by considering the number of potential dependency arcs that need to be processed when parsing these structures. This is a relevant metric for data-driven approaches, where parsing time is often dominated by the computation of model predictions or scores for such arcs. The parsing experiments are performed with a variant of Covington's algorithm for dependency parsing (Covington, 2001), using the treebank as an oracle in order to establish an upper bound on accuracy. However, the results are relevant for a larger class of algorithms that derive non-projective dependency graphs by treating every possible word pair as a potential dependency arc.

The paper is structured as follows. In section 2 we define dependency graphs, and in section 3 we formulate a number of constraints that can be used to define different classes of dependency graphs, ranging from unrestricted non-projective to strictly projective. In section 4 we introduce the parsing algorithm used in the experiments, and in section 5 we describe the experimental setup. In section 6 we present the results of the experiments and discuss their implications for non-projective dependency parsing. We conclude in section 7.

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. $V = \mathbf{Z}_{n+1}$
2. $E \subseteq V \times V$
3. $L : E \rightarrow R$

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

1. The node 0 is a root (ROOT).
2. G is connected (CONNECTEDNESS).[1]

The set V of *nodes* (or *vertices*) is the set $\mathbf{Z}_{n+1} = \{0, 1, 2, \ldots, n\}$ $(n \in \mathbf{Z}^+)$, i.e., 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 does not correspond to any token of the sentence and which will always be a root of the dependency graph (normally the only root).

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

[1] To be more exact, we require G to be *weakly connected*, which entails that the corresponding undirected graph is connected, whereas a *strongly connected* graph has a *directed* path between any pair of nodes.

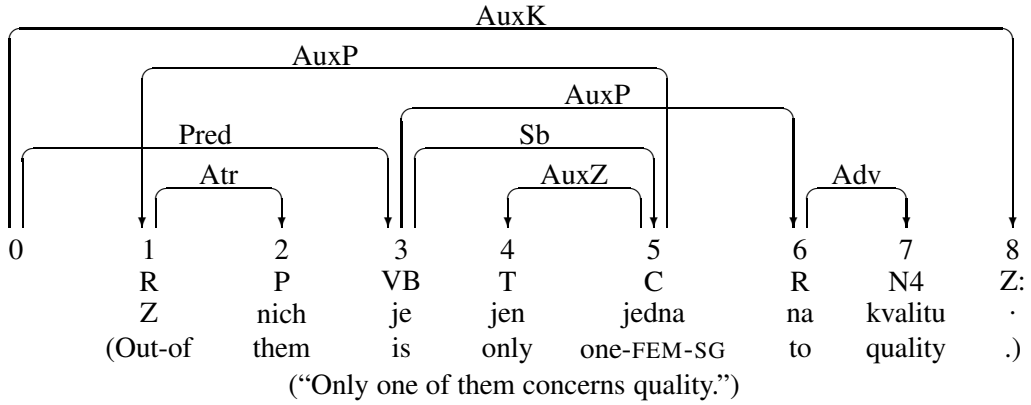

Figure 1: Dependency graph for Czech sentence from the Prague Dependency Treebank

say that i is the *head* and j is the *dependent* of the arc (i, j). As usual, we will use the notation $i \rightarrow j$ to mean that there is an arc connecting i and j (i.e., $(i, j) \in E$) and we will use the notation $i \rightarrow^* j$ for the reflexive and transitive closure of the arc relation E (i.e., $i \rightarrow^* j$ if and only if $i = j$ or there is a path of arcs connecting i to j).

The function L assigns a dependency type (arc label) $r \in R$ to every arc $e \in E$. Figure 1 shows a Czech sentence from the Prague Dependency Treebank with a well-formed dependency graph according to Definition 1–2.

3 Constraints

The only conditions so far imposed on dependency graphs is that the special node 0 be a root and that the graph be connected. Here are three further constraints that are common in the literature:

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).
4. The graph G is acyclic, i.e., if $i \rightarrow j$ then not $j \rightarrow^* i$ (ACYCLICITY).
5. 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).

Note that these conditions are independent in that none of them is entailed by any (combination) of the others. However, the conditions SINGLE-HEAD and ACYCLICITY together with the basic well-formedness conditions entail that the graph is a tree rooted at the node 0. These constraints are assumed in almost all versions of dependency grammar, especially in computational systems.

By contrast, the PROJECTIVITY constraint is much more controversial. Broadly speaking, we can say that whereas most practical systems for dependency parsing do assume projectivity, most dependency-based linguistic theories do not. More precisely, most theoretical formulations of dependency grammar regard projectivity as the norm but also recognize the need for non-projective representations to capture non-local dependencies (Mel'čuk, 1988; Hudson, 1990).

In order to distinguish classes of dependency graphs that fall in between arbitrary non-projective and projective, we define 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, satisfying SINGLE-HEAD and ACYCLICITY, and let G_e be the subgraph of G that only contains nodes between i and j for the arc $e = (i, j)$ (i.e., $V_e = \{i+1, \ldots, j-1\}$ if $i < j$ and $V_e = \{j+1, \ldots, i-1\}$ if $i > j$).

1. The *degree* of an arc $e \in E$ is the number of connected components c in G_e such that the root of c is not dominated by the head of 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 (which satisfies SINGLE-HEAD and ACYCLICITY) has degree 1. The only non-projective arc in the graph is $(5, 1)$ and $G_{(5,1)}$ contains three connected components, each of which consists of a single root node (2, 3 and 4). Since only one of these, 3, is not dominated by 5, the arc $(5, 1)$ has degree 1.

4 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** $i = 1$ **up to** n
2 **for** $j = i - 1$ **down to** 1
3 LINK(i, j)

The operation LINK(i, j) nondeterministically chooses between (i) adding the arc $i \rightarrow j$ (with some label), (ii) adding the arc $j \rightarrow i$ (with some label), and (iii) adding no arc at all. In this way, the algorithm builds a graph by systematically trying to link every pair of nodes (i, j) $(i > j)$. This graph will be a well-formed dependency graph, provided that we also add arcs from the root node 0 to every root node in $\{1, \ldots, n\}$. Assuming that the LINK(i, j) operation can be performed in some constant time c, the running time of the algorithm is $\sum_{i=1}^{n} c(n-1) = c(\frac{n^2}{2} - \frac{n}{2})$, which in terms of asymptotic complexity is $O(n^2)$.

In the experiments reported in the following sections, we modify this algorithm by making the performance of LINK(i, j) conditional on the arcs (i, j) and (j, i) being permissible under the given graph constraints:

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

The function PERMISSIBLE(i, j, C) returns **true** iff $i \rightarrow j$ and $j \rightarrow i$ are permissible arcs relative to the constraint C and the partially built graph G. For example, with the constraint SINGLE-HEAD, LINK(i, j) will not be performed if both i and j already have a head in the dependency graph. We call the pairs (i, j) $(i > j)$ for which LINK(i, j) is performed (for a given sentence and set of constraints) the *active pairs*, and we use the number of active pairs, as a function of sentence length, as an abstract measure of running time. This is well motivated if the time required to compute PERMISSIBLE(i, j, C) is insignificant compared to the time needed for LINK(i, j), as is typically the case in data-driven systems, where LINK(i, j) requires a call to a trained classifier, while PERMISSIBLE(i, j, C) only needs access to the partially built graph G.

The results obtained in this way will be partially dependent on the particular algorithm used, but they can in principle be generalized to any algorithm that tries to link all possible word pairs and that satisfies the following condition:

> For any graph $G = (V, E, L)$ derived by the algorithm, if $e, e' \in E$ and e covers e', then the algorithm adds e' before e.

This condition is satisfied not only by Covington's incremental algorithm but also by algorithms that add arcs strictly in order of increasing length, such as the algorithm of Eisner (2000) and other algorithms based on dynamic programming.

5 Experimental Setup

The experiments are based on data from two treebanks. The Prague Dependency Treebank (PDT) contains 1.5M words of newspaper text, annotated in three layers (Hajič, 1998; Hajič et al., 2001) according to the theoretical framework of Functional Generative Description (Sgall et al., 1986). Our experiments concern only the analytical layer and are based on the dedicated training section of the treebank. The Danish Dependency Treebank (DDT) comprises 100K words of text selected from the Danish PAROLE corpus, with annotation of primary and secondary dependencies based on Discontinuous Grammar (Kromann, 2003). Only primary dependencies are considered in the experiments, which are based on 80% of the data (again the standard training section).

The experiments are performed by parsing each sentence of the treebanks while using the gold standard dependency graph for that sentence as an oracle to resolve the nondeterministic choice in the LINK(i, j) operation as follows:

LINK(i, j)
1 **if** $(i, j) \in E_g$
2 $E \leftarrow E \cup \{(i, j)\}$
3 **if** $(j, i) \in E_g$
4 $E \leftarrow E \cup \{(j, i)\}$

where E_g is the arc relation of the gold standard dependency graph G_g and E is the arc relation of the graph G built by the parsing algorithm.

Conditions are varied by cumulatively adding constraints in the following order:

1. SINGLE-HEAD
2. ACYCLICITY
3. Degree $d \leq k$ $(k \geq 1)$
4. PROJECTIVITY

Table 1: Proportion of dependency arcs and complete graphs correctly parsed under different constraints in the Prague Dependency Treebank (PDT) and the Danish Dependency Treebank (DDT)

Constraint	PDT		DDT	
	Arcs $n = 1255590$	**Graphs** $n = 73088$	**Arcs** $n = 80193$	**Graphs** $n = 4410$
PROJECTIVITY	96.1569	76.8498	97.7754	84.6259
$d \leq 1$	99.7854	97.7507	99.8940	98.0272
$d \leq 2$	99.9773	99.5731	99.9751	99.5238
$d \leq 3$	99.9956	99.9179	99.9975	99.9546
$d \leq 4$	99.9983	99.9863	100.0000	100.0000
$d \leq 5$	99.9987	99.9945	100.0000	100.0000
$d \leq 10$	99.9998	99.9986	100.0000	100.0000
ACYCLICITY	100.0000	100.0000	100.0000	100.0000
SINGLE-HEAD	100.0000	100.0000	100.0000	100.0000
None	100.0000	100.0000	100.0000	100.0000

The purpose of the experiments is to study how different constraints influence expressivity and running time. The first dimension is investigated by comparing the dependency graphs produced by the parser with the gold standard dependency graphs in the treebank. This gives an indication of the extent to which naturally occurring structures can be parsed correctly under different constraints. The results are reported both as the proportion of individual dependency arcs (per token) and as the proportion of complete dependency graphs (per sentence) recovered correctly by the parser.

In order to study the effects on running time, we examine how the number of active pairs varies as a function of sentence length. Whereas the asymptotic worst-case complexity remains $O(n^2)$ under all conditions, the average running time will decrease with the number of active pairs if the LINK(i, j) operation is more expensive than the call to PERMISSIBLE(i, j, C). For data-driven dependency parsing, this is relevant not only for parsing efficiency, but also because it may improve training efficiency by reducing the number of pairs that need to be included in the training data.

6 Results and Discussion

Table 1 displays the proportion of dependencies (single arcs) and sentences (complete graphs) in the two treebanks that can be parsed exactly with Covington's algorithm under different constraints. Starting at the bottom of the table, we see that the unrestricted algorithm (None) of course reproduces all the graphs exactly, but we also see that the constraints SINGLE-HEAD and ACYCLICITY do not put any real restrictions on expressivity with regard to the data at hand. However, this is primarily a reflection of the design of the treebank annotation schemes, which in themselves require dependency graphs to obey these constraints.[2]

If we go to the other end of the table, we see that PROJECTIVITY, on the other hand, has a very noticeable effect on the parser's ability to capture the structures found in the treebanks. Almost 25% of the sentences in PDT, and more than 15% in DDT, are beyond its reach. At the level of individual dependencies, the effect is less conspicuous, but it is still the case in PDT that one dependency in twenty-five cannot be found by the parser even with a perfect oracle (one in fifty in DDT). It should be noted that the proportion of lost dependencies is about twice as high as the proportion of dependencies that are non-projective in themselves (Nivre and Nilsson, 2005). This is due to error propagation, since some projective arcs are blocked from the parser's view because of missing non-projective arcs.

Considering different bounds on the degree of non-projectivity, finally, we see that even the tightest possible bound ($d \leq 1$) gives a much better approximation than PROJECTIVITY, reducing the

[2] It should be remembered that we are only concerned with one layer of each annotation scheme, the analytical layer in PDT and the primary dependencies in DDT. Taking several layers into account simultaneously would have resulted in more complex structures.

Table 2: Quadratic curve estimation for $y = ax + bx^2$ (y = number of active pairs, x = number of words)

	PDT			DDT		
Constraint	a	b	r^2	a	b	r^2
PROJECTIVITY	1.9181	0.0093	0.979	1.7591	0.0108	0.985
$d \leq 1$	3.2381	0.0534	0.967	2.2049	0.0391	0.969
$d \leq 2$	3.1467	0.1192	0.967	2.0273	0.0680	0.964
ACYCLICITY	0.3845	0.2587	0.971	1.4285	0.1106	0.967
SINGLE-HEAD	0.7187	0.2628	0.976	1.9003	0.1149	0.967
None	−0.5000	0.5000	1.000	−0.5000	0.5000	1.000

proportion of non-parsable sentences with about 90% in both treebanks. At the level of individual arcs, the reduction is even greater, about 95% for both data sets. And if we allow a maximum degree of 2, we can capture more than 99.9% of all dependencies, and more than 99.5% of all sentences, in both PDT and DDT. At the same time, there seems to be no principled upper bound on the degree of non-projectivity, since in PDT not even an upper bound of 10 is sufficient to correctly capture all dependency graphs in the treebank.[3]

Let us now see how different constraints affect running time, as measured by the number of active pairs in relation to sentence length. A plot of this relationship for a subset of the conditions can be found in Figure 2. For reasons of space, we only display the data from DDT, but the PDT data exhibit very similar patterns. Both treebanks are represented in Table 2, where we show the result of fitting the quadratic equation $y = ax + bx^2$ to the data from each condition (where y is the number of active words and x is the number of words in the sentence). The amount of variance explained is given by the r^2 value, which shows a very good fit under all conditions, with statistical significance beyond the 0.001 level.[4]

Both Figure 2 and Table 2 show very clearly that, with no constraints, the relationship between words and active pairs is exactly the one predicted by the worst case complexity (cf. section 4) and that, with each added constraint, this relationship becomes more and more linear in shape. When we get to PROJECTIVITY, the quadratic coefficient b is so small that the average running time is practically linear for the great majority of sentences. However, the complexity is not much worse for the bounded degrees of non-projectivity ($d \leq 1$, $d \leq 2$). More precisely, for both data sets, the linear term ax dominates the quadratic term bx^2 for sentences up to 50 words at $d \leq 1$ and up to 30 words at $d \leq 2$. Given that sentences of 50 words or less represent 98.9% of all sentences in PDT and 98.3% in DDT (the corresponding percentages for 30 words being 88.9% and 86.0%), it seems that the average case running time can be regarded as linear also for these models.

7 Conclusion

We have investigated a series of graph-theoretic constraints on dependency structures, aiming to find a better approximation than PROJECTIVITY for the structures found in naturally occurring data, while maintaining good parsing efficiency. In particular, we have defined the *degree* of non-projectivity in terms of the maximum number of connected components that occur under a dependency arc without being dominated by the head of that arc. Empirical experiments based on data from two treebanks, from different languages and with different annotation schemes, have shown that limiting the degree d of non-projectivity to 1 or 2 gives an average case running time that is linear in practice and allows us to capture about 98% of the dependency graphs actually found in the treebanks with $d \leq 1$, and about 99.5% with $d \leq 2$. This is a substantial improvement over the projective approximation, which only allows 75–85% of the dependency graphs to be captured exactly. This suggests that the integration of such constraints into non-projective parsing algorithms will improve both accuracy and efficiency, but we have to leave the corroboration of this hypothesis as a topic for future research.

[3] The single sentence that is not parsed correctly at $d \leq 10$ has a dependency arc of degree 12.

[4] The curve estimation has been performed using SPSS.

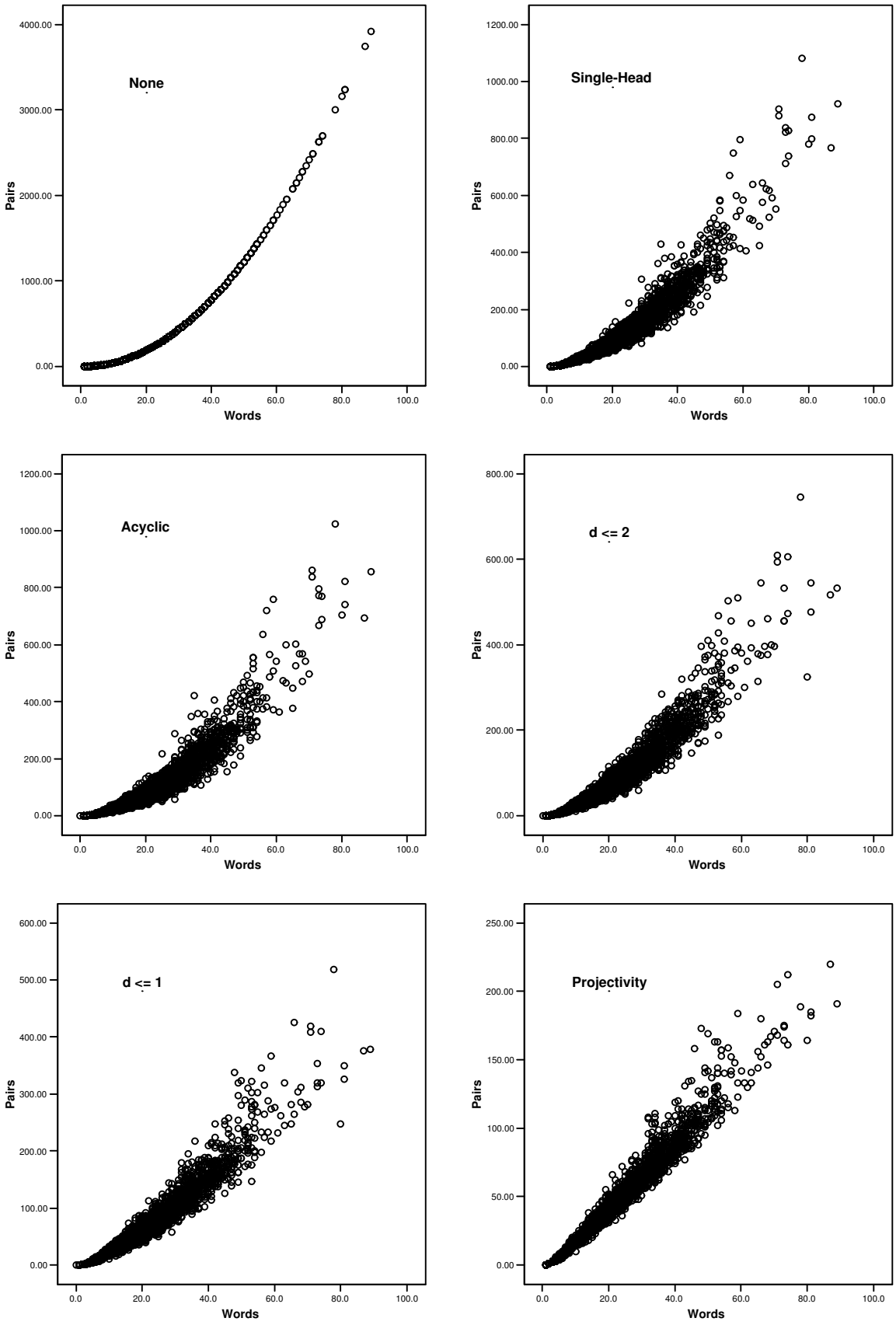

Figure 2: Number of active pairs as a function of sentence length under different constraints (DDT)

Acknowledgments

The research reported in this paper was partially funded by the Swedish Research Council (621-2002-4207). The insightful comments of three anonymous reviewers helped improve the final version of the paper.

References

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef Van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. *Proceedings of ACL*, pp. 320–327.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. *Proceedings of ACL*, pp. 646–653.

Michael Collins, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. *Proceedings of ACL*, pp. 505–512.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. *Proceedings of the 39th Annual ACM Southeast Conference*, pp. 95–102.

Péter Dienes and Amit Dubey. 2003. Deep syntactic processing by combining shallow methods. *Proceedings of ACL*, pp. 431–438.

Jason M. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pp. 29–62. Kluwer.

Haim Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.

Jan Hajič, Barbora Vidova Hladka, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

Jan Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. *Issues of Valency and Meaning*, pp. 106–132. Karolinum.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

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

Valentin Jijkoun and Maarten De Rijke. 2004. Enriching the output of a parser using memory-based learning. *Proceedings of ACL*, pp. 312–319.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. *Proceedings of ACL*, pp. 136–143.

Sylvain Kahane, Alexis Nasr and Owen Rambow. Pseudo-Projectivity: A Polynomially Parsable Non-Projective Dependency Grammar. *Proceedings of ACL-COLING*, pp. 646–652.

Matthias Trautner Kromann. 2003. The Danish Dependency Treebank and the DTAG treebank tool. *Proceedings of TLT*, pp. 217–220.

Roger Levy and Christopher Manning. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. *Proceedings of ACL*, pp. 328–335.

Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. *Proceedings of ACL*, pp. 31–38.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. *Proceedings of ACL*, pp. 91–98.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. *Proceedings of HLT/EMNLP*, pp. 523–530.

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

Peter Neuhaus and Norbert Bröker. 1997. The complexity of recognition of linguistically adequate dependency grammars. *Proceedings of ACL-EACL*, pages 337–343.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. *Proceedings ACL*, pp. 99–106.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. *Proceedings of CoNLL*, pp. 49–56.

Oliver Plaehn. 2000. Computing the most probably parse for a discontinuous phrase structure grammar. *Proceedings of IWPT*.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Pragmatic Aspects*. Reidel.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. *Proceedings of IWPT*, pp. 195–206.