Bare-Bones Dependency Parsing – A Case for Occam's Razor?

Joakim Nivre
Uppsala University
Uppsala, Sweden
joakim.nivre@lingfil.uu.se

Abstract

If all we want from a syntactic parser is a dependency tree, what do we gain by first computing a different representation such as a phrase structure tree? The principle of parsimony suggests that a simpler model should be preferred over a more complex model, all other things being equal, and the simplest model is arguably one that maps a sentence directly to a dependency tree – a bare-bones dependency parser. In this paper, I characterize the parsing problem faced by such a system, survey the major parsing techniques currently in use, and begin to examine whether the simpler model can in fact rival the performance of more complex systems. Although the empirical evidence is still limited, I conclude that bare-bones dependency parsers fare well in terms of parsing accuracy and often excel in terms of efficiency.

1 Introduction

The notion of dependency has come to play an increasingly central role in natural language parsing in recent years. On the one hand, lexical dependencies have been incorporated in statistical models for a variety of syntactic representations such as phrase structure trees (Collins, 1999), LFG representations (Riezler et al., 2002), and CCG derivations (Clark and Curran, 2004). On the other hand, dependency relations extracted from such representations have been exploited in many practical applications, for example, information extraction (Culotta and Sorensen, 2004), question answering (Bouma et al., 2005), and machine translation (Ding and Palmer, 2004). Given these developments, it is not surprising that there has also been a growing interest in parsing models that map sentences directly to dependency trees, an approach that will be referred to as *bare-bones dependency parsing* to distinguish it from parsing methods where dependencies are embedded into or extracted from other types of syntactic representations.

The bare-bones model can be motivated by the principle known as Occam's razor, which says that entities should not be postulated beyond necessity. If we can show that bare-bones dependency parsers produce dependency trees with at least the same accuracy and efficiency as more complex models, then they would be preferred on grounds of simplicity. In this paper, I will begin by explaining how the parsing problem for bare-bones dependency parsers differs from the more familiar parsing problem for phrase structure parsers. I will go on to survey the main techniques that are currently in use, grouped into four broad categories: chart parsing, constraint-based parsing, transition-based parsing, and hybrid methods. Finally, I will examine a number of recent studies that compare the performance of different types of parsers and conclude that bare-bones dependency parsers fare well in terms of accuracy as well as efficiency.

2 Parsing Problem

A dependency structure for a sentence $w_1, \ldots, w_n$ is a directed graph whose nodes represent the input tokens $w_1, \ldots, w_n$ and whose arcs represent syntactic relations from head to dependent. Arcs are normally labeled with dependency types, although unlabeled dependency graphs are also used. Depending on what formal constraints are adopted, we get different classes of dependency graphs, with different expressivity and complexity. If we only require graphs to be connected and acyclic, then words can have more than one head, which is convenient for representing deep syntactic relations. If we require the graph to be a tree, then each word can have at most one head, but we can still represent extraction phenomena using non-

Bolette Sandford Pedersen, Gunta Nešpore and Inguna Skadiņa (Eds.)
NODALIDA 2011 Conference Proceedings, pp. 6–11

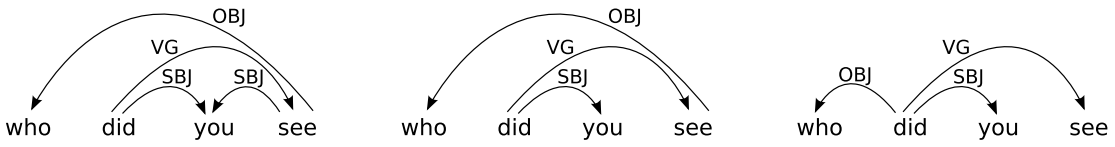

Figure 1: Dependency graphs: directed acyclic graph (left), tree (middle), projective tree (right).

projective arcs. If we require every subtree to have a contiguous yield, finally, we get the class of projective trees. The different classes are illustrated in Figure 1.

Regardless of what restrictions we put on dependency graphs, the parsing problem consists in finding the optimal set of arcs, given the nodes as input. This is different from phrase structure parsing, where only the terminal nodes are given as input and both internal nodes and edges have to be inferred during parsing. Many algorithms for dependency parsing are restricted to projective trees, which reduces the complexity of the parsing problem, but a number of systems are capable of handling non-projective trees, either by using non-standard algorithms or through post-processing. Very few systems can deal with directed acyclic graphs. Dependency parsers are generally evaluated by measuring precision and recall on dependency relations, with or without labels. When dependency graphs are restricted to trees, precision and recall coincide and are normally referred to as the attachment score.

3 Parsing Techniques

3.1 Chart Parsing Techniques

A straightforward method for dependency parsing is to view it as a restricted form of context-free parsing and reuse chart parsing algorithms like CKY and Earley, an idea that is implicit already in Hays (1964). Thanks to the constraints on dependency trees, it is possible to reduce complexity to $O(n^3)$ for lexicalized parsing using the span-based representation proposed by Eisner (1996). Coupled with statistical models of increasing complexity, this technique has resulted in excellent parsing accuracy for projective trees, with features defined over single arcs (McDonald et al., 2005a), pairs of arcs (McDonald and Pereira, 2006; Carreras, 2007) or even triples of arcs (Koo and Collins, 2010). These models are usually referred to as first-, second- and third-order models. One limitation of this parsing approach is that it does not easily extend to non-projective trees, let alone directed acyclic graphs. However, as shown by McDonald and Pereira (2006), it is possible to recover both non-projective arcs and multiple heads through post-processing.

3.2 Parsing as Constraint Satisfaction

A different approach is to view parsing as a constraint satisfaction problem, starting from a compact representation of all dependency graphs compatible with the input and successively eliminating invalid graphs through the propagation of grammatical constraints, as originally proposed by Maruyama (1990). By adding numerical weights to constraints and defining the score of a graph as a function of the weights of violated constraints, Menzel and Schröder (1998) turned this into an optimization problem where the goal is to find the highest-scoring dependency graph. Constraint-based parsing can easily accommodate different classes of dependency graphs and do not have the same inherent limitations on features or constraints as chart parsing, but the parsing problem is computationally intractable in general, so exact search methods cannot be used except in special cases. An interesting special case is the arc-factored model defined by McDonald et al. (2005b), where the score of a dependency tree is a sum of independent arc weights. Under these assumptions, finding the highest scoring dependency tree is equivalent to finding the maximum directed spanning tree in a complete graph containing all possible dependency arcs, a problem that can be computed in $O(n^2)$ time using algorithms from graph theory. Unfortunately, any attempt to extend the scope of weighted constraints beyond single arcs makes the parsing problem NP complete. Another variation of the constraint-based approach is the use of integer linear programming, which was pioneered by Riedel et al. (2006) and further improved by Martins et al. (2009).

3.3 Transition-Based Parsing

A third prominent method is to view parsing as deterministic search through a transition system (or state machine), guided by a statistical

Parser	Type	UAS
Yamada and Matsumoto (2003)	Trans-Local	90.3
McDonald et al. (2005a)	Chart-1st	90.9
Collins (1999)	PCFG	91.5
McDonald and Pereira (2006)	Chart-2nd	91.5
Charniak (2000)	PCFG	92.1
Koo et al. (2010)	Hybrid-Dual	92.5
Sagae and Lavie (2006)	Hybrid-MST	92.7
Zhang and Nivre (2011)	Trans-Global	92.9
Koo and Collins (2010)	Chart-3rd	93.0

Table 1: Dependency parsing for English (WSJ-PTB, Penn2Malt); unlabeled attachment scores.

Parser	Type	UAS
Collins (1999)	PCFG	82.2
McDonald et al. (2005a)	Chart-1st	83.3
Charniak (2000)	PCFG	84.3
McDonald et al. (2005b)	MST	84.4
Hall and Novák (2005)	PCFG+Post	85.0
McDonald and Pereira (2006)	Chart-2nd+Post	85.2
Nivre (2009)	Trans-Local	86.1
Zeman and Žabokrtský (2005)	Hybrid-Greedy	86.3
Koo et al. (2010)	Hybrid-Dual	87.3

Table 2: Dependency parsing for Czech (PDT); unlabeled attachment scores.

model for predicting the next transition, an idea first proposed by Yamada and Matsumoto (2003). Transition-based parsing can be very efficient, with linear running time for projective dependency trees (Nivre, 2003) and limited subsets of non-projective trees (Attardi, 2006). For arbitrary non-projective trees, the worst-case complexity is quadratic, but observed running time can still be linear with an appropriate choice of transition system Nivre (2009), and transition systems can be extended to handle directed acyclic graphs (Sagae and Tsujii, 2008). Transition-based parsers can base their decisions on very rich representations of the derivation history (including the partially built dependency graph) but may suffer from error propagation due to search errors especially if the statistical model is trained to maximize the accuracy of local transitions rather than complete transition sequences. Zhang and Clark (2008) showed how these problems can be alleviated by global optimization and beam search, and Huang and Sagae (2010) obtained further improvements through ambiguity packing.

3.4 Hybrid Methods

For parsing as for many other problems, it is often possible to improve accuracy by combining methods with different strengths. Thus, Zeman and Žabokrtský (2005) reported substantial improvements in parsing Czech by letting a number of parsers vote for the syntactic head of each word. A drawback of this simple voting scheme is that the output may not be a well-formed dependency graphs even if all the component parsers output well-formed graphs. This problem was solved by Sagae and Lavie (2006), who showed that we can use the spanning tree method of McDonald et al. (2005b) for parser combination by letting parsers vote for arcs in the complete graph and then extract the maximum spanning tree. Another hybrid technique is parser stacking, where one parser is used to generate input features for another parser, a method that was used by Nivre and McDonald (2008) to combine chart parsing and transition-based parsing, with further improvements reported by Torres Martins et al. (2008). Finally, Koo et al. (2010) used dual decomposition to combine third-order chart parsing and arc-factored spanning tree parsing with excellent empirical results.

4 Comparative Evaluation

When Yamada and Matsumoto (2003) presented the first comparative evaluation of dependency parsing for English, using data from the WSJ section of the Penn Treebank (Marcus et al., 1993) with what has later become known as the Penn2Malt conversion to dependencies, they observed that although their own bare-bones dependency parser had the advantage of simplicity and efficiency, it was not quite as accurate as the parsers of Collins (1999) and Charniak (2000). However, as the results reported in Table 1 clearly show, there has been a tremendous development since then, and the third-order chart parser of Koo and Collins (2010) is now as accurate as any phrase structure parser. Bare-bones dependency parsers are also the most efficient parsers available, with an average parsing time per sentence of 20 msec for the parser of Zhang and Nivre (2011), for example. As shown in Table 2, a very similar development has taken place in the case of Czech dependency parsing, as evaluated on the Prague Dependency Treebank (Hajič et al., 2001).

Cer et al. (2010) evaluated a number of systems for producing Stanford typed dependencies (de Marneffe et al., 2006) and found that bare-bones dependency parsers like MaltParser (Nivre et al., 2006) and MSTParser (McDonald and Pereira, 2006) had considerably lower accuracy than the best phrase structure parsers like the Berkeley

parser (Petrov et al., 2006; Petrov and Klein, 2007) and the parser of Charniak and Johnson (2005). However, the evaluation was performed after converting the parser output to so-called collapsed dependencies, a conversion process that is less accurate for dependency trees than for phrase structure trees. More importantly, the bare-bones dependency parsers were run without proper optimization, whereas most of the phrase structure parsers have been optimized for a long time not only for English but in particular for the type of Wall Street Journal text that was used in the evaluation. It is therefore likely that the evaluation results, although representative for out-of-the-box comparisons on this particular data set, do not generalize to other settings. Evidence for this conclusion comes from a similar study by Candito et al. (2010), where different types of parsers were evaluated on data from the French Treebank, and where there was practically no difference in accuracy between the best bare-bones dependency parsers (MaltParser, MSTParser) and the best phrase structure parser (Berkeley). With respect to efficiency, the transition-based MaltParser was found to be about ten times faster than the other two parsers.

Rimell et al. (2009) evaluated a number of statistical parsers specifically on their capacity to recover unbounded dependencies like those involved in different types of relative clauses, interrogative clauses and right node raising. The evaluation was extended to bare-bones dependency parsers in Nivre et al. (2010), and the overall results show that systems like MaltParser and MSTParser, augmented with simple post-processing for inferring multiple heads, perform at least as well as other types of treebank parsers, although not quite as well as grammar-driven systems like those of Clark and Curran (2004) and Miyao and Tsujii (2005).

5 Conclusion

Although the available evidence is still scattered and incomplete, the empirical results so far seem to support the hypothesis that bare-bones dependency parsers can achieve the same level of accuracy as more complex systems. Since they have the advantage of simplicity and are often highly efficient, they clearly seem to merit their place in contexts where the main requirement on syntactic analysis is to produce a dependency tree. To what extent they are also adequate as theoretical models of natural language syntax in general is of course a completely different question.

References

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 166–170.

Gosse Bouma, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jörg Tiedemann. 2005. Question answering for dutch using dependency relations. In *Working Notes of the 6th Workshop of the Cross-Language Evaluation Forum (CLEF 2005)*.

Marie Candito, Joakim Nivre, Pascal Denis, and Enrique Henestroza Anguiano. 2010. Benchmarking of statistical dependency parsers for french. In *Coling 2010: Posters*, pages 108–116.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 957–961.

Daniel Cer, Marie-Catherine de Marneffe, Dan Jurafsky, and Chris Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 173–180.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 132–139.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 104–111.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Aron Culotta and Jeffery Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 423–429.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed

dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*.

Yuan Ding and Martha Palmer. 2004. Synchronous dependency insertion grammars: A grammar formalism for syntax based statistical MT. In *Proceedings of the Workshop on Recent Advances in Dependency Grammar*, pages 90–97.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

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

Keith Hall and Vaclav Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pages 42–52.

David G. Hays. 1964. Dependency theory: A formalism and some observations. *Language*, 40:511–525.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1077–1086.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–11.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 342–350.

Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Meeting of the Association for Computational Linguistics (ACL)*, pages 31–38.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.

Wolfgang Menzel and Ingo Schröder. 1998. Decision procedures for dependency parsing using graded constraints. In *Proceedings of the Workshop on Processing of Dependency-Based Grammars (ACL-COLING)*, pages 78–87.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 83–90.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 950–958.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 833–841.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 351–359.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 404–411.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440.

Sebastian Riedel, Ruket Çakıcı, and Ivan Meza-Ruiz. 2006. Multi-lingual dependency parsing with incremental integer linear programming. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 226–230.

Stephan Riezler, Margaret H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–278.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 753–760.

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 157–166.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Daniel Zeman and Zdeněk Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pages 171–178.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 562–571.

Yue Zhang and Joakim Nivre. 2011. Transition-based parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

ISSN 1736-6305 Vol. 11
http://hdl.handle.net/10062/16955