Research Report AI-1994-02

Discontinuous Dependency Parsing of Free and Fixed Word Order: Work in Progress

Michael A. Covington

Artificial Intelligence Programs
The University of Georgia
Athens, Georgia 30602 U.S.A.

Discontinuous Dependency Parsing of Free and Fixed Word Order: Work in Progress

Michael A. Covington
Artificial Intelligence Programs
The University of Georgia
Athens, GA 30602–7415 U.S.A.
mcovingt@ai.uga.edu

April 1994

Abstract

The dependency–based free–word–order parsing algorithm of Covington (1987, 1990, 1992) can be extended to handle partly or completely fixed word order, while preserving its psychologically realistic preference for near attachment. By adding predictivity, this algorithm can be adapted to parse left–branching and right–branching structures in less stack space than center–embedded structures, just as the human parser appears to do.

1 Introduction

Most parsing algorithms assume that words occur in a fixed order, and that the input string can be divided into continuous substrings which are constituents. Discontinuous dependency parsing (DDP; Covington 1987, 1990, 1992) makes no such assumptions. It treats free word order as the simplest case, and treats restrictions on word order as additional constraints. Thus, DDP has an advantage in parsing languages with extensive word order variability.

In this paper I briefly note how to extend DDP to handle fixed as well as variable word order, and then briefly analyze some aspects of its psychological reality and suggest improvements.

2 Dependency grammar

The relation of head to dependent (governor to governed, subcategorizand to argument) has become increasingly important in modern grammatical theory. Dependency grammar dispenses with phrase structure trees and takes the head–dependent relation as basic. Specifically:

- Every word is the dependent of exactly one head (except the head of the sentence, which is not a dependent at all).
- Any number of words can be dependents of the same head.
- Thus, the dependency links form a tree whose root is the head of the sentence (normally taken to be the main verb).

Given two words X and Y, where X is the head of Y, or the head of the head of Y, and so on, we say that X is SUPERIOR to Y and Y is INFERIOR to X.

Constituents still exist, but they are a derived rather than a fundamental concept. A *phrase* (constituent) consists of any word together with all words that are inferior to it.

Suitably constrained dependency grammars are equivalent to suitably constrained X–bar phrase–structure grammars (Covington 1990). The intent of DDP, however, is to relax the constraints and thereby handle free word order.

In this paper I am not arguing for any particular theory of dependency grammar. Accordingly, I will assume only that there is a grammar which, given two words, can tell us whether they can be linked, and if so which is the head and which is the dependent.[1]

3 Original DDP algorithm

The original DDP algorithm (Covington 1987, 1990, 1992) closely resembles bottom–up shift–reduce parsing. The parser accepts words in order, one by one, holding them until they can be attached to the dependency tree. Two lists are maintained: *Wordlist,* containing all words that have been accepted

[1]For more concrete proposals see Hellwig (1986); Jäppinen, Lahtola, and Valkonen (1986); Mel'čuk (1988); Schubert (1987); and others. The full version of this paper will contain a more ample review of the literature. The most thorough dependency analysis of English is that of Hudson (1991).

so far, and *Headlist,* containing words that have been accepted but have not yet been attached as dependents of anything else. When the whole sentence has been parsed, *Headlist* contains only one word, the head of the sentence. The algorithm is as follows:

```
while there are words remaining in the input string:
   (1) Accept a word; call it W and push it onto Wordlist.
   (2) Search the previous elements of Wordlist, most recent first,
         and attach W as a dependent of one of them if possible.
         Otherwise, push W onto Headlist.
   (3) Search Headlist, most recent first, and attach zero or more
         of its elements as dependents of W, removing them from Headlist.
end
```

This is a nondeterministic algorithm; it was implemented in Prolog so that backtracking would be done automatically. Tests with parsing of Latin and Russian showed comparatively little backtracking in ordinary sentences, even those with very scrambled word order.[2]

Despite being able to handle unlimited word order freedom, DDP has a psychologically realistic preference for near attachment, because it searches both lists in most–recent–first order and therefore tries near attachments first.

4 Adaptation for fixed word order

To handle fixed word order, DDP needs to be able to:

- Specify that a phrase must be continuous;
- Specify the order of the head relative to each of its dependents;
- Specify the mutual order of the dependents of a head.

4.1 Continuity (adjacency, projectivity)

I define continuity (*alias* contiguity, adjacency, projectivity) as a property of individual head–to–dependent links, not of phrases:

[2]DDP is also the basis of the Korean parsers of Kwon, Yoon & Kim (1990) and Kwon & Yoon (1991).

The link between head H and dependent D is continuous if and only if every word between H and D is subordinate to H.

Implicit in this definition is the hypothesis that the grammar can account for partly fixed word order by specifying continuity requirements for individual links, rather than for whole phrases. This appears to be the case, although of course in many cases continuity requirements will apply to all the links in a phrase.

As an example of a head that must be continuous with one of its dependents but not with another, consider the English sentences:

John pressed the key quickly.
John quickly pressed the key.
Quickly John pressed the key.

The verb has two dependents, *quickly* and *the key.* The link from *pressed* to *the key* is continuous; that from *pressed* to *quickly* is not.

Continuity is implemented in the parser as follows:

- When searching previous words for the head of W, do not search all of *Wordlist*; look only at the word preceding W, its head, its head's head, and so on. That is, instead of following *Wordlist,* follow a chain of dependency links.
- When searching *Headlist* for dependents of W, use only the most recent element of *Headlist.* (Having done so, the parser can then use the next most recent element, etc., always working from the top of the stack and never skipping an element.)

This is essentially the "adjacency" requirement as implemented by Hudson (1989) and Fraser (1989).

4.2 Head–dependent order

As is well known, the linear order of head and dependent plays a major role in word order typology: some languages are predominantly head–initial, and others are head–final.

Implementing head–initial or head–final order in DDP is easy: leave out half of the search process, depending on what restriction is to be enforced. Specifically:

- When searching *Headlist* for dependents of W, do not attempt links that are specified as head–initial.
- When searching previous words for the head of W, do not attempt links that are specified as head–final.

These restrictions can also be implemented as preferences rather than absolute constraints: instead of leaving out parts of the search, the parser could attempt those parts of the search last, after other possibilities have been exhausted.

4.3 Dependent–dependent order

The mutual order of two dependents of the same head is harder to implement cleanly. It is also much less often needed in the description of languages, although English does provide one apparently good example:

Give the students an example.
**Give an example the students.*

The indirect object of the verb always comes before the direct object.[3]

¿From the dependency standpoint, this is a *global* order requirement — it involves at least two dependency links, not just one. The appropriate place to account for it is in the lexical entry of the verb, which already must mention both links in order to state that the verb is ditransitive.

Specifically, the mutual order of these two arguments can be accounted for as part of the subcategorization mechanism. Koch (1993) added a verb–class–based subcategorization system for DDP. If this is replaced with a list–based system along the lines of Shieber (1986), the grammar can require the parser to fulfill subcategorization requirements in a specific order, and thus require that one specific dependent must occur closer to the head than another.

This analysis presumes that only subcategorized dependents will be subject to mutual–order constraints.

[3]This construction is also problematic for GB theory: if both objects are complements of the same verb and hang from the same node, they should have the same grammatical relation to the verb, which they do not.

5 Further remarks

5.1 Predictivity

Conspicuously absent from present versions of DDP is any form of predictivity; the parser accepts words one by one but does not attempt to predict what will come next. Accordingly, DDP cannot exploit Hawkins' (1993) principle that the type of each constituent (or, in dependency terms, the head of each word) should be identified as soon as possible.

The performance of DDP could be improved by adding predictivity. This would make DDP into an analog of left–corner parsing rather than pure bottom–up parsing. The mechanism would be that whenever a word is accepted and is not a dependent of a word already seen, its head should be entered into the dependency tree as a node that is not yet lexically instantiated.

5.2 Psychological reality

The appeal of the DDP shift–reduce algorithm is that it provides a way to accept free word order while maintaining a realistic preference for near attachment. The need for such a preference is shown by phrases such as *mother's father's brother's house,* in which the order of the three possessives does not appear to be free in any known language (if it were, a series of N possessives would be N–factorial ways ambiguous, as noted by Johnson 1985).

Indeed, DDP overcomes a well–known psycholinguistic objection to shift–reduce parsing. Ordinary shift–reduce parsing is not symmetrical with regard to left and right branching: it uses up stack space when parsing right–branching but not left–branching structures. By contrast, the parsers in our heads apparently parse right– and left–branching structures with equal ease, and run out of stack space only on center–embedded structures (Johnson–Laird 1983, Abney and Johnson 1991, Resnik 1992).

DDP overcomes this limitation because it attaches dependents (arguments or modifiers) to heads one at a time, rather than waiting for constituents to be complete. Thus, it can deal with the words one at a time in either the left–branching structure[4]

[4]Pretending for the moment that the English *'s* possessive is a true genitive case. Taking *'s* to be a clitic, the example works equally well but the processing involves more steps.

$[_{NP}\ [_{NP}\ [_{NP}\ [_{NP}$ *Lincoln's* $]$ *doctor's* $]$ *dog's* $]$ *dinner* $]$

or its right–branching Latin counterpart:

$[_{NP}$ *statua* $[_{NP}$ *imperatoris* $[_{NP}$ *legionis* $[_{NP}$ *Gallorum* $]]]]$
'statue of the commander of the legion of the Gauls'

In the English example, the parser accepts *Lincoln's*, then accepts *doctor's* and attaches *Lincoln's* to it as a dependent, then accepts *dog's* and attaches *doctor's* as a dependent of it, and so on. In the Latin case, the parser accepts *statua*, then accepts and attaches *imperatoris* as a dependent of *statua*, then accepts and attaches *legionis* as a dependent of *imperatoris,* and so on. In neither case is a series of words held on the stack.[5]

DDP does use stack space when parsing center–embeddings, discontinuous constituents, and sequences of premodifiers. For example, *big fuzzy black dog* would be parsed by holding *big, fuzzy,* and *black* on the stack, and then attaching them all at once to *dog.* But if predictivity were added to DDP, these premodifier sequences could be parsed without using stack space, because DDP could then hypothesize the head noun and start attaching dependents to it before it is actually found.

References

Abney, Steven, and Johnson, Mark. (1991). "Memory requirements and local ambiguities for parsing strategies." *Journal of Psycholinguistic Research* 20: 233–250.

Covington, Michael A. (1987). "Parsing variable–word–order languages with unification–based dependency grammar." Research Report 01–0022, Advanced Computational Methods Center (now Artificial Intelligence Programs), University of Georgia.

Covington, Michael A. (1990). "Parsing discontinuous constituents in dependency grammar." *Computational Linguistics,* 16: 234–236.

[5]DDP is, then, "arc–eager" in the sense of Abney and Johnson (1991), but it is not clear that Resnik's (1992) formalization of the concept of "arc–eager" applies to it, because there is no composition of top–down and bottom–up expectations. Rather, what makes DDP "arc–eager" is the fact that it deals with grammatical relations one by one rather than waiting to complete constituents.

Covington, Michael A. (1992). "A dependency parser for variable–word–order languages." In *Computer assisted modeling on the IBM 3090: Papers from the 1989 IBM Supercomputing Competition,* ed. by K. R. Billingsley, Hilton U. Brown III, and Ed Derohanes, vol. 2, pp. 799–845, Athens, Ga.: Baldwin Press.

Fraser, Norman M. (1989). "Parsing and dependency grammar." *UCL Working Papers in Linguistics,* 1: 296–319.

Hawkins, John A. (1993). "Heads, parsing and word–order universals." In *Heads in Grammatical Theory,* ed. Greville G. Corbett, Norman M. Fraser, and Scott McGlashan, 231–265, Cambridge University Press.

Hellwig, Peter. (1986). "Dependency unification grammar." In *Proceedings, COLING–86,* 195–198.

Hudson, Richard. (1989). "Towards a computer–testable Word Grammar of English." *UCL Working Papers in Linguistics,* 1: 321–339.

Hudson, Richard. (1991). *English Word Grammar.* Cambridge, Mass.: Blackwell.

Jäppinen, Harri; Lahtola, Aarno; and Valkonen, Kari. (1986). "Functional structures for parsing dependency constraints." In *Proceedings, COLING–86,* 461–463.

Johnson, Mark. (1985). "Parsing with discontinuous constituents." In *Proceedings, ACL–85*, 127–132.

Johnson–Laird, Philip N. (1983). *Mental models.* Cambridge, Mass.: Harvard University Press.

Koch, Ulrich. (1993). "The enhancement of a dependency parser for Latin." Research Report AI-1993-03, Artificial Intelligence Programs, The University of Georgia.

Kwon, H. C., and Yoon, A. (1991). "Unification–based dependency parsing of governor–final languages." In *Proceedings, Second International Workshop on Parsing Technologies,* 182–192.

Kwon, H. C.; Yoon, A.; and Kim, Y. T. (1990). "A Korean analysis system based on unification and chart." In *Proceedings, Pacific Rim International Conference on Artificial Intelligence '90,* 251–256.

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

Resnik, Philip. (1992). "Left–corner parsing and psychological plausibility." In *Proceedings, COLING–92,* 191–197.

Schubert, Klaus. (1987). *Metataxis: Contrastive Dependency Syntax for Machine Translation.* Dordrecht: Foris.

Shieber, Stuart M. (1986). *An Introduction to Unification–Based Approaches to Grammar.* (CSLI Lecture Notes, 4.) Stanford: CSLI.