Linguistic Exploitation of Syntactic Databases: The Use of the Nijmegen Linguistic DataBase Program

Hans van Halteren and Theo van den Heuvel
(University of Nijmegen)

Amsterdam and Atlanta: Editions
Rodopi (Language and Computers:
Studies in Practical Linguistics 5), 1990,
ix+207 pp.
Paperbound, ISBN 90-6203-809-3,
$33.50, Dfl 65.00.[1]

Reviewed by
Ian Lancashire
University of Toronto

This book describes an important tool for the storage and exploration of syntactic structures that may be used under VAX/VMS, UNIX, and MS/PC-DOS in both interactive and batch modes. Linguistic DataBase, developed by the authors within the TOSCA Group at the University of Nijmegen, retrieves, displays, and counts constituent structures from syntactic trees. The book cover shows why LDB is original: a magnifying glass, hovering over a book opened to two pages of text, shows an inverted seven-level tree, root at the top, leaves corresponding to the words in a sentence at the bottom. Hans van Halteren and Theo van den Heuvel of the TOSCA group have created a primary tool for corpus linguistics: a generic displayer of syntactic trees from diverse databases with different encoding schemes.

LDB and its manual give students of language use — including teachers of English as a second language, computational linguists, researchers of style in literature, and language historians — a common retrieval system for syntactic phenomena. Interactive concordance programs retrieve words and their tags from sizable texts and corpora. Such tags sometimes encode morphological, part-of-speech, and even functional information. Yet these tools cannot conveniently retrieve any word or word group that satisfies a pattern of syntactic tags, let alone display the results of an inquiry as a tree.

The authors also describe the LDB as "a model of a database appropriate to a corpus-linguistic setting" (p. 3). Their manual outlines the current range of possibilities in syntactic corpus research, and LDB exemplifies the kind of editorial capabilities, displays, and file management facilities that a working database management system for syntactic trees should have. The book also tests both the function-category description model developed by F. Aarts and J. Aarts in 1982 and the contemporary English grammar published by Randolph Quirk and others ten years earlier. Both are important contributions to corpus linguistics.

1. The Program

The authors make LDB available free of charge to academic institutions (and for Dfl 5000 to others) together with the 130,000-word Nijmegen corpus of modern

1 An MS-DOS demonstration diskette for the examples and exercises in the book is available free of charge from the authors.

English.[2] Remarkably, the MS/PC-DOS demo version has all the tree-display and scrolling functions and yet requires no graphics card, just a one-line addition to CONFIG.SYS. It has to be run from the C: (hard) drive, but needs only about 650KB of available disk space. Just the first 200 trees are distributed for each of the two sample syntactic databases, taken from the 1.5 million-word TOSCA Corpus and from the Nijmegen Corpus, but these are quite sufficient for the purpose.

I installed and used the demo version under DOS 5.0 without encountering error messages or program failure. LDB employs a self-explanatory system of embedded menus, clearly labeled and more than adequately supported by online help screens. It uses mnemonic commands — letters and numbers, such as s for *son of* — rather than the function or arrow keys (or the mouse) conventionally employed now in commercial software. This gives LDB a somewhat old-fashioned look. These mnemonic commands appear at the bottom of every screen for convenient reference, but a user will have them by rote after a few days' work.

The interface has been designed so that novices will be able to navigate tree structures after a few minutes of practice with the mnemonic commands. Illegal menu choices are automatically blocked, it is hard to get lost because the exits are well posted, and sample "exploration schemes" (inquiries for retrieval of specific structures) permit anyone to see results quickly. Experts will be able to make their own exploration schemes without trouble and to review how a function-category grammar model works with the standard grammar of contemporary English.

The book gives a clear, generously illustrated tutorial to the inquiry program, followed by a valuable reference section where menus, displays, commands, and command editors are documented, and where a grammar of expressions (specifications, declarations, and actions) appears. Although the LDB demo diskette includes programs to create new databases with different function-category-attribute models, the book and demo do not describe how to implement them, other than advise on the limitations of the system. LDB cannot handle data in nontree (network) form, process trees with more than 1000 nodes, or manage more than two fixed labels per node (in the LDB databases, those two fixed labels are function and category), but the number of nonfixed labels (e.g., attributes) is unlimited. The authors ask users interested in developing their own databases for LDB to contact them in person for assistance.

2. Names in the Model

Among the first of Adam's important responsibilities was to systematize nature by "the naming of parts." The choice of names in LDB says much about its subject too.

The LDB model, conventionally, calls each separable thing in a syntactic analysis of an utterance (or sentence) a *node*, for which word my 1991 *Webster's College Dictionary* gives 11 definitions, most of them, including "a labeled point in a tree diagram at which subordinate lines branch off," drawn metaphorically from human anatomy and from botany. Van Halteren and van den Heuvel, like others in syntactic analysis, consciously mix both these metaphors in representing utterances as quasi-living societies (p. 10), although in display the nodes are of course only inanimate boxes, with or without labels inside them, and (where more than one node exists) connected by lines.

The most important metaphor is botanical: an utterance is like a tree. The first node (to the right or at the top), called the root, stands for the entire utterance. It is linked

2 Those interested in obtaining the complete system should contact Hans van Halteren, Department of Language and Speech, University of Nijmegen, P.O. Box 9103, 6500 HD Nijmegen, The Netherlands; e-mail: cor_hvh@kunrcl.urc.kun.nl

by branching lines to other nodes for successively smaller parts of the whole, until a group of end nodes are reached, called *leaves*, which represent the actual words in the utterance. The tree metaphor is based on visual similarities only (each is one root *below*, connected to many leaves *above* by branches), but otherwise the comparison breaks down. An LDB root *abstractly contains* all other nodes, and an LDB branch abstractly contains all the nodes and leaves descending from it, whereas in a tree the root, the branch, and the leaf are separate parts of the whole.

Tree metaphors lend a comforting if misleading concreteness to syntactic abstractions such as "sentence" and "noun phrase." These, the root and its branches, are made by the program's (the discipline's) own metalanguage to appear rather more solid than what speakers actually say and listeners hear, the "leaves." Yet of course syntactic analysis is about hypothetical abstractions. This is not to say that it does not truly describe language use, only that researchers sometimes may invest theoretical forms with more life than the stone Samuel Johnson kicked in refuting Bishop Berkeley.

The second metaphor used by LDB compares an utterance to an extended family of males. (Van Halteren and van den Heuvel, sensitive to sexist language, explain that they chose the masculine gender because it was "easier to associate the letters in the alphabet with command names in a mnemonic way" [p. 11]. For example, `D` for *daughter* would rule out using `D` for *down*.) Nodes are *fathers* by virtue of having one or more nodes coming out of them; and these nodes are thus *sons* of the father from which they come. Nodes sharing a father are then *brothers*. Most "males" in this family tree have one function (cf. job), one category (cf. class: upper, working, middle, etc.), and an unlimited number of attributes. Like the botanical metaphor, this one humanizes all nodes, whether representing abstractions or words themselves. The genealogical tree metaphor also implies a greater age and wisdom the farther back one traces toward the root. Old religions made the numinous familiar by representing it in human form. So it is with a new science, corpus linguistics.

Van Halteren and van den Heuvel also characterize the user (female, let us say in all fairness) metaphorically. By taking a "tree map view," she activates exploration schemes. These involve *actions* developed by an *activity editor*. In viewing the tree, she adopts a *focus*. By adopting the metaphor of an adventurer exploring mapped territory, LDB thus suggests that all trees and nodes are natural, the invaluable in need of conservation. Again naming drives this home. The user takes "the environment view" when looking at the whole tree, without labels.

The authors say that LDB should be used only with static, definitive analyses of text. The scientific terminology of the grammar itself, with its hyperrules, constants, variables, atomic specifiers, monadic and dyadic operators, etc., indeed suggests that the syntactically tagged corpus is immutable. Of course the anthropomorphic names have quite different connotations of flux. It would be unfair to expect more of a system that pioneers in its field, but linguists will in time want to edit LDB databases such as van Halteren and van den Heuvel and others supply and to transform them dynamically. Most of us like to ask "what-if" questions.

3. Exploring with LDB

The LDB demo starts the user off with a blank node, one without labels that matches every node in each database. With this, the user can amble through all sample trees to see how the authors have applied their function, category, and attribute names.

Her real work comes when she modifies this blank node by adding labels to it and by linking it with other (new) labeled nodes so as to create an exploration scheme.

For this purpose, van Halteren and van den Heuvel have devised a series of *editors*, first for the node(s) and then for various actions — procedures — that will determine how the actual inquiry is processed.

First, the user normally chooses an existing (stored) scheme to use or to modify. A *pattern editor* permits her to create the node boxes and the lines connecting them (fathers, sons, brothers). Once this task has been saved, she then activates label and expression editors to add, delete, or change the function, category, and attribute labels that make up the patterns in each node box. These patterns are nodes associated with constituents whenever one or more Boolean expressions are satisfied. Customary string, Boolean (*true, false, not, implies*), and numeric operators, and constants or other expressions as operands, are available.

The following are sample expression schemes:

1. `NOT ROOT`
   `CAT = 'S'`

2. `FUN = 'V'`
   `SNO = 2`

3. `LEAF`
   `CON = 'to IFM'`

4. `CAT = 'NP'` $\longrightarrow$ `CAT = 'NP'`
   `ANYDEPTH`

Expression 1 represents subclauses, and expression 2 verbs in second position (pp. 49, 51). The third requests the contents of leaves (codes and words) with the word *to* acting as infinitive marker; and expression 4 presents father–son nodes representing nested (adjacent or nonadjacent) noun phrases (`ANYDEPTH`; p. 54). `CAT` (category) and `FUN` (function) are string specifiers and work with labels (e.g., `S`, `NP`, and `V`) that are corpus-specific. The other specifiers are `ATT` (attribute), `WOR` (word), `COD` (code), and `CON` (contents, i.e., both word and code). Concatenation and substrings are supported. `ROOT` and `SNO` (son number) are structural properties; others are `LEAF`, `LVL` (level of node, where the root is level 1), `SCT` (son count), and `WCT` (word count). The user may also look for ambiguities both within nodes and in the branching between them.

After saving the pattern, the user calls on the *activity editors* to specify what actions are to be taken with this pattern when the search begins (*corpus intro*), when the utterance is encountered (*tree intro*), when a match is found (*match intro*), when the end of the utterance is reached (*tree effect*), and when the search ends (*corpus effect*). Possible actions, for instance, direct control to the user, output data to the screen or to a file in list or tabular form, and add reference numbers for the tree, the subcorpus, the location, the tree ambiguity number, etc., or the number of matches found. Once these actions have been specified, the exploration scheme is done. LDB will check the scheme for conformity to the program's rules before running it. Without leaving LDB, she may explore various corpora with this same scheme.

The authors have added many features to LDB for advanced users. For instance, it can filter successful matches into a subcorpus for further processing. Expressions may use tables and sets and may specify ambiguity and discontinuity. Batch processing with command files will be a preferred way of analysis with advanced users.

The Nijmegen LDB belongs to a growing number of valuable corpora and tools produced by (largely) northern European researchers in corpus linguistics in English

(Lancashire 1991) and documented now in *The ICAME Journal*.[3] Pieter de Haan (1989) has already published research on the postmodifying noun phrase with LDB and the Nijmegen corpus. The release of this manual and its easy-to-use software brings LDB to a much broader population of nonprogramming linguists and students of modern English, particularly teachers developing class exercises. Researchers in stylistics like myself will be grateful to the TOSCA group for a tool for studying an author's distinctive or shared syntactic habits. Even experienced computational linguists stand to benefit from having LDB and its corpora available as at-hand reference materials for points-of-usage not illustrated by standard grammars and dictionaries.

By all means, consider adding this valuable book and its program and corpus to your electronic library.

References

Aarts, Flor, and Aarts, Jan (1982). *English Syntactic Structures: Functions and Categories in Sentence Analysis*. London: Pergamon Press.

De Haan, Pieter (1989). *Postmodifying Clauses in the English Noun Phrase: A Corpus-Based Study*. Amsterdam: Rodopi.

Lancashire, Ian (1991). "Corpus linguistics." In *The Humanities Computing Yearbook 1989–1990: A Comprehensive Guide to Software and Other Resources*, edited by Ian Lancashire. Oxford: Clarendon Press.

Quirk, Randolph; Greenbaum, Sidney; Leech, Geoffrey; and Svartik, Jan (1972). *A Grammar of Contemporary English*. Longman.

Van Halteren, Hans (1989). "Possible use of syntactic analysis trees in syntax teaching." In *Computer Applications in Language Learning*, edited by Theo Bongaerts. Dordrecht: Foris, 147–161.

Ian Lancashire is Professor of English and Director of the Centre for Computing in the Humanities at the University of Toronto. He is a member of the Literary Texts Group of the Text Encoding Initiative. His research interests include medieval and Renaissance English drama, dictionaries and literature, bibliography, and literary content and style analysis. Lancashire's address is Department of English, New College, University of Toronto, Toronto, Ontario, Canada M5S 1A1; e-mail ian@epas.utoronto.ca.

3 *The ICAME Journal: International Computer Archive of Modern English*; edited by Stig Johansson, Department of English, University of Oslo, P.O. Box 1003, Blindern, N-0315 Oslo 3, Norway; published by the Norwegian Computing Centre for the Humanities, Harald Harfagres gate 31, P.O. Box 53, Universitetet, N-5027 Bergen, Norway.